

Sanjay Jain  
Hans Ulrich Simon  
Etsuji Tomita (Eds.)

LNAI 3734

# Algorithmic Learning Theory

16th International Conference, ALT 2005  
Singapore, October 2005  
Proceedings



 Springer

Lecture Notes in Artificial Intelligence 3734

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Sanjay Jain Hans Ulrich Simon  
Etsuji Tomita (Eds.)

# Algorithmic Learning Theory

16th International Conference, ALT 2005  
Singapore, October 8-11, 2005  
Proceedings



Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA  
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Sanjay Jain  
National University of Singapore  
School of Computing, Department of Computer Science  
117543 Singapore  
E-mail: sanjay@comp.nus.edu.sg

Hans Ulrich Simon  
Ruhr-Universität Bochum  
Fakultät für Mathematik, Lehrstuhl Mathematik und Informatik  
44780 Bochum, Germany  
E-mail: simon@lmi.ruhr-uni-bochum.de

Etsuji Tomita  
The University of Electro-Communications  
Department of Information and Communication Engineering  
Chofugaoka 1-5-1, Chofu, Tokyo 182-8585, Japan  
E-mail: tomita@ice.uec.ac.jp

Library of Congress Control Number: 2005933042

CR Subject Classification (1998): I.2.6, I.2.3, F.1, F.2, F.4, I.7

ISSN 0302-9743  
ISBN-10 3-540-29242-X Springer Berlin Heidelberg New York  
ISBN-13 978-3-540-29242-5 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media  
springeronline.com

© Springer-Verlag Berlin Heidelberg 2005  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 11564089 06/3142 5 4 3 2 1 0

# Preface

This volume contains the papers presented at the 16th Annual International Conference on Algorithmic Learning Theory (ALT 2005), which was held in Singapore (Republic of Singapore), October 8–11, 2005. The main objective of the conference is to provide an interdisciplinary forum for the discussion of the theoretical foundations of machine learning as well as their relevance to practical applications. The conference was co-located with the 8th International Conference on Discovery Science (DS 2005). The conference was also held in conjunction with the centennial celebrations of the National University of Singapore.

The volume includes 30 technical contributions, which were selected by the program committee from 98 submissions. It also contains the ALT 2005 invited talks presented by Chih-Jen Lin (National Taiwan University, Taipei, Taiwan) on “Training Support Vector Machines via SMO-type Decomposition Methods,” and by Vasant Honavar (Iowa State University, Ames, Iowa, USA) on “Algorithms and Software for Collaborative Discovery from Autonomous, Semantically Heterogeneous, Distributed, Information Sources.” Furthermore, this volume includes an abstract of the joint invited talk with DS 2005 presented by Gary L. Bradshaw (Mississippi State University, Starkville, USA) on “Invention and Artificial Intelligence,” and abstracts of the invited talks for DS 2005 presented by Ross D. King (The University of Wales, Aberystwyth, UK) on “The Robot Scientist Project,” and by Neil Smalheiser (University of Illinois at Chicago, Chicago, USA) on “The Arrowsmith Project: 2005 Status Report.” The complete versions of these papers are published in the DS 2005 proceedings (Lecture Notes in Computer Science Vol. 3735).

Since 1999, ALT has been awarding the E. Mark Gold Award for the most outstanding paper by a student author. This year the award was given to Rotem Bennet for his paper, “Learning Attribute-Efficiently with Corrupt Oracles,” co-authored by Nader Bshouty.

This conference was the 16th in a series of annual conferences established in 1990. Continuation of the ALT series is supervised by its steering committee consisting of: Thomas Zeugmann (Hokkaido University, Sapporo, Japan), Chair; Arun Sharma (Queensland University of Technology, Brisbane, Australia), Co-chair; Naoki Abe (IBM Thomas J. Watson Research Center, Yorktown, USA); Klaus Peter Jantke (FIT Leipzig e.V., Germany); Roni Khardon (Tufts University, Medford, USA); Phil Long (Columbia University, New York, USA); Hiroshi Motoda (Osaka University, Japan); Akira Maruoka (Tohoku University, Sendai, Japan); Luc De Raedt (Albert-Ludwigs-University, Freiburg, Germany); Takeshi Shinohara (Kyushu Institute of Technology, Iizuka, Japan); Osamu Watanabe (Tokyo Institute of Technology, Japan).

We would like to thank all the individuals and institutions that contributed to the success of the conference: the authors for submitting papers; the invited

speakers for accepting our invitation and lending us their insight into recent developments in their research areas; the Lee Foundation and AOARD (Asian Office of Aerospace Research and Development, US Air Force) for their generous financial support; the Division of Computer Science of Hokkaido University for maintaining the ALT 2005 Web-page together with software and for providing the poster; the Mathematics Department of the Ruhr-University Bochum for printing and distributing the posters; the Institute for Theoretical Computer Science, University of Lübeck, where Frank Balbach provided useful software; and the School of Computing, National University of Singapore for providing secretarial support, seed funding and equipment; as well as MBZ Marketing Büro Zeugmann for making the ALT 2005 Logo.

We are very grateful to Thomas Zeugmann for his continuous support including maintenance of the ALT 2005 Program Committee Web-page, and for providing his experience and active help in the process of publishing the proceedings.

Furthermore, we would like to express our gratitude to all program committee members for their hard work in reviewing the submitted papers and participating in on-line discussions. We are also grateful to the external referees whose reviews made a considerable contribution to this process.

We would also like to thank the DS 2005 chairs Achim Hoffmann (PC Chair, University of New South Wales, Australia), Tobias Scheffer (PC Chair, Humboldt University, Berlin, Germany) and Hiroshi Motoda (Conference Chair, Osaka University, Japan) for their effort in coordinating with ALT 2005, and Lee Wee Sun (National University of Singapore, Singapore) for his excellent work as the local arrangements chair. Last but not least, Springer provided excellent support in preparing this volume.

August 2005

Sanjay Jain  
Hans Ulrich Simon  
Etsuji Tomita

# Organization

## Conference Chair

Sanjay Jain  
National University of Singapore, Republic of  
Singapore

## Program Committee

Hans Ulrich Simon	Ruhr-Universität Bochum, Germany (Chair)
Etsuji Tomita	The University of Electro-Communications, Japan (Chair)
Dana Angluin	Yale University, USA
Hiroki Arimura	Hokkaido University, Japan
John Case	University of Delaware, USA
Nello Cristianini	University of California at Davis, USA
Victor Dalmau	Universitat Pompeu Fabra, Spain
Claudio Gentile	Università dell'Insubria, Italy
Vasant Honavar	Iowa State University, USA
Satoshi Kobayashi	The University of Electro-Communications, Japan
Phil Long	Columbia University, USA
Frank Stephan	National University of Singapore
Esko Ukkonen	University of Helsinki, Finland
Vladimir Vovk	Royal Holloway, UK
Manfred Warmuth	University of California at Santa Cruz, USA
Kenji Yamanishi	NEC Corporation, Japan
Takashi Yokomori	Waseda University, Japan
Thomas Zeugmann	Hokkaido University, Japan

## Local Arrangements

Lee Wee Sun  
National University of Singapore, Republic of  
Singapore

## Subreferees

Ilkka Autio

Francis Bach

Asa Ben-Hur

Tijl de Bie

Jorge Castro

Nicolò Cesa-Bianchi

Jiang Chen

Ricard Gavaldà

Kazuhiro Hotta

Matti Kääriäinen

Jyrki Kivinen

Mikko Koivisto

Dima Kuzmin

Michael Last

Jun Liao

Huma Lodhi

Satoshi Morinaga

Sayan Muckherjee

Jaakko Peltonen

Jan Poland

Roman Rosipal

Yasubumi Sakakibara

Johan Suykens

Yasuhiro Tajima

Haruhisa Takahashi

Jun-ichi Takeuchi

Noriyuki Tanida

Jason Weston

## Sponsoring Institutions

Lee Foundation, Singapore, Republic of Singapore

Asian Office of Aerospace Research and Development, US Air Force

School of Computing, National University of Singapore

Fakultät für Mathematik, Ruhr-Universität Bochum

Division of Computer Science, Hokkaido University

Institute for Theoretical Computer Science, University of Lübeck

MBZ Marketing Büro Zeugmann



# Table of Contents

Editors' Introduction	1
<b>Invited Papers</b>	
Invention and Artificial Intelligence	10
The Arrowsmith Project: 2005 Status Report	11
The Robot Scientist Project	12
Algorithms and Software for Collaborative Discovery from Autonomous, Semantically Heterogeneous, Distributed Information Sources	13
Training Support Vector Machines via SMO-Type Decomposition Methods	45
<b>Kernel-Based Learning</b>	
<b>Regular Contributions</b>	
Measuring Statistical Dependence with Hilbert-Schmidt Norms	63
An Analysis of the Anti-learning Phenomenon for the Class Symmetric Polyhedron	78
<b>Bayesian and Statistical Models</b>	
Learning Causal Structures Based on Markov Equivalence Class	92

Stochastic Complexity for Mixture of Exponential Families  
in Variational Bayes ..... 107

ACME: An Associative Classifier Based on Maximum Entropy Principle  
..... 122

**PAC-Learning**

Constructing Multiclass Learners from Binary Learners:  
A Simple Black-Box Analysis of the Generalization Errors  
..... 135

On Computability of Pattern Recognition Problems  
..... 148

PAC-Learnability of Probabilistic Deterministic Finite State Automata  
in Terms of Variation Distance ..... 157

Learnability of Probabilistic Automata via Oracles  
..... 171

**Query-Learning**

Learning Attribute-Efficiently with Corrupt Oracles  
..... 183

Learning DNF by Statistical and Proper Distance Queries Under  
the Uniform Distribution ..... 198

Learning of Elementary Formal Systems with Two Clauses Using Queries  
..... 211

Gold-Style and Query Learning Under Various Constraints  
on the Target Class ..... 226

**Inductive Inference**

Non U-Shaped Vacillatory and Team Learning  
..... 241

Learning Multiple Languages in Groups	256
---------------------------------------	-----

## Language Learning

Inferring Unions of the Pattern Languages by the Most Fitting Covers	269
--	-----

Identification in the Limit of Substitutable Context-Free Languages	283
---	-----

Algorithms for Learning Regular Expressions	297
---	-----

## Learning and Logic

A Class of Prolog Programs with Non-linear Outputs Inferable from Positive Data	312
---	-----

Absolute Versus Probabilistic Classification in a Logical Setting	327
---	-----

## Learning from Expert Advice

Online Allocation with Risk Information	343
---	-----

Defensive Universal Learning with Experts	356
---	-----

On Following the Perturbed Leader in the Bandit Setting	371
---	-----

Mixture of Vector Experts	386
---------------------------	-----

## Online Learning

On-line Learning with Delayed Label Feedback	399
--	-----

Monotone Conditional Complexity Bounds on Future Prediction Errors ..... 414

**Defensive Forecasting**

Non-asymptotic Calibration and Resolution ..... 429

Defensive Prediction with Expert Advice ..... 444

Defensive Forecasting for Linear Protocols ..... 459

**Teaching**

Teaching Learners with Restricted Mind Changes ..... 474

**Author Index** ..... 491

# Editors' Introduction

Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita

“Learning” is a complex phenomenon that is studied in different scientific disciplines. A computer program with the ability to “learn” contains mechanisms for gathering and evaluating information and, consequently, for improving its performance. Algorithmic Learning Theory provides a mathematical foundation for the study of learning programs. It is concerned with the design and analysis of learning algorithms. The analysis proceeds in a formal model such as to provide measures for the performance of a learning algorithm or for the inherent hardness of a given learning problem.

The variety of applications for algorithms that learn is reflected in the variety of formal learning models. For instance, we can distinguish between a passive mode of “learning from examples” and active modes of learning where the algorithm has more control about the information that is gathered. As for learning from examples, a further decision is whether we impose statistical assumptions on the sequence of examples or not. Furthermore, we find different success criteria in different models (like “approximate learning” versus “exact learning”).

The papers collected in this volume offer a broad view on the current research in the field including studies on several learning models (such as kernel-based learning, pac-learning, query learning, inductive inference, learning from expert advice, online learning and teaching). It comes without saying that these models are getting more and more refined as a response to new challenges in computer science (like taking advantage of the large amount of real-world data that become available in digital form, or like serving the growing need for adaptive techniques in the management and control of Internet applications).

The volume is structured as follows. It first presents the invited lectures and then the regular contributions. The latter are grouped according to thematic categories. In the remainder of the introduction, we provide the reader with a rough “road map”.

The invited lecture for ALT 2005 and DS 2005 by Gary Bradshaw compares Invention and Artificial Intelligence and points out some analogies. Starting from two case studies, the invention of the air-plain (1799–1909) and the invention of a model rocket by a group of high school students in rural West Virginia in the late 1950's, it is argued that general principals of invention may be applied to expedite the development of AI systems.

The invited lecture for DS 2005 by Neil R. Smalheiser gives a report on the current status of the Arrowsmith Project. The roots of this project trace back to the 1980's when Don Swanson proposed the concept of “undiscovered public knowledge” and published several examples in which two disparate literatures held complementary pieces of knowledge that, when brought together, made compelling and testable predictions about potential therapies for human disorders. In continuation of this work in the 1990's, Smalheiser and Swanson created

a computer-assisted search strategy (“Arrowsmith”). The lecture reviews the development until today.

Ross D. King, presenting the second invited lecture for DS 2005, pursues the question whether it is possible to automate the scientific process. This question is of increasing importance because, in many scientific areas, data are being generated much faster than they can be effectively analyzed by humans. In his lecture, King describes a physically implemented robotic system that applies techniques from artificial intelligence to carry out cycles of scientific experimentation. This is fleshed out by describing the application of the system to complex research problems in bioinformatics.

The invited lecture for ALT 2005 by Vasant Honavar (co-authored by Doina Caragea, Jun Zhang, Jie Bao, and Jyotishman Pathak) is concerned with the problem of data-driven knowledge acquisition and decision-making. Honavar describes the hurdles represented by massive size, semantic heterogeneity, autonomy, and distributed nature of the data repositories. He introduces some of the algorithmic and statistical problems that arise in such a setting and presents algorithms for learning classifiers from distributed data that offer rigorous performance guarantees relative to their centralized counterparts. The lecture furthermore presents techniques that help to cope with the problem of semantic heterogeneity.

The second invited lecture for ALT 2005 by Chih-Jen Lin (co-authored by Pai-Hsuen Chen and Rong-En Fan) concerns the convex quadratic optimization problem that has to be solved by Support Vector Machines (SVMs). Since the underlying cost matrix is high-dimensional and dense, this cannot be done by classical methods (like, for example, “Interior Point”). Instead a decomposition method is used. It proceeds iteratively and has, in each iteration, to solve a small-dimensional subproblem. Lin focuses on decomposition methods of the SMO-type and elaborates how the implementation of the procedure for “Working Set Selection” affects the speed of convergence to the optimal solution. In a sense, Lin’s lecture offers the opportunity of looking deeply inside a widely used learning machine.

We now turn our attention to the regular contributions contained in this volume.

During the last decade there has been a lot of interest in learning systems that express the “similarity” between two “instances” as an inner product of vectors in an appropriate feature space. The inner product operation is often not carried out explicitly, but reduced to the evaluation of a so-called kernel function which operates on instances of the original space. This offers the opportunity to handle high-dimensional feature spaces in an efficient manner. This strategy, introduced by Vapnik and co-workers in connection with the so-called Support Vector Machine, is a theoretically well founded and very powerful method that, in the years since its introduction, has already outperformed many other learning systems in a wide variety of applications.

Gretton, Bousquet, Smola and Schölkopf continue a line of research where the kernel-based approach is used as a tool for the detection of statistical de-

dependencies. More precisely, they provide a measure of statistical dependence between random variables based on the Hilbert-Schmidt norm of a cross covariance operator. Their method has some theoretically appealing features while being experimentally competitive to existing methods.

Kowalczyk and Chapelle study the surprising phenomenon of “anti-learning”, where data sets are represented in such a way that some well-known standard learning techniques lead to hypotheses performing much worse than random guessing. While there seem to exist “real-life” data sets which are of that type, the authors consider artificial data whose analysis sheds some light on this (at first glance) counterintuitive phenomenon. They identify some abstract properties of a given positive definite kernel which inevitably lead to anti-learning (for some standard learning techniques including “large-margin classification” or “nearest neighbour”). They furthermore explain which kernel-transformations convert “anti-learning” into “learning” and which do not.

Causal networks are graphical representations for “causality” between random variables. Like Bayesian networks or other graphical models they represent knowledge being not as strict as formal logical statements but being quite helpful for what is called “reasoning under uncertainty”. Bayesian Inference leads to decisions that minimize a risk function. Bayesian learning refers to the problem of inferring the unknown parameters of a distribution (chosen from a known parameterized class of distributions). Typically the a priori distribution for the unknown parameters gives support to a wide range of parameters, whereas the a posteriori distribution is peaked around the true parameter values.

By means of statistical information we can determine causal networks only up to Markov-equivalence. An equivalence class can be represented by a chordal chain graphs that contains directed and undirected edges. Each undirected edge represents two mutually correlated variables where we cannot know whether causality (if any) goes in one or the other direction. He, Geng and Liang describe a hierarchical partitioning of a class of Markov-equivalent causal networks (given by a chordal chain graph) into finer and finer subclasses up to the point where the partition consists of the single causal networks. They prove that, at each stage of the refinement process, an equivalence class can be again represented as a chordal chain graph.

Variational Bayesian Learning results from Bayesian Learning by introducing a simplifying assumption (in case there are hidden variables) that makes the approach computationally more tractable. Empirically it is known to have good generalization performance in many applications. Watanabe and Watanabe provide some additional theoretical support by proving lower and upper bounds on the stochastic complexity in the Variational Bayesian learning of the mixture of exponential families.

As for classification tasks, the Bayes classifier chooses the label that has the maximum a posteriori probability. In order to apply the formula of Bayes, we have to know the a priori probability and (an approximation of) the class-conditional densities. Thonangi and Pudi suggest a new method for the approxi-

mation of the class-conditional densities: they use a class-conditional probability model of the data (based on certain conjunctions of binary features and the corresponding relative frequencies) and determine the density function of maximum entropy that satisfies this model.

In the PAC-learning model, the learner receives as input training examples, drawn at random according to an unknown distribution and labeled according to an unknown target function  $f$ , and returns a “hypothesis”  $h$  that (with high probability of success) is a close approximation of  $f$ . While the first papers on pac-learning focussed on binary classification problems with the probability of misclassification as an underlying pseudo-metric, there have been many extensions of the basic model since then.

Fakcharoenphol and Kijirikul revisit the well-studied problem of converting base binary learners (coping with binary classification problems) into multi-class learners. They reconsider various classical constructions, including “one-versus-all” and the “(adaptive) directed acyclic graph approach”, and come up with some new generalization error bounds.

The paper by Ryabko relates “PAC-learnability” and “computability” by showing the following result: the number of examples needed by a recursive PAC-learner to learn a computable function cannot be upper-bounded by any computable function. The result is obtained by applying a negative result on data compression based on Kolmogorov complexity.

The papers by Palmer and Goldberg and by Guttman, Vishwanathan and Williamson are both dealing with so-called Probabilistic Deterministic Finite State Automata (PDFA). A PDFA, in contrast to a DFA (its deterministic counterpart), performs random state transitions and thus represents a probability distribution over strings. In a recent paper from 2004, it was shown by Clark and Thollard that the class of PDFAs is polynomially pac-learnable where the polynomial (which bounds sample size and run-time) depends on the size of the target PDFA, a measure for the “distinguishability” of states and the expected length of strings generated from any state. Their pseudo-metric (measuring the distance between target distribution and the hypothesis) is the KL-divergence. By passing from KL-divergence to the variation distance (still useful for pattern classification tasks), Palmer and Goldberg are able to considerably simplify the proof for PAC-learnability and to remove the dependence on the expected string length. Guttman, Vishwanathan and Williamson extend the result by Clark and Thollard into another direction by generalizing the notion of “distinguishability”.

In the query-learning model, the learner is allowed to ask certain queries to an oracle. The learning performance is measured by the number and type of queries needed to exactly (sometimes approximately) identify an unknown target concept  $f$  (where “concept” means “binary-valued function”). Among the most popular query types are “membership queries (MQs)” (asking for the class label of an instance) and “equivalence queries (EQs)” (asking for a “counterexample” to the current hypothesis). Statistical queries can be used to get a statistical estimate for the probability of an event that involves a labeled random example. This query type plays a special role because an algorithm



that learns by means of statistical queries can be converted into a noise-tolerant pac-learning algorithm.

Bennet and Bshouty investigate the possibility of learning “attribute-efficiently” (such that the time-bound grows only sub-linearly with the number of irrelevant attributes) with “corrupted oracles” (whose answers may occasionally be wrong or missing). Their main result is that an attribute-efficient learner expecting a perfectly reliable oracle of type MQ or EQ can be converted into an attribute-efficient learner that copes with corrupted oracles. (This extends a result from FOCS 2004, where attribute-efficiency had not been an issue.)

As shown by Köbler and Lindner in their paper presented at ALT 2002, DNF-formulas can be learned by statistical queries under the uniform distribution. The paper by Lindner in this volume improves on this result by showing that the learning algorithm can cope with larger inaccuracies of the empirical estimates provided by the oracle. A similar remark holds for the related model of “learning by proper distance queries”.

Kato, Matsumoto and Miyahara discuss the learnability of certain logic programs, called “Elementary Formal Systems (EFS)”, that represent formal languages. These programs can, for instance, succinctly represent the popular “regular pattern languages” whose learnability has been well studied in the past. It is shown that EFS-definable languages can be learned in polynomial time with membership and superset queries. Furthermore, it is shown that various other combinations of query types are insufficient for this purpose.

Jain, Lange and Zilles extend work by Lange and Zilles from COLT 2004 and ALT 2004. The latter papers revealed surprising relations between Gold-style and query learning of indexable classes of recursive languages. In their contribution to this volume, they examine (arbitrary) classes of recursively enumerable languages and analyze which of the relations still hold. It turns out that, although some of the relations are lost, there is still a nice hierarchy with several cross-connections between Gold-style and query learning.

In Gold’s model of “learning in the limit from positive data”, the learner should identify an unknown recursive (or recursively enumerable) target language  $L$  (where  $L$  belongs to a class  $C$  that is known to the learner). To this end, it receives a “text” (basically an infinite list of the words in  $L$ ). It keeps track of a (description of a) “hypothesis” that is updated after the reception of the next word from the text. We say that  $C$  is learned in the limit, if the learner is able to identify every language  $L \in C$  after finitely many steps. Within this model, one distinguishes between “explanatory learning” (learner converges to a description of the target concept) and “behaviourally correct learning” (learner converges to the target concept but may, in principle, use arbitrarily many descriptions for it). Between these two extremes, we find the possibility of converging to the target and using (in the limit) at most  $k$  different descriptions. This leads to what is called “vacillatory learning” (actually an infinite hierarchy of models that is in-between explanatory and behaviourally correct learning). Another notion of interest is “U-shaped” versus “Non U-shaped” learning, where a learner exhibits U-shaped behaviour if it passes from the cor-

rect hypothesis to an incorrect-one, but later returns to the correct-one. This kind of learning behaviour is also discussed in the cognitive psychology literature.

Carlucci, Case, Jain and Stephan continue a line of research that explores whether U-shaped learning behaviour is logically necessary within the model of learning in the limit from positive data. It was known that U-shaped learning behaviour is, indeed, necessary for “behaviourally correct learning” but not for “explanatory learning”. The authors refine this result in several ways. They show that non U-shaped vacillatory learning collapses to explanatory learning. Moreover, a U-shaped learner on the second level of the hierarchy can be transformed into a non U-shaped behaviourally correct learner, whereas a transformation of this kind is not in general possible if the U-shaped learner is located on the third (or higher) level of the hierarchy.

Jain and Kinber consider the following variant of learning in the limit from positive data: given a text for a union of  $n$  (pairwise disjoint) languages, find in the limit  $m$  grammars such that each of them generates some union of the  $n$  languages and every of the  $n$  languages occurs in exactly one of these unions. The paper provides considerable insight in this model.

The next group of papers is concerned with learning languages in the limit from positive data. It is known, for instance, that bounded unions of non-erasable pattern languages and erasable regular pattern languages are learnable in this model whereas regular languages are not. Furthermore, a learning strategy that selects a language from the version space that is minimal with respect to inclusion ensures identification in the limit provided that each language  $L$  in the target class has a characteristic set (i.e., a finite subset  $S$  with  $L$  as the unique minimal language from the target class that contains  $S$ ).

Ng and Shinohara introduce another notion of minimality that refers to the number of elements in a language being shorter than a specified length. They show that this notion of minimality and a suitably adapted notion of a characteristic set fit the same purpose for learning languages in the limit from positive data as the classical notions (with a range of applicability that has not become smaller). Thus, the selection of a language from the version space that is minimal in the new sense is an interesting alternative to the selection of an inclusion-minimal language.

Clark and Eyraud consider the class of substitutable context-free languages (building on the notion of substitutability by Harris from the 1950’s). They show that every substitutable context-free language has a characteristic set, which implies their learnability in the limit from positive data.

Fernau identifies a non-trivial subclass of regular languages (representable by special regular expressions called “simple looping expressions” in the paper) and presents an algorithm that has a polynomial update time and learns simple looping expressions in the limit from positive data.

The following group of papers again deals with the model of learnability in the limit from positive data. The target classes under consideration are Prolog programs and logical structures, respectively.

The paper by Rao continues with the research project of identifying fragments of Prolog that are learnable in the limit from positive data. The new fragment studied in the paper (“recursion bounded” Prolog programs) is incomparable with the previously identified fragments and contains programs that neither belong to the class of “linearly-moded programs” (introduced by Rao himself) nor to the class of “safe programs” (introduced by Martin and Sharma).

Jain, Martin and Stephan discuss the problem of learning logical structures in the limit from positive examples, where the positive examples are logical formulas that are satisfied by the unknown target structure. They prove learnability in this model except for a set of logical structures of measure zero.

In the classical model of learning an unknown binary sequence from expert advice, the learner predicts the sequence bitwise such that learning proceeds in rounds. In every round, it first receives  $N$  expert predictions  $p_1, \dots, p_N \in [0, 1]$ . Then, it makes its own prediction  $p \in [0, 1]$  (in dependence of the experts predictions). At the end of the round, it receives the next bit  $b$  of the sequence and suffers loss  $|p - b|$  (the probability of making a wrong prediction when 1 is chosen with probability  $p$ ). The learner is exposed to a worst-case analysis (like playing against an intelligent adversary) in an “agnostic” setting (where any bit sequence is conceivable). The goal of the learner is to keep the difference between its own total loss and the total loss of the best expert as small as possible. This difference is called the “regret” of the learner. Many version of this basic model are found in the literature. The goal within the theoretical analysis is finding tight bounds on the smallest possible regret.

The paper by Harada, Takimoto and Maruoka studies the problem of sequentially assigning a probability vector  $p$  over  $k$  options. Then a cost vector  $c$  over the  $k$  options is revealed and the learner suffers loss  $p^\top c$  (cost averaged over  $p$ ). Vector  $p$  is chosen in dependence of (an average of)  $N$  probability vectors that represent the “expert advice” in this setting. The authors generalize the known performance bounds for two well-known algorithms (Hedge Algorithm and Aggregating Algorithm) to the case where each available option has different range of possible losses. As mentioned in the paper, the setting can be applied to the problem of finding a route with minimum total expected delay through a network.

The paper by Poland and Hutter and the paper by Kujala and Elomaa both study a variation of a problem that was introduced by Kalai and Vempala at COLT 2003. The latter authors consider a broad class of online optimization problems that contains “learning from expert advice” as a special case. They furthermore present and analyze a strategy named “follow the perturbed leader”. Poland and Hutter extend this work in the bandit setting to the case of a countably infinite pool of experts, and to the case in which, instead of uniform upper bounds on the cost in each round, there are (slowly) increasing bounds. Kujala and Elomaa likewise consider the framework of Kalai and Vempala. They apply the strategy of Mc Mahan and Blum from COLT 2004 (which is designed such as to cope with an adaptive adversary) to the more restricted case of an oblivious adversary. This leads to stronger regret bounds.

The paper by Henderson, Shawe-Taylor and Žerovnik generalizes the results in the standard model of learning a binary sequence with expert advice to the multi-dimensional case of a binary vector sequence.

In the classical model of “online learning”, there is a class of target functions that is fixed in advance. Learning proceeds in rounds. In every round, the learner first receives an instance  $x$ , then makes a prediction  $\tilde{y}$  and, at the end of the round, receives the true label  $y = f(x)$  and suffers loss  $l(\tilde{y}, y)$ . In case of a Boolean target class, a natural choice for  $l$  is the zero-one loss (such that the accumulated loss counts the number of mistakes). Many versions of this basic model are found in the literature. The goal within the analysis is finding the best possible loss bounds (or mistake bounds, respectively) for a given class of target functions.

The paper by Mesterharm extends the standard worst-case online learning framework by allowing delayed label feedback. It is shown how a given standard online learner can be converted into a learner being able to cope with delay and analyzes how the mistake bounds are related. The analysis is provided for both the adversarial setting and the setting when the instances are drawn according to a (slowly) drifting distribution.

Chernov and Hutter establish bounds on future prediction errors in online prediction of computably stochastic sequences. This is a setting where fairly tight bounds exist for the total number of errors leading one to suspect that the future errors must necessarily decrease with time. The main contribution of this paper is a quantitative treatment of this observation in terms of (a variant of) Kolmogorov complexity.

The next series of papers investigates the problem of predicting (in an online fashion) the binary label associated with a given instance. The setting is agnostic, i.e., it is assumed that there is a hidden function (or distribution) which maps instances to labels. Instead of using a comparison class or expert advice (such that the learner competes with the best function from the comparison class or with the best expert), the analysis aims at verifying two abstract properties of the Forecaster’s prediction algorithm: being “well-calibrated” and having “high resolution”. If a prediction  $p \in [0, 1]$  is viewed as the Forecaster’s “confidence” that the (unknown) binary label associated with a given instance  $x$  is 1 (where the label is revealed by the adversary after the Forecaster made its commitment), then being well calibrated roughly means that the Forecaster’s confidence estimates are accurate on average. Having high resolution roughly means that the Forecaster is “sufficiently specific” to the given instance  $x$ .

Vovk describes a prediction algorithm for the Forecaster that provably is well-calibrated and has high resolution. The algorithm is kernel-based. The analysis is quite involved with results given in terms of the Fermi-Sobolev norm of Lipschitzian functions.

The next paper in the volume (again by Vovk) demonstrates how one can derive classical results in the “learning with expert advice” setting starting from

results of the preceding paper. Here, the “expert class” is infinite-dimensional. Basically the learner competes with any decision rule of bounded Fermi-Sobolev norm. The main result is quite general and applies to a wide variety of decision-theoretic scenarios.

The third result in this series about “defensive forecasting” is the paper by Vovk, Nouretdinov, Takemura and Shafer which goes one step further. It considers multi-class forecasting and inserts an additional player called “Skeptic”. Loosely speaking, the Skeptic tries to make money on any lack of agreement between Forecaster and Reality. The main result basically states that, for any continuous strategy of Skeptic, there exists a strategy of Forecaster that does not allow Skeptic’s capital to grow, regardless of what Reality is doing.

In the teaching model, a teacher delivers examples to a learner. The examples are labeled according to a target function (known to the teacher but unknown to the learner). In order to avoid collusion between a learner and a teacher (such as agreement on a coding scheme which allows the teacher to encode a representation of the target function), the success criterion for teaching must be carefully defined. One popular definition prescribes that the target should be the only function which is left in the version space after all examples in the teaching sequence have been processed. It turns out, unfortunately, that, with this definition, many harmless looking concept classes may require intolerably long teaching sequences in the worst-case. For this reason, one finds several alternative success criteria in the literature.

Balbach and Zeugmann make a new suggestion for a teaching model where collusion is avoided in a different fashion. They augment the concept class by a neighbourhood relation and let the teaching process proceed in rounds. In each round, the teacher presents a new example  $x$ . If the learner’s current hypothesis errs on  $x$ , the learner may pass from  $h$  to a new hypothesis  $h'$  from the neighbourhood (where  $h'$  must be chosen such as to make a minimal number of mistakes on the seen examples). Balbach and Zeugman analyze several variants of this approach and relate it to the classical models. It is a nice feature of their approach that it leads to meaningful statements about teachability of classes over an infinite domain.

# Invention and Artificial Intelligence\*

Gary Bradshaw

Psychology Department, P.O. Box 6161,  
Mississippi State University, MS 39762, USA  
[glb2@ra.msstate.edu](mailto:glb2@ra.msstate.edu)

**Abstract.** Invention, like scientific discovery, sometimes occurs through a heuristic search process where an inventor seeks a successful invention by searching through a space of inventions. For complex inventions, such as the airplane or model rockets, the process of invention can be expedited by an appropriate strategy of invention. Two case studies will be used to illustrate these general principles: the invention of the airplane (1799-1909) and the invention of a model rocket by a group of high school students in rural West Virginia in the late 1950's. Especially during the invention of the airplane, inventors were forced to make scientific discoveries to complete the invention. Then we consider the enterprise of artificial intelligence and argue that general principles of invention may be applied to expedite the development of AI systems.

---

\* The full version of this paper is published in the Proceedings of the 8th International Conference on Discovery Science, Lecture Notes in Artificial Intelligence Vol. 3735.

# The Arrowsmith Project: 2005 Status Report\*

Neil R. Smalheiser

UIC Psychiatric Institute, University of Illinois-Chicago,  
MC912, 1601 W. Taylor Street, Chicago, IL 60612, USA  
neils@uic.edu

**Abstract.** In the 1980s, Don Swanson proposed the concept of “undiscovered public knowledge,” and published several examples in which two disparate literatures (i.e., sets of articles having no papers in common, no authors in common, and few cross-citations) nevertheless held complementary pieces of knowledge that, when brought together, made compelling and testable predictions about potential therapies for human disorders. In the 1990s, Don and I published more predictions together and created a computer-assisted search strategy (“Arrowsmith”). At first, the so-called one-node search was emphasized, in which one begins with a single literature (e.g., that dealing with a disease) and searches for a second unknown literature having complementary knowledge (e.g. that dealing with potential therapies). However, we soon realized that the two-node search is better aligned to the information practices of most biomedical investigators: in this case, the user chooses two literatures and then seeks to identify meaningful links between them. Could typical biomedical investigators learn to carry out Arrowsmith analyses? Would they find routine occasions for using such a sophisticated tool? Would they uncover significant links that affect their experiments? Four years ago, we initiated a project to answer these questions, working with several neuroscience field testers. Initially we expected that investigators would spend several days learning how to carry out searches, and would spend several days analyzing each search. Instead, we completely re-designed the user interface, the back-end databases, and the methods of processing linking terms, so that investigators could use Arrowsmith without any tutorial at all, and requiring only minutes to carry out a search. The Arrowsmith Project now hosts a suite of free, public tools. It has launched new research spanning medical informatics, genomics and social informatics, and has, indeed, assisted investigators in formulating new experiments, with direct impact on basic science and neurological diseases.

---

\* The full version of this paper is published in the Proceedings of the 8th International Conference on Discovery Science, Lecture Notes in Artificial Intelligence Vol. 3735.

# The Robot Scientist Project\*

Ross D. King

Department of Computer Science, The University of Wales,  
Aberystwyth Penglais, Aberystwyth, Ceredigion, Wales, U.K.  
rdk@aber.ac.uk

**Abstract.** The question of whether it is possible to automate the scientific process is of both great theoretical interest and increasing practical importance because, in many scientific areas, data are being generated much faster than they can be effectively analysed. We describe a physically implemented robotic system that applies techniques from artificial intelligence to carry out cycles of scientific experimentation. The system automatically originates hypotheses to explain observations, devises experiments to test these hypotheses, physically runs the experiments using a laboratory robot, interprets the results to falsify hypotheses inconsistent with the data, and then repeats the cycle. We applied this system to the determination of gene function using deletion mutants of yeast (*Saccharomyces cerevisiae*) and auxotrophic growth experiments. We built and tested a detailed logical model (involving genes, proteins and metabolites) of the aromatic amino acid synthesis pathway. In biological experiments that automatically reconstruct parts of this model, we show that an intelligent experiment selection strategy is competitive with human performance and significantly outperforms, with a cost decrease of 3-fold and 100-fold (respectively), both cheapest and random-experiment selection. We have now scaled up this approach to discover novel biology. To achieve this we combined our logical reasoning approach with bioinformatics. We have built a logical model of all known *S. cerevisiae* metabolism. In this model there are still reactions where the gene encoding the enzyme is unknown.

We demonstrate that we can automatically hypothesize these genes, and generate “wet” biological evidence that either confirms or contradicts this hypothesis. This approach can also be used to automatically test biological genome annotations.

---

\* The full version of this paper is published in the Proceedings of the 8th International Conference on Discovery Science, Lecture Notes in Artificial Intelligence Vol. 3735.



# Algorithms and Software for Collaborative Discovery from Autonomous, Semantically Heterogeneous, Distributed Information Sources

Doina Caragea, Jun Zhang, Jie Bao, Jyotishman Pathak, and Vasant Honavar

Artificial Intelligence Research Laboratory,  
Center for Computational Intelligence, Learning, and Discovery,  
Department of Computer Science, Iowa State University,  
226 Atanasoff Hall, Ames, IA 50011  
[honavar@cs.iastate.edu](mailto:honavar@cs.iastate.edu)

**Abstract.** Development of high throughput data acquisition technologies, together with advances in computing, and communications have resulted in an explosive growth in the number, size, and diversity of potentially useful information sources. This has resulted in unprecedented opportunities in data-driven knowledge acquisition and decision-making in a number of emerging increasingly data-rich application domains such as bioinformatics, environmental informatics, enterprise informatics, and social informatics (among others). However, the massive size, semantic heterogeneity, autonomy, and distributed nature of the data repositories present significant hurdles in acquiring useful knowledge from the available data. This paper introduces some of the algorithmic and statistical problems that arise in such a setting, describes algorithms for learning classifiers from distributed data that offer rigorous performance guarantees (relative to their centralized or batch counterparts). It also describes how this approach can be extended to work with autonomous, and hence, inevitably semantically heterogeneous data sources, by making explicit, the ontologies (attributes and relationships between attributes) associated with the data sources and reconciling the semantic differences among the data sources from a user's point of view. This allows user or context-dependent exploration of semantically heterogeneous data sources. The resulting algorithms have been implemented in INDUS - an open source software package for collaborative discovery from autonomous, semantically heterogeneous, distributed data sources.

## 1 Introduction

Recent development of high throughput data acquisition technologies in a number of domains (e.g., biological sciences, environmental sciences, atmospheric sciences, space sciences, commerce) together with advances in digital storage, computing, and communications technologies have resulted in the proliferation of a multitude of physically distributed data repositories created and maintained by autonomous entities (e.g., scientists, organizations). The resulting increasingly

data rich domains offer unprecedented opportunities in computer assisted data-driven knowledge acquisition in a number of applications including in particular, data-driven scientific discovery in bioinformatics (e.g., characterization of protein sequence-structure-function relationships in computational molecular biology), environmental informatics, health informatics; data-driven decision making in business and commerce, monitoring and control of complex systems (e.g., load forecasting in electric power networks), and security informatics (discovery of and countermeasures against attacks on critical information and communication infrastructures). Machine learning algorithms [1, 2, 3, 4, 5, 6, 7] offer some of the most cost-effective approaches to knowledge acquisition (discovery of features, correlations, and other complex relationships and hypotheses that describe potentially interesting regularities) from large data sets. However, the applicability of current approaches to machine learning in emerging data rich applications in practice is severely limited by a number of factors:

- (a) Data repositories are large in size, dynamic, and physically distributed. Consequently, it is neither desirable nor feasible to gather all of the data in a centralized location for analysis. Hence, there is a need for efficient algorithms for learning from multiple distributed data sources without the need to transmit large amounts of data. In other domains, the ability of autonomous organizations to share raw data may be limited due to a variety of reasons (e.g., privacy considerations). In such cases, there is a need for knowledge acquisition algorithms that can learn from statistical summaries of data (e.g., counts of instances that match certain criteria) that are made available as needed from the distributed data sources in the absence of access to raw data.
- (b) Autonomously developed and operated data sources often differ in their structure and organization (relational databases, flat files, etc.) and the operations that can be performed on the data source (e.g., types of queries - relational queries, restricted subsets of relational queries, statistical queries, keyword matches; execution of user-supplied code to compute answers to queries that are not directly supported by the data source; storing results of computation at the data source for later use) and the precise mode of allowed interactions can be quite diverse. Hence, there is a need for theoretically well-founded strategies for efficiently obtaining the information needed for learning within the operational constraints imposed by the data sources.
- (c) Autonomously developed data sources differ in terms of their underlying ontological commitments [8], i.e., assumptions concerning the objects that exist in the world, the properties or attributes of the objects, the possible values of attributes, and their intended meaning, as well as the granularity or level of abstraction at which objects and their properties are described. The increasing need for information sharing between organizations, individuals and scientific communities have led to significant community-wide efforts aimed at the construction of ontologies in many areas: Gene Ontology - GO ([www.geneontology.org](http://www.geneontology.org)) [9] for molecular biology, Unified Medical Language System -UMLS ([www.nlm.nih.gov/research/umls](http://www.nlm.nih.gov/research/umls)) for health infor-

matics, Semantic Web for Earth and Environmental Terminology - SWEET ([sweet.jpl.nasa.gov](http://sweet.jpl.nasa.gov)) for geospatial informatics. While explicit declaration of the ontology associated with a data repository helps standardize the semantics to an extent, because the ontological commitments associated with a data source (and hence its implied semantics) are typically determined by the data source designers based on their understanding of the intended use of the data repository and because data sources that are created for use in one context or application often find use in other contexts or applications, semantic differences among autonomously designed, owned, and operated data repositories are simply unavoidable. Effective use of multiple sources of data in a given context requires reconciliation of such semantic differences from the user's perspective [10, 11]. Collaborative scientific discovery applications often require users to be able to analyze data from multiple, semantically disparate data sources there is no single privileged perspective that can serve all users, or for that matter, even a single user, in every context. Hence, there is a need for methods that can dynamically and efficiently extract and integrate information needed for learning (e.g., statistics) from distributed, semantically heterogeneous data based on user-specified ontologies and user-specified mappings between ontologies.

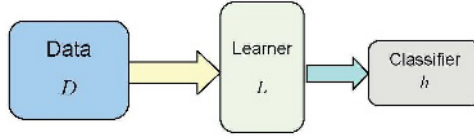
Against this background, we consider the problem of data driven knowledge acquisition from autonomous, distributed, semantically heterogeneous, data sources. The rest of this paper is organized as follows:

## 2 Learning from Distributed Data

### 2.1 Problem Formulation

Given a data set  $D$ , a hypothesis class  $H$ , and a performance criterion  $P$ , an algorithm  $L$  for learning (from centralized data  $D$ ) outputs a hypothesis  $h \in H$  that optimizes  $P$ . In pattern classification applications,  $h$  is a classifier (e.g., a decision tree, a support vector machine, etc.) (See Figure 1). The data  $D$  typically consists of a set of training examples. Each training example is an ordered tuple of attribute values, where one of the attributes corresponds to a class label and the remaining attributes represent inputs to the classifier. The goal of learning is to produce a hypothesis that optimizes the performance criterion e.g., minimizing classification error (on the training data) and the complexity of the hypothesis.

In a distributed setting, a data set  $D$  is distributed among the sites  $1, \dots, p$  containing data set fragments  $D_1, \dots, D_p$ . Two common types of data fragmentation are: horizontal fragmentation and vertical fragmentation. In the case of horizontal fragmentation, each site contains a subset of the data tuples that make up  $D$ , i.e.,  $D = \cup_{i=1}^p D_i$ . In the case of vertical fragmentation each site stores the subtuples of data tuples (corresponding to a subset of the attributes used to define data tuples in  $D$ ). In this case,  $D$  can be constructed by taking the  $\cup_{i=1}^p$  of the individual data sets  $D_1, \dots, D_p$  (assuming a unique identifier for each



**Fig. 1.** Learning from centralized data

data tuple is stored with the corresponding subtuples). More generally, the data may be fragmented into a set of relations (as in the case of tables of a relational database, but distributed across multiple sites) i.e.,  $D = \bigotimes_{i=1}^p D_i$  (where  $\bigotimes$  denotes the  $\otimes$  operation). If a data set  $D$  is distributed among the sites  $1, \dots, p$  containing data set fragments  $D_1, \dots, D_p$ , we assume that the individual data sets  $D_1, \dots, D_p$  collectively contain (in principle) all the information needed to construct the dataset  $D$ . More generally,  $D$  may be fragmented across multiple relations [12, 13, 14, 15, 16].

The distributed setting typically imposes a set of constraints  $Z$  on the learner that are absent in the centralized setting. For example, the constraints  $Z$  may prohibit the transfer of raw data from each of the sites to a central location while allowing the learner to obtain certain types of statistics from the individual sites (e.g., counts of instances that have specified values for some subset of attributes). In some applications of data mining (e.g., knowledge discovery from clinical records),  $Z$  might include constraints designed to preserve privacy.

The problem of learning from distributed data can be stated as follows: Given the fragments  $D_1, \dots, D_p$  of a data set  $D$  distributed across the sites  $1, \dots, p$ , a set of constraints  $Z$ , a hypothesis class  $H$ , and a performance criterion  $P$ , the task of the learner  $L_d$  is to output a hypothesis that optimizes  $P$  using only operations allowed by  $Z$ . Clearly, the problem of learning from a centralized data set  $D$  is a special case of learning from distributed data where  $p = 1$  and  $Z = \phi$ . Having defined the problem of learning from distributed data, we proceed to define some criteria that can be used to evaluate the quality of the hypothesis produced by an algorithm  $L_d$  for learning from distributed data relative to its centralized counterpart. We say that an algorithm  $L_d$  for learning from distributed data sets  $D_1, \dots, D_p$  is exact relative to its centralized counterpart  $L$  if the hypothesis produced by  $L_d$  is identical to that obtained by  $L$  from the data set  $D$  obtained by appropriately combining the data sets  $D_1, \dots, D_p$ .

Example: Let  $L_d$  be an algorithm for learning a Support Vector Machine (SVM) classifier [Cortes and Vapnik, 1995]  $h_d : R^N \rightarrow \{-1, 1\}$  under a set of specified constraints  $Z$  from horizontally fragmented distributed data  $D_1, \dots, D_p$ , where each  $D_i \subset D \subset R^N \times \{-1, 1\}$ . Let  $L$  be a centralized algorithm for learning an SVM classifier  $h : R^N \rightarrow \{-1, 1\}$  from data set  $D \subset R^N \times \{-1, 1\}$ . Suppose further that  $D = \cup_{i=1}^p D_i$ . Then we say that  $L_d$  is exact with respect to  $L$  if and only if  $\forall X \in R^N, h(X) = h_d(X)$ .

Proof of exactness of an algorithm for learning from distributed data relative to its centralized counterpart ensures that a large collection of existing theoretical (e.g., sample complexity, error bounds) as well as empirical results

obtained in the centralized setting apply in the distributed setting. We can define exactness of learning from distributed data with respect to other criteria of interest (e.g., expected accuracy of the learned hypothesis). More generally, it might be useful to consider algorithms for learning from distributed data that are provably approximate relative to their centralized counterparts. However, in the discussion that follows, we focus on algorithms for learning from distributed data that are provably exact with respect to their centralized counterparts in the sense defined above.

## 2.2 A General Framework for Designing Algorithms for Learning from Distributed Data

Our general strategy for designing an algorithm for learning from distributed data that is provably exact with respect to its centralized counterpart (in the sense defined above) follows from the observation that most of the learning algorithms use only certain statistics computed from the data  $D$  in the process of generating the hypotheses that they output. (Recall that a statistic is simply a function of the data. Examples of statistics include mean value of an attribute, counts of instances that have specified values for some subset of attributes, the most frequent value of an attribute, etc.) This yields a natural decomposition of a learning algorithm into two components:

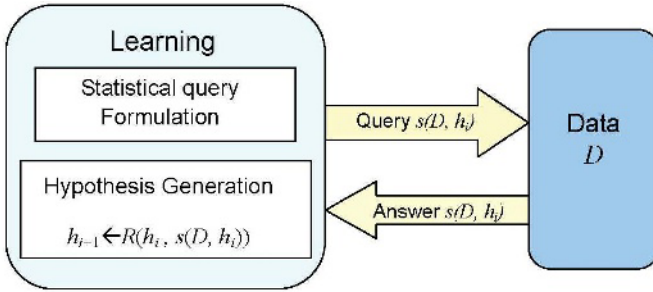
- (a) an information extraction component that formulates and sends a statistical query to a data source and
- (b) a hypothesis generation component that uses the resulting statistic to modify a partially constructed hypothesis (and further invokes the information extraction component as needed).

A statistic  $s(D)$  is called a sufficient statistic for a parameter  $\theta$  if  $s(D)$ , loosely speaking, provides all the information needed for estimating the parameter from data  $D$ . Thus, sample mean is a sufficient statistic for the mean of a Gaussian distribution. A sufficient statistic  $s$  for a parameter  $\theta$  is called a minimal sufficient statistic if for every sufficient statistic  $s_\theta$  for  $\theta$ , there exists a function  $g_{s_\theta}(s_\theta(D)) = s(D)$  [17, 18].

We have, inspired by theoretical work on PAC learning from statistical queries [19], generalized this notion of a sufficient statistic for a parameter  $\theta$  into a sufficient statistic  $s_{L,h}(D)$  for learning a hypothesis  $h$  using a learning algorithm  $L$  applied to a data set  $D$  [20].

Trivially, the data  $D$  is a sufficient statistic for learning  $h$  using  $L$ . However, we are typically interested in statistics that are minimal or at the very least, substantially smaller in size (in terms of the number of bits needed for encoding) than the data set  $D$ . In some simple cases, it is possible to extract a sufficient statistic  $s_{L,h}(D)$  for constructing a hypothesis  $h$  in one step (e.g., by querying the data source for a set of conditional probability estimates when  $L$  is the standard algorithm for learning a Naive Bayes classifier). In such a case, we say that  $s_{L,h}(D)$  is a sufficient statistic for learning  $h$  using the learning algorithm  $L$  if there exists an algorithm that accepts  $s_{L,h}(D)$  as input and outputs  $h = L(D)$ .

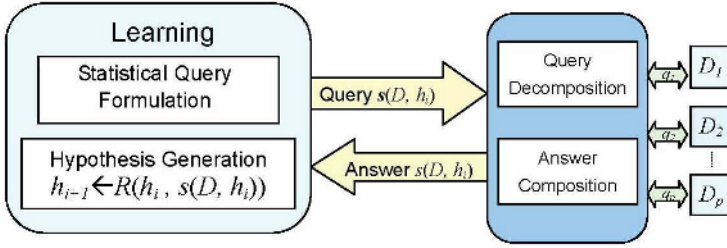
In a more general setting,  $h$  is constructed by  $L$  by interleaving information extraction (statistical query) and hypothesis generation operations. For example, a decision tree learning algorithm would start with an empty initial hypothesis  $h_0$ , obtain the sufficient statistics (expected information concerning the class membership of an instance associated with each of the attributes) for the root of the decision tree (a partial hypothesis  $h_1$ ), and recursively generate queries for additional statistics needed to iteratively refine  $h_1$  to obtain a succession of partial hypotheses  $h_1, h_2$  culminating in  $h$  (See Figure 2).



**Fig. 2.** Learning = Statistical Query Answering + Hypothesis Generation

We say that  $s(D, h_i)$  is a sufficient statistic for the refinement of a hypothesis  $h_i$  into  $h_{i+1}$  (denoted by  $s_{h_i \rightarrow h_{i+1}}$ ) if there exists an algorithm  $R$  which accepts  $h_i$  and  $s(D, h_i)$  as inputs and outputs  $h_{i+1}$ . We say that  $s_h(D, h_1, \dots, h_m)$  is a sufficient statistic for the refinement of the hypotheses  $(h_1 \dots h_m)$  into  $h$  (denoted by  $s_{(h_1, \dots, h_m) \rightarrow h}$ ) if there exists an algorithm  $C$  which accepts as inputs  $h_1 \dots h_m$  and  $s_h(D, h_1, \dots, h_m)$  and outputs the hypothesis  $h$ . We say that  $s_{h_i \rightarrow h_{i+k}}$  (where  $i \geq 0$  and  $k > 0$  are positive integers) is a sufficient statistic for iteratively refining a hypothesis  $h_i$  into  $h_{i+k}$  if  $h_{i+k}$  can be obtained through a sequence of refinements starting with  $h_i$ . We say that  $s_{(h_1, \dots, h_m) \rightarrow h}$  is a sufficient statistic for obtaining hypothesis  $h$  starting with hypotheses  $h_1, \dots, h_m$  if  $h$  can be obtained from  $h_1, \dots, h_m$  through some sequence of applications of composition and refinement operations. Assuming that the relevant sufficient statistics (and the procedures for computing them) can be defined, the application of a learning algorithm  $L$  to a data set  $D$  can be reduced to the computation of  $s_{(h_0, \dots, h_m) \rightarrow h}$  through some sequence of applications of hypothesis refinement and composition operations starting with the hypothesis  $h$  (See Figure 3). In this model, the only interaction of the learner with the repository of data  $D$  is through queries for the relevant statistics. Information extraction from distributed data entails decomposing each statistical query  $q$  posed by the information extraction component of the learner into sub queries  $q_1, \dots, q_n$  that can be answered by the individual data sources  $D_1, \dots, D_p$  respectively, and a procedure for combining the answers to the sub queries into an answer for the original query  $q$ . (See Figure 3).

It is important to note that the general strategy for learning classifiers from distributed data is applicable to a broad class of algorithms for learning classi-



**Fig. 3.** Learning from Distributed Data = Statistical Query Answering + Hypothesis generation

fiers from data[20]. This follows from the fact that the output  $h$  of any learning algorithm is in fact a function of the data  $D$ , and hence by definition, a statistic. Consequently, we can devise a strategy for computing  $h$  from the data  $D$  through some combination of refinement and composition operations starting with an initial hypothesis (or an initial set of hypotheses). When the learner's access to data sources is subject to constraints  $Z$ , the resulting plan for information extraction has to be executable without violating the constraints  $Z$ . The exactness of the algorithm  $L_d$  for learning from distributed data relative to its centralized counterpart, which requires access to the complete data set  $D$  follows from the correctness (soundness) of the query decomposition and answer composition procedure. The separation of concerns between hypothesis construction and extraction of sufficient statistics from data makes it possible to explore the use of sophisticated techniques for query optimization that yield optimal plans for gathering sufficient statistics from distributed data sources under a specified set of constraints that describe the query capabilities of the data sources, operations permitted by the data sources (e.g., execution of user supplied procedures), and available computation, bandwidth, and memory resources.

### 2.3 Representative Algorithms for Learning Classifiers from Distributed Data

We have applied the general framework described above for construction of algorithms for learning classifiers from distributed data to design provably exact algorithms for learning Naive Bayes, Nearest Neighbor, and Decision Tree classifiers from distributed data under horizontal as well as vertical data fragmentation [21], and Support Vector Machine (SVM) Classifiers under horizontal data fragmentation [22, 23]. We briefly summarize our results on learning decision tree classifiers and SVM classifiers from distributed data. We have obtained similar results for algorithms for learning Naive Bayes, Neural Network, and Bayesian Network classifiers [24].

**Algorithms for Learning Decision Tree Classifiers from Distributed Data.** Algorithms that construct decision tree classifiers [25, 26] search in a greedy fashion for attributes that yield the maximum amount of information for determining the class membership of instances in a training set  $D$  of labeled

instances. The information gain associated with an attribute under consideration at a particular node can be expressed in terms of the relative frequencies of instances that satisfy certain constraints on attribute values (determined by the path from the root to each of the nodes resulting from the split) for each possible class [21, 27, 28]. We have devised provably sound strategies for gathering the necessary statistics from distributed data sets, thereby obtaining distributed decision tree learning algorithms that are provably exact relative to their centralized counterparts [21]. This approach to learning decision trees from distributed data provides an effective way to learn classifiers in scenarios in which the distributed data sources provide only statistical summaries of the data and the set of unique keys on demand but prohibit access to data instances. Even when it is possible to access the raw data, the proposed algorithm compares favorably with the centralized counterpart which needs access to the entire data set whenever the communication cost incurred by the former is lower than the cost of collecting the entire data set in a central location. Let  $|D|$  be the total number of examples in the distributed data set;  $|A|$ , the number of attributes;  $V$  the maximum number of possible values per attribute;  $p$  the number of sites across which the data set  $D$  is distributed;  $M$  the number of classes; and  $size(T)$  the number of nodes in the decision tree. Our analysis [20] has shown that in the case of horizontally fragmented data, the distributed algorithm has an advantage when  $MVp\ size(T) < |D|$ . In the case of vertically fragmented data, the corresponding conditions are given by  $size(T) < |A|$ . Our experiments have shown that these conditions are often met in the case of large, high-dimensional data sets that are encountered in several applications (e.g., construction of decision trees for classification of protein sequences into functional families) [29, 30] in computational biology.

**Learning Support Vector Machine Classifiers from Distributed Data.** Support Vector Machine (SVM) algorithm [31, 32] constructs a binary classifier that corresponds to a separating hyperplane that maximizes the margin of separation in  $R^N$  between instances belonging two classes. Because the weight vector that defines the maximal margin hyperplane can be expressed as a weighted sum of a subset of training instances (called support vectors), the support vectors and the associated weights also constitute a sufficient statistic for SVM. In a distributed setting under horizontal fragmentation of data, it is possible to compute the maximal margin separating hyperplane by making several passes through the distributed data sets (without having to gather all of the data in a centralized place), and updating the hyperplane on each pass so as to maximize the margin of separation. We have shown (based on convergence results for SVM algorithms proved by [33]) that this strategy yields a provably exact algorithm for learning an SVM classifier from distributed data under horizontal fragmentation [22, 23].

#### 2.4 Related Work on Learning Classifiers from Distributed Data

Srivastava et al. [34] propose methods for distributing a large centralized data set to multiple processors to exploit parallel processing to speed up learning.



Grossman and Guo [35], and Provost and Kolluri [36] survey several methods that exploit parallel processing for scaling up data mining algorithms to work with large data sets. In contrast, the focus of our work is on learning classifiers from a set of autonomous distributed data sources. The autonomous nature of the data sources implies that the learner has little control over the manner in which the data are distributed among the different sources.

Distributed data mining has received considerable attention in the literature [37]. Domingos [38] and Prodromidis et al. [39] propose an ensemble approach to learning from horizontally fragmented distributed data which essentially involves learning separate classifiers from each data set and combining them typically using a weighted voting scheme. This requires gathering a subset of data from each of the data sources at a central site to determine the weights to be assigned to the individual hypotheses (or shipping the ensemble of classifiers and associated weights to the individual data sources where they can be executed on local data to set the weights). In contrast, our approach is applicable even in scenarios which preclude transmission of data or execution of user-supplied code at the individual data sources but allow transmission of minimal sufficient statistics needed by the learning algorithm. A second potential drawback of the ensemble of classifiers approach to learning from distributed data is that the resulting ensemble of classifiers is typically much harder to comprehend than a single classifier. A third important limitation of the ensemble classifier approach to learning from distributed data is the lack of strong guarantees concerning accuracy of the resulting hypothesis relative to the hypothesis obtained in the centralized setting.

Bhatnagar and Srinivasan [40] propose an algorithm for learning decision tree classifiers from vertically fragmented distributed data. Kargupta et al. [41] describe an algorithm for learning decision trees from vertically fragmented distributed data using a technique proposed by Mansour [42] for approximating a decision tree using  $\sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^m \sum_{l=1}^m \sum_{p=1}^m \sum_{q=1}^m \sum_{r=1}^m \sum_{s=1}^m \sum_{t=1}^m \sum_{u=1}^m \sum_{v=1}^m \sum_{w=1}^m \sum_{x=1}^m \sum_{y=1}^m \sum_{z=1}^m \sum_{\dots}^m$  corresponding to attribute combinations whose size is at most logarithmic in the number of nodes in the tree. At each data source, the learner estimates the Fourier coefficients from the local data, and transmits them to a central site. These estimates are combined to obtain a set of Fourier coefficients for the decision tree (a process which requires a subset of the data from each source to be transmitted to the central site). A given set of Fourier coefficients can correspond to multiple decision trees. At present, such approaches offer no guarantees concerning the performance of the hypothesis obtained in the distributed setting relative to that obtained in the centralized setting.

Unlike the papers summarized above, our approach summarized in Section 2.2 [20] offers a general framework for the design of algorithms for learning from distributed data that is provably exact with respect to its centralized counterpart. Central to our approach is a clear separation of concerns between hypothesis construction and extraction of sufficient statistics from data, making it possible to explore the use of sophisticated techniques for query optimization that yield optimal plans for gathering sufficient statistics from distributed data

sources under specified set of constraints  $Z$  that describe the query capabilities and operations permitted by the data sources (e.g., execution of user supplied procedures). Our approach also lends itself to adaptation to learning from semantically heterogeneous data sources (see below). Provided the needed mappings between ontologies can be specified, our approach to learning from distributed data can be extended to yield a sound approach to learning from heterogeneous distributed data encountered in practical applications (see Section 3).

### 3 Information Integration from Semantically Heterogeneous Distributed Data

#### 3.1 Semantic Data Integration Problem

In order to extend our approach (summarized in Section 2.2) to learning from distributed data (which assumes a common ontology that is shared by all of the data sources) into effective algorithms for learning classifiers from distributed data sources, techniques need to be developed for answering the statistical queries posed by the learner in terms of the learner’s ontology  $O$  from the heterogeneous data sources (where each data source  $D_i$  has an associated ontology  $O_i$ ). Thus, we have to solve a variant of the problem of integrated access to distributed data repositories - the data integration problem [43] in order to be able to use machine learning approaches to acquire knowledge from semantically heterogeneous data. This problem is best illustrated by an example: Consider two academic departments that independently collect information about their students. Suppose a data set  $D_1$  collected by the first department is described by the attributes *ID*, *Student Level*, *Monthly Income*, and *Internship*, and it is stored into a table as the one corresponding to  $D_1$  in Table 1. Suppose a data set  $D_2$  collected by the second department is described by the attributes *SID*, *Student Program*, *Hourly Income*, and *Intern*, and it is stored into a table as the one corresponding to  $D_2$  in Table 1.

**Table 1.** Student data collected by two departments and a university statistician

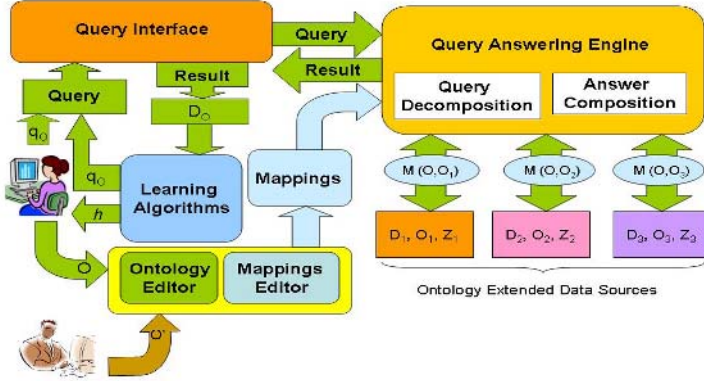
$D_1$	<i>ID</i>	<i>Student Level</i>	<i>Monthly Income</i>	<i>Internship</i>
	34	M.S.	1530	yes
	49	1st Year	600	no
	23	Ph.D.	1800	no
$D_2$	<i>SID</i>	<i>Student Program</i>	<i>Hourly Income</i>	<i>Intern</i>
	1	Master	14	yes
	2	Doctoral	17	no
	3	Undergraduate	8	yes
$D_U$	<i>SSN</i>	<i>Student Status</i>	<i>Yearly Income</i>	<i>Intern</i>
	475	Master	16000	?
	287	Ph.D.	18000	?
	530	Undergrad	7000	?

Consider a user, e.g., a university statistician, who wants to construct a predictive model based on data from two departments of interest from his or her own perspective, where the representative attributes are  $\text{Completed Internship}$  and  $\text{Department}$ . For example, the statistician may want to construct a model that can be used to infer whether a typical student (represented as in the entry corresponding to  $D_U$  in Table 1) drawn from the same population from which the two departments receive their students is likely to have completed an internship. This requires the ability to perform queries over the two data sources associated with the departments of interest from the user's perspective (e.g.,  $\text{Completed Internship} = \text{Completed Internship}$ ). However, because the two data sources differ in terms of semantics from the user's perspective the user must recognize the semantic correspondences between the attributes  $\text{Completed Internship}$  in the first data source,  $\text{Completed Internship}$  in the second data source and  $\text{Completed Internship}$  in the user data; the attributes  $\text{Department}$  and  $\text{Department}$ , etc. From our perspective, a data integration system should: allow users to specify what information is needed instead of how it can be obtained; allow each user to impose his or her own points of view (ontological commitments) on the data sources and post queries specified using terms in that ontology; hide the complexity of communication and interaction with heterogeneous distributed data sources; automatically transform user queries into queries that are understood by the respective data sources; map the results obtained into the form expected by the user and store them for future analysis; allow incorporation of new data sources as needed; and support sharing of ontologies (hence ontological commitments) and among users as needed [10].

### 3.2 INDUS: An Ontology Based Federated Query Centric Data Integration System

Our recent work has led to the development of a *query centric* approach to information integration from heterogeneous, distributed information sources which has been implemented in the data integration component of INDUS (Intelligent Data Understanding System) prototype [10, 11, 44] (See Figure 4).

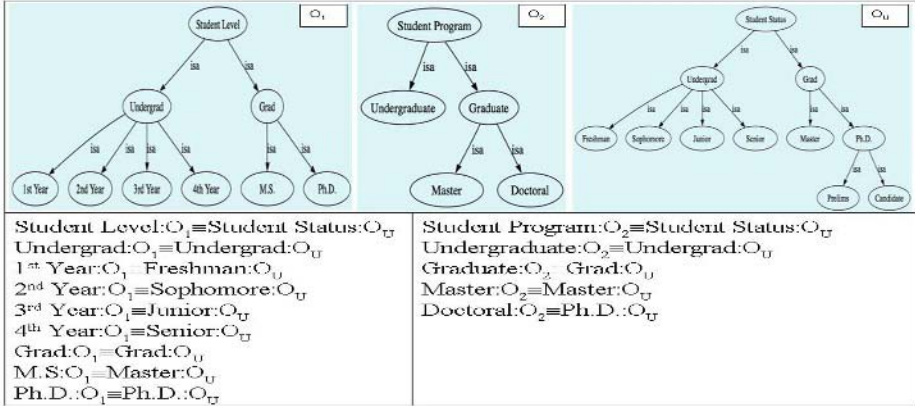
The choice of the federated (as opposed to data warehouse) and query centric (as opposed to source centric) approach to information integration was motivated by characteristics of a class of scientific applications of data-driven knowledge acquisition. A detailed discussion of the design rationale of INDUS can be found in [10, 20, 44]. In brief, a federated approach lends itself much better to settings where it is desirable to postpone specification of user ontology  $O$  and the mapping  $M(O, O_1, \dots, O_p) = \{M(O, O_1), \dots, M(O, O_p)\}$  between  $O$  and data source specific ontologies  $O_1, \dots, O_p$  until when the user is ready to use the system. The choice of a query centric approach in INDUS enables users the desired flexibility in integrating data from multiple autonomous sources in ways that match their own context or application specific ontological commitments whereas in a source centric approach, the semantics of the data (what the data from a source should mean to a user) are determined by the source. INDUS enables a scientist



**Fig. 4.** INDUS (Intelligent Data Understanding System) for Information Integration from Heterogeneous, Distributed, Autonomous Information Sources.  $D_1, D_2, D_3$  are data sources with associated ontologies  $O_1, O_2, O_3$  and  $O$  is a user ontology. Queries posed by the user are answered by a query answering engine in accordance with the mappings between user ontology and the data source ontologies, specified using a user-friendly editor.

to view a collection of physically distributed, autonomous, heterogeneous data sources (regardless of their location, internal structure, and query interfaces) as if they were relational databases, (i.e. a collection of inter-related tables). Each data source in INDUS has associated with it, a data source description which includes the ontology of the data source and a description of the query capabilities of the data source (i.e., the schema of the data source). INDUS makes explicit the (sometimes implicit) ontologies associated with data sources. This allows the specification of semantic correspondences between data sources [11] which can be expressed in ontology-extended relational algebra (independently developed by [45]).

We assume that each data source has associated with it, an ontology that includes hierarchies corresponding to attribute value taxonomies (AVT) (See Figure 5). We specify the correspondence between semantically similar attributes, by mapping the domain of the type of one attribute to the domain of the type of the semantically similar attribute (e.g.,  $person.age$  to  $person.age$  or  $person.age$  to  $person.age$ ) [11]. Explicit specification of mappings between AVTs in the user ontology  $O_U$  and data source ontologies  $O_1$  and  $O_2$  allows the user to view data  $D_1$  and  $D_2$  from his or her own perspective. Such mappings can be used to answer user queries that are expressed in terms of  $O_U$  from the data sources  $D_1$  and  $D_2$ . Let  $\langle D_1, O_1, S_1 \rangle, \dots, \langle D_p, O_p, S_p \rangle$  be an ordered set of  $p$  ontology-extended data sources and  $U$  a user that poses queries against the heterogeneous data sources  $D_1, \dots, D_p$ . A user perspective  $P_U$  is given by a user ontology  $O_U$  and a set of  $\{r_1, \dots, r_n, \dots, r_m\}$  or  $\{r_1, \dots, r_n, \dots, r_m\}$   $IC$  that define relationships between terms in  $O_1, \dots, O_p$ , respectively, and the terms in  $O_U$ . The semantic correspondences take one of the two forms:  $x \leq y$



**Fig. 5.** Attribute value taxonomies (ontologies)  $O_1$  and  $O_2$  associated with the attributes *Student Level*, *Student Program* in two data sources of interest.  $O_U$  is the ontology for *Student Status* from the user’s perspective. An example of user-specified semantic correspondences between the user ontology  $O_U$  and data source ontologies  $O_1$  and  $O_2$  respectively is also shown.

( $x$  is semantically subsumed by  $y$ ),  $x \geq y$  ( $x$  semantically subsumes  $y$ ),  $x \equiv y$  ( $x$  is semantically equivalent to  $y$ ),  $x \neq y$  ( $x$  is semantically incompatible with  $y$ ),  $x \approx y$  ( $x$  is semantically compatible with  $y$ ) (inspired by bridge rules introduced by Bouquet et al. [46]). See 5 for an illustration of user-defined semantic correspondences between data sources  $O_1$  and  $O_2$ , respectively, and  $O_U$ .

Let  $O_1, \dots, O_p$  (respectively) be the ontologies associated with the data sources  $D_1, \dots, D_p$ . Let  $P_U = (O_U, IC)$  a user perspective with respect to these ontologies. We say that the ontologies  $O_1, \dots, O_p$ , are integrable according to the user ontology  $O_U$  in the presence of semantic correspondences  $IC$  if there exist  $p$  partial injective mappings  $M(O_U, O_1), \dots, M(O_U, O_p)$  from  $O_1, \dots, O_p$ , respectively, to  $O_U$ . Examples of such mappings include functions for converting monthly income and hourly income (respectively) from the ontologies associated with data sources  $D_1$  and  $D_2$  (see Figure 5) into yearly income in terms of user ontology  $O_U$ ; or for mapping instances corresponding to  $\dots$  students from data source  $D_1$  into instances described as  $\dots$  from the user perspective. We have completed the implementation of a working prototype of the INDUS system to enable users with some familiarity with the relevant data sources to rapidly and flexibly assemble data sets from multiple data sources and to query these data sets. This can be done by specifying a user ontology, simple semantic mappings between data source specific ontologies and the user ontology and queries - all without having to write any code. The current implementation of INDUS which has been released under Gnu public license (<http://www.cild.iastate.edu/software/indus.html>) includes support for:

- (a) Import and reuse of selected fragments of existing ontologies and editing of ontologies [47].

- (b) Specification of semantic correspondences between a user ontology  $O_U$  and data source ontologies [11]. Semantic correspondences between ontologies can be defined at two levels: schema level (between attributes that define data source schemas) and attribute level (between values of attributes). Consistency of semantic correspondences is verified by reasoning about subsumption and equivalence relationships [48]
- (c) Registration of a new data source using a data-source editor for defining the schema of the data source (the names of the attributes and their corresponding ontological types), location, type of the data source and access procedures that can be used to interact with a data source.
- (d) Specification and execution of queries across multiple semantically heterogeneous, distributed data sources with different interfaces, functionalities and access restrictions. Each user may choose relevant data sources from a list of data sources that have been previously registered with the system and specify a user ontology (by selecting an ontology from a list of available ontologies or by invoking the ontology editor and defining a new ontology). The user can select mappings between data source ontologies and user ontology from the available set of existent mappings (or invoke the mappings editor to define a new set of mappings). The data needed for answering a query is specified by selecting (and possibly restricting) attributes from the user ontology, through a user-friendly interface. Queries posed by the user are sent to a query-answering engine (QAE) that automatically decomposes the user query expressed in terms of the user ontology into queries that can be answered by the individual data sources. QAE combines (after applying the necessary mappings) to generate the answer for the user query. The soundness of the data integration process (relative to a set of user-specified mappings between ontologies) follows from the soundness of the query decomposition procedure, and the correctness of the behavior of the query answering engines associated with the individual data sources, and the answer composition procedure [11].
- (e) Storage and further manipulation of results of queries. The results returned by a user query can be temporarily stored in a local relational database. This in effect, represents a materialized relational view (modulo the mappings between user and data source specific ontologies) across distributed, heterogeneous (and not necessarily relational) data repositories. The current design of INDUS supports further analysis (e.g., by applying machine learning algorithms) on the retrieved data.

In summary, INDUS offers the functionality necessary to flexibly integrate information from multiple heterogeneous data sources and structure the results according to a user-supplied ontology. INDUS has been used to assemble several data sets used in the exploration of protein sequence-structure-function relationships [44].

### 3.3 Related Work on Data Integration

Hull [49], Davidson et al. [50] and Eckman [51] survey alternative approaches to data integration. A wide range of approaches to data integration have been

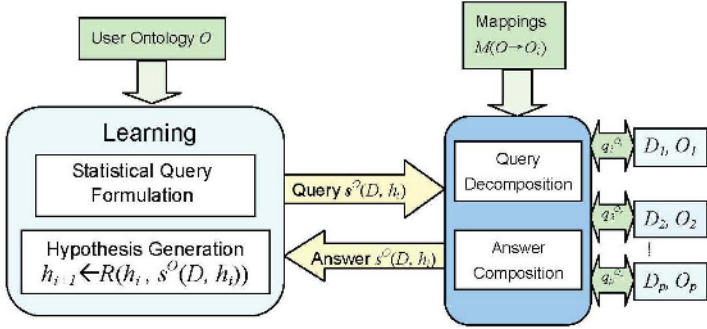
considered including multi-database systems [52, 53, 54], mediator based approaches [55, 56, 57, 58, 59, 60, 61, 62]. Several data integration projects have focused specifically on integration of biological data [63, 64, 65, 66, 67]. Tomasic et al. [68] proposed an approach to scaling up access to heterogeneous data sources. Haas et al. [69] investigated optimization of queries across heterogeneous data sources. Space does not permit a detailed survey of the extensive literature on data integration. Rodriguez-Martinez and Roussoloulos [70] proposed a code shipping approach to design of an extensible middleware system for distributed data sources. Lambrecht et al. [71] proposed a planning framework for gathering information from distributed sources. These efforts addressed, and to varying degrees, solved the following problems in data integration: design of query languages and rules for decomposing queries into sub queries and composing the answers to sub queries into answers to the initial query. Maluf and Wiederhold [72] proposed an ontology algebra for merging of ontologies. Our group developed an approach to specifying semantic correspondences between ontologies and for querying semantically heterogeneous data using ontologies and inter-ontology mappings [10]. This approach is similar to the ontology-extended relational algebra developed by Bonatti et al. [45]. The design of INDUS [10, 11, 44] was necessitated by the lack of publicly available data integration platforms that could be used as a basis for learning classifiers from semantically heterogeneous distributed data. INDUS draws on much of the existing literature on data integration and hence shares some of the features of existing data integration platforms. But it also includes some relatively novel features (See Section 3.2).

## 4 Knowledge Aquisition from Semantically Heterogeneous Distributed Data

The stage is now set for developing sound approaches to learning from semantically heterogeneous, distributed data (See Figure 6). While it is possible to retrieve the data necessary for learning from a set of heterogeneous data sources using INDUS, store the retrieved data in a local database, and then apply standard (centralized) learning algorithms, such approach is not feasible when the amounts of data involved are large, and bandwidth and memory are limited, or when the query capabilities of the data sources are limited to answering a certain class of statistical queries (e.g., counts of instances that satisfy certain constraints on the values of their attributes). Hence, the development of sound approaches to answering statistical queries from semantically heterogeneous data sources a variety of constraints and assumptions motivated by application scenarios encountered in practice is a key element of our research plan.

### 4.1 Partially Specified Data

Our approach to design of algorithms for learning classifiers from semantically heterogeneous distributed data is a natural extension of our approach to learning from distributed data discussed in Section 2 (See Figure 3) which assumed a common ontology that is shared by all of the data sources. We propose to extend this



**Fig. 6.** General Framework for learning classifiers from semantically heterogeneous distributed data. The learner generates statistical queries expressed in terms of user ontology). These queries have to be mapped into queries expressed in terms of data source specific ontologies that can be executed directly on the data sources and the results combined and mapped into the answer to the query posed by the learner.

framework to work with semantically heterogeneous distributed data sources by developing techniques for answering the statistical queries posed by the learner in terms of the learner’s ontology  $O$  using the heterogeneous data sources (where each data source  $D_i$  has an associated ontology  $O_i$ ) (See Figure 6).

Before we can discuss approaches for answering statistical queries from semantically heterogeneous data, it is useful to explore what it means to answer a statistical query in a setting in which autonomous data sources differ in terms of the levels of abstraction at which data are described. We illustrate some of the issues that have to be addressed using an example: Consider the data source ontologies  $O_1$  and  $O_2$  and the user ontology  $O_U$  shown in Figure 5. The attribute  $\text{year}$  in data source  $D_1$  is specified in greater detail (lower level of abstraction) than in  $D_2$ . That is, data source  $D_1$  carries information about the precise categorization of students into 1<sup>st</sup> year, 2<sup>nd</sup> year, 3<sup>rd</sup> year, and 4<sup>th</sup> year students, whereas data source  $D_2$  makes no such distinctions among students. Now suppose that  $D_1$  contains 5, 10, 15, 10 instances (respectively) of 1<sup>st</sup> year, 2<sup>nd</sup> year, 3<sup>rd</sup> year, and 4<sup>th</sup> year (undergrad) students and 20 instances of students. Suppose  $D_2$  contains 20 instances of students, 40 instances of students respectively.

Suppose a statistical query  $q^{O_U}$  is posed by the user against the two data sources  $D_1$  and  $D_2$  based on ontology  $O_U$ : What fraction of students are students? The answer to this query can be computed in a straightforward fashion as the ratio of number of students  $\text{year}$  ((5+10+15+10)+20=60) divided by the total number of students whose  $\text{year}$  is recorded in the two data sources (60+20+40=120) yielding an answer of 0.5.

Now consider a different statistical query  $r^{O_U}$ : What fraction of the students in the two data sources are students? The answer to this query is not as straightforward as the answer to the previous query  $q^{O_U}$ . This is due to the fact that the student records in data source  $D_2$  are only



[73, 74] with respect to the ontology  $O$ . Consequently, we can never know the precise fraction of students that are  $\{freshman, sophomore\}$  based on the information available in the two data sources. However, if it is reasonable to assume that the data contained in both  $D_1$  and  $D_2$  are drawn from the same,  $\{freshman, sophomore\}$  (i.e., can be modeled by the same underlying distribution), we can estimate the fraction of students that are  $\{freshman, sophomore\}$  in the data source  $D_2$  based on the fraction of  $\{freshman, sophomore\}$  students that are  $\{freshman, sophomore\}$  in the data source  $D_1$  (i.e., 10 out of 40) and use the result to answer the query  $r^{O_U}$ . Under the assumption that the population of students in  $D_1$  and  $D_2$  can be modeled by the same distribution, the number of  $\{freshman, sophomore\}$  students in  $D_2$  is given by  $(\frac{10}{40})(20) = 5$ . Hence, the number of  $\{freshman, sophomore\}$  students in  $D_1$  and  $D_2 = 10 + 5 = 15$ . Thus, the answer to the query  $r^{O_U}$  is  $\frac{15}{120} = 0.125$ . Note that the answer to query  $q^{O_U}$  is completely determined by the data source ontologies  $O_1, O_2$ , the user ontology  $O_U$  and the mappings shown in Figure 5. However, answer to the query  $r^{O_U}$  is only partially determined by the ontologies and the mappings shown in Figure 5. In such cases, answering statistical queries from semantically heterogeneous data sources requires the user to supply not only the mapping between the ontology and the ontologies associated with the data sources but also additional assumptions of a statistical nature (e.g., that data in  $D_1$  and  $D_2$  can be modeled by the same underlying distribution) and the validity of the answer returned depends on the validity of the assumptions and the soundness of the procedure that computes the answer based on the supplied assumptions.

Hence, the development of algorithms for learning classifiers from semantically heterogeneous data requires addressing the problem of learning classifiers from partially specified data. Specifically, this entails development provably sound methods based extensions to our current formulations of ontology-based query decomposition and answer composition methods in INDUS [11] for answering statistical queries from semantically heterogeneous data sources under alternative statistical assumptions.

## 4.2 The Problem of Learning Classifiers from Partially Specified Data

Let us start by considering a centralized data set  $D$  with an associated ontology  $O$ . Let  $\{A_1, A_2, \dots, A_n\}$  be an ordered set of nominal attributes, and let  $dom(A_i)$  denote the set of values (the domain) of attribute  $A_i$ . An attribute value taxonomy  $T_i$  for attribute  $A_i$  is a tree structured concept hierarchy in the form of a partially order set  $(dom(A_i), \leq)$ , where  $dom(A_i)$  is a finite set that enumerates all attribute values in  $A_i$ , and  $\leq$  is the partial order that specifies relationships among attribute values in  $dom(A_i)$  (see any of the ontologies in Figure 5). Collectively,  $O = \{T_1, T_2, \dots, T_n\}$  represents the ordered set of attribute value taxonomies associated with attributes  $\{A_1, A_2, \dots, A_n\}$  (see Figure 7).

Let  $Nodes(T_i)$  represent the set of all values in  $T_i$ , and  $Root(T_i)$  stand for the root of  $T_i$ . The set of leaves of the tree,  $Leaves(T_i)$ , corresponds to the set of  $\{freshman, sophomore, \dots, junior\}$  of attribute  $A_i$  (e.g., Freshman, Sophomore, etc. in the hierarchy corresponding to the attribute  $\{freshman, sophomore, \dots, junior\}$  in Figure 5). The internal

nodes of the tree (i.e.,  $Nodes(T_i) - Leaves(T_i)$ ) correspond to abstract values of attribute  $A_i$  (e.g., Undergrad, Grad, Ph.D. in in the hierarchy corresponding to the attribute "Education" in Figure 5). Each arc of the tree corresponds to a  $\succ_{A_i}$  relationship over attribute values in the AVT. Thus, an AVT defines an abstraction hierarchy over values of an attribute.

The set of abstract values at any given level in the tree  $T_i$  form a partition of the set of values at the next level (and hence, the set of primitive values of  $A_i$ ). For example, in Figure 5, the nodes at level 1, i.e., Undergrad, Grad, define a partition of attribute values that correspond to nodes at level 2 (and hence, a partition of all primitive values of the "Education" attribute). After Haussler [75], we define a cut  $\gamma_i$  of an AVT  $T_i$  as a subset of nodes in  $T_i$  satisfying the following two properties: (1) For any leaf  $l \in Leaves(T_i)$ , either  $l \in \gamma_i$  or  $l$  is a descendent of a node  $n \in \gamma_i$ ; and (2) For any two nodes  $f, g \in \gamma_i$ ,  $f$  is neither a descendent nor an ancestor of  $g$ . Cuts through AVT  $T_i$  correspond to a partition of  $Leaves(T_i)$ . Thus, the global cut  $\Gamma$  corresponding to  $\{\gamma_1, \gamma_2, \gamma_3\}$  defines a partition over the primitive values of the "Education" attribute.

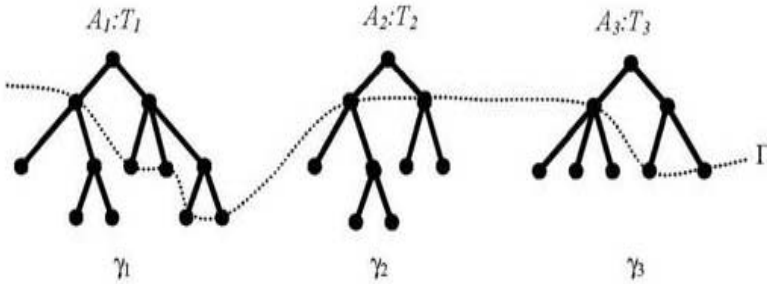


Fig. 7. Global cut through a set of attribute value taxonomies

The original instance space  $I$  in the absence of AVT is an instance space defined over the domains of all attributes. Let  $\Gamma = \{\gamma_1, \dots, \gamma_n\}$  be a global cut through  $T$ , where  $\gamma_i$  stands for a cut through  $T_i$  (see Figure 7). The cut  $\Gamma$  defines an instance space  $I_\Gamma$ . A set of AVT  $O = \{T_1, T_2, \dots, T_n\}$  associated with a set of attributes  $A = \{A_1, A_2, \dots, A_n\}$  induces an instance space  $I_O = \cup_\Gamma I_\Gamma$  (the union of instance spaces induced by all the cuts through the set of AVT  $O$ ). We say that an instance  $x \in I_O$  is partially specified if one or more of its attribute values are not primitive. A data set  $D_O$  (relative to a set  $O$  of AVT) is a collection of instances drawn from  $I_O$  where each instance is labeled with the appropriate class label from  $C = \{c_1, c_2, \dots, c_k\}$ , a finite set of mutually disjoint classes. Thus,  $D_O \subset I_O \times C$ .

The problem of learning classifiers from AVT and partially specified data can be formulated as follows: Given a user-supplied set of AVT  $O$  and a data set  $D_O$  of (possibly) partially specified labeled instances, construct a classifier  $h_O : I_O \rightarrow C$  for assigning appropriate class labels to each instance in the instance space  $I_O$ .

### 4.3 Learning from Partially Specified Semantically Heterogeneous Data

Suppose that a data set  $D$  is distributed over the data sources  $D_1, \dots, D_p$ , where each data source  $D_i$  contains only a horizontal fragment (subset of data tuples) of the data  $D$ . Each distributed data set  $D_i$  is described by the set of attributes  $\{A_1^i, \dots, A_n^i\}$  and their corresponding AVT  $O_i = \{T_1^i, \dots, T_n^i\}$ . Let  $\{A_1^U, \dots, A_n^U\}$  be the set of attributes that describe the data  $D$  in a user view and let  $O_U = T_1^U, \dots, T_n^U$  be a user-supplied collection of taxonomies over the set of attributes  $A_1^U, \dots, A_n^U$ . Let  $\Psi = \{\varphi_1, \varphi_2, \dots, \varphi_p\}$  be a collection of user-defined mappings from data source taxonomies  $O_i$  to user taxonomies  $O_U$ , respectively. A global cut  $\Gamma^U$  in the user's collection of taxonomies  $O_U = \{T_1^U, \dots, T_n^U\}$  determines cuts  $\{\Gamma_1, \Gamma_2, \dots, \Gamma_n\}$  in the data source taxonomies, through the means of user-defined mappings  $\Psi$ . The abstract instance space defined by  $\Gamma^U$  is denoted by  $I_{\Gamma^U}$  and is given by  $I_{\Gamma^U} = \varphi_1(I_{\Gamma_1}) \cup \varphi_2(I_{\Gamma_2}) \dots \cup \varphi_p(I_{\Gamma_n})$ . The set of user AVT  $O_U = T_1^U, \dots, T_n^U$  induces an instance space  $I_{O_U} = \cup_{\Gamma^U} I_{\Gamma^U}$ . We say that an instance  $x \in I_{O_U}$  is *primitive* if one or more of its attribute values are not primitive. A *primitive instance space*  $D_{O_U}$  (relative to a set  $O_U$  of user AVT) is a collection of instances drawn from  $I_{O_U}$  where each instance is labeled with the appropriate class label from  $C = \{c_1, c_2, \dots, c_k\}$ , a finite set of mutually disjoint classes. Thus,  $D_{O_U} \subset I_{O_U} \times C$ .

The problem of learning classifiers from distributed, semantically heterogeneous data sources can be formulated as follows: Given a collection of (possibly) partially specified data sources  $D_1, \dots, D_p$  and their associated collections of taxonomies  $\{O_1, \dots, O_p\}$ , a user collection of taxonomies  $O_U$  and a set of mappings  $\Psi$  from data source taxonomies  $O_i$  to user taxonomies  $O_U$ , construct a classifier  $h_{O_U} : I_{O_U} \rightarrow C$  for assigning appropriate class labels to each instance in the instance space  $I_{O_U}$ .

### 4.4 AVT-Guided Learning Algorithms

AVT-guided learning algorithms extend standard learning algorithms in principled ways so as to exploit the information provided by AVT. We have designed and implemented AVT-NBL [74] and AVT-DTL [73] for learning AVT-guided Naive Bayes and Decision Tree classifiers, respectively. The standard Decision Trees or Naive Bayes learning algorithms can be viewed as special cases of AVT-DTL or AVT-NBL, where the AVT associated with each attribute has only one level. The root of such an AVT corresponds to the value of the attribute being unknown and the leaves correspond to the primitive values of the attribute. We will use Naive Bayes Learner (NBL) as an example to illustrate our approach to AVT-guided learning algorithms. Naive Bayes classifier operates under the assumption that each attribute is independent of the others given the class. Thus, the joint class conditional probability of an instance can be written as the product of individual class conditional probabilities corresponding to each attribute defining the instance. The Bayesian approach to classifying an instance  $x = \{v_1, \dots, v_n\}$  is to assign it to the most probable class  $c_{MAP}(x)$ . We have:

$$c_{MAP}(x) = \operatorname{argmax}_{c_j \in C} p(v_1, \dots, v_n | c_j) p(c_j) = \operatorname{argmax}_{c_j \in C} p(c_j) \prod_i p(v_i | c_j).$$

Therefore, the task of the Naive Bayes Learner (NBL) is to estimate the class probabilities  $p(c_j)$  and the class conditional probabilities  $p(v_i|c_j)$ , for all classes  $c_j \in \mathbf{C}$  and for all attribute values  $v_i \in \text{dom}(A_i)$ . These probabilities can be estimated from a training set  $D$  using standard probability estimation methods [1] based on relative frequency counts. We denote by  $\sigma(v_i|c_j)$  the frequency count of a value  $v_i$  of the attribute  $A_i$  given the class label  $c_j$ , and by  $\sigma(c_j)$  the frequency count of the class label  $c_j$  in a training set  $D$ .

AVT-guided NBL, called AVT-NBL [74] efficiently exploits taxonomies defined over values of each attribute in a data set to find a Naive Bayes classifier that optimizes the Conditional Minimum Description Length (CMDL) score [Friedman et al., 1997]. More precisely, the task of AVT-NBL is to construct a Naive Bayes classifier for assigning an unlabeled instance  $x \in I_O$  to its most probable class  $c_{MAP}(x)$ . As in the case of NBL, we assume that each attribute is independent of the other attributes given the class. A Naive Bayes classifier defined on the instance space  $I_O$  is completely specified by a set of class conditional probabilities for each value of each attribute. Suppose we denote the table of class conditional probabilities associated with values in  $\gamma_i$  by  $CPT(\gamma_i)$ . Then the Naive Bayes classifier defined over the instance space  $I_O$  is specified by  $h(\Gamma) = \{CPT(\gamma_1), \dots, CPT(\gamma_n)\}$ .

AVT-NBL starts with the Naive Bayes Classifier that is based on the most abstract value of each attribute (the most general hypothesis) and successively refines the classifier (hypothesis) using a criterion that is designed to tradeoff between the accuracy of classification and the complexity of the resulting Naive Bayes classifier. Successive refinements of  $\Gamma$  correspond to an ordering of Naive Bayes classifiers based on the structure of the AVTs in  $O$ . For example, in Figure 8,  $\Gamma'$  is a refinement of  $\Gamma$ , and hence the corresponding hypothesis  $h(\Gamma')$  is a refinement of  $h(\Gamma)$  [74].

The scoring function that we use to evaluate a candidate AVT-guided refinement of a Naive Bayes Classifier is an adaptation (for the case of classifiers constructed from partially specified data) of the Conditional Minimum Description Length (CMDL) criterion [76] and captures the tradeoff between the accuracy and the complexity of the resulting Naive Bayes classifier [74].

The parameters that define the classifier can be estimated from the observed class distribution in the data  $D$  based on frequency counts  $\sigma_i(c_j)$  and  $p_i(v|c_j)$  is the class conditional probability of value  $v$  of attribute  $A_i$  given the class label  $c_j$ .

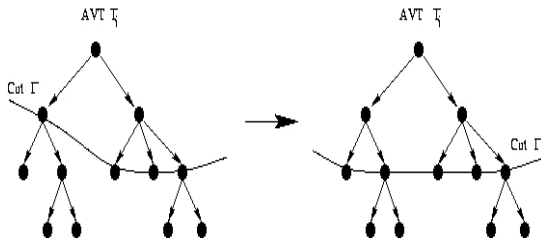


Fig. 8. Global cut through a set of attribute value taxonomies

The value of  $p_i(v|c_j)$  can similarly be estimated from the frequency counts  $\sigma_i(v|c_j)$  obtained from the data set  $D$ . When some of the data are partially specified, we can use a 2-step process for computing  $\sigma_i(v|c_j)$ . First we make an upward pass through the AVT for each attribute aggregating the class conditional frequency counts based on the specified attribute values in the data set; Then we propagate the counts associated with partially specified attribute values down through the tree, augmenting the counts at lower levels according to the distribution of values along the branches based on the subset of the data for which the corresponding values are fully specified [73, 74]. This procedure can be seen as a special case of EM (Expectation Maximization) algorithm for estimation of  $\sigma_i(v|c_j)$  under the assumption that the attributes are independent given the class [74].

Thus, AVT-NBL produces a hypothesis  $h$  that intuitively trades off the complexity of Naive Bayes classifier (in terms of the number of parameters used to describe the relevant class conditional probabilities) against accuracy of classification. The algorithm terminates when none of the candidate refinements of the classifier yield statistically significant improvement in the  $CMDL$  score [74]. Our experiments with several synthetic as well as real-world data sets have demonstrated the efficacy of AVT-NBL [74] and AVT-DTL [73].

#### 4.5 Learning Classifiers from Partially Specified Semantically Heterogeneous Data

Our approach to AVT-guided learning from partially specified semantically heterogeneous data [77] relies on our general strategy for transforming algorithms for learning from data into algorithms for learning from distributed, semantically heterogeneous data [11, 20]. As mentioned before, this strategy is based on the decomposition of the learning task into an information extraction component (when the information needed for learning are gathered) and a hypothesis generation component (that uses the information to generate or refine a current hypothesis).

Recall that a statistic  $s_L(D)$  is a sufficient statistic for a hypothesis  $h$  using a learning algorithm  $L$  applied to a data set  $D$  if there exists a procedure that takes  $s_L(D)$  as input and outputs  $h$  [20]. For example, in the case of NBL, the frequency counts  $\sigma(v_i|c_j)$  of the value  $v_i$  of the attribute  $A_i$  given the class label  $c_j$  in a training set  $D$ , and the frequency count  $\sigma(c_j)$  of the class label  $c_j$  in a training set  $D$  completely summarize the information needed for constructing a Naive Bayes classifier from  $D$ , and thus, they constitute sufficient statistics for NBL. As noted in Section 2, some simple learning algorithms such as NBL  $s_L(D)$ , the sufficient statistics required for constructing the classifier can be computed in one step, in general, a learning algorithm may require assembly of  $s_L(D)$  through an interleaved execution of the information extraction and hypothesis generation components [20].

We illustrate our approach to using this strategy to design AVT-guided algorithms for learning classifiers from semantically heterogeneous data using the Naive Bayes classifier as an example. However, the proposed approach can be extended to a broad range of machine learning algorithms including variants

of Decision Tree, Bayesian networks (Naive Bayes and Tree-Augmented Naive Bayes classifiers), generalized linear models, support vector machines.

**Sufficient Statistics for AVT-NBL.** As we have shown, AVT-NBL starts with a Naive Bayes classifier  $h_0 = h(\Gamma_0)$  corresponding to the most abstract cut  $\Gamma_0$  in the attribute value taxonomy associated with the data (i.e., the most general classifier that simply assigns each instance to the class that is a priori most probable) and it iteratively refines the classifier by refining the corresponding cut until a best cut, according to the performance criterion, is found. More precisely, let  $h_i$  be the current hypothesis corresponding to the current cut  $\Gamma$  (i.e.,  $h_i = h(\Gamma)$ ) and  $\Gamma'$  a (one-step) refinement of  $\Gamma$  (see Figure 8).

Let  $h(\Gamma')$  be the Naive Bayes classifier corresponding to the cut  $\Gamma'$  and let  $CMDL(\Gamma|D)$  and  $CMDL(\Gamma'|D)$  be the CMDL scores corresponding to the hypotheses  $h(\Gamma)$  and  $h(\Gamma')$ , respectively. If  $CMDL(\Gamma) > CMDL(\Gamma')$  then  $h_{i+1} = h(\Gamma')$ , otherwise  $h_{i+1} = h(\Gamma)$ . This procedure is repeated until no (one-step) refinement  $\Gamma'$  of the cut  $\Gamma$  results in a significant improvement of the CMDL score, and the algorithm ends by outputting the classifier  $h(\Gamma)$ .

Thus, the classifier that the AVT-NBL finds is obtained from  $h_0 = h(\Gamma_0)$  through a sequence of refinement operations. The refinement sufficient statistics  $s_L(D, h_i \rightarrow h_{i+1})$  are identified below.

Let  $h_i$  be the current hypothesis corresponding to a cut  $\Gamma$  and  $CMDL(\Gamma|D)$  its score. If  $\Gamma'$  is a refinement of the cut  $\Gamma$ , then the refinement sufficient statistics needed to construct  $h_{i+1}$  are given by the frequency counts needed to construct  $h(\Gamma')$  together with the probabilities needed to compute  $CLL(h(\Gamma')|D)$  (calculated once we know  $h(\Gamma')$ ). If we denote by  $dom_{\Gamma'}(A_i)$  the domain of the attribute  $A_i$  when the cut  $\Gamma'$  is considered, then the frequency counts needed to construct  $h(\Gamma')$  are  $\sigma(v_i|c_j)$  for all values  $v_i \in dom_{\Gamma'}(A_i)$  of all attributes  $A_i$  and for all class values  $c_j \in dom_{\Gamma'}(C)$ , and  $\sigma(c_j)$  for all class values  $c_j \in dom_{\Gamma'}(C)$ . To compute  $CLL(h(\Gamma')|D)$  the products  $\prod_j p_{h(\Gamma')}(v_{ij}|c_k)$  for all examples  $x_i = (v_{i1}, \dots, v_{in})$  and for all classes  $c_k \in C$  are needed.

The step  $i + 1$  of the algorithm corresponding to the cut  $\Gamma'$  can be briefly described in terms of information gathering and hypothesis generation components as follows:

- (1) Compute  $\sigma(v_i|c_j)$  and  $\sigma(c_j)$  corresponding to the cut  $\Gamma'$  from the training data  $D$
- (2) Generate the NB classifier  $h(\Gamma')$
- (3) Compute  $\prod_j p_{h(\Gamma')}(v_{ij}|c_k)$  from  $D$
- (4) Generate the hypothesis  $h_{i+1}$

**Learning Naive Bayes Classifiers from Semantically Heterogeneous Data.** Let  $\{D_1, \dots, D_p\}$  be a set of semantically heterogeneous data sources with associated ontologies  $\{O_1, \dots, O_p\}$ . Let  $O_U$  be a user collection of AVT and  $\Gamma$  a cut through the user AVT.

The step  $i + 1$  (corresponding to the cut  $\Gamma'$  in the user ontology) of the algorithm for learning Naive Bayes classifiers from distributed, semantically heterogeneous data sources  $D_1, \dots, D_p$  can be described in terms of information gathering and hypothesis generation components as follows:

- (1) Compute  $\sigma(v_i|c_j)$  and  $\sigma(c_j)$  corresponding to the cut  $\Gamma'$  from the distributed data sources  $D_1, \dots, D_p$
- (2) Generate the NB classifier  $h(\Gamma')$  at the user location and send it to the data sources  $D_1, \dots, D_p$
- (3) Compute  $\prod_j p_{h(\Gamma')}(v_{ij}|c_k)$  from  $D_1, \dots, D_p$
- (4) Generate the hypothesis  $h_{i+1}$  at the user location

Thus, using the decomposition of an AVT-guided algorithm for learning classifier from partially specified data into information extraction and hypothesis generation components, we reduce the problem of learning classifiers from distributed, semantically heterogeneous data sources to the problem of querying for sufficient statistics from such data sources (e.g., frequency counts  $\sigma(v_i|c_j)$  and  $\sigma(c_j)$  corresponding to a cut). This involves design of procedures for decomposing statistical queries into sub-queries corresponding to the distributed data sources and procedures for combining the partial answers into a final answer to the initial queries (e.g., adding up counts) [77].

#### 4.6 Related Work on Learning Classifiers from Partially Specified Data

Walker [78] first used attribute value taxonomies in information retrieval from large databases. DeMichiel [79], and Chen and Tseng [80] proposed database models to handle imprecision using partial values and associated probabilities where a partial value refers to a set of possible values for an attribute. McClean et al. [81] proposed aggregation operators defined over partial values. While this work suggests ways to aggregate statistics so as to minimize information loss, it does not address the problem of learning from AVT and partially specified data. The problem of learning classifiers from AVT and partially specified data was formulated and solved in the case of decision tree classifiers by Zhang and Honavar [73] and in the case of Naive Bayes classifiers by Zhang and Honavar [74]. Development of approaches to exploit abstractions over attribute values and class labels to optimally exploit partially specified data. The use of prior knowledge or domain theories specified typically in first order logic or propositional logic to guide learning from data has been explored in ML-SMART [82], FOCL [83] and KBANN [84] systems as well as in the work of Aronis et al. [85] and Aronis and Provost [86]. However, the work on exploiting domain theories in learning has not focused on the effective use of AVT to learn classifiers from partially specified data. Approaches to exploiting abstractions in learning from fully specified data have been studied by several authors [87, 88, 89, 90, 91, 92, 93, 94, 95]. We have developed simple algorithms for learning decision tree [73] and Naive Bayes [74] classifiers from partially specified data. These methods assume independence of attributes in estimating answers to statistical queries from partially specified data based on the distribution of observed values. in fully specified instances. It is also of interest to investigate methods based on multiple imputation [96, 97, 98] which has been used with success in a number of applications such as studies of air quality [99], employment [100], and health care [101] to cope with missing observations. Multiple imputation aims to: (a) use available information to make

good predictions of the missing values and (b) reflect uncertainty due to the fact that some of the data were in fact not observed. Some causes of missing data such as when an individual does not answer a particular question, and when an individual refuses to answer any questions, but some demographic information such as the identity of the data source that the person is associated with is available, have been considered in the statistical literature [102, 103, 104].

## 5 Summary and Discussion

Biological, environmental, ecological, engineering, social, and biomedical sciences are in the midst of being transformed from data poor sciences into data rich sciences, in large part, due to rapid advances in experimental and data acquisition methods. Recent advances in computer science, statistical methods, and information theory provide powerful conceptual tools for extracting knowledge from data and for developing algorithmic models of causal interactions within and across multiple levels of organization in complex systems. Advances in computing, storage, communication, and software technologies (e.g., web services that can be invoked and on the Internet and executed on remote computers or data repositories) provide unprecedented opportunities for exploiting disparate data and knowledge to address fundamental scientific questions. Because data sources that are created for use by one scientific community (e.g., structural biologists) find use in other contexts (e.g. exploration of macromolecular function), given the prevalence of discipline-specific terminologies (ontologies), semantic differences among autonomous data repositories are simply unavoidable. Effective use of multiple sources of data in a given context requires reconciliation of such semantic differences from the user's point of view. This is especially true in emerging areas of scientific inquiry at the boundaries of established disciplines (e.g., computational biology) that draw on multiple areas of inquiry (e.g., molecular biology, biophysics, structural biology). Furthermore, because many of the data sources of interest are autonomous and geographically distributed, it is neither desirable nor feasible to gather all of the data in a centralized location for analysis. Hence, there is an urgent need for algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources. Against this background, the research summarized in this paper has led to:

- (a) The development of a general theoretical framework for learning predictive models (e.g., classifiers) from large, physically distributed data sources where it is neither desirable nor feasible to gather all of the data in a centralized location for analysis [20]. This framework offers a general recipe for the design of algorithms for learning from distributed data that are provably exact with respect to their centralized counterparts (in the sense that the model constructed from a collection of physically distributed data sets is provably identical to that obtained in the setting where the learning algorithm has access to the entire data set). A key feature of this framework is the clear separation of concerns between hypothesis construction and extraction and



refinement of sufficient statistics needed by the learning algorithm from data which reduces the problem of learning from data to a problem of decomposing a query for sufficient statistics across multiple data sources and combining the answers returned by the data sources to obtain the answer for the original query. This work has resulted in the identification of sufficient statistics for a large family of learning algorithms including in particular, algorithms for learning decision trees [20], neural networks, support vector machines [23] and Bayesian networks, and consequently, provably exact algorithms for learning the corresponding classifiers from distributed data.

- (b) The development of theoretically sound yet practical variants of a large class of algorithms [20, 23] for learning predictive models (classifiers) from distributed data sources under a variety of assumptions (motivated by practical applications) concerning the nature of data fragmentation, and the query capabilities and operations permitted by the data sources (e.g., execution of user supplied procedures), and precise characterization of the complexity (computation, memory, and communication requirements) of the resulting algorithms relative to their centralized counterparts.
- (c) The development of a theoretically sound approach to formulation and execution of statistical queries across semantically heterogeneous data sources [11]. This work has demonstrated how to use semantic correspondences and mappings specified by users from a set of terms and relationships among terms (user ontology) to terms and relations in data source specific ontologies to construct a sound procedure for answering queries for sufficient statistics needed for learning classifiers from semantically heterogeneous data. An important component of this work has to do with the development of statistically sound approaches to learning classifiers from data described at different levels of abstraction across different data sources [73, 74].
- (d) The design and implementation of INDUS, a modular, extensible, open-source software toolkit (<http://www.cild.iastate.edu/software/indus.html>) for data-driven knowledge acquisition from large, distributed, autonomous, semantically heterogeneous data sources [44, 11].
- (e) Applications of the resulting approaches to data-driven knowledge acquisition tasks that arise in bioinformatics [30, 44, 105, 106].

Work in progress is aimed at:

- (a) Extending the INDUS query answering engine to flexibly interact with different data sources that might support different functionalities or impose different constraints on users (For example, some data sources might answer only restricted classes of statistical queries. Others might allow retrieval of raw data. Still others might allow execution of user-supplied procedures at the data source, there by allowing the users to effectively extend the query capabilities of the data source);
- (b) Investigation of resource-bounded approximations of answers to statistical queries generated by the learner; develop approximation criteria for evaluation of the quality of classifiers obtained in the distributed setting under

a given set of resource constraints and query capabilities relative to that obtained in the centralized setting with or without such constraints. This is especially important in application scenarios in which it is not feasible to obtain exact answers to statistical queries posed under the access and resource constraints imposed by the distributed setting;

- (c) Development of tools to support modular development of ontologies, interactive specification of mappings between ontologies including automated generation of candidate mappings for consideration by users, and reasoning algorithms for ensuring semantic integrity of user-specified mappings between ontologies;
- (d) Development of sophisticated approaches to estimation from partially specified data, of the statistics needed by learning algorithms; and
- (e) Application of the resulting algorithms and software to collaborative discovery problems that arise in areas such as computational biology e.g., discovery of relationships between macromolecular sequence, structure, expression, interaction, function, and evolution; discovery of genetic regulatory networks from multiple sources of data (e.g., gene expression, protein localization, protein-protein interaction).

## Acknowledgements

This research was supported in part by grants from the National Science Foundation (NSF IIS 0219699) and the National Institutes of Health (GM 066387) to Vasant Honavar. This work has benefited from discussions with Adrian Silvescu, Jaime Reinoso-Castillo, and Drena Dobbs.

## References

- [1] Mitchell, T.: *Machine Learning*. McGraw Hill (1997)
- [2] Duda, R., Hart, E., Stork, D.: *Pattern Recognition*. Wiley (2000)
- [3] Thrun, S., Faloutsos, C., Mitchell, M., Wasserman, L.: *Automated learning and discovery: State-of-the-art and research topics in a rapidly growing field*. *AI Magazine* (1999)
- [4] Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer-Verlag (2001)
- [5] Bishop, C.M.: *Neural Networks for Pattern Recognition*. New York: Oxford University Press (1995)
- [6] Baldi, P., Frasconi, P., Smyth, P.: *Modeling the Internet and the Web - Probabilistic Methods and Algorithms*. New York: Wiley (2003)
- [7] Baldi, P., Brunak, S.: *Bioinformatics - A Machine Learning Approach*. Cambridge, MA: MIT Press (2003)
- [8] Sowa, J.: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. New York: PWS Publishing Co. (1999)
- [9] Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J., Richardson, J., Ringwald, M., Rubin, G., Sherlock, G.: *Gene ontology: tool for unification of biology*. *Nature Genetics* **25** (2000) 25–29

- [10] Reinoso-Castillo, J., Silvescu, A., Caragea, D., Pathak, J., Honavar, V.: Information extraction and integration from heterogeneous, distributed, autonomous information sources: a federated, query-centric approach. In: IEEE International Conference on Information Integration and Reuse, Las Vegas, Nevada (2003)
- [11] Caragea, D., Pathak, J., Honavar, V.: Learning classifiers from semantically heterogeneous data. In: Proceedings of the International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems. (2004)
- [12] Dzeroski, S., Lavrac, N., eds.: Relational Data Mining. Springer-Verlag (2001)
- [13] Getoor, L., Friedman, N., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In Dzeroski, S., N. Lavrac, E., eds.: Relational Data Mining. Springer-Verlag (2001)
- [14] Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Orlando, FL, Morgan Kaufmann Publishers Inc. (1999) 1300–1309
- [15] Atramentov, A., Leiva, H., Honavar, V.: Learning decision trees from multi-relational data. In Horvth, T., Yamamoto, A., eds.: Proceedings of the 13th International Conference on Inductive Logic Programming. Volume 2835 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2003) 38–56
- [16] Neville, J., Jensen, D., Gallagher, B.: Simple estimators for relational bayesian classifiers. In: ICDM 2003. (2003)
- [17] Casella, G., Berger, R.: Statistical Inference. Duxbury Press, Belmont, CA (2001)
- [18] Davidson, A.: Statistical Models. London: Cambridge University Press (2003)
- [19] Kearns, M.: Efficient noise-tolerant learning from statistical queries. Journal of the ACM **45** (1998) 983–1006
- [20] Caragea, D., Silvescu, A., Honavar, V.: A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. International Journal of Hybrid Intelligent Systems **1** (2004)
- [21] Caragea, D., Silvescu, A., Honavar, V.: Decision tree induction from distributed heterogeneous autonomous data sources. In: Proceedings of the International Conference on Intelligent Systems Design and Applications, Tulsa, Oklahoma (2003)
- [22] Caragea, D., Silvescu, A., Honavar, V.: Agents that learn from distributed dynamic data sources. In: Proceedings of the Workshop on Learning Agents, Agents 2000/ECML 2000, Barcelona, Spain (2000) 53–61
- [23] Caragea, C., Caragea, D., , Honavar, V.: Learning support vector machine classifiers from distributed data. extended abstract. In: Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2005). (2005)
- [24] Caragea, D.: Learning classifiers from Distributed, Semantically Heterogeneous, Autonomous Data Sources. Ph.d. thesis, Department of Computer Science. Iowa State University, Ames, Iowa, USA (2004)
- [25] Quinlan, R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
- [26] Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees. Wadsworth, Monterey, CA (1984)
- [27] Graefe, G., Fayyad, U., Chaudhuri, S.: On the efficient gathering of sufficient statistics for classification from large sql databases. In: Proceedings of the Fourth International Conference on KDD, Menlo Park, CA, AAAI Press (1998) 204–208
- [28] Moore, A.W., Lee, M.S.: Cached sufficient statistics for efficient machine learning with large datasets. Journal of Artificial Intelligence Research **8** (1998) 67–91

- [29] Wang, X., Schroeder, D., Dobbs, D., Honavar, V.: Data-driven discovery of rules for protein function classification based on sequence motifs: Rules discovered for peptidase families based on meme motifs outperform those based on prosite patterns and profiles. In: Proceedings of the Conference on Computational Biology and Genome Informatics. (2002)
- [30] Andorf, C., Silvescu, A., Dobbs, D., Honavar, V.: Learning classifiers for assigning protein sequences to gene ontology functional families. In: Fifth International Conference on Knowledge Based Computer Systems (KBCS 2004), India (2004)
- [31] Cortes, C., Vapnik, V.: Support vector networks. *Machine Learning* **20** (1995) 273–297
- [32] Cristianini, N., Shawe-Taylor, J.: *An Introduction to Support Vector Machines*. Cambridge University Press (2000)
- [33] Bradley, P.S., Mangasarian, O.L.: Massive data discrimination via linear support vector machines. *Optimization Methods and Software* **13(1)** (2000) 1–10
- [34] Srivastava, A., Han, E., Kumar, V., Singh, V.: Parallel formulations of decision-tree classification algorithms. *Data Mining and Knowledge Discovery* **3** (1999) 237–261
- [35] Grossman, L., Gou, Y.: Parallel methods for scaling data mining algorithms to large data sets. In Zytkow, J., ed.: *Handbook on Data Mining and Knowledge Discovery*. Oxford University Press (2001)
- [36] Provost, F.J., Kolluri, V.: A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery* **3** (1999) 131–169
- [37] Park, B., Kargupta, H.: Constructing simpler decision trees from ensemble models using Fourier analysis. In: Proceedings of the 7th Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD’2002), Madison, WI, ACM SIGMOD (2002) 18–23
- [38] Domingos, P.: Knowledge acquisition from examples via multiple models. In: Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, TN, Morgan Kaufmann (1997) 98–106
- [39] Prodromidis, A., Chan, P., Stolfo, S.: Meta-learning in distributed data mining systems: issues and approaches. In Kargupta, H., Chan, P., eds.: *Advances of Distributed Data Mining*. AAAI Press (2000)
- [40] Bhatnagar, R., Srinivasan, S.: Pattern discovery in distributed databases. In: Proceedings of the Fourteenth AAAI Conference, Providence, RI, AAAI Press/The MIT Press (1997) 503–508
- [41] Kargupta, H., Park, B., Hershberger, D., Johnson, E.: Collective data mining: A new perspective toward distributed data mining. In Kargupta, H., Chan, P., eds.: *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press (1999)
- [42] Mansour, J.: Learning boolean functions via the fourier transform. In: *Theoretical Advances in Neural Computation and Learning*. Kluwer (1994)
- [43] Levy, A.: Logic-based techniques in data integration. In: *Logic-based artificial intelligence*. Kluwer Academic Publishers (2000) 575–595
- [44] Caragea, D., Silvescu, A., Pathak, J., Bao, J., Andorf, C., Dobbs, D., Honavar, V.: Information integration and knowledge acquisition from semantically heterogeneous biological data sources. In: Proceedings of the Second International Workshop on Data Integration in Life Sciences, (DILS 2005), San Diego, CA, Berlin: Springer-Verlag. *Lecture Notes in Computer Science* (2005)
- [45] Bonatti, P., Deng, Y., Subrahmanian, V.: An ontology-extended relational algebra. In: Proceedings of the IEEE Conference on Information Integration and Reuse, IEEE Press (2003) 192–199

- [46] Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: C-owl: Contextualizing ontologies. In: Proceedings of the Second International Semantic Web Conference, Springer Verlag, LNCS 2870 (2003)
- [47] Bao, J., Honavar, V.: Collaborative ontology building with wiki@nt - a multi-agent based ontology building environment. In: Proceedings of the Third International Workshop on Evaluation of Ontology based Tools, at the Third International Semantic Web Conference ISWC, Hiroshima, Japan (2004)
- [48] Bao, J., Honavar, V.: An efficient algorithm for reasoning about subsumption and equivalence relationships to support collaborative editing of ontologies and inter-ontology mappings. under review. (2005)
- [49] Hull, R.: Managing semantic heterogeneity in databases: A theoretical perspective. In: PODS, Tucson, Arizona (1997) 51–61
- [50] Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, G., Stoeckert, C.: K2/Kleisli and GUS: experiments in integrated access to genomic data sources. *IBM Journal* **40** (2001)
- [51] Eckman, B.: A practitioner's guide to data management and data integration in bioinformatics. *Bioinformatics* (2003) 3–74
- [52] Sheth, A., Larson, J.: Federated databases: architectures and issues. *ACM Computing Surveys* **22** (1990) 183–236
- [53] Barsalou, T., Gangopadhyay, D.: M(dm): An open framework for interoperation of multimodel multidatabase systems. *IEEE Data Engineering* (1992)
- [54] Bright, M., Hurson, A., Pakzad, S.: A taxonomy and current issues in multi-database systems. *Computer Journal* **25** (1992) 5–60
- [55] Wiederhold, G., Genesereth, M.: The conceptual basis for mediation services. *IEEE Expert* **12** (1997) 38–47
- [56] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., Vassalos, V., Widom, J.: The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems, Special Issue on Next Generation Information Technologies and Systems* **8** (1997)
- [57] Chang, C.K., Garcia-Molina, H.: Mind your vocabulary: query mapping across heterogeneous information sources. In: ACM SIGMOD International Conference On Management of Data, Philadelphia, PA (1999)
- [58] Arens, Y., Chin, C., Hsu, C., Knoblock, C.: Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems* **2** (1993) 127–158
- [59] Knoblock, C., Minton, S., Ambite, J., Ashish, N., Muslea, I., Philpot, A., Tejada, S.: The ariadne approach to Web-based information integration. *International Journal of Cooperative Information Systems* **10** (2001) 145–169
- [60] Lu, J., Moerkotte, G., Schue, J., Subrahmanian, V.: Efficient maintenance of materialized mediated views. In: Proceedings of 1995 ACM SIGMOD Conference on Management of Data, San Jose, CA (1995)
- [61] Levy, A.: The information manifold approach to data integration. *IEEE Intelligent Systems* **13** (1998)
- [62] Draper, D., Halevy, A.Y., Weld, D.S.: The nimble XML data integration system. In: ICDE. (2001) 155–160
- [63] Etzold, T., Harris, H., Beulah, S.: SRS: An integration platform for databanks and analysis tools in bioinformatics. *Bioinformatics Managing Scientific Data* (2003) 35–74
- [64] Haas, L., Schwarz, P., Kodali, P., Kotlar, E., Rice, J., Swope, W.: DiscoveryLink: a system for integrated access to life sciences data sources. *IBM System Journal* **40** (2001)

- [65] Stevens, R., Goble, C., Paton, N., Becchofer, S., Ng, G., Baker, P., Bass, A.: Complex query formulation over diverse sources in tambis. *Bioinformatics* (2003) 189–220
- [66] Chen, J., Chung, S., Wong, L.: The Kleisli query system as a backbone for bioinformatics data integration and analysis. *Bioinformatics* (2003) 147–188
- [67] Tannen, V., Davidson, S., Harker, S.: The information integration in K2. *Bioinformatics* (2003) 225–248
- [68] Tomasic, A., Rashid, L., Valduriez, P.: Scaling heterogeneous databases and design of DISCO. *IEEE Transactions on Knowledge and Data Engineering* **10** (1998) 808–823
- [69] Haas, L., Kossmann, D., Wimmers, E., Yan, J.: Optimizing queries across diverse sources. In: *Proceedings of the 23rd VLDB Conference, Athens, Greece* (1997) 267–285
- [70] Rodriguez-Martinez, M., Roussopoulos, R.: MOCHA: a self-extensible database middleware system for distributed data sources. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX* (2000) 213–224
- [71] Lambrecht, E., Kambhampati, S., Gnanaprakasam, S.: Optimizing recursive information-gathering plans. In: *Proceedings of the International Joint Conference on Artificial Intelligence, AAAI Press* (1999) 1204–1211
- [72] Maluf, D., Wiederhold, G.: Abstraction of representation in interoperation. *Lecture Notes in AI* **1315** (1997)
- [73] Zhang, J., Honavar, V.: Learning decision tree classifiers from attribute-value taxonomies and partially specified data. In Fawcett, T., Mishra, N., eds.: *Proceedings of the International Conference on Machine Learning, Washington, DC* (2003) 880–887
- [74] Zhang, J., Honavar, V.: Learning concise and accurate naive bayes classifiers from attribute value taxonomies and data. In: *Proceedings of the Fourth ICMD.* (2004)
- [75] Haussler, D.: Quantifying inductive bias: AI learning algorithms and Valiant’s learning framework. *Artificial Intelligence* **36** (1988) 177–221
- [76] Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. *Machine Learning* **29** (1997)
- [77] Caragea, D., Zhang, J., Pathak, J., Honavar, V.: Learning classifiers from distributed, ontology-extended data sources. under review. (2005)
- [78] Walker, A.: On retrieval from a small version of a large database. In: *VLDB Conference, 1989.* (1989)
- [79] DeMichiel, L.: Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Trans. Knowl. Data Eng.* **1** (1989)
- [80] Chen, A., Tseng, F.: Evaluating aggregate operations over imprecise data. *IEEE Trans. On Knowledge and Data Engineering* **8** (1996)
- [81] McClean, S., Scotney, B., Shapcott, M.: Aggregation of imprecise and uncertain information in databases. *IEEE Transactions on Knowledge and Data Engineering* **6** (2001)
- [82] Bergadano, F., Giordana, A.: Guiding induction with domain theories. In: *Machine Learning An Artificial Intelligence Approach. Volume 3.* Palo Alto, CA: Morgan Kaufmann (1990)
- [83] Pazzani, M., Kibler, D.: The role of prior knowledge in inductive learning. *Machine Learning* **9** (1992)

- [84] Towell, G., Shavlik, J.: Knowledge-based artificial neural networks. *Artificial Intelligence* **70** (1994)
- [85] Aronis, J., Kolluri, V., Provost, F., Buchanan, B.: The WoRLD: knowledge discovery from multiple distributed databases. Technical Report ISL-96-6, Intelligent Systems Laboratory, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA (1996)
- [86] Aronis, J., Provost, F.: Increasing the efficiency of inductive learning with breadth-first marker propagation. In: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. (1997)
- [87] Nunez, M.: The use of background knowledge in decision tree induction. *Machine Learning* **6** (1991)
- [88] H., A., Akiba, Y., Kaneda, S.: On handling tree-structured attributes. In: *Proceedings of the Twelfth International Conference on Machine Learning*. (1995)
- [89] Dhar, V., Tuzhilin, A.: Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering* **5** (1993)
- [90] Han, J., Fu, Y.: Exploration of the power of attribute-oriented induction in data mining, u.m. fayyad et al. (eds.). In: *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press (1996)
- [91] Hendler, J., Stoffel, K., Taylor, M.: *Advances in high performance knowledge representation* (1996)
- [92] Taylor, M., Stoffel, K., Hendler, J.: Ontology-based induction of high level classification rules. In: *SIGMOD Data Mining and Knowledge Discovery workshop proceedings*, Tuscon, Arizona (1997)
- [93] Pazzani, M., Mani, S., Shankle, W.: Beyond concise and colorful: Learning intelligible rules. In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, Newport Beach, CA (1997)
- [94] Pazzani, M., Mani, M., Shankle, W.: Comprehensible knowledge discovery in databases. In: *Proceedings of the the Cognitive Science Conference*. (1997)
- [95] desJardins, M., Getoor, L., Koller, D.: Using feature hierarchies in bayesian network learning. In: *Proceedings of the Symposium on Abstraction, Reformulation, Approximation*. Lecture Notes in Artificial Intelligence 1864: 260-270, Springer-Verlag (2000)
- [96] Rubin, D.: Multiple imputations in sample surveys: A phenomenological bayesian approach to nonresponse (c/r: p29-34). In: *Proceedings of the American Statistical Association, Section on Survey Research Methods*. ((1978))
- [97] Rubin, D.: *Multiple imputation for nonresponse in surveys*. John Wiley and Sons (New York; Chichester) (1987)
- [98] Rubin, D.: Multiple imputation after 18+ years. *Journal of the American Statistical Association* **91** (1996)
- [99] Junninen, H., Niska, H., Tuppurainen, K., Ruuskanen, J., Kolehmainen, M.: Methods for imputation of missing values in air quality data sets. *Atmospheric Environment* **38** (2004)
- [100] Longford, N.: Missing data and small area estimation in the uk labour force survey. *Journal of the Royal Statistical Society Series A-Statistics in Society* **167** (2004)
- [101] Raghunathan, T.: What do we do with missing data? some options for analysis of incomplete data. *Annual Review of Public Health* **25** (2004)
- [102] Little, R., Rubin, D.: *Statistical analysis with missing data*. John Wiley and Sons (New York; Chichester), 2nd edition (2002)
- [103] Madow, W., Olkin, I., Rubin, D.B., e.: *Incomplete data in sample surveys (Vol. 2): Theory and bibliographies*. Academic Press (New York; London) (1983)

- [104] Madow, W., Nisselson, J., Olkin, I.e.: Incomplete data in sample surveys (Vol. 1): Report and case studies. Academic Press (New York; London) (1983)
- [105] Yan, C., Dobbs, D., Honavar, V.: A two-stage classifier for identification of protein-protein interface residues. *Bioinformatics* **20** (2004) i371–378
- [106] Yan, C., Honavar, V., Dobbs, D.: Identifying protein-protein interaction sites from surface residues - a support vector machine approach. *Neural Computing Applications* **13** (2004) 123–129



# Training Support Vector Machines via SMO-Type Decomposition Methods

Pai-Hsuen Chen, Rong-En Fan, and Chih-Jen Lin

Department of Computer Science, National Taiwan University

**Abstract.** This article gives a comprehensive study on SMO-type (Sequential Minimal Optimization) decomposition methods for training support vector machines. We propose a general and flexible selection of the two-element working set. Main theoretical results include 1) a simple asymptotic convergence proof, 2) a useful explanation of the shrinking and caching techniques, and 3) the linear convergence of this method. This analysis applies to any SMO-type implementation whose selection falls into the proposed framework.

## 1 Introduction

Given training vectors  $\mathbf{x}_i \in R^n, i = 1, \dots, l$ , in two classes, and a vector  $\mathbf{y} \in R^l$  such that  $y_i \in \{1, -1\}$ , the support vector machines (SVM) [1, 5] require the solution of the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) &= \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, i = 1, \dots, l, \\ & \mathbf{y}^T \boldsymbol{\alpha} = 0, \end{aligned} \tag{1}$$

where  $\mathbf{e}$  is the vector of all ones,  $C < \infty$  is the upper bound of all variables, and  $Q$  is an  $l$  by  $l$  symmetric positive semi-definite (PSD) matrix. Training vectors  $\mathbf{x}_i$  are mapped into a high dimensional space by the function  $\phi$ ,  $Q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ , and  $K(\mathbf{x}_i, \mathbf{x}_j)$  is the kernel function.

Due to the density of the matrix  $Q$ , traditional optimization methods cannot be directly applied to solve (1). Currently the decomposition method is one of the major methods to train SVM (e.g. [3, 8, 17, 19]). This method, an iterative procedure, considers only a small subset of variables per iteration. Denoted as  $B$ , this subset is called the working set. Since each iteration involves only  $|B|$  columns of the matrix  $Q$ , the memory problem is solved.

A special type of decomposition methods is the Sequential Minimal Optimization (SMO) [19], which restricts  $B$  to have only two elements. Then at each iteration one solves a simple two-variable problem without needing optimization software. It is sketched in the following:

**Algorithm 1 (SMO-type Decomposition methods).**

1. Find  $\alpha^1$  as the initial feasible solution. Set  $k = 1$ .
2. If  $\alpha^k$  is a stationary point of (1), stop. Otherwise, find a working set  $B = \{i, j\} \subset \{1, \dots, l\}$ . Define  $N \equiv \{1, \dots, l\} \setminus B$  and  $\alpha_B^k$  and  $\alpha_N^k$  as sub-vectors of  $\alpha^k$  corresponding to  $B$  and  $N$ , respectively.
3. Solve the following sub-problem with the variable  $\alpha_B$ :

$$\begin{aligned}
\min_{\alpha_B} \quad & \frac{1}{2} [\alpha_B^T (\alpha_N^k)^T] \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} - [\mathbf{e}_B^T \ \mathbf{e}_N^T] \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} \\
& = \frac{1}{2} [\alpha_i \ \alpha_j] \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + (-\mathbf{e}_B + Q_{BN} \alpha_N^k)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \text{const.} \\
\text{subject to} \quad & 0 \leq \alpha_i, \alpha_j \leq C, \\
& y_i \alpha_i + y_j \alpha_j = -\mathbf{y}_N^T \alpha_N^k.
\end{aligned} \tag{2}$$

Set  $\alpha_B^{k+1}$  to be the optimal point of (2).

4. Set  $\alpha_N^{k+1} \equiv \alpha_N^k$ . Set  $k \leftarrow k + 1$  and goto Step 2.

Note that the set  $B$  changes from one iteration to another. To simplify the notation, we use  $B$  instead of  $B^k$ . If a proper working set is used at each iteration, the function value  $f(\alpha^k)$  strictly decreases. However, this property does not imply that the sequence  $\{\alpha^k\}$  converges to a stationary point of (1). Thus, proving the convergence of decomposition methods is usually a challenging task. Under certain rules for selecting the working set, the asymptotic convergence has been established (e.g. [13, 2, 7, 18, 16]).

In this article, we give a comprehensive study on SMO-type decomposition methods. The selection of the two-element working set  $B$  is very general. Thus, all results obtained here apply to any SMO-type implementation whose selection falls into the category of consideration. In Section 2, we discuss existing working set selections for SMO-type methods and propose a general scheme. Section 3 then proves the asymptotic convergence.

Shrinking and caching are two effective techniques to speed up the decomposition methods. Earlier [15] has given the theoretical foundation of these two techniques, but requires some assumptions. In Section 4, we provide better and more general explanation without assumptions. Convergence rates are another important issue which indicates how fast the method approaches an optimal solution. We establish the linear convergence of the proposed method in Section 5. An example showing that linear rate is the best worst case analysis is in Section 6. This paper is based on the reports [4, 12], which include all proofs. Note that here we assume  $Q$  to be PSD, but [4] also considers indefinite kernels (e.g. tanh kernel).

## 2 Existing and New Working Set Selections

In this Section, we discuss existing working set selections and then propose a general scheme for SMO-type methods.

### 2.1 Existing Selections

A popular way to select the working set  $B$  is via the “maximal violating pair:”

#### WSS 1 (Working set selection via the “maximal violating pair”).

1. Select

$$i \in \arg \max_{t \in I_{\text{up}}(\alpha^k)} -y_t \nabla f(\alpha^k)_t, j \in \arg \min_{t \in I_{\text{low}}(\alpha^k)} -y_t \nabla f(\alpha^k)_t,$$

where

$$\begin{aligned} I_{\text{up}}(\alpha) &\equiv \{t \mid \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}, \\ I_{\text{low}}(\alpha) &\equiv \{t \mid \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}. \end{aligned} \tag{3}$$

2. Return  $B = \{i, j\}$ .

This working set was first proposed in [11] and has been used in, for example, LIBSVM [3] and  $SVM^{light}$  [8]<sup>1</sup>. WSS1 can be derived through the Karush-Kuhn-Tucker (KKT) optimality condition of (1): A vector  $\alpha$  is a stationary point if and only if there is a scalar  $b$  and two nonnegative vectors  $\lambda$  and  $\mu$  such that

$$\begin{aligned} \nabla f(\alpha) + b\mathbf{y} &= \lambda - \mu, \\ \lambda_i \alpha_i &= 0, \mu_i (C - \alpha)_i = 0, \lambda_i \geq 0, \mu_i \geq 0, i = 1, \dots, l, \end{aligned}$$

where  $\nabla f(\alpha) \equiv Q\alpha - \mathbf{e}$  is the gradient of  $f(\alpha)$ . This can be rewritten as

$$\nabla f(\alpha)_i + by_i \geq 0 \quad \text{if } \alpha_i < C, \tag{4}$$

$$\nabla f(\alpha)_i + by_i \leq 0 \quad \text{if } \alpha_i > 0. \tag{5}$$

Since  $y_i = \pm 1$ , by defining  $I_{\text{up}}(\alpha)$  and  $I_{\text{low}}(\alpha)$  as in (3), and rewriting (4)-(5) to obtain the range of  $b$ , a feasible  $\alpha$  is a stationary point of (1) if and only if

$$m(\alpha) \equiv \max_{i \in I_{\text{up}}(\alpha)} -y_i \nabla f(\alpha)_i \leq M(\alpha) \equiv \min_{i \in I_{\text{low}}(\alpha)} -y_i \nabla f(\alpha)_i. \tag{6}$$

Following [11], we define a “violating pair” of the condition (6).

**Definition 1 (Violating pair).**  $\dots, i \in I_{\text{up}}(\alpha), j \in I_{\text{low}}(\alpha) \dots -y_i \nabla f(\alpha)_i > -y_j \nabla f(\alpha)_j \dots \{i, j\} \dots$

From (6), the indices  $\{i, j\}$  which most violate the condition are a natural choice of the working set. They are called a “maximal violating pair” in WSS1. Violating pairs are essential in making decomposition methods work:

**Theorem 1 ([7]).**  $\dots f(\alpha^{k+1}) < f(\alpha^k), \forall k \dots B \dots$

<sup>1</sup> Though  $SVM^{light}$  allows  $|B| > 2$ , if  $|B| = 2$ , it essentially implements WSS1.

Unfortunately, having a violating pair in  $B$  and then the strict decrease of  $f(\boldsymbol{\alpha}^k)$  do not guarantee the convergence to a stationary point. An interesting example is from [10]: Given five data  $\mathbf{x}^1 = [1, 0, 0]^T$ ,  $\mathbf{x}^2 = [0, 1, 0]^T$ ,  $\mathbf{x}^3 = [0, 0, 1]^T$ ,  $\mathbf{x}^4 = [0.1, 0.1, 0.1]^T$ , and  $\mathbf{x}^5 = [0, 0, 0]^T$ . If  $y_1 = \dots = y_4 = -1, y_5 = 1$ ,  $C = 100$ , and the linear kernel  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$  is used, then the optimal solution of (1) is  $[0, 0, 0, 200/3, 200/3]^T$  and the optimal objective value is  $-200/3$ .

Starting with  $\boldsymbol{\alpha}^1 = [2, 0, 0, 0, 2]^T$ , if in the next three iterations we choose the following working sets:

$$\{1, 2\} \rightarrow \{2, 3\} \rightarrow \{3, 1\} , \tag{7}$$

the next three  $\boldsymbol{\alpha}$  are:

$$[1, 1, 0, 0, 2]^T \rightarrow [1, 0.5, 0.5, 0, 2]^T \rightarrow [0.75, 0.5, 0.75, 0, 2]^T .$$

If we continue the same way in (7) to choose the working set, the algorithm requires an infinite number of iterations. In addition, the sequence converges to a non-optimal point  $[2/3, 2/3, 2/3, 0, 2]^T$  with the objective value  $-10/3$ . Note that all working sets used are violating pairs.

A careful look at the procedure reveals why it fails to obtain the optimal solution. We have the following from the second to the fourth iterations:

$-y_t \nabla f(\boldsymbol{\alpha}^k)_t, t = 1, \dots, 5$	$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j$	$m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)$
$[0, 0, -1, -0.8, 1]^T$	1	2
$[0, -0.5, -0.5, -0.8, 1]^T$	0.5	1.8
$[-0.25, -0.5, -0.25, -0.8, 1]^T$	0.25	1.8

Clearly, the selected  $\{i, j\}$  is a violating pair, but does not reduce the maximal violation  $m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)$ . This discussion shows the importance of the maximal violating pair in the decomposition method. When such a pair is used (i.e. WSS1), the convergence has been established in [13, 14]. However, if  $B$  is any other violating pair, it is unclear whether SMO-type methods still converge to a stationary point or not. Motivated from the above analysis, we conjecture that a “sufficiently violated” pair is enough and propose a general working set selection in the following subsection.

## 2.2 A General Working Set Selection for SMO-Type Methods

We propose choosing a “constant-factor” violating pair as the working set. That is, the difference between the two selected indices is larger than a constant fraction of the maximal violation.

### WSS 2 (Working set selection: constant-factor violating pair).

1. Consider a fixed  $0 < \sigma \leq 1$  for all iterations.
2. Return  $B = \{i, j\}$  via selecting any  $i \in I_{\text{up}}(\boldsymbol{\alpha}^k), j \in I_{\text{low}}(\boldsymbol{\alpha}^k)$  satisfying

$$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \geq \sigma(m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)) > 0 . \tag{8}$$

Clearly (8) maintains the quality of the selected pair by linking it to the maximal violating pair. We can consider an even more general relationship between the two pairs:

**WSS 3 (Working set selection: a generalization of WSS2).**

1. Throughout all iterations, let  $h : R^1 \rightarrow R^1$  be any function satisfying
  - (a)  $h$  strictly increases on  $x \geq 0$ , and
  - (b)  $h(x) \leq x, \forall x \geq 0, h(0) = 0$ .
2. Return  $B = \{i, j\}$  via selecting any  $i \in I_{\text{up}}(\alpha^k), j \in I_{\text{low}}(\alpha^k)$  satisfying

$$-y_i \nabla f(\alpha^k)_i + y_j \nabla f(\alpha^k)_j \geq h(m(\alpha^k) - M(\alpha^k)) > 0 . \tag{9}$$

The condition  $h(x) \leq x$  ensures that there is at least one pair  $\{i, j\}$  satisfying (9). Clearly,  $h(x) = \sigma x$  with  $0 < \sigma \leq 1$  fulfills all required conditions and WSS3 reduces to WSS2. The function  $h$  can be in a more complicated form. For example, if

$$h(m(\alpha^k) - M(\alpha^k)) \equiv \min (m(\alpha^k) - M(\alpha^k), (m(\alpha^k) - M(\alpha^k))^2) , \tag{10}$$

then it also satisfies all requirements. In the rest of this paper, we analyze the SMO-type method using WSS3 for the working set selection.

### 3 Asymptotic Convergence of Using WSS3

The decomposition method generates a sequence  $\{\alpha^k\}$ . If it is finite, then a stationary point has been obtained. Hence we consider only the case of an infinite sequence. First we require a lemma:

**Lemma 1** ([13, 14]).

$$\{ \alpha^k \}, k \in \mathcal{K} \quad \bar{\alpha} \\ \{ \alpha^{k+s} \}, k \in \mathcal{K} \quad \bar{\alpha} \tag{1}$$

Since the feasible region of (1) is compact, there is at least one convergent subsequence of  $\{\alpha^k\}$ . The main convergence result is in the following:

**Theorem 2.**

$$\{ \alpha^k \} \quad \{ \alpha^k \} \tag{1}$$

Assume that  $\bar{\alpha}$  is the limit point of any convergent subsequence  $\{\alpha^k\}, k \in \mathcal{K}$ . If  $\bar{\alpha}$  is not a stationary point of (1), it does not satisfy the KKT condition (6). Thus, there is at least one “maximal violating pair”:

$$\bar{i} \in \arg m(\bar{\alpha}), \bar{j} \in \arg M(\bar{\alpha}), \text{ and } \Delta \equiv -y_{\bar{i}} \nabla f(\bar{\alpha})_{\bar{i}} + y_{\bar{j}} \nabla f(\bar{\alpha})_{\bar{j}} > 0 . \tag{11}$$

Lemma 1, the continuity of  $\nabla f(\boldsymbol{\alpha})$ , and  $h(\Delta/2) > 0$  from (11) imply that for any given  $r$ , there is  $\bar{k} \in \mathcal{K}$  such that for all  $k \in \mathcal{K}, k \geq \bar{k}$ :

$$\text{For } u = 0, \dots, r, \quad -y_{\bar{i}} \nabla f(\boldsymbol{\alpha}^{k+u})_{\bar{i}} + y_{\bar{j}} \nabla f(\boldsymbol{\alpha}^{k+u})_{\bar{j}} > \Delta/2 . \quad (12)$$

$$\text{If } i \in I_{\text{up}}(\bar{\boldsymbol{\alpha}}), \text{ then } i \in I_{\text{up}}(\boldsymbol{\alpha}^k), \dots, i \in I_{\text{up}}(\boldsymbol{\alpha}^{k+r}) . \quad (13)$$

$$\text{If } i \in I_{\text{low}}(\bar{\boldsymbol{\alpha}}), \text{ then } i \in I_{\text{low}}(\boldsymbol{\alpha}^k), \dots, i \in I_{\text{low}}(\boldsymbol{\alpha}^{k+r}) . \quad (14)$$

$$\begin{aligned} \text{If } -y_i \nabla f(\bar{\boldsymbol{\alpha}})_i > -y_j \nabla f(\bar{\boldsymbol{\alpha}})_j, \text{ then for } u = 0, \dots, r, \\ -y_i \nabla f(\boldsymbol{\alpha}^{k+u})_i > -y_j \nabla f(\boldsymbol{\alpha}^{k+u})_j . \end{aligned} \quad (15)$$

$$\begin{aligned} \text{If } -y_i \nabla f(\bar{\boldsymbol{\alpha}})_i = -y_j \nabla f(\bar{\boldsymbol{\alpha}})_j, \text{ then for } u = 0, \dots, r, \\ | -y_i \nabla f(\boldsymbol{\alpha}^{k+u})_i + y_j \nabla f(\boldsymbol{\alpha}^{k+u})_j | < h(\Delta/2) . \end{aligned} \quad (16)$$

$$\text{If } -y_i \nabla f(\bar{\boldsymbol{\alpha}})_i > -y_j \nabla f(\bar{\boldsymbol{\alpha}})_j \text{ and } \{i, j\} \text{ is the working set at the } (k+u)\text{th} \\ \text{iteration, } 0 \leq u \leq r-1, \text{ then } i \notin I_{\text{up}}(\boldsymbol{\alpha}^{k+u+1}) \text{ or } j \notin I_{\text{low}}(\boldsymbol{\alpha}^{k+u+1}) . \quad (17)$$

We give the details of deriving (12), and (13)-(16) are similar. Lemma 1 implies sequences  $\{\boldsymbol{\alpha}^k\}, \{\boldsymbol{\alpha}^{k+1}\}, \dots, \{\boldsymbol{\alpha}^{k+r}\}, k \in \mathcal{K}$  all converge to  $\bar{\boldsymbol{\alpha}}$ . For  $\{\boldsymbol{\alpha}^{k+u}\}$ , there is  $k_u$  such that (12) holds for  $k \geq k_u, k \in \mathcal{K}$ . As  $r$  is finite, by selecting  $\bar{k}$  to be the largest of these  $k_u, u = 0, \dots, r$ , we have (12) for all  $u = 0, \dots, r$ . Next we derive (17) [13–Lemma IV.4]: Similar to the optimality condition (6) for problem (1), for the  $(k+u)$ th sub-problem, if  $\boldsymbol{\alpha}_B$  is a stationary point, then

$$\max_{t \in I_{\text{up}}(\boldsymbol{\alpha}_B)} -y_t \nabla f \left( \left[ \begin{array}{c} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N^{k+u} \end{array} \right]_t \right) \leq \min_{t \in I_{\text{low}}(\boldsymbol{\alpha}_B)} -y_t \nabla f \left( \left[ \begin{array}{c} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N^{k+u} \end{array} \right]_t \right) .$$

Now  $B = \{i, j\}$  and  $\boldsymbol{\alpha}_B^{k+u+1}$  is a stationary point of the sub-problem satisfying the above inequality. If  $i \in I_{\text{up}}(\boldsymbol{\alpha}^{k+u+1})$  and  $j \in I_{\text{low}}(\boldsymbol{\alpha}^{k+u+1})$ , the above inequality implies  $-y_i \nabla f(\boldsymbol{\alpha}^{k+u+1})_i \leq -y_j \nabla f(\boldsymbol{\alpha}^{k+u+1})_j$ , an inequality which contradicts (15). Thus, (17) is valid.

We then rearrange components in  $\bar{\boldsymbol{\alpha}}$  so that

$$-y_1 \nabla f(\bar{\boldsymbol{\alpha}})_1 \leq \dots \leq -y_l \nabla f(\bar{\boldsymbol{\alpha}})_l , \quad (18)$$

and define

$$S^1(k) \equiv \sum \{i \mid i \in I_{\text{up}}(\boldsymbol{\alpha}^k)\} \text{ and } S^2(k) \equiv \sum \{l-i \mid i \in I_{\text{low}}(\boldsymbol{\alpha}^k)\} .$$

Clearly,

$$l \leq S^1(k) + S^2(k) \leq l(l-1) . \quad (19)$$

If  $\{i, j\}$  is selected at the  $(k+u)$ th iteration ( $u = 0, \dots, r$ ), we claim that

$$-y_i \nabla f(\bar{\boldsymbol{\alpha}})_i > -y_j \nabla f(\bar{\boldsymbol{\alpha}})_j . \quad (20)$$

It is impossible that  $-y_i \nabla f(\bar{\boldsymbol{\alpha}})_i < -y_j \nabla f(\bar{\boldsymbol{\alpha}})_j$  as  $-y_i \nabla f(\boldsymbol{\alpha}^{k+u})_i < -y_j \nabla f(\boldsymbol{\alpha}^{k+u})_j$  from (15) then violates (8). If they are equal, then

$$\begin{aligned} -y_i \nabla f(\boldsymbol{\alpha}^{k+u})_i + y_j \nabla f(\boldsymbol{\alpha}^{k+u})_j < h(\Delta/2) < h(-y_{\bar{i}} \nabla f(\boldsymbol{\alpha}^{k+u})_{\bar{i}} + y_{\bar{j}} \nabla f(\boldsymbol{\alpha}^{k+u})_{\bar{j}}) \\ \leq h(m(\boldsymbol{\alpha}^{k+u}) - M(\boldsymbol{\alpha}^{k+u})) . \end{aligned} \quad (21)$$

The first two inequalities come from (16) and (12), while the last is from  $\bar{i} \in I_{\text{up}}(\bar{\alpha}), \bar{j} \in I_{\text{low}}(\bar{\alpha})$ , (13), and (14). Clearly, (21) contradicts (8), so we have (20).

Next we use a counting procedure to obtain the contradiction of (11). From the  $k$ th to the  $(k + 1)$ st iteration, (20) and then (17) show that

$$i \notin I_{\text{up}}(\alpha^{k+1}) \text{ or } j \notin I_{\text{low}}(\alpha^{k+1}) .$$

For the first case, (13) implies  $i \notin I_{\text{up}}(\bar{\alpha})$  and hence  $i \in I_{\text{low}}(\bar{\alpha})$ . From (14) and the selection rule (8),  $i \in I_{\text{low}}(\alpha^k) \cap I_{\text{up}}(\alpha^k)$ . With  $i \notin I_{\text{up}}(\alpha^{k+1})$  and  $j \in I_{\text{low}}(\alpha^k)$ ,

$$S^1(k + 1) \leq S^1(k) - i + j \leq S^1(k) - 1, \quad S^2(k + 1) \leq S^2(k) , \quad (22)$$

where  $-i + j \leq -1$  comes from (18). Similarly, for the second case,

$$S^1(k + 1) \leq S^1(k), \quad S^2(k + 1) \leq S^2(k) - (l - j) + (l - i) \leq S^2(k) - 1 . \quad (23)$$

From iteration  $(k + 1)$  to  $(k + 2)$ , we can repeat the same argument. Note that (17) can still be used because of (20). Using (22) and (23), in  $r \equiv l(l - 1)$  iterations,  $S^1(k) + S^2(k)$  is reduced to zero, a contradiction to (19). Therefore, the assumption (11) is wrong and the proof is complete.  $\square$

We can easily prove that if (1) has a unique optimal solution, then  $\{\alpha^k\}$  globally converges. This happens under very general circumstances. For example, if the RBF kernel  $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$  is used and all  $\mathbf{x}_i \neq \mathbf{x}_j$ ,  $Q$  is PD, so (1) is strictly convex. Hence it has a unique solution.

Earlier convergence work [13] of using WSS1 (i.e., maximal violating pair) can not be applied here for WSS3, so we develop a novel counting technique in the proof. Details of the differences are discussed in [4].

## 4 Stopping Condition, Shrinking and Caching

In this section we discuss other important properties of the proposed method. Due to space limitation, we omit all proofs, which can be found in [4]. To begin, we show a set  $I$  used later is well defined.

**Theorem 3.**  $\bar{\alpha} \neq \hat{\alpha}$  (1)  $M(\bar{\alpha}) = m(\bar{\alpha}) = M(\hat{\alpha}) = m(\hat{\alpha})$ .

$$I \equiv \{i \mid -y_i \nabla f(\bar{\alpha})_i > M(\bar{\alpha}), -y_i \nabla f(\bar{\alpha})_i < m(\bar{\alpha})\} , \quad (24)$$

$$(1) \quad \alpha_i \quad i \in I$$

### 4.1 Stopping Condition and Finite Termination

As the decomposition method only asymptotically approaches an optimum, in practice, it terminates after satisfying a stopping condition. For example, we can pre-specify a small stopping tolerance  $\epsilon > 0$  and check if

$$m(\alpha^k) - M(\alpha^k) \leq \epsilon \quad (25)$$

has been satisfied or not. Though there are other stopping conditions, (25) is commonly used due to its closeness to the optimality condition (6). To justify its use, we must have that under any  $\epsilon > 0$ , the proposed method stops in a finite number of iterations. Thus, an infinite loop never happens. To have (25), one can prove a stronger condition:

$$\lim_{k \rightarrow \infty} m(\alpha^k) - M(\alpha^k) = 0 . \tag{26}$$

This condition is not readily available as from the respective definitions of  $m(\alpha)$  and  $M(\alpha)$ , it is unclear if they are continuous functions of  $\alpha$  or not.

The first study on the stopping condition of SMO-type methods is [9]. They consider a selection rule involving the stopping tolerance  $\epsilon$ . Since our selection is independent of  $\epsilon$ , their analysis cannot be applied here. Another work [15] proves (26) for WSS1 under the assumption that the sequence  $\{\alpha^k\}$  globally converges. Here, for the more general WSS3, we prove (26) without any assumption.

**Theorem 4.** *If the sequence  $\{\alpha^k\}$  converges to  $\alpha^*$ , then (26) holds.*

### 4.2 Shrinking and Caching

Shrinking and caching are two effective techniques to make the decomposition method faster. If an  $\alpha_i^k$  remains at 0 or  $C$  for many iterations, eventually it may stay at the same value. Based on this principle, the shrinking technique [8] reduces the size of the optimization problem without considering some bounded variables. The procedure then works on a smaller problem. In the end we add shrunk components back and check if an optimal solution of the original problem is obtained.

Another technique to reduce the training time is caching. Since  $Q$  may be too large to be stored, its elements are calculated as needed. We can allocate some space (called cache) to store recently used  $Q_{ij}$  [8]. If in final iterations only few columns of  $Q$  are still needed and the cache contains them, we can save many kernel evaluations. [15–Theorem II.3] has proved that in the final iterations of using WSS1, only a small subset of variables are still updated. Such a result supports the use of shrinking and caching techniques. However, this proof considers only any convergent subsequence of  $\{\alpha^k\}$  or assumes the global convergence. Here, we provide a more general theory without such assumptions.

**Theorem 5.** *If the sequence  $\{\alpha^k\}$  converges to  $\alpha^*$ , then (24) holds for all  $k \geq \bar{k}$  and  $i \in I$ , where  $I = \{i \in \{1, \dots, l\} \mid \alpha_i^* \in (0, C)\}$  and  $\forall k \geq \bar{k}$*

$$i \notin \{t \mid M(\alpha^k) \leq -y_t \nabla f(\alpha^k)_t \leq m(\alpha^k)\} . \tag{27}$$

Theorem 5 implies that in all final iterations, the SMO-type method involves only indices in the set

$$I' \equiv \{1, \dots, l\} \setminus I . \tag{28}$$



Thus, caching could be very effective. This theorem also illustrates two possible shrinking implementations:

1. Elements not in the set (27) are removed.
2. Any  $\alpha_i$  which has stayed at the same bound for a certain number of iterations is removed.

LIBSVM [3] considers the former approach, while *SVM<sup>light</sup>* [8] uses the latter.

## 5 Linear Convergence

Though we have proved the asymptotic convergence, it is important to investigate how fast the method converges. Under some assumptions, [12] was the first to prove the linear convergence of certain decomposition methods. It allows the working set to have more than two elements and WSS1 is a special case. Here we show that when the SMO-type working set selection is extended from WSS1 to WSS2, the same analysis holds.

Note that WSS3 uses a function  $h$  to control the quality of the selected pair. We will see in the proof that it may affect the convergence rate. Proving the linear convergence requires the condition (8), so results established in this section are for WSS2 but not WSS3.

First we make a few assumptions.

**Assumption 1.**  $Q \succ 0, \mathbf{p} \neq \mathbf{0}, \mathbf{y} \neq \mathbf{0}, \mathbf{y}^T \mathbf{y} > 0$ .

Then (1) is a strictly convex programming problem and hence has a unique global optimum  $\bar{\alpha}$ .

By Theorem 5, after large enough iterations working sets are all from the set  $I'$  defined in (28). From the optimality condition (6), the scalar  $\bar{b}$  satisfies  $\bar{b} = m(\bar{\alpha}) = M(\bar{\alpha})$ , and the set  $I'$  corresponds to elements satisfying

$$\nabla f(\bar{\alpha})_i + \bar{b}y_i = 0 \quad . \tag{29}$$

From (4)-(5), another form of the optimality condition, if  $\bar{\alpha}_i$  is not at a bound, (29) holds. We further assume that this is the only case that (29) happens:

**Assumption 2 (Nondegeneracy).**  $\bar{\alpha} \in \text{int}(C), \nabla f(\bar{\alpha})_i + \bar{b}y_i = 0, 0 < \bar{\alpha}_i < C$

This assumption, commonly used in optimization analysis, implies that indices of all bounded  $\bar{\alpha}_i$  are exactly the set  $I$ . Therefore, after enough iterations, Theorem 5 and Assumption 2 imply that all bounded variables are fixed and are not included in the working set. By treating these variables as constants, essentially we solve a problem with the following form:

$$\begin{aligned} \min_{\alpha} f(\alpha) &= \frac{1}{2} \alpha^T Q \alpha + \mathbf{p}^T \alpha \\ \text{subject to} \quad & \mathbf{y}^T \alpha = \Delta \quad , \end{aligned} \tag{30}$$

where  $\mathbf{p}$  is a vector by combining  $-\mathbf{e}$  and other terms related to the bounded components, and  $\Delta$  is a constant. Moreover,  $0 < \alpha_i^k < C$  for all  $k$  and  $i$  even though we do not explicitly write down inequality constraints in (30). Then the optimal solution  $\bar{\alpha}$  with the corresponding  $\bar{b}$  can be obtained by the following linear system:

$$\begin{bmatrix} Q & \mathbf{y} \\ \mathbf{y}^T & 0 \end{bmatrix} \begin{bmatrix} \bar{\alpha} \\ \bar{b} \end{bmatrix} = \begin{bmatrix} -\mathbf{p} \\ \Delta \end{bmatrix} . \tag{31}$$

At each iteration, we consider minimizing  $f(\alpha_B^k + \mathbf{d})$ , where  $\mathbf{d}$  is the direction from  $\alpha_B^k$  to  $\alpha_B^{k+1}$ , so the sub-problem (2) is written as

$$\begin{aligned} \min_{\mathbf{d}} \quad & \frac{1}{2} \mathbf{d}^T Q_{BB} \mathbf{d} + \nabla f(\alpha^k)_B^T \mathbf{d} \\ \text{subject to} \quad & \mathbf{y}_B^T \mathbf{d} = 0 , \end{aligned} \tag{32}$$

where  $\nabla f(\alpha^k) = Q\alpha^k + \mathbf{p}$ . If an optimal solution of (32) is  $\mathbf{d}^k$ , then  $\alpha_B^{k+1} = \alpha_B^k + \mathbf{d}^k$  and  $\alpha_N^{k+1} = \alpha_N^k$ . With the corresponding  $b^k$ , this sub-problem is solved by the following equation:

$$\begin{bmatrix} Q_{BB} & \mathbf{y}_B \\ \mathbf{y}_B^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d}^k \\ b^k \end{bmatrix} = \begin{bmatrix} -\nabla f(\alpha^k)_B \\ 0 \end{bmatrix} . \tag{33}$$

Using (31),

$$\begin{aligned} Q(\alpha^k - \bar{\alpha}) &= Q\alpha^k + \mathbf{p} + \bar{b}\mathbf{y} \\ &= \nabla f(\alpha^k) + \bar{b}\mathbf{y} . \end{aligned} \tag{34}$$

By defining  $Y \equiv \text{diag}(\mathbf{y})$  to be a diagonal matrix with elements of  $\mathbf{y}$  on the diagonal, and using  $y_i = \pm 1$ ,

$$-YQ(\alpha^k - \bar{\alpha}) = -Y\nabla f(\alpha^k) - \bar{b}\mathbf{e} . \tag{35}$$

The purpose of checking  $Q(\alpha^k - \bar{\alpha})$  is to see how close the current solution is to the optimal one. Then (35) links it to  $-Y\nabla f(\alpha^k)$ , a vector used for the working set selection. Remember that for finding violating pairs, we first sort  $-y_i \nabla f(\alpha^k)_i$  in an ascending order.

The following two theorems are main results on the linear convergence.

**Theorem 6.**

(1) Let  $c < 1$  and  $\bar{k}$  such that for  $k \geq \bar{k}$

$$(\alpha^{k+1} - \bar{\alpha})^T Q(\alpha^{k+1} - \bar{\alpha}) \leq c(\alpha^k - \bar{\alpha})^T Q(\alpha^k - \bar{\alpha}) . \tag{36}$$

First, Theorem 5 implies that there is  $\bar{k}$  such that after  $k \geq \bar{k}$ , the problem is reduced to (30). We then directly calculate the difference between the  $(k+1)$ st and the  $k$ th iterations:

$$(\boldsymbol{\alpha}^{k+1} - \bar{\boldsymbol{\alpha}})^T Q(\boldsymbol{\alpha}^{k+1} - \bar{\boldsymbol{\alpha}}) - (\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}})^T Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) \quad (37)$$

$$= 2(\mathbf{d}^k)^T (Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))_B + (\mathbf{d}^k)^T Q_{BB} \mathbf{d}^k$$

$$= (\mathbf{d}^k)^T (2(Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))_B - \nabla f(\boldsymbol{\alpha}^k)_B - b^k \mathbf{y}_B) \quad (38)$$

$$= (\mathbf{d}^k)^T ((Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))_B + (\bar{b} - b^k) \mathbf{y}_B) \quad (39)$$

$$= (\mathbf{d}^k)^T ((Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))_B + (b^k - \bar{b}) \mathbf{y}_B) \quad (40)$$

$$= -[-(Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))_B + (\bar{b} - b^k) \mathbf{y}_B]^T Q_{BB}^{-1} [-(Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))_B + (\bar{b} - b^k) \mathbf{y}_B] ,$$

where (38) is from (33), (39) is from (34), (40) is by using the fact  $\mathbf{y}_B^T \mathbf{d}^k = 0$  from (33), and the last equality is from (33) and (34). If we define

$$\hat{Q} \equiv Y_B Q_{BB}^{-1} Y_B \text{ and } \mathbf{v} \equiv -Y(Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}})) , \quad (41)$$

where  $Y_B \equiv \text{diag}(\mathbf{y}_B)$ , then  $\mathbf{v}_B = -Y_B(Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))_B$  and (37) becomes

$$-[\mathbf{v}_B + (\bar{b} - b^k) \mathbf{e}_B]^T \hat{Q} [\mathbf{v}_B + (\bar{b} - b^k) \mathbf{e}_B] . \quad (42)$$

From (35), we define

$$\begin{aligned} v^1 &\equiv \max_t (v_t) = m(\boldsymbol{\alpha}^k) - \bar{b} , \\ v^l &\equiv \min_t (v_t) = M(\boldsymbol{\alpha}^k) - \bar{b} . \end{aligned} \quad (43)$$

Thus, the selection rule (8) of WSS2 implies

$$|v_i - v_j| \geq \sigma(v^1 - v^l) , \quad (44)$$

where  $\{i, j\}$  is the working set of the  $k$ th iteration.

We denote that  $\min(\text{eig}(\cdot))$  and  $\max(\text{eig}(\cdot))$  to be the minimal and maximal eigenvalues of a matrix, respectively. A further calculation of (42) shows

$$\begin{aligned} &[\mathbf{v}_B + (\bar{b} - b^k) \mathbf{e}_B]^T \hat{Q} [\mathbf{v}_B + (\bar{b} - b^k) \mathbf{e}_B] \\ &\geq \min(\text{eig}(\hat{Q})) [\mathbf{v}_B + (\bar{b} - b^k) \mathbf{e}_B]^T [\mathbf{v}_B + (\bar{b} - b^k) \mathbf{e}_B] \\ &\geq \min(\text{eig}(\hat{Q})) \max_{t \in B} (v_t + (\bar{b} - b^k))^2 \\ &\geq \min(\text{eig}(\hat{Q})) \left(\frac{v_i - v_j}{2}\right)^2, \text{ where } \{i, j\} \text{ is working set} \end{aligned} \quad (45)$$

$$\geq \min(\text{eig}(\hat{Q})) \sigma^2 \left(\frac{v^1 - v^l}{2}\right)^2 \quad (46)$$

$$\geq \min(\text{eig}(\hat{Q})) \left(\frac{\mathbf{y}^T Q^{-1} \mathbf{y}}{2 \sum_{i,j} |Q_{ij}^{-1}|}\right)^2 \sigma^2 \max(|v^1|, |v^l|)^2 \quad (47)$$

$$\geq \frac{\min(\text{eig}(\hat{Q}))}{l} \left(\frac{\mathbf{y}^T Q^{-1} \mathbf{y}}{2 \sum_{t,s} |Q_{ts}^{-1}|}\right)^2 \sigma^2 (Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))^T Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) \quad (48)$$

$$\begin{aligned} &\geq \frac{\min(\text{eig}(\hat{Q}))}{l \max(\text{eig}(Q^{-1}))} \left(\frac{\mathbf{y}^T Q^{-1} \mathbf{y}}{2 \sum_{t,s} |Q_{ts}^{-1}|}\right)^2 \sigma^2 (Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))^T Q^{-1} Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) \\ &\geq \frac{\min(\text{eig}(\hat{Q}))}{l \max(\text{eig}(Q^{-1}))} \left(\frac{\mathbf{y}^T Q^{-1} \mathbf{y}}{2 \sum_{t,s} |Q_{ts}^{-1}|}\right)^2 \sigma^2 (\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}})^T Q(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) , \end{aligned} \quad (49)$$

where (45) is from Lemma 4, (46) is from (44), (47) is from Lemma 5, and (48) follows from (43). Note that both lemmas are given in Appendix A.

Here we give more details about the derivation of (47): If  $v^1 v^l \leq 0$ , then of course

$$|v^1 - v^l| \geq \max(|v^1|, |v^l|) .$$

With  $y_i = \pm 1$ ,  $\frac{\mathbf{y}^T Q^{-1} \mathbf{y}}{\sum_{t,s} |Q_{ts}^{-1}|} \leq 1$  so (47) follows. On the other hand, if  $v^1 v^l \geq 0$ , we consider  $\mathbf{v} = (YQY)(-Y(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}))$  from (41). Since  $-\mathbf{e}^T Y(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) = -\mathbf{y}^T (\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) = 0$ , we can apply Lemma 5: With

$$\begin{aligned} |(YQY)_{ij}^{-1}| &= |Q_{ij}^{-1} y_i y_j| = |Q_{ij}^{-1}| \text{ and} \\ \mathbf{e}^T (YQY)^{-1} \mathbf{e} &= \mathbf{y}^T Q^{-1} \mathbf{y} , \end{aligned}$$

we have

$$\begin{aligned} |v^1 - v^l| &\geq \max(|v^1|, |v^l|) - \min(|v^1|, |v^l|) \\ &\geq \left( \frac{\mathbf{y}^T Q^{-1} \mathbf{y}}{\sum_{t,s} |Q_{ts}^{-1}|} \right) \max(|v^1|, |v^l|) , \end{aligned}$$

which implies (47).

Finally we can define a constant  $c$  as follows:

$$c \equiv 1 - \min_B \left( \frac{\min(\text{eig}(Q_{BB}^{-1}))}{l \max(\text{eig}(Q^{-1}))} \left( \frac{\mathbf{y}^T Q^{-1} \mathbf{y}}{2 \sum_{t,s} |Q_{ts}^{-1}|} \right)^2 \sigma^2 \right) < 1 ,$$

where  $B$  is any two-element subset of  $\{1, \dots, l\}$ . Combining (42) and (49), after  $k \geq \bar{k}$ , (36) holds.  $\square$

Note that the condition (8) of Algorithm 2 is used in (44) and then (46). If WSS3 is considered, in (46) we will have a term  $h((v^1 - v^l)/2)^2$ . Thus, the function  $h$  affects the convergence rate. Since  $h(x) \leq x$ , linear rate is the best using our derivation.

The linear convergence of the objective function is as follows:

**Theorem 7.** *If  $c < 1$ , then for any  $k \geq \bar{k}$*

$$f(\boldsymbol{\alpha}^{k+1}) - f(\bar{\boldsymbol{\alpha}}) \leq c(f(\boldsymbol{\alpha}^k) - f(\bar{\boldsymbol{\alpha}})) .$$

We will show that for any  $k \geq \bar{k}$ ,

$$f(\boldsymbol{\alpha}^k) - f(\bar{\boldsymbol{\alpha}}) = \frac{1}{2} (\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}})^T Q (\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) , \tag{50}$$

so the proof immediately follows from Theorem 6. Using (31),

$$\begin{aligned}
 & f(\boldsymbol{\alpha}^k) - f(\bar{\boldsymbol{\alpha}}) \\
 &= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q \boldsymbol{\alpha}^k + \mathbf{e}^T \boldsymbol{\alpha}^k - \frac{1}{2}(\bar{\boldsymbol{\alpha}})^T Q \bar{\boldsymbol{\alpha}} - \mathbf{e}^T \bar{\boldsymbol{\alpha}} \\
 &= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q \boldsymbol{\alpha}^k + (-Q\bar{\boldsymbol{\alpha}} - \bar{\mathbf{b}}\mathbf{y})^T \boldsymbol{\alpha}^k - \frac{1}{2}(\bar{\boldsymbol{\alpha}})^T Q \bar{\boldsymbol{\alpha}} - (-Q\bar{\boldsymbol{\alpha}} - \bar{\mathbf{b}}\mathbf{y})^T \bar{\boldsymbol{\alpha}} \\
 &= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q \boldsymbol{\alpha}^k - (\bar{\boldsymbol{\alpha}})^T Q \boldsymbol{\alpha}^k + \frac{1}{2}(\bar{\boldsymbol{\alpha}})^T Q \bar{\boldsymbol{\alpha}} \\
 &= \frac{1}{2}(\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}})^T Q (\boldsymbol{\alpha}^k - \bar{\boldsymbol{\alpha}}) .
 \end{aligned} \tag{51}$$

Since we always keep the feasibility of  $\boldsymbol{\alpha}^k$ , (51) comes from  $\mathbf{y}^T \boldsymbol{\alpha}^k = \mathbf{y}^T \bar{\boldsymbol{\alpha}}$ .  $\square$

## 6 An Example of Linear Convergence

We have shown that under some general conditions, the decomposition method using WSS 2 is at least linearly convergent. However, it is unclear whether the convergence is actually better than linear or not. Here, we present a simple example which exactly has the linear convergence. Hence, in theory, the linear convergence is already the best worst-case analysis.

Consider  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$  with  $\|\mathbf{x}_1 - \mathbf{x}_2\| = \|\mathbf{x}_1 - \mathbf{x}_3\| = \|\mathbf{x}_2 - \mathbf{x}_3\|$ ,  $\mathbf{y} = [1, 1, -1]^T$ , and  $C$  is large enough. If the RBF kernel is used, the dual SVM problem is

$$\begin{aligned}
 & \min_{\alpha_1, \alpha_2, \alpha_3} \frac{1}{2} [\alpha_1 \ \alpha_2 \ \alpha_3] \begin{bmatrix} 1 & a & -a \\ a & 1 & -a \\ -a & -a & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} - (\alpha_1 + \alpha_2 + \alpha_3) \\
 & \text{subject to } \alpha_1 + \alpha_2 - \alpha_3 = 0 , \\
 & \quad 0 \leq \alpha_1, \alpha_2, \alpha_3 ,
 \end{aligned} \tag{52}$$

where  $a = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$ . At the optimal solution,

$$\bar{\boldsymbol{\alpha}} = \left[ \frac{2}{3(1-a)} \ \frac{2}{3(1-a)} \ \frac{4}{3(1-a)} \right]^T . \tag{53}$$

It satisfies the linear constraint  $\alpha_1 + \alpha_2 - \alpha_3 = 0$ . With  $b \equiv 1/3$ , it also satisfies the following condition:

$$\begin{bmatrix} 1 & a & -a \\ a & 1 & -a \\ -a & -a & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + b \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} .$$

Thus,  $\bar{\boldsymbol{\alpha}}$  is an optimal solution. For convenience, we define

$$e_i^k \equiv \alpha_i^k - \bar{\alpha}_i, i = 1, 2, 3 .$$

For the discussion below, we require only that the working set is a violating pair. We then state two lemmas and give proofs in Appendix B.

**Lemma 2.** ,  $\alpha^1 = [0, 0, 0]^T$  ,  $k \geq 3$  ,  $\alpha^k$  ,

$$\begin{aligned} \alpha_1^k, \alpha_2^k, \alpha_3^k &> 0, \\ -y_i \nabla f(\alpha^k)_i &= -y_j \nabla f(\alpha^k)_j, \end{aligned}$$

,  $\{i, j\}$  ,  $(k-1)$  ,

**Lemma 3.**

,  $k \geq 3$  ,  $(k-1)$  ,  $\{1, 3\}$  ,

$$2e_1^k + e_2^k = 0 . \tag{54}$$

,  $\{2, 3\}$  ,

$$e_1^k + 2e_2^k = 0 . \tag{55}$$

,  $\{1, 2\}$  ,

$$e_1^k - e_2^k = 0 . \tag{56}$$

,  $k \geq 2$  ,  $e_i^k \neq 0, i = 1, 2, 3$

The following theorem proves the linear reduction of the objective value.

**Theorem 8.** ,  $\alpha^1 = [0, 0, 0]^T$  ,  $k \geq 3$  ,

$$(\alpha^{k+1} - \bar{\alpha})^T Q (\alpha^{k+1} - \bar{\alpha}) = \frac{1}{4} (\alpha^k - \bar{\alpha})^T Q (\alpha^k - \bar{\alpha}) . \tag{57}$$

Now the size of the working set is two, so the three possible sets are  $\{1, 2\}$ ,  $\{1, 3\}$ , and  $\{2, 3\}$ . With  $\alpha_3^k = \alpha_1^k + \alpha_2^k$ ,

$$\begin{aligned} &(\alpha^k - \bar{\alpha})^T Q (\alpha^k - \bar{\alpha}) \\ &= 2(1-a)((e_1^k)^2 + (e_2^k)^2 + e_1^k e_2^k) . \end{aligned}$$

If  $\{1, 3\}$  is the working set at the  $k$ th iteration, then from (54) and  $\alpha_2^{k+1} = \alpha_2^k$ ,

$$\begin{aligned} &(\alpha^{k+1} - \bar{\alpha})^T Q (\alpha^{k+1} - \bar{\alpha}) \\ &= 2(1-a)((e_1^{k+1})^2 + (e_2^{k+1})^2 + e_1^{k+1} e_2^{k+1}) \\ &= \frac{3}{2}(1-a)(e_2^k)^2 . \end{aligned} \tag{58}$$

To have (57), we thus need

$$\frac{\frac{3}{2}(1-a)(e_2^k)^2}{2(1-a)((e_1^k)^2 + (e_2^k)^2 + e_1^k e_2^k)} = \frac{1}{4} ,$$

which is equivalent to

$$(e_1^k - e_2^k)(e_1^k + 2e_2^k) = 0 . \quad (59)$$

The current working set  $\{1, 3\}$  can not be the working set of the  $(k - 1)$ st (i.e., the previous) iteration. Otherwise, Lemma 2 implies that it is not a violating pair at the  $k$ th iteration, a contradiction to our assumption on the selecting the working set. Therefore, at the  $(k - 1)$ st iteration, the working set must be either  $\{1, 2\}$  or  $\{2, 3\}$ . Thus, (59) follows from (56) or (55) under Lemma 3.

The proof for the case that  $\{2, 3\}$  is the working set is very similar. If  $\{1, 2\}$  is the working set, putting (56) into (58), (59) becomes

$$(2e_1^k + e_2^k)(e_1^k + 2e_2^k) = 0 ,$$

so the result also follows.  $\square$

Finally, using (50), we have the linear convergence:

$$f(\boldsymbol{\alpha}^{k+1}) - f(\bar{\boldsymbol{\alpha}}) = \frac{1}{4}(f(\boldsymbol{\alpha}^k) - f(\bar{\boldsymbol{\alpha}})) .$$

## 7 Conclusions

Optimization issues in training support vector machines are interesting and challenging. In this article we have studied many theoretical issues on solving the SVM quadratic programming problem. Practical implementations also benefit from such analysis. The recent paper [6] considers a working set selection based on WSS 2 and reports faster training than existing implementations.

## References

- [1] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.
- [2] C.-C. Chang, C.-W. Hsu, and C.-J. Lin. The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008, 2000.
- [3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [4] P.-H. Chen, R.-E. Fan, and C.-J. Lin. A study on SMO-type decomposition methods for support vector machines. Technical report, Department of Computer Science, National Taiwan University, 2005. <http://www.csie.ntu.edu.tw/~cjlin/papers/generalSMO.pdf>.
- [5] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.
- [6] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training SVM. Technical report, Department of Computer Science, National Taiwan University, 2005.
- [7] D. Hush and C. Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51:51–71, 2003.

[8] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

[9] S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46:351–360, 2002.

[10] S. S. Keerthi and C. J. Ong. On the role of the threshold parameter in SVM training algorithms. Technical Report CD-00-09, Department of Mechanical and Production Engineering, National University of Singapore, Singapore, 2000.

[11] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt’s SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

[12] C.-J. Lin. Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

[13] C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, 2001.

[14] C.-J. Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13(1):248–250, 2002.

[15] C.-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 13(5):1045–1052, 2002.

[16] N. List and H. U. Simon. A general convergence theorem for the decomposition method. In *Proceedings of the 17th Annual Conference on Learning Theory*, pages 363–377, 2004.

[17] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of CVPR’97*, pages 130–136, New York, NY, 1997. IEEE.

[18] L. Palagi and M. Sciandrone. On the convergence of a modified version of SVM<sup>light</sup> algorithm. *Optimization Methods and Software*, 20(2-3):315–332, 2005.

[19] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

## A Proof of Lemmas Used in Section 5

**Lemma 4.**  $v_1 \geq \dots \geq v_l$

$$\max_i (|v_i|) \geq \frac{v_1 - v_l}{2}$$

We notice that  $\max_i (|v_i|)$  must happen at  $v_1$  or  $v_l$ . It is easy to see

$$\frac{v_1 - v_l}{2} \leq \frac{|v_1| + |v_l|}{2} \leq \max(|v_1|, |v_l|) \quad \square$$

**Lemma 5.**  $Q \mathbf{v}_1, \dots, Q \mathbf{v}_l, \dots, Q \mathbf{x}_1, \dots, Q \mathbf{x}_n$

$$\begin{aligned} \mathbf{e}^T \mathbf{x} &= 0 \\ \mathbf{v} \equiv Q \mathbf{x} \quad \max_i ((Q \mathbf{x})_i) &= v^1 > v^l = \min_i ((Q \mathbf{x})_i) \quad v^1 v^l \geq 0 \end{aligned}$$



$$\min(|v^1|, |v^l|) \leq \left(1 - \frac{\mathbf{e}^T Q^{-1} \mathbf{e}}{\sum_{i,j} |Q_{ij}^{-1}|}\right) \max(|v^1|, |v^l|) .$$

Since  $v^1 > v^l$  and  $v^1 v^l \geq 0$ , we have either  $v^1 > v^l \geq 0$  or  $0 \geq v^1 > v^l$ . For the first case, if the result is wrong,

$$v^l > \left(1 - \frac{\mathbf{e}^T Q^{-1} \mathbf{e}}{\sum_{i,j} |Q_{ij}^{-1}|}\right) v^1 ,$$

so for  $j = 1, \dots, l$ ,

$$\begin{aligned} v^1 - v_j &\leq v^1 - v^l \\ &< \left(\frac{\mathbf{e}^T Q^{-1} \mathbf{e}}{\sum_{i,j} |Q_{ij}^{-1}|}\right) v^1 . \end{aligned} \tag{60}$$

With  $\mathbf{x} = Q^{-1} \mathbf{v}$  and (60),

$$\begin{aligned} \mathbf{e}^T \mathbf{x} &= \mathbf{e}^T Q^{-1} \mathbf{v} \\ &= \sum_{i,j} Q_{ij}^{-1} v_j \\ &= \sum_{i,j} Q_{ij}^{-1} (v^1 - (v^1 - v_j)) \\ &\geq v^1 \mathbf{e}^T Q^{-1} \mathbf{e} - (v^1 - v^l) \sum_{i,j} |Q_{ij}^{-1}| \\ &> v^1 \left( \mathbf{e}^T Q^{-1} \mathbf{e} - \left(\frac{\mathbf{e}^T Q^{-1} \mathbf{e}}{\sum_{i,j} |Q_{ij}^{-1}|}\right) \sum_{i,j} |Q_{ij}^{-1}| \right) \\ &= 0 \end{aligned}$$

causes a contradiction. The case of  $0 \geq v^1 > v^l$  is similar.  $\square$

## B Proof of Lemmas Used in Section 6

### B.1 Proof of Lemma 2

At the first iteration,  $\{1, 3\}$  and  $\{2, 3\}$  are violating pairs and can be selected as the working set. Without loss of generality, we assume it is  $\{1, 3\}$ . Thus,

$$\boldsymbol{\alpha}^2 = \left[ \frac{1}{1-a} \ 0 \ \frac{1}{1-a} \right]^T .$$

For  $k \geq 3$ ,  $\alpha_3^k$  must be larger than 0. Otherwise, from  $\mathbf{y}^T \boldsymbol{\alpha}^k = 0$ ,  $\alpha_1^k = \alpha_2^k = \alpha_3^k = 0$ . Then  $f(\boldsymbol{\alpha}^k) = f(\boldsymbol{\alpha}^1)$  contradicts that  $f(\boldsymbol{\alpha}^k)$  is strictly decreasing. Since  $\boldsymbol{\alpha}^2$  is the optimum when  $\alpha_2 = 0$ , if  $\alpha_2^k = 0$ , then  $f(\boldsymbol{\alpha}^k) \geq f(\boldsymbol{\alpha}^2)$ . This

result again violates the decreasing property. Hence  $\alpha_2^k > 0, \forall k \geq 3$ . Similarly, if  $\alpha_1^k = 0$ , then  $f(\boldsymbol{\alpha}^k) \geq f([0 \ \frac{1}{1-a} \ \frac{1}{1-a}]^T) = f(\boldsymbol{\alpha}^2)$  causes a contradiction.

Consider any  $k \geq 3$  and assume  $\{i, j\}$  is the working set at the  $(k-1)$ st iteration. By the above discussion,  $\alpha_i^k > 0$  and  $\alpha_j^k > 0$ . The optimality condition of the sub-problem then implies  $-y_i \nabla f(\boldsymbol{\alpha}^k)_i = -y_j \nabla f(\boldsymbol{\alpha}^k)_j$ .

## B.2 Proof of Lemma 3

We calculate the gradient at  $\boldsymbol{\alpha}$ :

$$\nabla f(\boldsymbol{\alpha})_1 = \alpha_1 + a\alpha_2 - a\alpha_3 - 1 \quad , \quad (61)$$

$$\nabla f(\boldsymbol{\alpha})_2 = a\alpha_1 + \alpha_2 - a\alpha_3 - 1 \quad , \quad (62)$$

$$\nabla f(\boldsymbol{\alpha})_3 = -a\alpha_1 - a\alpha_2 + \alpha_3 - 1 \quad . \quad (63)$$

If  $\{1, 3\}$  is the working set at the  $(k-1)$ st iteration, then Lemma 2 implies  $-y_1 \nabla f(\boldsymbol{\alpha}^k)_1 = -y_3 \nabla f(\boldsymbol{\alpha}^k)_3$ . We obtain

$$\alpha_1^k + \alpha_3^k = 2/(1-a) \quad , \quad (64)$$

Using (64) and  $\mathbf{y}^T \boldsymbol{\alpha}^k = 0$ ,

$$2\alpha_1^k + \alpha_2^k = 2/(1-a) \quad . \quad (65)$$

Since  $\bar{\boldsymbol{\alpha}}$  is optimal and  $\bar{\alpha}_1 > 0, \bar{\alpha}_3 > 0$ , we have  $-y_1 \nabla f(\bar{\boldsymbol{\alpha}})_1 = -y_3 \nabla f(\bar{\boldsymbol{\alpha}})_3$ . This and  $\mathbf{y}^T \bar{\boldsymbol{\alpha}} = 0$  implies

$$2\bar{\alpha}_1 + \bar{\alpha}_2 = 2/(1-a) \quad . \quad (66)$$

Combining (65) and (66), we obtain the desired result:

$$2e_1^k + e_2^k = 0 \quad . \quad (67)$$

For  $\{2, 3\}$  or  $\{1, 2\}$  to be the working set at the  $(k-1)$ st iteration, the deviation is similar. Thus the first result of this lemma is complete.

We prove the second result by induction. Directly from  $\boldsymbol{\alpha}^2 - \bar{\boldsymbol{\alpha}}$ , we have  $e_i^2 \neq 0, i = 1, 2, 3$ . Given  $k \geq 3$ , assume results hold for the  $(k-1)$ st iteration. If  $\{1, 3\}$  is the working set at the  $(k-1)$ st iteration, then  $e_2^k = e_2^{k-1} \neq 0$ . With (67) from the above discussion,  $e_1^k = -e_2^k/2 \neq 0$ . Using  $\mathbf{y}^T \boldsymbol{\alpha}^k = \mathbf{y}^T \bar{\boldsymbol{\alpha}} = 0$ ,  $e_1^k + e_2^k - e_3^k = 0$ , so

$$e_3^k = e_1^k + e_2^k = e_2^k/2 \neq 0 \quad .$$

For  $\{2, 3\}$  or  $\{1, 2\}$  to be the working set, the proof is similar.

# Measuring Statistical Dependence with Hilbert-Schmidt Norms

Arthur Gretton<sup>1</sup>, Olivier Bousquet<sup>2</sup>, Alex Smola<sup>3</sup>, and Bernhard Schölkopf<sup>1</sup>

<sup>1</sup> MPI for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany  
{arthur, bernhard.schoelkopf}@tuebingen.mpg.de

<sup>2</sup> Pertinence, 32, Rue des Jeûneurs, 75002 Paris, France  
olivier.bousquet@pertinence.com

<sup>3</sup> National ICT Australia, North Road, Canberra 0200 ACT, Australia  
alex.smola@nicta.com.au

**Abstract.** We propose an independence criterion based on the eigen-spectrum of covariance operators in reproducing kernel Hilbert spaces (RKHSs), consisting of an empirical estimate of the Hilbert-Schmidt norm of the cross-covariance operator (we term this a Hilbert-Schmidt Independence Criterion, or HSIC). This approach has several advantages, compared with previous kernel-based independence criteria. First, the empirical estimate is simpler than any other kernel dependence test, and requires no user-defined regularisation. Second, there is a clearly defined population quantity which the empirical estimate approaches in the large sample limit, with exponential convergence guaranteed between the two: this ensures that independence tests based on HSIC do not suffer from slow learning rates. Finally, we show in the context of independent component analysis (ICA) that the performance of HSIC is competitive with that of previously published kernel-based criteria, and of other recently published ICA methods.

## 1 Introduction

Methods for detecting dependence using kernel-based approaches have recently found application in a wide variety of areas. Examples include independent component analysis [3, 10], gene selection [20], descriptions of gait in terms of hip and knee trajectories [15], feature selection [9], and dependence detection in fMRI signals [11]. The principle underlying these algorithms is that we may define covariance and cross-covariance operators in RKHSs, and derive statistics from these operators suited to measuring the dependence between functions in these spaces.

In the method of Bach and Jordan [3], a regularised correlation operator was derived from the covariance and cross-covariance operators, and its largest singular value (the kernel canonical correlation, or KCC) was used as a statistic to test independence. The approach of Gretton et al. [11] was to use the largest singular value of the cross-covariance operator, which behaves identically to the

correlation operator at independence, but is easier to define and requires no regularisation — the resulting test is called the constrained covariance (COCO). Both these quantities fall within the framework set out by Rényi [17], namely that for sufficiently rich function classes, the functional correlation (or, alternatively, the cross-covariance) serves as an independence test, being zero only when the random variables tested are independent. Various empirical kernel quantities (derived from bounds on the mutual information that hold near independence)<sup>1</sup> were also proposed based on the correlation and cross-covariance operators in [3, 10], however their connection to the population covariance operators remains to be established (indeed, the population quantities to which these approximations converge are not yet known). Gretton *et al.* [11] showed that these various quantities are guaranteed to be zero for independent random variables only when the associated RKHSs are universal [19].

The present study extends the concept of COCO by using the  $\ell_2$  spectrum of the cross-covariance operator to determine when all its singular values are zero, rather than looking only at the largest singular value; the idea being to obtain a more robust indication of independence. To this end, we use the sum of the squared singular values of the cross-covariance operator (i.e., its squared Hilbert-Schmidt norm) to measure dependence — we call the resulting quantity the Hilbert-Schmidt Independence Criterion (HSIC).<sup>2</sup> It turns out that the empirical estimate of HSIC is identical to the  $\ell_2$  norm of the quadratic dependence measure of Achard *et al.* [1], although we shall see that their derivation approaches this criterion in a completely different way. Thus, the present work resolves the open question in [1] regarding the link between the quadratic dependence measure and kernel dependence measures based on RKHSs, and generalises this measure to metric spaces (as opposed to subsets of the reals). More importantly, however, we believe our proof assures that HSIC is indeed a dependence criterion under all circumstances (i.e., HSIC is zero if and only if the random variables are independent), which is not necessarily guaranteed in [1]. We give a more detailed analysis of Achard’s proof in Appendix B.

Compared with previous kernel independence measures, HSIC has several advantages:

- The empirical estimate is much simpler — just the trace of a product of Gram matrices — and, unlike the canonical correlation or kernel generalised variance [3], HSIC does not require extra regularisation terms for good finite sample behaviour.
- The empirical estimate converges to the population estimate at rate  $1/\sqrt{m}$ , where  $m$  is the sample size, and thus independence tests based on HSIC do not suffer from slow learning rates [8]. In particular, as the sample size increases, we are guaranteed to detect any existing dependence with high

---

<sup>1</sup> Respectively the Kernel Generalised Variance (KGV) and the Kernel Mutual Information (KMI).

<sup>2</sup> The possibility of using a Hilbert-Schmidt norm was suggested by Fukumizu *et al.* [9], although the idea was not pursued further in that work.

probability. Of the alternative kernel dependence tests, this result is proved only for the constrained covariance [11].

- The finite sample bias of the estimate is  $O(m^{-1})$ , and is therefore negligible compared to the finite sample fluctuations (which underly the convergence rate in the previous point). This is currently proved for , , other kernel dependence test, including COCO.
- Experimental results on an ICA problem show that the new independence test is superior to the previous ones, and competitive with the best existing specialised ICA methods. In particular, kernel methods are substantially more resistant to outliers than other specialised ICA algorithms.

We begin our discussion in Section 2, in which we define the cross-covariance operator between RKHSs, and give its Hilbert-Schmidt (HS) norm (this being the population HSIC). In Section 3, we give an empirical estimate of the HS norm, and establish the link between the population and empirical HSIC by determining the bias of the finite sample estimate. In Section 4, we demonstrate exponential convergence between the population HSIC and empirical HSIC. As a consequence of this fast convergence, we show in Section 5 that dependence tests formulated using HSIC do not suffer from slow learning rates. Also in this section, we describe an efficient approximation to the empirical HSIC based on the incomplete Cholesky decomposition. Finally, in Section 6, we apply HSIC to the problem of independent component analysis (ICA).

## 2 Cross-Covariance Operators

In this section, we provide the functional analytic background necessary in describing cross-covariance operators between RKHSs, and introduce the Hilbert-Schmidt norm of these operators. Our presentation follows [21, 12], the main difference being that we deal with cross-covariance operators rather than the covariance operators.<sup>3</sup> We also draw on [9], which uses covariance and cross-covariance operators as a means of defining conditional covariance operators, but does not investigate the Hilbert-Schmidt norm; and on [4], which characterises the covariance and cross-covariance operators for general Hilbert spaces.

### 2.1 RKHS Theory

Consider a Hilbert space  $\mathcal{F}$  of functions from  $\mathcal{X}$  to  $\mathbb{R}$ . Then  $\mathcal{F}$  is a reproducing kernel Hilbert space if for each  $x \in \mathcal{X}$ , the Dirac evaluation operator  $\delta_x : \mathcal{F} \rightarrow \mathbb{R}$ , which maps  $f \in \mathcal{F}$  to  $f(x) \in \mathbb{R}$ , is a bounded linear functional. To each point  $x \in \mathcal{X}$ , there corresponds an element  $\phi(x) \in \mathcal{F}$  such that  $\langle \phi(x), \phi(x') \rangle_{\mathcal{F}} = k(x, x')$ , where  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a unique positive definite kernel. We will require in particular that  $\mathcal{F}$  be separable (it must have a complete orthonormal system).

---

<sup>3</sup> Briefly, a cross-covariance operator maps from one space to another, whereas a covariance operator maps from a space to itself. In the linear algebraic case, the covariance is  $C_{xx} := \mathbf{E}_{\mathbf{x}}[\mathbf{x}\mathbf{x}^{\top}] - \mathbf{E}_{\mathbf{x}}[\mathbf{x}]\mathbf{E}_{\mathbf{x}}[\mathbf{x}^{\top}]$ , while the cross-covariance is  $C_{xy} := \mathbf{E}_{\mathbf{x},\mathbf{y}}[\mathbf{x}\mathbf{y}^{\top}] - \mathbf{E}_{\mathbf{x}}[\mathbf{x}]\mathbf{E}_{\mathbf{y}}[\mathbf{y}^{\top}]$ .

As pointed out in [12–Theorem 7], any continuous kernel on a separable  $\mathcal{X}$  (e.g.  $\mathbb{R}^n$ ) induces a separable RKHS.<sup>4</sup> We likewise define a second separable RKHS,  $\mathcal{G}$ , with kernel  $l(\cdot, \cdot)$  and feature map  $\psi$ , on the separable space  $\mathcal{Y}$ .

Denote by  $C : \mathcal{G} \rightarrow \mathcal{F}$  a linear operator. Then provided the sum converges, the Hilbert-Schmidt (HS) norm of  $C$  is defined as

$$\|C\|_{\text{HS}}^2 := \sum_{i,j} \langle Cv_i, u_j \rangle_{\mathcal{F}}^2, \tag{1}$$

where  $u_i$  and  $v_j$  are orthonormal bases of  $\mathcal{F}$  and  $\mathcal{G}$  respectively. It is easy to see that this generalises the Frobenius norm on matrices.

A linear operator  $C : \mathcal{G} \rightarrow \mathcal{F}$  is called a Hilbert-Schmidt operator if its HS norm exists. The set of Hilbert-Schmidt operators  $\text{HS}(\mathcal{G}, \mathcal{F}) : \mathcal{G} \rightarrow \mathcal{F}$  is a separable Hilbert space with inner product

$$\langle C, D \rangle_{\text{HS}} := \sum_{i,j} \langle Cv_i, u_j \rangle_{\mathcal{F}} \langle Dv_i, u_j \rangle_{\mathcal{F}}.$$

Let  $f \in \mathcal{F}$  and  $g \in \mathcal{G}$ . Then the tensor product operator  $f \otimes g : \mathcal{G} \rightarrow \mathcal{F}$  is defined as

$$(f \otimes g)h := f \langle g, h \rangle_{\mathcal{G}} \text{ for all } h \in \mathcal{G}. \tag{2}$$

Moreover, by the definition of the HS norm, we can compute the HS norm of  $f \otimes g$  via

$$\begin{aligned} \|f \otimes g\|_{\text{HS}}^2 &= \langle f \otimes g, f \otimes g \rangle_{\text{HS}} = \langle f, (f \otimes g)g \rangle_{\mathcal{F}} \\ &= \langle f, f \rangle_{\mathcal{F}} \langle g, g \rangle_{\mathcal{G}} = \|f\|_{\mathcal{F}}^2 \|g\|_{\mathcal{G}}^2 \end{aligned} \tag{3}$$

## 2.2 The Cross-Covariance Operator

We assume that  $(\mathcal{X}, \Gamma)$  and  $(\mathcal{Y}, \Lambda)$  are furnished with probability measures  $p_x, p_y$  respectively ( $\Gamma$  being the Borel sets on  $\mathcal{X}$ , and  $\Lambda$  the Borel sets on  $\mathcal{Y}$ ). We may now define the mean elements with respect to these measures as those members of  $\mathcal{F}$  and  $\mathcal{G}$  respectively for which

$$\begin{aligned} \langle \mu_x, f \rangle_{\mathcal{F}} &:= \mathbf{E}_x [\langle \phi(x), f \rangle_{\mathcal{F}}] = \mathbf{E}_x [f(x)], \\ \langle \mu_y, g \rangle_{\mathcal{G}} &:= \mathbf{E}_y [\langle \psi(y), g \rangle_{\mathcal{G}}] = \mathbf{E}_y [g(y)], \end{aligned} \tag{4}$$

where  $\phi$  is the feature map from  $\mathcal{X}$  to the RKHS  $\mathcal{F}$ , and  $\psi$  maps from  $\mathcal{Y}$  to  $\mathcal{G}$ . Finally,  $\|\mu_x\|_{\mathcal{F}}^2$  can be computed by applying the expectation twice via

$$\|\mu_x\|_{\mathcal{F}}^2 = \mathbf{E}_{x,x'} [\langle \phi(x), \phi(x') \rangle_{\mathcal{F}}] = \mathbf{E}_{x,x'} [k(x, x')]. \tag{5}$$

<sup>4</sup> For more detail on separable RKHSs and their properties, see [12] and references therein.

Here the expectation is taken over independent copies  $x, x'$  taken from  $p_x$ . The means  $\mu_x, \mu_y$  exist as long as their respective norms in  $\mathcal{F}$  and  $\mathcal{G}$  are bounded, which is true when the kernels  $k$  and  $l$  are bounded (since then  $\mathbf{E}_{x,x'}[k(x, x')] < \infty$  and  $\mathbf{E}_{y,y'}[l(y, y')] < \infty$ ). We are now in a position to define the cross-covariance operator.

Following [4, 9],<sup>5</sup> the operator associated with the joint measure  $p_{x,y}$  on  $(\mathcal{X} \times \mathcal{Y}, \Gamma \times \Lambda)$  is a linear operator  $C_{xy} : \mathcal{G} \rightarrow \mathcal{F}$  defined as

$$C_{xy} := \mathbf{E}_{x,y} [(\phi(x) - \mu_x) \otimes (\psi(y) - \mu_y)] = \underbrace{\mathbf{E}_{x,y} [\phi(x) \otimes \psi(y)]}_{:=\tilde{C}_{xy}} - \underbrace{\mu_x \otimes \mu_y}_{:=M_{xy}}. \quad (6)$$

Here (6) follows from the linearity of the expectation. We will use  $\tilde{C}_{xy}$  and  $M_{xy}$  as the basis of our measure of dependence. Our next goal is to derive the Hilbert-Schmidt norm of the above quantity; the conditions under which  $C_{xy}$  is a HS operator will then follow from the existence of the norm.

### 2.3 Hilbert-Schmidt Independence Criterion

**Definition 1 (HSIC).** Let  $p_{xy}$  be a joint measure on  $(\mathcal{X} \times \mathcal{Y}, \Gamma \times \Lambda)$  and let  $\mathcal{F}, \mathcal{G}$  be Hilbert spaces. The Hilbert-Schmidt Independence Criterion (HSIC) is defined as

$$\text{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) := \|C_{xy}\|_{\text{HS}}^2. \quad (7)$$

To compute it we need to express HSIC in terms of kernel functions. This is achieved by the following lemma:

**Lemma 1 (HSIC in terms of kernels).**

$$\text{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) = \mathbf{E}_{x,x',y,y'}[k(x, x')l(y, y')] + \mathbf{E}_{x,x'}[k(x, x')]\mathbf{E}_{y,y'}[l(y, y')] - 2\mathbf{E}_{x,y}[\mathbf{E}_{x'}[k(x, x')]\mathbf{E}_{y'}[l(y, y')]] \quad (8)$$

Here  $\mathbf{E}_{x,x',y,y'}$  denotes the expectation over independent pairs  $(x, y)$  and  $(x', y')$  drawn from  $p_{xy}$ . This lemma is proved in Appendix A. It follows from Lemma 8 that the HS norm of  $C_{xy}$  exists when the various expectations over the kernels are bounded, which is true as long as the kernels  $k$  and  $l$  are bounded.

## 3 Empirical Criterion

In order to show that HSIC is a practical criterion for testing independence, and to obtain a formal independence test on the basis of HSIC, we need to perform three more steps. First, we need to approximate  $\text{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G})$  given a finite

<sup>5</sup> Our operator (and that in [9]) differs from Baker’s in that Baker defines all measures directly on the function spaces.

number of observations. Second, we need to show that this approximation converges to HSIC sufficiently quickly. Third, we need to show that HSIC is, indeed, an indicator for the independence of random variables (subject to appropriate choice of  $\mathcal{F}$  and  $\mathcal{G}$ ). We address the first step in this section, and the remaining two steps in the two sections that follow.

### 3.1 Estimator of HSIC

**Definition 2 (Empirical HSIC).** Let  $Z := \{(x_1, y_1), \dots, (x_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$  be a sample of size  $m$  from a joint distribution  $p_{xy}$  on  $\mathcal{X} \times \mathcal{Y}$ . The empirical HSIC is defined as

$$\text{HSIC}(Z, \mathcal{F}, \mathcal{G}) := (m - 1)^{-2} \text{tr} K H L H \tag{9}$$

$$H, K, L \in \mathbb{R}^{m \times m} \quad K_{ij} := k(x_i, x_j), L_{ij} := l(y_i, y_j) \quad H_{ij} := \delta_{ij} - m^{-1}$$

An advantage of  $\text{HSIC}(Z, \mathcal{F}, \mathcal{G})$  is that it can be computed in  $O(m^2)$  time, whereas other kernel methods cost at least  $O(m^3)$  before approximations are made (although in practice, this advantage is somewhat academic, since good approximations to all kernel dependence criteria can be computed in similar time: see [3–Section 4] and Section 5.2). What we now need to show is that it is indeed related to  $\text{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G})$ :

**Theorem 1 ( $O(m^{-1})$  Bias of Estimator).** Let  $\mathbf{E}_Z$  denote the expectation over  $Z$  from  $p_{xy}$ . Then

$$\text{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) = \mathbf{E}_Z [\text{HSIC}(Z, \mathcal{F}, \mathcal{G})] + O(m^{-1}).$$

This means that if the variance of  $\text{HSIC}(Z, \mathcal{F}, \mathcal{G})$  is larger than  $O(m^{-1})$  (and indeed, the uniform convergence bounds we derive with respect to  $p_{x,y}$  will be  $O(m^{-1/2})$ ), the bias arising from the definition of  $\text{HSIC}(Z, \mathcal{F}, \mathcal{G})$  is negligible in the overall process. The proof is in Appendix A.

## 4 Large Deviation Bounds

As a next step we need to show that the deviation between  $\text{HSIC}[Z, \mathcal{F}, \mathcal{G}]$  and its expectation is not too large. This section repeatedly uses a bound from [13–p.25], which applies to U-statistics of the form we encounter in the previous section.

**Theorem 2 (Deviation bound for U-statistics).** Let  $u$  be a U-statistic of order  $r$  with kernel  $g$  and mean  $\mathbf{E}_u[u] = a$ . Then for any  $t > 0$ ,

$$u := \frac{1}{\binom{m}{r}} \sum_{i_1^r} g(x_{i_1}, \dots, x_{i_r}),$$

$$g \text{ is bounded, } a \leq g \leq b \text{ and } t > 0$$

$$\mathbf{P}_u \{u - \mathbf{E}_u[u] \geq t\} \leq \exp \left( -\frac{2t^2 \lceil m/r \rceil}{(b - a)^2} \right).$$

We now state our main theorem. The proof is in Appendix A.

<sup>6</sup> We denote  $\binom{m}{n} := \frac{m!}{(m-n)!}$ .



**Theorem 3 (Bound on Empirical HSIC).**

Let  $k, l$  be the number of samples for  $X$  and  $Y$  respectively. Let  $m > 1$  and  $\delta > 0$ . Let  $p_{x,y}$  be a joint distribution on  $\mathcal{X} \times \mathcal{Y}$  and  $Z$  be a sample from  $p_{x,y}$ . Then

$$|\text{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) - \text{HSIC}(Z, \mathcal{F}, \mathcal{G})| \leq \sqrt{\frac{\log(6/\delta)}{\alpha^2 m}} + \frac{C}{m},$$

where  $\alpha^2 > 0.24$  and  $C$  is a constant.

## 5 Independence Tests Using HSIC

In this section, we describe how HSIC can be used as an independence measure, and as the basis for an independence test. We also describe an approximation to HSIC which is more efficient to compute. We begin by demonstrating that the Hilbert-Schmidt norm can be used as a measure of independence, as long as the associated RKHSs are universal [19].

**Theorem 4 ( $C_{xy}$  and Independence).**

Let  $k, l$  be the number of samples for  $\mathcal{X}$  and  $\mathcal{Y}$  respectively. Let  $\|f\|_\infty \leq 1$  and  $\|g\|_\infty \leq 1$  for  $f \in \mathcal{F}$  and  $g \in \mathcal{G}$ . Let  $\|C_{xy}\|_{\text{HS}} = 0$ . Then  $x$  and  $y$  are independent.

According to Gretton [11], the largest singular value (i.e., the  $\|C_{xy}\|_{\text{S}}$ ) is zero if and only if  $x$  and  $y$  are independent, under the conditions specified in the theorem. Since  $\|C_{xy}\|_{\text{S}} = 0$  if and only if  $\|C_{xy}\|_{\text{HS}} = 0$ , it follows that  $\|C_{xy}\|_{\text{HS}} = 0$  if and only if  $x$  and  $y$  are independent. ■

### 5.1 Independence Tests

We now describe how to use HSIC as the basis of an independence test. Consider a set  $\mathcal{P}$  of probability distributions  $p_{x,y}$ . We may decompose  $\mathcal{P}$  into two subsets:  $\mathcal{P}_i$  contains distributions  $p_{x,y}^{(i)}$  under which  $x$  and  $y$  are independent, and  $\mathcal{P}_d$  contains distributions  $p_{x,y}^{(d)}$  under which  $x$  and  $y$  are dependent.

We next introduce a test  $\Delta(Z)$ , which takes a data set  $Z \sim p_Z$ , where  $p_Z$  is the distribution corresponding to  $m$  independent draws from  $p_{x,y}$ , and returns

$$\Delta(Z) = \begin{cases} 1 & \text{if } Z \sim p_Z^{(d)} \\ 0 & \text{if } Z \sim p_Z^{(i)} \end{cases}$$

Given that the test sees only a finite sample, it cannot determine with complete certainty from which class of distributions the data are drawn. We call  $\Delta$  an  $\alpha$ -test when

$$\sup_{p_{x,y}^{(i)} \in \mathcal{P}_i} \mathbf{E}_{Z \sim p_Z^{(i)}}[\Delta(Z) = 1] \leq \alpha.$$

In other words  $\alpha$  upper bounds the probability of a Type I error. It follows from Theorem 3 that the empirical HSIC converges to the population HSIC at speed

$1/\sqrt{m}$ . This means that if we define the independence test  $\Delta(Z)$  as the indicator that HSIC is larger than a term of the form  $C\sqrt{\log(1/\alpha)/m}$ , with  $C$  a suitable constant, then  $\Delta(Z)$  is an  $\alpha$ -test with Type II error upper bounded by a term approaching zero as  $1/\sqrt{m}$ .

## 5.2 Efficient Computation

Computational cost is another factor in using HSIC as an independence criterion. As in [3], we use a low rank decomposition of the Gram matrices via an incomplete Cholesky decomposition, which permits an accurate approximation to HSIC as long as the kernel has a fast decaying spectrum. This results in the following cost saving, which we use in our experiments. The proof is in Appendix A.

**Lemma 2 (Efficient approximation to HSIC).**  $K \approx AA^\top$ ,  $L \approx BB^\top$ ,  $A \in \mathbb{R}^{m \times d_f}$ ,  $B \in \mathbb{R}^{m \times d_g}$ ,  $\text{tr}HKHL = O(m(d_f^2 + d_g^2))$ .

Finally, note that although the present measure of dependence pertains only to the two-variable case, a test of pairwise dependence for a greater number of variables may easily be defined by summing HSIC over every pair of variables — this quantity vanishes if and only if the random variables are pairwise independent. We use this generalisation in the experiments of Section 6.

## 6 Experimental Results

We apply our estimates of statistical dependence to the problem of linear instantaneous independent component analysis [14]. In this setting, we assume a random source vector  $\mathbf{s}$  of dimension  $n$ , where  $s_i \in \mathbb{R}$ , such that the components are mutually independent;  $p_{\mathbf{s}}(\mathbf{s}) = \prod_{i=1}^n p_{s_i}(s_i)$ . We observe a vector  $\mathbf{t}$  that corresponds to a linear mixing  $\mathbf{t} = \mathbf{A}\mathbf{s}$  of the sources  $\mathbf{s}$ , where  $\mathbf{A}$  is an  $n \times n$  matrix with full rank.<sup>7</sup> We wish to recover an estimate  $\mathbf{x}$  of the unmixed elements  $\mathbf{s}$  given  $m$  i.i.d. samples from  $p_{\mathbf{t}}(\mathbf{t})$ , and using the linear mixing model and the fact that the unmixed components are independent. This problem is indeterminate in certain respects: for instance, the ordering and scale of the sources cannot be recovered using independence alone.

It is clear that the various cross-covariance based kernel dependence tests, including HSIC, can each be used to determine when the inverse  $\mathbf{V}$  of  $\mathbf{A}$  is found,<sup>8</sup> by testing the pairwise independence of the components in  $\mathbf{x} = \mathbf{V}\mathbf{t}$  (bearing in mind Theorem 4 and its implications for the various kernel dependence tests). This requires a gradient descent procedure in which the kernel contrasts are minimised as a function of  $\mathbf{V}$ ; see [3, 10] for details. The Amari divergence [2],

<sup>7</sup> This assumes the number of sources is equal to the number of sensors, and the sources are spatially distinct.

<sup>8</sup> Up to permutation and scaling, and assuming no more than one source is Gaussian [14].

which is invariant to permutation and scaling, is used to compare  $\mathbf{V}$  and  $\mathbf{A}^{-1}$ . We acknowledge that the application of a general dependence function to linear ICA is not an optimal non-parametric approach to the problem of estimating the entries in  $\mathbf{A}$ , as discussed in [18]. Indeed, most specialised ICA algorithms exploit the linear mixing structure of the problem to avoid having to conduct a general test of independence, which makes the task of recovering  $\mathbf{A}$  easier. That said, ICA is in general a good benchmark for dependence measures, in that it applies to a problem with a known “ground truth”, and tests that the dependence measures approach zero gracefully as dependent random variables are made to approach independence (through optimisation of the unmixing matrix).

As well as the kernel algorithms, we also compare with three standard ICA methods (FastICA [14], Jade [6], and Infomax [5]); and two recent state of the art methods, neither of them based on kernels: RADICAL [16], which uses order statistics to obtain entropy estimates; and characteristic function based ICA (CFICA) [7].<sup>9</sup> It was recommended to run the CFICA algorithm with a good initialising guess; we used RADICAL for this purpose. All kernel algorithms were initialised using Jade (except for the 16 source case, where Fast ICA was used due to its more stable output). RADICAL is based on an exhaustive grid search over all the Jacobi rotations, and does not require an initial guess.

Our first experiment consisted in demixing data drawn independently from several distributions chosen at random with replacement from Table 1, and mixed with a random matrix having condition number between 1 and 2. In the case of the KCC and KGV, we use the parameters recommended in [3]: namely,  $\kappa = 2 \times 10^{-2}$  and  $\sigma = 1$  for  $m \leq 1000$ ,  $\kappa = 2 \times 10^{-3}$  and  $\sigma = 0.5$  for  $m > 1000$  ( $\sigma$  being the kernel size, and  $\kappa$  the coefficient used to scale the regularising terms). In the case of our dependence tests (COCO, KMI, HSIC), we used  $\sigma = 1$  for the Gaussian kernel, and  $\sigma = 3$  for the Laplace kernel. After convergence, the kernel size was halved for all methods, and the solution refined in a “polishing” step. Results are given in Table 2.

**Table 1.** Densities used, and their respective kurtoses. Densities have zero mean and unit variance.

Density	Kurtosis
Student, 3 DOF	$\infty$
Double exponential	3.00
Uniform	-1.20
Student, 5 DOF	6.00
Exponential	6.00
2 double exponentials	-1.70
Symmetric. 2 Gaussians, multimodal	-1.85
As above, transmodal	-0.75
As above, unimodal	-0.50
Asymmetric. 2 Gaussians, multimodal	-0.57
As above, transmodal	-0.29
As above, unimodal	-0.20
Symmetric. 4 Gaussians, multimodal	-0.91
As above, transmodal	-0.34
As above, unimodal	-0.40
Asymmetric. 4 Gaussians, multimodal	-0.67
As above, transmodal	-0.59
As above, unimodal	-0.82

<sup>9</sup> We are aware that the same authors propose an alternative algorithm, “Efficient ICA”. We did not include results from this algorithm in our experiments, since it is unsuited to mixtures of Gaussians (which have fast decaying tails) and discontinuous densities (such as the uniform density on a finite interval), which both occur in our benchmark set.

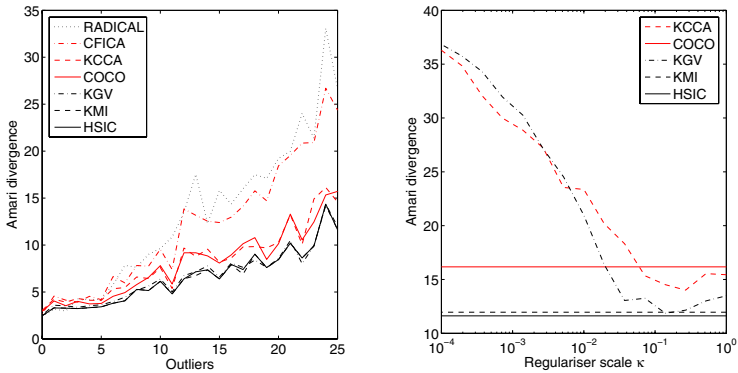
We note that HSIC with a Gaussian kernel performs on par with the best alternatives in the final four experiments, and that HSIC with a Laplace kernel gives joint best performance in six of the seven experiments. On the other hand, RADICAL and the KGV perform better than HSIC in the  $m = 250$  case. While the Laplace kernel clearly gives superior performance, this comes at an increased computational cost, since the eigenvalues of the associated Gram matrices decay more slowly than for the Gaussian kernel, necessitating the use of a higher rank in the incomplete Cholesky decomposition. Interestingly, the Laplace kernel can improve on the Gaussian kernel even with sub-Gaussian sources, as seen for instance in [10–Table 6.3] for the KMI and COCO.<sup>10</sup> This is because the slow decay of the eigenspectrum of the Laplace kernel improves the detection of dependence encoded at higher frequencies in the probability density function, which need not be related to the kurtosis — see [11–Section 4.2].

**Table 2.** Demixing of  $n$  randomly chosen i.i.d. samples of length  $m$ , where  $n$  varies from 2 to 16. The Gaussian kernel results are denoted  $g$ , and the Laplace kernel results  $l$ . The column *Rep.* gives the number of runs over which the average performance was measured. Note that some algorithm names are truncated: Fica is Fast ICA, IMAX is Infomax, RAD is RADICAL, CFIC is CFICA, CO is COCO, and HS is HSIC. Performance is measured using the Amari divergence (smaller is better).

n	m	Rep.	FICA	Jade	IMAX	RAD	CFIC	KCC	COg	COI	KGV	KMIg	KMIl	HSg	HSl
2	250	1000	10.5 ± 0.4	9.5 ± 0.4	44.4 ± 0.9	5.4 ± 0.2	7.2 ± 0.3	7.0 ± 0.3	7.8 ± 0.3	7.0 ± 0.3	5.3 ± 0.2	6.0 ± 0.2	5.7 ± 0.2	5.9 ± 0.2	5.8 ± 0.3
2	1000	1000	6.0 ± 0.3	5.1 ± 0.2	11.3 ± 0.6	2.4 ± 0.1	3.2 ± 0.1	3.3 ± 0.1	3.5 ± 0.1	2.9 ± 0.1	2.3 ± 0.1	2.6 ± 0.1	2.3 ± 0.1	2.6 ± 0.1	2.4 ± 0.1
4	1000	100	5.7 ± 0.4	5.6 ± 0.4	13.3 ± 1.1	2.5 ± 0.1	3.3 ± 0.2	4.5 ± 0.4	4.2 ± 0.3	4.6 ± 0.6	3.1 ± 0.6	4.0 ± 0.7	3.5 ± 0.7	2.7 ± 0.1	2.5 ± 0.2
4	4000	100	3.1 ± 0.2	2.3 ± 0.1	5.9 ± 0.7	1.3 ± 0.1	1.5 ± 0.1	2.4 ± 0.5	1.9 ± 0.1	1.6 ± 0.1	1.4 ± 0.1	1.4 ± 0.05	1.2 ± 0.05	1.3 ± 0.05	1.2 ± 0.05
8	2000	50	4.1 ± 0.2	3.6 ± 0.2	9.3 ± 0.9	1.8 ± 0.1	2.4 ± 0.1	4.8 ± 0.9	3.7 ± 0.9	5.2 ± 1.3	2.6 ± 0.3	2.1 ± 0.1	1.9 ± 0.1	1.9 ± 0.1	1.8 ± 0.1
8	4000	50	3.2 ± 0.2	2.7 ± 0.1	6.4 ± 0.9	1.3 ± 0.05	1.6 ± 0.1	2.1 ± 0.2	2.0 ± 0.1	1.9 ± 0.1	1.7 ± 0.2	1.4 ± 0.1	1.3 ± 0.05	1.4 ± 0.05	1.3 ± 0.05
16	5000	25	2.9 ± 0.1	3.1 ± 0.3	9.4 ± 1.1	1.2 ± 0.05	1.7 ± 0.1	3.7 ± 0.6	2.4 ± 0.1	2.6 ± 0.2	1.7 ± 0.1	1.5 ± 0.1	1.5 ± 0.1	1.3 ± 0.05	1.3 ± 0.05

In our next experiment, we investigated the effect of outlier noise added to the observations. We selected two generating distributions from Table 1, randomly and with replacement. After combining  $m = 1000$  samples from these distributions with a randomly generated matrix having condition number between 1 and 2, we generated a varying number of outliers by adding  $\pm 5$  (with equal probability) to  $s_j$  signals at random locations. All kernels used were Gaussian with size  $\sigma = 1$ ; Laplace kernels resulted in decreased performance for this noisy data. Results are shown in Figure 1. Note that we used  $\kappa = 0.11$  for the KGV and KCC in this plot, which is an order of magnitude above the level recommended in [3]: this resulted in an improvement in performance (broadly

<sup>10</sup> COCO is referred to in this table as KC.



**Fig. 1. Left:** Effect of outliers on the performance of the ICA algorithms. Each point represents an average Amari divergence over 100 independent experiments (smaller is better). The number of corrupted observations in *both* signals is given on the horizontal axis. **Right:** Performance of the KCC and KGV as a function of  $\kappa$  for two sources of size  $m = 1000$ , where 25 outliers were added to each source following the mixing procedure.

speaking, an increase in  $\kappa$  causes the KGV to approach the KMI, and the KCC to approach COCO [10]).<sup>11</sup>

An additional experiment was also carried out on the same data, to test the sensitivity of the KCC and KGV to the choice of the regularisation constant  $\kappa$ . We observe in Figure 1 that too small a  $\kappa$  can cause severe underperformance for the KCC and KGV. On the other hand,  $\kappa$  is required to be small for good performance at large sample sizes in Table 2. A major advantage of HSIC, COCO, and the KMI is that these do not require any additional tuning beyond the selection of a kernel.

In conclusion, we emphasise that ICA based on HSIC, despite using a more general dependence test than in specialised ICA algorithms, nonetheless gives joint best performance on all but the smallest sample size, and is much more robust to outliers. Comparing with other kernel algorithms (which are also based on general dependence criteria), HSIC is simpler to define, requires no regularisation or tuning beyond kernel selection, and has performance that meets or exceeds the best alternative on all data sets besides the  $m = 250$  case.

**Acknowledgements.** The authors would like to thank Kenji Fukumizu and Matthias Hein for helpful discussions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. National ICT Australia is funded through the Australian Government’s *Research Infrastructure* initiative, in part through the Australian Research Council.

<sup>11</sup> The results presented here for the KCC and KGV also improve on those in [16, 3] since they include a polishing step for the KCC and KGV, which was not carried out in these earlier studies.

## References

- [1] S. Achard, D.-T. Pham, and C. Jutten, *Quadratic dependence measure for nonlinear blind source separation*, 4th International Conference on ICA and BSS, 2003.
- [2] S.-I. Amari, A. Cichoki, and Yang H., *A new learning algorithm for blind signal separation*, Advances in Neural Information Processing Systems, vol. 8, MIT Press, 1996, pp. 757–763.
- [3] F. Bach and M. Jordan, *Kernel independent component analysis*, Journal of Machine Learning Research **3** (2002), 1–48.
- [4] C. R. Baker, *Joint measures and cross-covariance operators*, Transactions of the American Mathematical Society **186** (1973), 273–289.
- [5] A. Bell and T. Sejnowski, *An information-maximization approach to blind separation and blind deconvolution*, Neural Computation **7** (1995), no. 6, 1129–1159.
- [6] J.-F. Cardoso, *Blind signal separation: statistical principles*, Proceedings of the IEEE **90** (1998), no. 8, 2009–2026.
- [7] A. Chen and P. Bickel, *Consistent independent component analysis and prewhitening*, Tech. report, Berkeley, 2004.
- [8] L. Devroye, L. Györfi, and G. Lugosi, *A probabilistic theory of pattern recognition*, Applications of mathematics, vol. 31, Springer, New York, 1996.
- [9] K. Fukumizu, F. R. Bach, and M. I. Jordan, *Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces*, Journal of Machine Learning Research **5** (2004), 73–99.
- [10] A. Gretton, R. Herbrich, and A. Smola, *The kernel mutual information*, Tech. report, Cambridge University Engineering Department and Max Planck Institute for Biological Cybernetics, 2003.
- [11] A. Gretton, A. Smola, O. Bousquet, R. Herbrich, A. Belitski, M. Augath, Y. Murayama, J. Pauls, B. Schölkopf, and N. Logothetis, *Kernel constrained covariance for dependence measurement*, AISTATS, vol. 10, 2005.
- [12] M. Hein and O. Bousquet, *Kernels, associated structures, and generalizations*, Tech. Report 127, Max Planck Institute for Biological Cybernetics, 2004.
- [13] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association **58** (1963), 13–30.
- [14] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent component analysis*, John Wiley and Sons, New York, 2001.
- [15] S. E. Leurgans, R. A. Moyeed, and B. W. Silverman, *Canonical correlation analysis when the data are curves*, Journal of the Royal Statistical Society, Series B (Methodological) **55** (1993), no. 3, 725–740.
- [16] E. Miller and J. Fisher III, *ICA using spacings estimates of entropy*, JMLR **4** (2003), 1271–1295.
- [17] A. Rényi, *On measures of dependence*, Acta Math. Acad. Sci. Hungar. **10** (1959), 441–451.
- [18] A. Samarov and A. Tsybakov, *Nonparametric independent component analysis*, Bernoulli **10** (2004), 565–582.
- [19] I. Steinwart, *On the influence of the kernel on the consistency of support vector machines*, JMLR **2** (2001).
- [20] Y. Yamanishi, J.-P. Vert, and M. Kanehisa, *Heterogeneous data comparison and gene selection with kernel canonical correlation analysis*, Kernel Methods in Computational Biology (Cambridge, MA) (B. Schölkopf, K. Tsuda, and J.-P. Vert, eds.), MIT Press, 2004, pp. 209–229.
- [21] L. Zwald, O. Bousquet, and G. Blanchard, *Statistical properties of kernel principal component analysis*, Proceedings of the 17th Conference on Computational Learning Theory (COLT), 2004.

## A Proofs

### A.1 Proof of Lemma 1

We expand  $C_{xy}$  via (6) and using (3):

$$\begin{aligned} \|C_{xy}\|_{\text{HS}}^2 &= \langle \tilde{C}_{xy} - M_{xy}, \tilde{C}_{xy} - M_{xy} \rangle_{\text{HS}} \\ &= \mathbf{E}_{x,y,x',y'} [\langle \phi(x) \otimes \psi(y), \phi(x) \otimes \psi(y) \rangle_{\text{HS}}] \\ &\quad - 2\mathbf{E}_{x,y} [\langle \mu_x \otimes \mu_y, \phi(x) \otimes \psi(y) \rangle_{\text{HS}}] + \langle \mu_x \otimes \mu_y, \mu_x \otimes \mu_y \rangle_{\text{HS}} \end{aligned}$$

Substituting the definition of  $\mu_x$  and  $\mu_y$  and using that  $\langle \phi(x), \phi(x') \rangle_{\mathcal{F}} = k(x, x')$  (and likewise for  $l(y, y')$ ) proves the claim.

### A.2 Proof of Theorem 1

The idea underlying this proof is to expand  $\text{tr}HKKHL$  into terms depending on pairs, triples, and quadruples  $(i, j)$ ,  $(i, j, q)$  and  $(i, j, q, r)$  of non-repeated terms, for which we can apply uniform convergence bounds with U-statistics.

By definition of  $H$  we can write

$$\text{tr}KHLH = \underbrace{\text{tr}KL}_{(a)} - 2m^{-1} \underbrace{\mathbf{1}^\top K L \mathbf{1}}_{(b)} + m^{-2} \underbrace{\text{tr}K \text{tr}L}_{(c)}$$

where  $\mathbf{1}$  is the vector of all ones, since  $H = \mathbf{1} - m^{-1}\mathbf{1}\mathbf{1}^\top$  and since  $K, L$  are symmetric. We now expand each of the terms separately and take expectations with respect to  $Z$ .

For notational convenience we introduce the Pochhammer symbol  $(m)_n := \frac{m!}{(m-n)!}$ . One may check that  $\frac{(m)_n}{m^n} = 1 + O(m^{-1})$ . We also introduce the index set  $\mathbf{i}_r^m$ , which is the set of all  $r$ -tuples drawn without replacement from  $\{1, \dots, m\}$ .

We expand  $\mathbf{E}_Z[\text{tr}K L]$  into

$$\mathbf{E}_Z \left[ \sum_i K_{ii} L_{ii} + \sum_{(i,j) \in \mathbf{i}_2^m} K_{ij} L_{ji} \right] = O(m) + (m)_2 \mathbf{E}_{x,y,x',y'} [k(x, x')l(y, y')] \quad (10)$$

Normalising terms by  $\frac{1}{(m-1)^2}$  yields the first term in (8), since  $\frac{m(m-1)}{(m-1)^2} = 1 + O(m^{-1})$ .

We expand  $\mathbf{E}_Z[\mathbf{1}^\top K L \mathbf{1}]$  into

$$\begin{aligned} &\mathbf{E}_Z \left[ \sum_i K_{ii} L_{ii} + \sum_{(i,j) \in \mathbf{i}_2^m} (K_{ii} L_{ij} + K_{ij} K_{jj}) \right] + \mathbf{E}_Z \left[ \sum_{(i,j,r) \in \mathbf{i}_3^m} K_{ij} L_{jr} \right] \\ &= O(m^2) + (m)_3 \mathbf{E}_{x,y} [\mathbf{E}_{x'} [k(x, x')] \mathbf{E}_{y'} [l(y, y')]] \end{aligned}$$

Again, normalising terms by  $\frac{2}{m(m-1)^2}$  yields the second term in (8). As with (a) we used that  $\frac{m(m-1)(m-2)}{m(m-1)^2} = 1 + O(m^{-1})$ .

As before we expand  $\mathbf{E}_Z[\mathbf{tr}K\mathbf{tr}L]$  into terms containing varying numbers of identical indices. By the same argument we obtain

$$O(m^3) + \mathbf{E}_Z \left[ \sum_{(i,j,q,r) \in \mathbf{i}_4^m} K_{ij}L_{qr} \right] = O(m^3) + (m)_4 \mathbf{E}_{x,x'}[k(x,x')] \mathbf{E}_{y,y'}[l(y,y')]. \quad (11)$$

Normalisation by  $\frac{1}{m^2(m-1)^2}$  takes care of the last term in (8), which completes the proof.

### A.3 Proof of Theorem 3

As in the proof in Appendix A.2, we deal separately with each of the three terms in (8), omitting for clarity those terms that decay as  $O(m^{-1})$  or faster.<sup>12</sup> Denote by  $\mathbf{P}_Z$  the probability with respect to  $m$  independent copies  $(x_i, y_i)$  drawn from  $p_{xy}$ . Moreover, we split  $t$  into  $\alpha t + \beta t + (1 - \alpha - \beta)t$  where  $\alpha, \beta > 0$  and  $\alpha + \beta < 1$ . The probability of a positive deviation  $t$  has bound

$$\begin{aligned} & \mathbf{P}_Z \{ \text{HSIC}(Z, \mathcal{F}, \mathcal{G}) - \text{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) \geq t \} \\ & \leq \mathbf{P}_Z \left\{ \mathbf{E}_{x,y,x',y'}[k(x,x')l(y,y')] - \frac{1}{(m)_2} \sum_{\mathbf{i}_2^m} K_{i_1 i_2} L_{i_1 i_2} \geq \alpha t \right\} \\ & \quad + \mathbf{P}_Z \left\{ \mathbf{E}_{x,y}[\mathbf{E}_{x'}[k(x,x')]] \mathbf{E}_{y'}[l(y,y')] - \frac{1}{(m)_3} \sum_{\mathbf{i}_3^m} K_{i_1 i_2} L_{i_2 i_3} \geq \frac{\beta}{2} t \right\} \\ & \quad + \mathbf{P}_Z \left\{ \mathbf{E}_{x,x'}[k(x,x')] \mathbf{E}_{y,y'}[l(y,y')] - \frac{1}{(m)_4} \sum_{\mathbf{i}_4^m} K_{i_1 i_2} L_{i_3 i_4} \geq \frac{1 - \alpha - \beta}{t} \right\} \end{aligned}$$

Using the shorthand  $z := (x, y)$  we define the kernels of the U-statistics in the three expressions above as  $g(z_i, z_j) = K_{ij}L_{ij}$ ,  $g(z_i, z_j, z_r) = K_{ij}L_{jr}$  and  $g(z_i, z_j, z_q, z_r) = K_{ij}L_{qr}$ . Finally, employing Theorem 2 allows us to bound the three probabilities as

$$e^{-2mt^2 \frac{\alpha^2}{2}}, e^{-2mt^2 \frac{\beta^2}{3 \times 4}}, \text{ and } e^{-2mt^2 \frac{(1-\alpha-\beta)^2}{4}},$$

Setting the argument of all three exponentials equal yields  $\alpha^2 > 0.24$ : consequently, the positive deviation probability is bounded from above by  $3e^{-\alpha^2 mt^2}$ . The bound in Theorem 2 also holds for deviations in the opposite direction, thus the overall probability is bounded by doubling this quantity. Solving for  $t$  yields the desired result.

<sup>12</sup> These terms are either sample means or U-statistics scaled as  $m^{-1}$  or worse, and are thus guaranteed to converge at rate  $m^{-1/2}$  according to reasoning analogous to that employed below. Thus, we incorporate them in the  $C/m$  term.



#### A.4 Proof of Lemma 2

Computing  $A$  and  $B$  costs  $O(md_f^2)$  and  $O(md_g^2)$  time respectively. Next note that

$$\begin{aligned} \text{tr}H(AA^\top)H(BB^\top) &= \text{tr}(B^\top(HA))(B^\top(HA))^\top \\ &= \|(HA)^\top B\|_{\text{HS}}^2 \end{aligned}$$

Here computing  $(HA)$  costs  $O(md_f)$  time. The dominant term in the remainder is the matrix-matrix multiplication at  $O(md_f d_g)$  cost. Hence we use

$$\widetilde{\text{HSIC}}(Z; \mathcal{F}, \mathcal{G}) := (m-1)^{-2} \|(HA)^\top B\|_{\text{HS}}^2.$$

## B HSIC Derivation of Achard *et al.*

Achard *et al.* [1] motivate using HSIC to test independence by associating the empirical HSIC with a particular population quantity, which they claim is zero if and only if the random variables being tested are independent. We now examine their proof of this assertion. The derivation begins with [1–Lemma 2.1], which states the components  $x_i$  of the random vector  $\mathbf{x}$  are mutually independent if and only if

$$\mathbf{E}_{\mathbf{x}} \left[ \prod_{i=1}^n k(x_i - y_i) \right] = \prod_{i=1}^n [\mathbf{E}_{x_i} k(x_i - y_i)] \quad \forall y_1, \dots, y_n, \quad (12)$$

as long as the kernel  $k$  has Fourier transform everywhere non-zero (here  $y_i$  are real valued offset terms). Achard *et al.* claim that testing the above is equivalent to testing whether  $Q(\mathbf{x}) = 0$ , where

$$Q(\mathbf{x}) = \frac{1}{2} \int \left( \mathbf{E}_{\mathbf{x}} \left[ \prod_{i=1}^n k \left( \frac{x_i}{\sigma_i} - y_i \right) \right] - \prod_{i=1}^n \left[ \mathbf{E}_{x_i} k \left( \frac{x_i}{\sigma_i} - y_i \right) \right] \right)^2 dy_1 \dots dy_n, \quad (13)$$

for scale factors  $\sigma_i > 0$  (the empirical HSIC is then recovered by replacing the population expectations with their empirical counterparts, and some additional manipulations). However  $Q(\mathbf{x}) = 0$  tells us only that (12) holds almost surely, whereas a test of independence requires (12) to hold pointwise. In other words,  $Q(\mathbf{x}) = 0$  does not imply  $\mathbf{x}$  are mutually independent, even though mutual independence implies  $Q(\mathbf{x}) = 0$ .

# An Analysis of the Anti-learning Phenomenon for the Class Symmetric Polyhedron

Adam Kowalczyk<sup>1</sup> and Olivier Chapelle<sup>2</sup>

<sup>1</sup> National ICT Australia and RISE,  
The Australian National University, Canberra, Australia  
adam.kowalczyk@nicta.com.au

<sup>2</sup> Max Planck Institute for Biological Cybernetics,  
Tübingen, Germany  
olivier.chapelle@tuebingen.mpg.de

**Abstract.** This paper deals with an unusual phenomenon where most machine learning algorithms yield good performance on the training set but systematically *worse than random performance* on the test set. This has been observed so far for some natural data sets and demonstrated for some synthetic data sets when the classification rule is learned from a small set of training samples drawn from some high dimensional space.

The initial analysis presented in this paper shows that anti-learning is a property of data sets and is quite distinct from over-fitting of a training data. Moreover, the analysis leads to a specification of some machine learning procedures which can overcome anti-learning and generate machines able to classify training and test data consistently.

## 1 Introduction

The goal of a supervised learning system for binary classification is to classify instances of an independent test set as well as possible on the basis of a model learned from a labeled training set. Typically, the model has similar classification behavior on both the training and test sets, i.e., it classifies training and test instances with precision higher than the expected accuracy of the random classifier. Thus it has what we refer to as “*good performance*”. However, there are real life situations where better than random performance on the training set yields systematically worse than random performance on the off-training test set. One example is the Aryl Hydrocarbon Receptor classification task in KDD Cup 2002 [3, 9, 11]. These systems exhibit what we call “*anti-learning*”. As it has been discussed in [8], anti-learning can be observed in publicly available microarray data used for prediction of cancer outcomes, which can show both learning and anti-learning mode, depending on the features selected.

In this paper however, we focus on synthetic data which facilitates rigorous analysis. The aim is to demonstrate rigorously that anti-learning can occur, and can be primarily a feature of the data as it happens for many families of algorithms, universally across all setting of tunable parameters. In particular, we analyse a task of classification of binary labeled vertices of a class symmetric polyhedron embedded in a sphere (Section 2). The classification task seems to

be very easy: the data is linearly separable and any two labeled vertices can unravel all labels. However, this simplicity is very deceptive: we prove in Section 3 that none of the wide range of well establish algorithms such as perceptron, Support Vector Machines, generalised regression,  $\kappa$ -nearest neighbours can learn to classify consistently the data. In fact, given a proper subset of the domain to train, they can easily learn to classify it, but they always totally misclassify the remaining data. This effect is very different from poor generalization abilities where a classifier would perform close to random: here the predictions on the test set are not random, they are exactly the *opposite* of what they should be. In Section 3.2 we show that there exists kernel transformations which can actually change perfect anti-learning data into perfectly learnable data. Finally, Section 4 discusses the results.

## 2 Geometry of Class Symmetric Kernels

Let  $S$  be a set of labeled examples  $\{(\mathbf{x}_i, y_i)\}_{i \in \mathbb{S}} \subset X \times \{-1, +1\}$  indexed uniquely by the index set  $\mathbb{S}$ , with  $X \subset \mathbb{R}^N$ . We are interested in classification rules of the form  $\text{sign} \circ f : X \rightarrow \{-1, +1\}$ , where  $f = \mathcal{A}(T)$ .  $\mathcal{A}$  may also depend on some hyperparameters such as kernel  $k : X \times X \rightarrow \mathbb{R}$ , the regularization constant, etc.

### 2.1 Performance Measures

Assume we are given  $f : X \rightarrow \mathbb{R}$  and a non-void test subset

$$T = \{(\mathbf{x}_i, y_i)\}_{i \in \mathbb{T}} \subset S$$

indexed uniquely by  $\mathbb{T} \subset \mathbb{S}$  and containing samples from both labels.

**Accuracy.** We define the *accuracy* of the decision rule  $\mathbf{x} \mapsto \text{sign}(f(\mathbf{x}))$  as

$$\text{ACC}(f, T) = \frac{1}{2} \sum_{y=\pm 1} \mathbb{P}_T(y_j f(\mathbf{x}_j) > 0 \mid y_j = y)$$

Here  $\mathbb{P}_T$  denotes the frequency calculated for the subset  $T \subset S$ . Note this is the *test accuracy* performance measure, independent of prior distribution of data classes.

**Area Under ROC.** For  $f$  as above, we use the *Area Under the Receiver Operating Characteristic Curve*,  $\text{AROC}(f, T)$ <sup>1</sup>, the plot of true vs. false positive rate, as another performance measure. Following [1] we use the formula

$$\begin{aligned} \text{AROC}(f, T) &= \mathbb{P}_T(f(\mathbf{x}_i) < f(\mathbf{x}_j) \mid y_i = -1, y_j = 1) \\ &\quad + \frac{1}{2} \mathbb{P}_T(f(\mathbf{x}_i) = f(\mathbf{x}_j) \mid y_i \neq y_j) \end{aligned}$$

---

<sup>1</sup> Also known as *the area under the curve*,  $AUC$ ; it is essentially the well known order statistics  $U$ .

Note that the second term in the above formula takes care of ties, when instances from different labels are mapped to the same value.

The expected value of both  $\text{AROC}(f, T)$  and  $\text{ACC}(f, T)$  for the trivial, uniformly random predictor  $f$ , is 0.5. This is also the value for these metrics for the trivial constant classifier mapping all  $T$  to a constant value,  $\pm 1$ . Note the following fact:

$$\text{AROC}(f, T) = 1 \iff \exists C, \forall i \in \mathbb{T}, y_i(f(\mathbf{x}_i) - C) > 0; \tag{1}$$

$$\text{AROC}(f, T) = 0 \iff \exists C, \forall i \in \mathbb{T}, y_i(f(\mathbf{x}_i) - C) < 0. \tag{2}$$

There are at least two reasons why we use AROC in this paper.

1. AROC is a widely used measure of classifier performance in practical applications, especially biological and biomedical classification. As we have indicated in the introduction, this paper is step toward understanding anti-learning in biomedical classification problems, so explicit usage of AROC makes a direct link to such applications.
2. AROC is independent of an additive bias term while accuracy or error rate are critically dependent on a selection of such a term. For instance,  $\text{ACC}(f + b, T) = 0.5$  for any  $b \geq \max(f(T))$ , even if  $\text{ACC}(f, T) = 0$ . Typically, in such a case other intermediate values for  $\text{ACC}(f + b', T)$  could be also obtained for other values of the bias  $b'$ . However,  $\text{AROC}(f + b, T) = \text{const}$ , since AROC depends on the order in the set  $(f + b)(T) \subset \mathbb{R}$  and this is independent of the additive constant  $b$ . (Note that modulo a constant factor, AROC is a well known order statistic  $U$  [1].)  $\square$

### 2.2 Class Symmetric Matrices

Now we introduce the basic object for theoretical analysis in this paper. In order to simplify deliberations we consider synthetic datasets for which the entries in the Gram matrix depend only on the classes of the corresponding points.

**Definition 1.** A matrix  $[k_{ij}]_{i,j \in \mathbb{S}}$  is called class symmetric if  $r > 0, c_y \in \mathbb{R}, y \in \{0, \pm 1\}, i, j \in \mathbb{S}$

$$k_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} r^2, & i = j \\ r^2 c_{y_i}, & y_i = y_j, i \neq j \\ r^2 c_0, & y_i \neq y_j \end{cases} \tag{3}$$

where  $k_{ij}$  is class symmetric  $i, j \in \mathbb{S}$

Now we establish a necessary and sufficient condition on the coefficients of this matrix for it to be a positive definite kernel matrix.

**Lemma 1.** (i)  $D_+ := \frac{1-c_y}{n_y} + c_y, n_y := |\{i \in \mathbb{S}; y_i = y\}|, y = \pm 1$

$$D_+ D_- > c_0^2, D_y > 0, 1 - c_y > 0, y = \pm 1. \tag{4}$$

(ii)  $[k_{ij}]_{i,j \in \mathbb{S}}$  is class symmetric (3)

(iii)  $\mathbf{z}_i \in \mathbb{R}^{|\mathbb{S}|}, k_{ij} = \mathbf{z}_i \cdot \mathbf{z}_j, i, j \in \mathbb{S}$

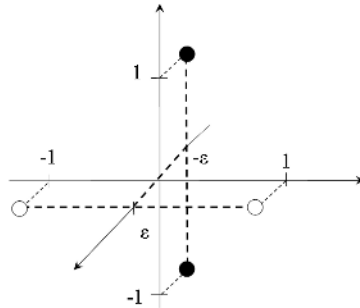
See Appendix for the proof.

The points  $\mathbf{z}_i$  as above belong to the sphere of radius  $r$  centered at the origin. They are vertices of  $\dots$ , of  $n = |S|$  vertices and  $n(n + 1)/2$  edges. The vertices of the same label  $y$  form an  $n_y$ -simplex, with all edges of constant length  $d_y := r\sqrt{2 - 2c_y}$ ,  $y = \pm 1$ . The distances between any pair of vertices of opposite labels are equal to  $d_0 := r\sqrt{2 - 2c_0}$ . Note that the linear independence in Lemma 1 insures that the different labels on CS-polyhedron are  $\dots$ .

It is interesting to have a geometrical picture of what happens in the case  $c_y < c_0$ ,  $y \pm 1$ . In that case, each point is nearer to all the points of the opposite class than to any point of the same class. In this kind of situation, the nearest  $\kappa$ -neighbors classifier would obviously lead to anti-learning. Thus we have:

**Proposition 1.**  $\dots \kappa > 1, \dots k, \dots S \dots T$   
 $\dots > \kappa/2, \dots c_y < c_0, \dots y \pm 1, \dots \kappa, \dots$   
 $\dots \rho(\mathbf{x}, \mathbf{x}') :=$   
 $\sqrt{2r^2 - 2k(\mathbf{x}, \mathbf{x}')}, \dots S, \text{ACC}(f_\kappa, S)$   
 $= \text{AROC}(f_\kappa, S) = 0$

An example of four point perfect anti-learning subset  $S \subset \mathbb{R}^3$  is given in Figure 1.



**Fig. 1.** Elevated XOR - an example of the perfect anti-learning data in 3-dimensions. The  $z$ -values are  $\pm\epsilon$ . The linear kernel satisfies the CS-condition (3) with  $r^2 = 1 + \epsilon^2$ ,  $c_0 = -\epsilon^2 r^{-2}$  and  $c_{-1} = c_{+1} = (-1 + \epsilon^2)r^{-2}$ . Hence the perfect anti-learning condition (6) holds if  $\epsilon < 0.5$ . It can be checked directly, that any linear classifier such as perceptron or maximal margin classifier, trained on a proper subset misclassify all the off-training points of the domain. This can be especially easily visualized for  $0 < \epsilon \ll 1$ .

The geometry of CS-polyhedron can be hidden in the data. An example which will be used in our simulations follows.

$\dots$  Hadamard matrices are special (square) orthogonal matrices of 1's and -1's. They have applications in combinatorics, signal

processing, numerical analysis. An  $n \times n$  Hadamard matrix,  $H_n$ , with  $n > 2$  exists only if  $n$  is divisible by 4. The Matlab function `hadamard(n)` handles only cases where  $n$ ,  $n/12$  or  $n/20$  is a power of 2. Hadamard matrices give rise to  $CS$ -polyhedrons  $S^o(H_n) = \{(\mathbf{x}_i, y_i)\}_{i=1, \dots, n} \subset \mathbb{R}^{n-1}$ . The recipe is as follows. Choose a non-constant row and use its entries as labels  $y_i$ . For data points,  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^{n-1}$ , use the columns of the remaining  $(n-1) \times n$  matrix. An example of  $4 \times 4$ -Hadamard matrix, the corresponding data for the 3-rd row used as labels and the kernel matrix follows :

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}; y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}; [\mathbf{x}_1, \dots, \mathbf{x}_4] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix};$$

$$[k_{ij}] = \begin{bmatrix} 3 & -1 & 1 & 1 \\ -1 & 3 & 1 & 1 \\ 1 & 1 & 3 & -1 \\ 1 & 1 & -1 & 3 \end{bmatrix}.$$

Since the columns of Hadamard matrix are orthogonal, from the above construction we obtain  $\mathbf{x}_i \cdot \mathbf{x}_j + y_i y_j = n \delta_{ij}$ . Hence the dot-product kernel  $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$  satisfies (3) with  $c_0 = -c_y = 1/(n-1)$  and  $r^2 = n-1$ .

Note that the vectors of this data set are linearly dependent, hence this set does not satisfy Lemma 1 (condition (iii) does not hold). It is instructive to check directly that the first inequality in (4) is violated as well. Indeed, in such case we have  $D_y = \frac{1}{n-1}$ , hence the equality  $D_+ D_- = c_0^2$  holds.

In order to comply strictly with Lemma 1, one of the vectors from  $S^o(H_n)$  has to be removed. After such removal we will have a set of  $n-1$  vectors in the  $n-1$  dimensional space which are linearly independent. This can be easily checked directly, but also we check equivalent condition (ii) of Lemma 1. Indeed, in such a case we obtain  $D_+ = \frac{n+2}{(n-1)(n-2)}$  and  $D_- = \frac{1}{n-1}$  assuming that the first, constant vector (with the positive label) has been removed. Hence,

$$D_+ D_- = \frac{n+2}{(n-1)^2(n+2)} > \frac{1}{(n-1)^2} = c_0^2$$

and all inequalities in (4) hold.

We shall denote this truncated Hadamard set by  $S(H_n)$ .

### 3 Perfect Learning/Anti-learning Theorem

**Standing Assumption:** In order to simplify our considerations, from now on we assume that  $\emptyset \neq T \subset S$  is a subset such that both  $T$  and  $S \setminus T$  contain examples from  $S$ .

Now we consider the class of kernel machines [4, 12, 13]. We say that the function  $f : X \rightarrow \mathbb{R}$  has  $T$  or write  $f \in \text{CONE}(k, T)$ , if there exists coefficients  $\alpha_i \geq 0, i \in \mathbb{T}$ , such that  $\alpha \neq 0$  and

$$f(\mathbf{x}) = \sum_{i \in \mathbb{T}} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) \quad \text{for every } \mathbf{x} \in X. \tag{5}$$

As for the  $\kappa$ -nearest neighbours algorithm (see Proposition 1) the learning mode for kernel machines with CS-kernel depends on the relative values of  $c_y$  and  $c_0$ . More precisely, the following theorem holds.

**Theorem 1.** *Let  $k$  be a CS-kernel on  $X$  and  $S$  a subset of  $X$ . Then, the perfect anti-learning*

$$c_y < c_0, \quad y = \pm 1, \tag{6}$$

$$\forall T \subset S, \forall f \in \text{CONE}(k, T), \text{AROC}(f, S \setminus T) = 0, \tag{7}$$

$$\forall T \subset S, \forall f \in \text{CONE}(k, T), \exists b \in \mathbb{R}, \text{ACC}(f + b, S \setminus T) = 0, \tag{8}$$

the perfect learning

$$c_y > c_0, \quad y = \pm 1, \tag{9}$$

$$\forall T \subset S, \forall f \in \text{CONE}(k, T), \text{AROC}(f, S \setminus T) = 1, \tag{10}$$

$$\forall T \subset S, \forall f \in \text{CONE}(k, T), \exists b \in \mathbb{R}, \text{ACC}(f + b, S \setminus T) = 1, \tag{11}$$

For  $f$  as in (5), (3) holding for the CS-kernel  $k$  and  $b := \sum_{i \in \mathbb{T}} \alpha_i y_i c_0$ , we have

$$\begin{aligned} y_j (f(\mathbf{x}_j) - b) &= y_j \sum_{i \in \mathbb{T}} \alpha_i y_i (k_{ij} - c_0) = \sum_{i \in \mathbb{T}, y_i = y_j} \alpha_i (c_{y_j} - c_0) - \sum_{i \in \mathbb{T}, y_i \neq y_j} \alpha_i (c_0 - c_0) \\ &= \begin{cases} < 0, & \text{if (6) holds;} \\ > 0, & \text{if (9) holds;} \end{cases} \end{aligned}$$

for  $j \in \mathbb{S} - \mathbb{T}$ . This proves immediately the equivalences (6)  $\Leftrightarrow$  (8) and (9)  $\Leftrightarrow$  (11), respectively. Application of (1) and (2) completes the proof.  $\square$

Note that for the Hadamard matrix example given above,  $c_y = -\frac{1}{n-1} < \frac{1}{n-1} = c_0$  and the perfect anti-learning condition holds. See also the caption of figure 1 for the XOR problem.

A number of popular supervised learning algorithms outputs solutions  $f(\mathbf{x}) + b$  where  $f \in \text{CONE}(k, T)$  and  $b \in \mathbb{R}$ . This includes the support vector machines (both with linear and non-linear kernels) [2, 4, 12], the classical, kernel or voting perceptron [5]. The class  $\text{CONE}(k, T)$  is convex, hence boosting of weak learners from  $\text{CONE}(k, T)$  produces also a classifier in this class. Others algorithms, such as regression or the generalized (ridge) regression [4, 12] applied to the CS-kernel  $k$  on  $T$ , necessarily output such a machine. Indeed the following proposition holds

**Proposition 2.** *Let  $k$  be a CS-kernel on  $X$  and  $S$  a subset of  $X$ . Then, the perfect learning*

$$R = \lambda \|\mathbf{w}\|^2 + \sum_{i \in \mathbb{T}} \xi_i^2, \quad \xi_i := 1 - y_i (\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b),$$

the perfect learning  $\mathbf{x} \mapsto \mathbf{w} \cdot \Phi(\mathbf{x}) \in \text{CONE}(k, T)$

It is sufficient to consider the linear case, i.e.  $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ . Due to the linear independence and class symmetry in vectors  $\mathbf{x}_i$ , the vector  $\mathbf{w}$  has the unique expansion

$$\lambda \mathbf{w} = \sum_i y_i \xi_i \mathbf{x}_i = \xi_+ \sum_{i, y_i=+1} \mathbf{x}_i - \xi_- \sum_{i, y_i=-1} \mathbf{x}_i.$$

Both slacks  $\xi_+$  and  $\xi_-$  are  $\geq 0$  at the minimum. Indeed, if one of them is  $< 0$ , we can “shrink”  $\|\mathbf{w}\|$ , i.e. the replacement  $\mathbf{w} \leftarrow a\mathbf{w}$ , where  $0 < a < 1$ , and adjust  $b$ , in a way which will decrease the magnitude of this slack leaving the other one unchanged. So  $R$  would decrease. This contradicts that  $(\mathbf{w}, b)$  minimizes  $R$ .  $\square$

### 3.1 Transforming the Kernel Matrix

Consider the case where the linear kernel  $k_{lin}(\mathbf{x}_i, \mathbf{x}_j) := \mathbf{x}_i \cdot \mathbf{x}_j$  applied to some data (such as the Hadamard matrix) yields a CS-kernel for which anti-learning occurs (i.e.  $c_y < c_0$ ,  $y \pm 1$  from the Theorem 1). A natural question arises: can we instead apply a non-linear kernel which would suppress anti-learning?

First note that several non-linear kernels can be expressed as a composition with the linear kernel  $k = \varphi \circ k_{lin}$ . For instance, the polynomial kernel of degree  $d$ ,  $k_d = (k_{lin} + b)^d$  or the Gaussian kernel

$$k_\sigma(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) = \exp\left(-\frac{\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2k_{lin}(\mathbf{x}_i, \mathbf{x}_j)}{\sigma^2}\right).$$

Indeed, if  $k$  is a class symmetric kernel satisfying (3) then  $k_\sigma = \varphi_1 \circ k$ , where  $\varphi_1 : \xi \in \mathbb{R} \mapsto \exp(-2(r^2 - \xi)/\sigma^2)$  is a monotonically increasing function. Similarly, for the odd degree  $d = 1, 3, \dots$  we have  $k_d = \varphi_2 \circ k$ , where  $\varphi_2 : \xi \in \mathbb{R} \mapsto (\xi + b)^d$ ,  $\xi \in \mathbb{R}$ , is a monotonically increasing function.

Note that when this composition function is a monotonically increasing one, the relative order of  $c_y$  and  $c_0$  in the new non-linear kernel matrix will be unchanged and anti-learning will persist.

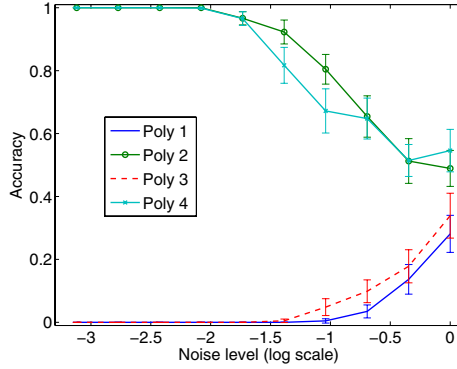
**Corollary 1.** Let  $k, k'$  be class symmetric kernels with  $S_{k'} = S_k$  and  $k' = \varphi \circ k$  where  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  is a monotonically increasing function. If  $(a, b) \subset \mathbb{R}$  is an interval containing  $c_{-1}, c_{+1}$  and  $c_0$  then  $c_y < c_0$  for  $y \in \{\pm 1\}$  in  $k$  implies  $c_y < c_0$  in  $k'$ .

In the next section, we introduce a non-monotonic modification of the kernel matrix which can overcome anti-learning.

### 3.2 From Anti-learning to Learning

Consider the special case of a CS-kernel with  $c_0 = 0$ , i.e. when examples from opposite labels are positioned on two mutually orthogonal hyperplanes. According to theorem 1, anti-learning will occur if  $c_y < 0$ , for  $y \in \{\pm 1\}$ . A way to reverse this behavior would be for instance to take the absolute values of the





**Fig. 2.** Switch from anti-learning to learning upon non-monotonic transformation of the kernel, Theorem 2. Plots show independent test set accuracy, average over 30 trials. For the experiments we have used Hadamard data set  $S(H_{128})$ , see Example 1 of Section 2.2, with the gaussian noise  $\mathcal{N}(0, \sigma)$  added independently to each data matrix entry. The data set has been split randomly into training and test sets (respectively two thirds and one third). We have used the hard margin SVM exclusively, so the training accuracy was always 1. We plot averages for 30 repeats of the experiments and also the standard deviation bars. We have used four different kernels, the linear kernel  $k_1 = k_{lin}$  and its three non-monotonic transformations,  $k_d := (k_{lin} - \hat{c}_0)^d$ ,  $d = 2, 3, 4$ , where  $\hat{c}_0 := \text{mean}_{y_i \neq y_j} \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$  was estimated from the training data.

kernel matrix such that the new  $c_y$  becomes positive. Then, according to Theorem 1 again, perfect learning could take place.

This is the main idea behind the following theorem which makes it possible to go from anti-learning to learning, for the CS-kernel case at least. Its main task here is to establish that the new kernel matrix is positive definite.

**Theorem 2.** Let  $k$  be a symmetric kernel matrix and  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  a function satisfying  $\varphi(0) = 0$

$$0 < \varphi(-\theta) \leq \varphi(\theta) \leq \varphi(\theta') \quad \text{for } 0 < \theta \leq \theta' \quad (12)$$

Then the kernel  $k_\varphi := \varphi(k - c_0)$  is a symmetric CS-kernel matrix for  $S$  with  $c_\varphi = 0$ . (9)

Note that this theorem shows an advantage of using kernels for formulating the results of this paper. Indeed, the claims of this section are straightforward to formulate and prove using CS-kernels but next to impossible directly.

It is easy to see that the kernel  $k_\varphi$  satisfies conditions (3) of CS-symmetry with coefficients  $r_\varphi^2 := \varphi(r^2(1 - c_0)) > 0$ ,  $c_{\varphi,0} = 0$  and  $c_{\varphi,y} := \varphi(r^2(c_y - c_0))/\varphi(r^2(1 - c_0)) > c_{\varphi,0} = 0$  for  $y = \pm 1$ . Now a straightforward algebra gives the relation

$$D_{\varphi,y} = \frac{1 - c_{\varphi,y}}{n_y} + c_{\varphi,y} \geq c_{\varphi,y} \left( 1 - \frac{1}{n_y} \right) > 0 = |c_{\varphi,0}|,$$

hence the first two conditions of (4) holds. The last condition of (4) is equivalent to  $\varphi(r^2(c_y - c_0)) < \varphi(r^2(1 - c_0))$ . When  $c_y - c_0 \geq 0$ , this is satisfied thanks to (12) and the last condition of (4). When  $c_y - c_0 \leq 0$ , we have  $2c_0 - c_y < c_y + 2\frac{1-c_y}{n_y} = c_y(1 - \frac{2}{n_y}) + \frac{2}{n_y} < 1$ , the first (resp. second) inequality coming from the first (resp. second) condition of (4). This gives  $-(c_y - c_0) < 1 - c_0$  and the desired result through (12).  $\square$

The examples of functions  $\varphi$  satisfying the above assumptions are  $\theta \mapsto |\theta|^d$ ,  $d = 1, 2, \dots$ , leading to a family of Laplace kernels  $|k - c_0|^d$  [13]; this family includes polynomial kernels  $(k - c_0)^d$  of even degree  $d$ . This is illustrated by simulation results in Figure 2.

## 4 Discussion

**CS-Polyhedrons in Real Life.** The CS-polyhedron is a very special geometrical object. Can it emerge in more natural settings? Surprisingly the answer is positive. For instance, [6] have shown that it emerges for a high dimensional dataset with low sample size. They studied a non-standard asymptotic, where the dimension tends to infinity while the sample size is fixed. Their analysis shows a tendency for the data to lie deterministically at the vertices of a CS-polyhedron. Essentially all the randomness in the data appears as a random rotation of this polyhedron. This CS-polyhedron is always of the perfect learning type, in terms of Theorem 1. All abnormalities of support vector machine the authors have observed, reduce to sub-optimality of the bias term selected by the maximal margin separator.

CS-polyhedron structure can be also observed in some biologically motivated models of growth under constraints of competition for limited resources [7, 8]. In this case the models can generate both perfectly learning and perfectly anti-learning data, depending on modeling assumptions.

**Relation to Real Life Data.** As mentioned already in the introduction, the example of the Aryl Hydrocarbon Receptor used in KDD'02 Cup competition is unusual: this dataset shows strong anti-learning while modeled by ordinary two-class SVM, but is “learnable” if non-standard one-class SVM is used, and this behavior changes abruptly if the continuous transition from one model to the other is used [9, 11]. In [7] this has been also shown to be the case for a dedicated CS-polyhedron model, which with an addition noise, reproduces closely also other results observed for the KDD'02 data.

**Anti-learning is not Over-Fitting.** In a complete over-fitting situation, a supervised learning algorithm generates a classifier performing on a level of random guessing. Such a classifier does not allow to discern any useful information for classification of independent data, yielding  $AROC \approx 0.5$ . This is not the case of anti-learning. In its perfect form (Theorem 1) we obtain a predictor  $f$  which perfectly misclassify the test data,  $AROC(f, S \setminus T) = 0$ , hence its negation,  $-f$ , allows to recover the ordering fully consistent with labels,  $AROC(-f, T - S) = 1$ .

**Noise Suppresses Anti-learning.** Furthermore, the anti-learning occurs for a data set, or more generally a kernel, with the special “symmetries”. Addition of random noise suppresses these symmetries and kills the anti-learning effect. This is vividly demonstrated in Figure 2, where in tune with the increased variance of noise in the data, the average accuracy on the test set for the linear SVM increases from  $\approx 0$  to the average level of random guessing,  $\approx 0.5$ . At the same time, for the transformed quadratic kernel SVM, for which learning occurs in line with the predictions of Theorem 2, the average accuracy decreases from  $\approx 1$  to the average level of random guessing,  $\approx 0.5$ .

**Relation to Some Other Research.** Anti-learning should not be mixed with No-Free-Lunch theorems promoted by Wolpert [15]. The “No-Free-Lunch” type results make statements about averages across all target concepts (this is the crux of all its incarnations), while anti-learning deals with a single, very special, target concepts at a time and makes statements on behavior of wide classes of algorithms. However, anti-learnable datasets are constructive examples of data where many standard supervised learning algorithms have high error rate on the test data, a phenomenon envisaged by the No-Free-Lunch Theorems.

There are some similarities of this paper with another recent one [14]. Both studies make use of Hadamard matrices and discuss problems with using classifiers within the span of the training set. However, the main differences are as follows. (i) The setting of [14] is more about feature selection since the target vector is still part of the data matrix and the goal is to identify it. (ii) Their results are about regression (square loss). (iii) Like in the case of No-Free-Lunch Theorem, claims in [14] are about averages across multiple learning target concepts and also across both, the combined the training and the test sets. However, in this paper we are in position to evaluate the performance on each of these data sets separately and for a single target concept at a time. (iv) In our study Hadamard matrix is only one example of  $CS$ -polyhedrons displaying anti-learning. This is clear from the main Theorem 1. Thus our results are applicable to a wider class of datasets.

**Deceptive Simplicity of Classification of  $CS$ -Polyhedrons.** In the perfect anti-learning setting two labeled samples are sufficient to classify perfectly all data. To be concrete, let us consider the perfect anti-learning  $CS$ -kernel  $k$  on  $(\mathbf{x}_i, y_i)_{i \in \mathcal{S}} \subset \mathbb{R}^N$ , see (6). Given two labeled examples, say  $i_o$  and  $j_o$ , the classification rule

$$\mathbf{x} \in \mathbb{R}^N \mapsto \begin{cases} y_{j_o}, & \text{if } k(\mathbf{x}_{i_o}, \mathbf{x}) = k(\mathbf{x}_{i_o}, \mathbf{x}_{j_o}); \\ -y_{j_o}, & \text{otherwise,} \end{cases}$$

perfectly allocates labels the whole space  $S$ . However, many well established “work-horses” of machine learning, such as SVM, ridge regression, perceptron, voting perceptron,  $k$ -nearest neighbours is provably unable to discover solutions (existing in their hypothesis spaces) doing this task, even when given all but two points for the training. Thus this is not the problem of “inadequate” hypothesis space or too poor representation. In fact the algorithms always find rules which

systematically miss-order any two data points of different labels not included in the training. And boosting these “weak learners” leads to the similar result.

All this can be interpreted as follows. These learning algorithms are unable to estimate the distribution of labels in the data space, since the samples is too small. However, they are capable of discerning hidden “patterns” in the data very efficiently. Thus they learn although in a different way than expected. How to harness this capability is another issue.

**Non-linear Classifiers.** Theorem 2 shows however, that at least in some situations, a non-linear transformation of the kernel can convert an anti-learning task into a learning task. In the future we will also present some ensemble based machine learning algorithms capable of achieving this goal.

**Relation to Machine Learning Theory.** Anti-learning is relevant to special cases of the classification, when inference is to be done from a very small training sample from a very high dimensional space. This occurs especially in bio-informatics [9, 11, 8, 7]. It is worth to note that this is the regime where ordinary machine theory and statistics is void. In particular, this paper does not contradict the VC theory [13] which states that the training error and the test error should be close to each other when the number of training samples is significantly larger than the VC dimension (i.e. capacity) of the learning system. In all the anti-learning examples we encountered here the number of learning examples is always not larger than the number of dimensions (i.e. VC-dimension).

**Learning Distributed Concepts.** An interesting observation on anti-learning has been made by J. Langford [10], where anti-learning is linked to learning concepts which are distributed, rather than concentrated.

**Final Summary.** This study is a primarily introduction to a phenomenon of anti-learning. We have concentrated on a very simple synthetic data set for which anti-learning can be demonstrated formally for a large classes of learning algorithms. This dataset class is so simple that it can be analyzed analytically, but it is also rich enough to demonstrate unexpected and novel behavior of many “standard” learning algorithms. But we must stress, that the motivation for this research comes from real life cancer genomic data sets (unpublished at this stage) which consistently display anti-learning behavior. Obviously, this is not exactly the perfect anti-learning, but rather a consistent performance below random guessing on an independent test set. So this paper is an initial step in an attempt to understand properties of some real datasets and ultimately to work out the practical ways to deal with such non-standard leaning problems. Its aim is also to build awareness and initial acceptance for this class of learning problems and to encourage other researchers to come forward with datasets which do display such “counter-intuitive” properties, rather than dismissing them as non-sense. (This last point reflects also our personal experience.)

It is too early to draw any definite conclusions in this paper, as to whether and how anti-learning data sets should be dealt with. Our Theorem 2 says how to deal with some classes of anti-learning data, i.e. the *CS*-kernels or, equivalently, of *CS*-polyhedrons. However, such transformations are ineffective for noisy real life

anti-learning data we are interested in. Thus alternative, more robust techniques to deal with this issue have still to be investigated. We would like to add that the standard approach to learning from a small size sample, namely aggressive feature selection, does not solve the problem, at some real life cases at least. In particular, this is demonstrated by experimental results reported in [8, 11].

**Future Research.** There is a number of directions this research can be extended to in future. We shall list some of our current preferences now.

1. Identify and research novel examples of anti-learning datasets, both synthetic and natural.
2. Develop techniques for consistent classification of anti-learning data.
3. Research techniques capable of seamless learning from both learnable and anti-learnable datasets.
4. Study the problem of regression (square loss) for the anti-learning datasets.
5. Research iid sampling models, in particular learning curves, for the anti-learning datasets.

## 5 Conclusions

Anti-learning does occur in some machine learning tasks when inference is done from very low sample sizes in high dimensional feature spaces. This warrants radical re-thinking of basic concepts of learnability and generalization which are currently totally biased towards the “learning mode” of discerning the knowledge from data. It also warrants further research into theoretical analysis and development of practical methods for dealing with anti-learning problems, since such do occur in important real life applications.

## Acknowledgements

Many thanks to Cheng Soon Ong, Alex Smola and Grant Baldwin for help in preparation of this paper and to Manfred Warmuth and S.V.N. Vishwanathan for clarifying discussions.

National ICT Australia is funded by the Australian Government’s Department of Communications, Information Technology and the Arts and the Australian Council through Baking Australia’s Ability and the ICT Center of Excellence program.

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors’ views.

## References

- [1] D. Bamber, *The area above the ordinal dominance graph and the area below the receiver operating characteristic graph*, *J. Math. Psych.* **12** (1975), 387 – 415.
- [2] C. Cortes and V. Vapnik, *Support vector networks*, *Machine Learning* **20** (1995), 273 – 297.

- [3] M. Craven, *The Genomics of a Signaling Pathway: A KDD Cup Challenge Task*, SIGKDD Explorations **4(2)** (2002).
- [4] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge University Press, Cambridge, 2000.
- [5] Y. Freund and R.E. Schapire, *Large margin classification using the perceptron algorithm*, Machine Learning **37** (1999), 277–296.
- [6] P. Hall, J. S. Marron, and A. Neeman, *Geometric representation of high dimension low sample size data*, preprint, to appear in the Journal of the Royal Statistical Society, Series B, 2005.
- [7] A. Kowalczyk, O. Chapelle, and G. Baldwin, *Analysis of the anti-learning phenomenon*, <http://users.rsise.anu.edu.au/~akowalczyk/antilearning/>, 2005.
- [8] A. Kowalczyk and C.S. Ong, *Anti-learning in binary classification*, <http://users.rsise.anu.edu.au/~akowalczyk/antilearning/>, 2005.
- [9] A. Kowalczyk and B. Raskutti, *One Class SVM for Yeast Regulation Prediction*, SIGKDD Explorations **4(2)** (2002).
- [10] J. Langford, 2005, <http://hunch.net/index.php?p=35>.
- [11] B. Raskutti and A. Kowalczyk, *Extreme re-balancing for SVMs: a case study*, SIGKDD Explorations **6(1)** (2004), 60–69.
- [12] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*, MIT Press, 2001.
- [13] V. Vapnik, *Statistical learning theory*, Wiley, New York, 1998.
- [14] M.K. Warmuth and S.V.N. Vishwanathan, *Leaving the Span*, COLT 2005, to appear.
- [15] D.H. Wolpert, *The supervised learning no-free-lunch theorems*, World Conference on Soft Computing 2001.

## Appendix: Proof of Lemma 1, Section 2

Equivalence (ii)  $\iff$  (iii) is a standard linear algebra result. The vectors  $\mathbf{z}_i$  can be found with a Cholesky decomposition of  $k$ . Note that they are linearly independent if and only if the Gram matrix  $k$  is non-singular.

We prove the crucial equivalence (i)  $\iff$  (ii) now. In order to simplify the notation, without loss of generality we may assume that  $r^2 = 1$ . Let us assume that indices are ordered in such a fashion that  $y_i = +1$  for  $1 \leq i \leq n_+$  and  $y_i = -1$  for  $n_+ < i \leq n_+ + n_-$ . The kernel matrix can be written as follows

$$k = [k_{ij}] = \begin{bmatrix} (1 - c_+) \mathbb{I}_{n_+} + c_+ & c_0 \\ c_0 & (1 - c_-) \mathbb{I}_{n_-} + c_- \end{bmatrix},$$

where  $\mathbb{I}_n$  is the  $(n \times n)$  identity matrix. First we observe that  $k$  being symmetric has  $n_+ + n_-$  linearly independent eigenvectors with real eigenvalues. Now observe that for any vector  $\mathbf{v} = (v_i) = ((\mathbf{v}_{+,i}), 0) \in \mathbb{R}^{n_+} \times \mathbb{R}^{n_-}$  such that  $\sum_{i=1}^{n_+} v_{+,i} = 0$  we have  $k\mathbf{v} = (1 - c_+)\mathbf{v}$ . Thus this is an eigenvector with eigenvalue  $\lambda = 1 - c_+$ . Obviously the subspace of such vectors has dimensionality  $(n_+ - 1)$ . Similarly we find  $(n_- - 1)$  dimensional subspace of vectors of the form  $(0, \mathbf{v}_-) \in \mathbb{R}^{n_+} \times \mathbb{R}^{n_-}$  with eigenvalues  $\lambda = 1 - c_-$ .

The remaining two linearly independent eigenvectors are of the form  $\mathbf{v} = (v_+, \dots, v_+, v_-, \dots, v_-) \in \mathbb{R}^{n_+} \times \mathbb{R}^{n_-}$ , where  $v_+, v_- \in \mathbb{R}$ . For such a vector the eigenvector equation  $kbv = \lambda \mathbf{v}$  reduces to two linear equations:

$$\begin{bmatrix} n_+ D_+ - \lambda & n_- c_0 \\ n_+ c_0 & n_- D_- - \lambda \end{bmatrix} \begin{bmatrix} v_+ \\ v_- \end{bmatrix} = \lambda \begin{bmatrix} v_+ \\ v_- \end{bmatrix}.$$

This  $2 \times 2$  matrix has positive eigenvalues if and only if its determinant and trace are positive or equivalently if  $D_+ D_- > c_0^2$  and  $D_y > 0$ ,  $y = \pm 1$ .  $\square$

# Learning Causal Structures Based on Markov Equivalence Class

Yang-Bo He<sup>1,2</sup>, Zhi Geng<sup>2</sup>, and Xun Liang<sup>1</sup>

<sup>1</sup> Institute of Computer Science and Technology,  
Peking University, Beijing 100871, China  
{heyangbo, liangxun}@icst.pku.edu.cn

<sup>2</sup> The Mathematic College of Peking University, Beijing 100871, China  
zgend@math.pku.edu.cn

**Abstract.** Because causal learning from observational data cannot avoid the inherent indistinguishability for causal structures that have the same Markov properties, this paper discusses causal structure learning within a Markov equivalence class. We present that the additional causal information about a given variable and its adjacent variables, such as knowledge from experts or data from randomization experiments, can refine the Markov equivalence class into some smaller constrained equivalent subclasses, and each of which can be represented by a chain graph. Those sequential characterizations of subclasses provide an approach for learning causal structures. According to the approach, an iterative partition of the equivalent class can be made with data from randomization experiments until the exact causal structure is identified.

**Keywords:** Bayesian networks; Causal structure; Directed acyclic graphs; Constrained essential graph; Randomization experiments.

## 1 Introduction

Bayesian network is a powerful tool to represent an interested system in many fields such as sociology, epidemiology, business and biology [2, 9]. The structure of Bayesian network can be used to represent both dependent relations and causal relations. For example, in bioinformatics, Friedman [3] used them to provide a concise representation of complex cellular networks by composing simpler submodels, and Jansen et al. [5] developed an approach using them to predict protein-protein interactions genome-wide in yeast. In those applications, directed acyclic graphs (DAG) are used to represent both dependent and causal structures [7, 8]. It's crucial to discover the structure of a DAG for understanding the interested system represented by it or for doing some uncertainty inference with it [6].

There are many methods of causal structure discovery, and the main methods are Bayesian methods [2, 4] and constraint-based methods [8, 9]. From observational studies, neither Bayesian methods nor constraint-based methods can avoid the inherent indistinguishability for DAGs that have the same Markov properties [2, 4]. When a DAG is used to depict a causal structure, its directed edges



denote causal relationships from causes to effects. To discover exact causal relationships among variables, we need to distinguish the DAGs in the same Markov equivalence class with additional information. Knowledge from domain experts and data generated by interventional experiments can provide this kind of information. Heckerman, Geiger and Chickering [4] discussed the problem of learning Bayesian networks with the combination of knowledge and statistical data. Based on this idea, Cooper and Yoo [2] presented a method of causal discovery from a mixture of experimental and observational data. Tian and Pearl [9, 10] proposed a method of discovering causal structures based on dynamic environment. All of those methods try to discover casual structure based on Bayesian methods by utilizing the additional information from domain experts or experimental data.

Randomization interventional experiment is often used to generate experimental data that can provide the additional information to distinguishing underlying causal structure from other structures in the same Markov equivalence class. Repeated experiments on a given variable can produce the data that contains the causal knowledge about the given variable and its adjacent variables. Data from this kind of experiments can identify the causal directions of the edges adjacent to the given variable. Because the constrain of directed acyclic graph, the causal directions of other edges can also be identified. Consequently, the original Markov equivalence class is refined into smaller subsets with additional knowledge. Does there exist a graph representation of those subsets? Does there exist a repeatable framework to identify the underlying causal structure from Markov equivalence class based on randomization experiments? All of those questions motivate us to discuss the properties of Markov equivalence class.

In this paper, we discuss a partition of a Markov equivalence class of directed acyclic graphs based on the causal knowledge of a given variable. This partition divides the DAGs in the same Markov equivalence class into several subclasses. We also present the characterizations of the partition. Those results provide an approach for learning structures of directed acyclic graphs with data from Markov Equivalence Class based on randomization experiments.

This paper is organized as follows. In Section 2, we introduce notation and definitions and then review two foundational results in literatures of Markov equivalence class. In Section 3, the theoretical results of our approach for partitioning a Markov equivalence class and the sequential properties of the iterative partition will be discussed. Based on those results, a framework of learning causal structure from Markov equivalence class will be presented. In Section 4, we discuss the influence of randomization experiment on learning causal structure. Then an approach with experimental data to discover causal structure is presented. We also give some examples to illuminate the approach. Conclusion is discussed in Section 5.

## 2 Markov Equivalence Class

Verma and Pearl [12] presented a criterion to judge whether two DAGs are Markov equivalent. This criterion provides a method for portioning DAGs into

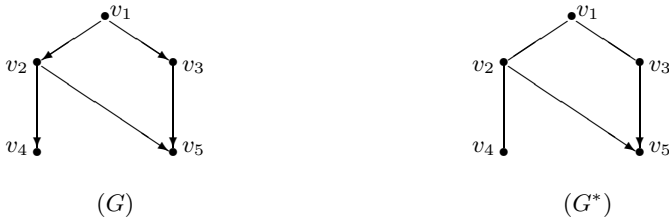
classes in each of which DAGs have the same Markov properties. Andersson, Madigan and Perlman [1] discussed characterization of Markov equivalence classes of DAGs and presented essential graphs to depict Markov equivalence classes. Below we introduce notations, definitions and related results.

A graph  $G$  is a pair  $(V, E)$ , where  $V$  is a set of variables and  $E$  is a set of edges. Let  $(v_1, v_2)$  denote a directed edge in  $E$ , depicted as an arrow  $v_1 \rightarrow v_2 \in E$  if  $(v_1, v_2) \in E$  and  $(v_2, v_1) \notin E$ . If both  $(v_1, v_2) \in E$  and  $(v_2, v_1) \in E$ , then  $v_1$  and  $v_2$  are connected with an undirected edge, depicted as a line. The graph  $G$  is *directed* if all edges of it are directed. The triple  $(v_1, v_2, v_3)$  is called an *immorality* of  $G$  if the subgraph  $G_{\{v_1, v_2, v_3\}}$  induced by  $(v_1, v_2, v_3)$  is  $v_1 \rightarrow v_2 \leftarrow v_3$ , where  $v_1$  and  $v_3$  are not connected by direct edge. The skeleton  $G^u$  of a graph  $G = (V, E)$  is defined as  $G^u = (V, E^u)$ , where  $E^u = \{(v_1, v_2) | (v_1, v_2) \in E \text{ or } (v_2, v_1) \in E\}$ . A *path*  $\pi$  of length  $n \geq 1$  from  $v$  to  $w$  in  $G$  is a sequence  $\pi = \{v_0, v_1, \dots, v_n\} \subseteq V$  of distinct variables such that  $v_0 = v$ ,  $v_n = w$  and either  $v_{i-1} \rightarrow v_i \in G$  or  $v_{i-1} - v_i \in G$  for  $i = 1, \dots, n$ . A *cycle* is a path of  $v_0 = v_n$ . A DAG is a directed graph that contains no directed cycle, and a chain graph is a graph that contains no directed cycle. A graph is *chordal* if each circle of length larger than or equal four has a chord. Let  $G_1 \sim G_2$  denote that DAGs  $G_1$  and  $G_2$  are Markov equivalent. The *essential graph*  $G^*$  is defined as

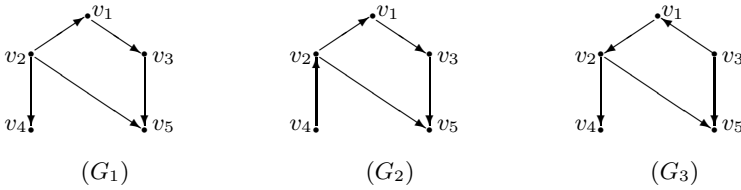
$$G^* = \cup\{G' | G' \sim G\}. \tag{1}$$

Here  $\cup$  is an operation with some equivalent DAGs as input. The output of this operation is a graph with the same skeleton as any input and its edge is directed if and only if it occurs as a directed edge with the same orientation in every DAGs of the input set. The *essential graph*  $G^*$  represents a set of DAGs that have the same Markov properties. Let  $pa_G(v) = \{w \in V | w \rightarrow v \in G\}$  denote the set of parents of  $v$ . We say that a variable  $v$  is a *root* in  $G$  if  $pa_G(v) = \emptyset$ . The set of edges adjacent to a variable  $v$  is denoted as  $E(v)$ , and  $e(v)$  is one of configuration of it.

Figure 1 shows two graphs, one is a directed acyclic graph  $G$  with five variables and the other is its essential graph  $G^*$  that represents all DAGs equivalent to  $G$ . For the graph  $G$  in Figure 1,  $V = (v_1, \dots, v_5)$  and  $E = \{(v_1, v_2), \dots, (v_3, v_5)\}$ . Because all edges are directed edges and there is no directed circle, the graph  $G$  is a directed acyclic graph. The triple  $(v_2, v_5, v_3)$ , with an induced subgraph  $v_2 \rightarrow v_5 \leftarrow v_3$  and without direct edge between  $v_2$  and  $v_3$ , is an immorality of  $G$ . The graph  $G^*$  in Figure 1 has the same variables and skeleton with  $G$ . There are three undirected edges, so  $G^*$  is not a directed graph. Because there is no directed circle,  $G^*$  is a chain graph. An important concept in chain graph is the *chain component*, it is the biggest variable set that can induce an undirected subgraph. The variable set  $\tau = (v_1, v_2, v_3, v_4)$  is a chain component of  $G^*$  and  $v_5$  is a chain component with one variable. According to the properties of chain graphs, there is no directed edge within a given chain component and no undirected edge between any two chain components. In Figure 1,  $G^*$  has two chain components  $\tau$  and  $v_5$ , and there is no directed edge in  $G_\tau^*$  and all edges between  $\tau$  and  $v_5$  are directed.



**Fig. 1.** Directed acyclic graph  $G$  with five variables and its essential graph  $G^*$ . The essential graph  $G^*$  has two essential edges, and three undirected edges.



**Fig. 2.** Three DAGs,  $G_1, G_2, G_3$  that are equivalent to the DAG  $G$  in Fig 1, Those four DAGs have the same skeleton and the same immorality

A well-known graph-theoretic criterion for Markov equivalence of DAG is presented by Verma and Pearl [12] as follows.

**Lemma 1.** *Let  $G_1$  and  $G_2$  be DAGs with the same skeleton and the same immorality. Then  $G_1$  and  $G_2$  are Markov equivalent.*

If  $G_1$  and  $G_2$  have the same skeleton and the same immoralities, they are Markov equivalence, denoted as  $G_1 \sim G_2$ . The equivalence class containing  $G$  is denoted by  $[G]$ . This lemma provides a practical criterion for deciding whether two DAGs are Markov equivalent. Figure 2 lists all DAGs that are equivalent to  $G$  in Figure 1. In Figure 2, three DAGs,  $G_1, G_2$  and  $G_3$  have the same edges as  $G$  regardless of their directions, that is, they have the same skeleton as  $G$ . Moreover, there is the same immorality,  $v_2 \rightarrow v_5 \leftarrow v_3$ , in each DAG, so we can get that  $\{G, G_1, G_2, G_3\}$  form a Markov equivalence class  $[G]$  represented by  $G^*$ .

Lemma 1 presents the relationships among DAGs in a Markov equivalence class. However, it does not concern the properties of an equivalence class as a whole. Andersson et al. [1] introduced an essential graph to represent a Markov equivalence class and showed the properties of essential graphs.

**Lemma 2.** *Let  $G = (V, E)$  be a DAG and  $G^*$  be its essential graph. Let  $G_1, \dots, G_r$  be DAGs such that  $G_1 \sim \dots \sim G_r \sim G$ . Then  $G^*$  is the unique graph such that  $G^* \sim G_i$  for all  $i = 1, \dots, r$ . Moreover,  $G^*$  is the unique graph such that  $G^* \sim G$  and  $G^*$  has the same skeleton as  $G$ .*

From this lemma, we can get that each Markov equivalence class can be represented by a unique chain graph defined by (1). So the chain graph  $G^*$  is

the essential graph and can represent uniquely the Markov equivalence class  $\{G, G_1, G_2, G_3\}$  showed in Figures 1 and 2.

### 3 The Framework of Learning Causal Structure Based on Markov Equivalence Class

In this section, we will present the framework of learning causal structure based on a Markov equivalence class using additional causal information of a given variable. In order to do so, some theoretical results about the partition of Markov equivalence class will be given. We can get that, based on the additional causal information, the original Markov equivalence class can be partitioned into smaller constrained equivalence subclasses iteratively until each subclass contains only one causal structure.

Let  $E(v)_G$  be the edges adjacent to  $v$  in graph  $G$ , and a constrained subclass of Markov equivalence class  $[G]$  is defined as

$$C_{e(v)} = \{G' | G' \in [G], E(v)_{G'} = e(v)\}. \tag{2}$$

Thus  $C_v$  is a set that consists of DAGs with  $E(v) = e(v)$  in the Markov equivalence class  $[G]$ . A Markov equivalence class can be divided into several subclasses by a given  $e(v)$ . If  $v$  is not in any chain component of an essential graph  $G^*$ , the directions of edges in  $E(v)$  of  $G^*$  are determined, and thus we have  $C_{e(v)} = [G]$ . But if  $v$  is in some chain component of  $G^*$ , we can partition the Markov equivalence class into several subclasses because there are undirected edges adjacent to  $v$  in a chain graph, that is, the configuration of  $E(v)$  is not determined.

We use a graph to represent each subclass. Defining  $e(v)$  as following,

$$G_{e(v)}^* = \cup\{G' | G' \in [G], E(v)_{G'} = e(v)\}. \tag{3}$$

Andersson et al. [1] showed that the essential graph  $G^*$  satisfies the three conditions mentioned in Lemma 2. Now we show that the  $e(v)$  also satisfies those conditions.

**Theorem 1.**  $G_{e(v)}^*$  satisfies the three conditions of Lemma 2. For any  $v \in \tau$ ,  $\tau \in \mathcal{T}$ ,  $H$  is a graph with  $G_{e(v)}^*$  as skeleton and  $H$  is a graph with  $G_{e(v)}^*$  as skeleton.

$$H \text{ is a graph with } G_{e(v)}^* \text{ as skeleton and } H \text{ is a graph with } G_{e(v)}^* \text{ as skeleton. } v_1 \rightarrow v_2 - v_3 \text{ is a path in } H$$

From the definition of  $G_{e(v)}^*$ , we have that  $G_{e(v)}^*$  has the same skeleton as the essential graph  $G^*$  and contains all directed edges of  $G^*$ . That is, all directed edges in  $G^*$  is also directed in  $G_{e(v)}^*$ . So the second condition of Theorem 1 holds obviously. The third condition of Theorem 1 also holds because all DAGs represented by  $G_{e(v)}^*$  are Markov equivalent.

In order to prove that  $G_{e(v)}^*$  is also a chain graph, we first introduce an algorithm to construct a graph, in which some undirected edges of the initial essential graph are oriented according to  $e(v)$ .

**Algorithm 1.**  $G^* = (V, E)$  is a chain graph,  $v \in \tau$ ,  $\tau$  is a chain component of  $G^*$ ,  $e(v)$  is a directed edge in  $G^*$ .

- 1)  $H = (V, E)$  is a graph,  $v \in \tau$ ,  $\tau$  is a chain component of  $G^*$ ,  $e(v)$  is a directed edge in  $H$ .
- 2)  $H = (V, E)$  is a graph,  $v \in \tau$ ,  $\tau$  is a chain component of  $G^*$ ,  $e(v)$  is a directed edge in  $H$ .
  - $v_1 \rightarrow v_2 - v_3 \in H$ ,  $v_1 - v_3 \in H$ ,  $v_1 \rightarrow v_3 \in H$
  - $v_2 - v_3 \in H$ ,  $v_2 \rightarrow v_3 \in H$
  - $v_1 \rightarrow v_2 \rightarrow v_3 \in H$ ,  $v_1 - v_3 \in H$ ,  $v_1 - v_3 \in H$ ,  $v_1 \rightarrow v_3 \in H$

It can be shown that the graph  $H$  constructed by Algorithm 1 is a chain graph and  $H$  is equal to the constrained essential graph  $G_{e(v)}^*$ . We show those results as following two Lemma.

**Lemma 3.**  $H$  is a chain graph.

If  $H$  is not a chain graph, there must be a directed circle in subgraph  $H_\tau$  for some chain component  $\tau$  of  $G^*$ . Moreover,  $G_\tau^*$  be chordal and  $H \subset G^*$ , so  $H_\tau$  is chordal too. we can educe a three-edge directed circle in  $H_\tau$  as Figure 3 or Figure 4.

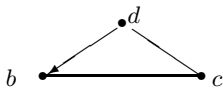


Fig. 3.  $SG_6$

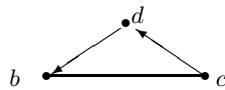


Fig. 4.  $SG_{61}$

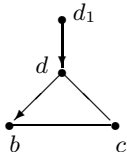


Fig. 5.  $SG_7$

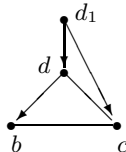


Fig. 6.  $SG_8$

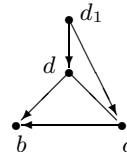


Fig. 7.  $SG_9$

If Figure 4 is a subgraph of  $H$  at some stage of the algorithm, the undirected edge  $b-c$  will be oriented to  $b \leftarrow c$  according to the Algorithm 1. So, only Figure 3 could be a subgraph of  $H$ .

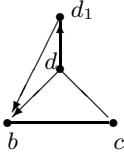


Fig. 8.  $SG_{10}$

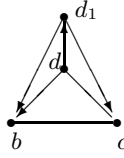


Fig. 9.  $SG_{11}$

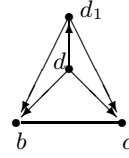


Fig. 10.  $SG_{12}$

Because all edges adjacent to  $a$  have been oriented in the first step of Algorithm 1 and the directed edge  $d \rightarrow b$  be not in  $G^*$ ,  $d \rightarrow b$  also not a edge adjacent to  $a$  according to the Algorithm 1. So  $d \rightarrow b$  must be identified by the second step of Algorithm 1. There are two situation, one is in order to avoiding immorality as Figure 5, the other is in order to avoiding directed circle as Figure 8.

We can get an order of all directed edges in  $H_\tau$  in which the preceding directed edges be oriented before the posterior directed edges according to the Algorithm 1. Firstly, we prove that the directed edge  $d \rightarrow b$  in Figure 3 is not the first directed edge oriented in the second step of Algorithm 1.

In the first case as Figure 5, if  $d \rightarrow b$  is the first directed edge oriented in the second step of Algorithm 1, we have  $d_1 = a$ . Because  $b$  and  $a$  are not adjacent, and  $d-c$  be a undirected edge in  $H$ , so  $d_1 \rightarrow c$  must be in  $H$  as Figure 6, where  $d_1 = a$ . Now we consider the subgraph  $b-c \leftarrow d_1$ , according to the rule in Algorithm 1,  $b \leftarrow c$  be in  $G_{e(a)}^*$  as Figure 7, it's a contradiction to the assumption that  $b-c \in H$ .

In the second case as Figure 8, if  $d \rightarrow b$  is the first directed edge oriented in the second step of Algorithm 1, we also have  $d_1 = a$ .

Considering the structure  $d_1 \rightarrow b-c$ , and  $d-c$  be a undirected edge in  $H$ , we have  $d_1 \rightarrow c$  must be in  $H$  as Figure 9. Now we consider the subgraph of  $\{d, d_1, c\}$  and, according to the rule in Algorithm 1,  $d \rightarrow c$  be in  $H$  as Figure 10, it's a contradiction to the assumption that  $d-c \in H$ .

So, we have the first directed edge oriented in the second step of Algorithm 1 be not in a directed circle. Supposing that the first  $k$  directed edges oriented in the second step of Algorithm 1 be not in any directed circle, we intend to prove that the  $(k + 1)th$  directed edge also be not in a directed circle.

Let  $d \rightarrow b$  be the  $(k + 1)th$  directed edge oriented in the second step of Algorithm 1, Suppose Figure 3 be a subgraph of  $H$ . There also have only two situations as Figure 5 and Figure 8 for orienting  $d \rightarrow b$ .

In situation as Figure 5, Because  $d_1 \rightarrow d$  be in the first  $k$  directed edges and  $d-c \in H$ , so  $d_1 \rightarrow c$  must be in  $H$ . We also get that  $b \leftarrow c$  must be in  $H$  as Figure 7. It's a contradiction to the assumption that  $b-c \in H$ .

In situation as Figure 5, Because  $d_1 \rightarrow b$  and  $d \rightarrow d_1$  be in the first  $k$  directed edges and  $b-c \in H$ , so  $d_1 \rightarrow c$  must be in  $H$ . We also get that  $d \leftarrow c$  must be in  $H$  as Figure 10. It's a contradiction to the assumption that  $d-c \in H$ .

So, the  $(k + 1)th$  directed edge also be not in any directed circle. Now, we can get that any directed edges in  $H_\tau$  be not in directed circle. It imply that there be no directed circles in  $H_\tau$ . So  $H$  be a chain graph.  $\square$

**Lemma 4.**  $G_{e(v)}^* \subseteq H \subseteq G_{e(v)}^* \cup e(v) \subseteq G_{e(v)}^* \cup e(v) \subseteq H \subseteq G_{e(v)}^* \cup e(v) \subseteq G_{e(v)}^* \cup e(v) \subseteq H$   
 $G_{e(v)}^* \cup e(v) = H$

We first prove  $G_{e(a)}^* \subseteq H$ . We just need to prove that all directed edges in  $H$  must be in  $G_{e(a)}^*$  too. We use induction to finish the proof.

Obviously, after the first step of Algorithm 1, all directed edges in  $H$  be in  $G_{e(a)}^*$ . We now prove that the first directed edge identified by the second step of Algorithm 1, such as  $b \leftarrow c$ , be in  $G_{e(a)}^*$ .  $b \leftarrow c$  must be identified by the first rule of the second step of Algorithm 1. There must be a variable  $d \notin \tau$  such that  $b \leftarrow c \leftarrow d$  be the subgraph of  $H$  in the same step. Because all directed edges in  $H$  be in  $G_{e(a)}^*$  in the same step,  $b \leftarrow c \leftarrow d$  must be the same subgraph in any  $G' \in G_{e(a)}^*$ . Otherwise  $b \rightarrow c \leftarrow d$  form a v-structure such that  $G' \notin [G]$ . So  $b \leftarrow c \in G_{e(a)}^*$ .

Suppose that the first  $k$  directed edges identified by the second step of Algorithm 1 be in  $G_{e(a)}^*$ , we now prove that the  $k + 1$  directed edge identified by the second step of Algorithm 1 also be in  $G_{e(a)}^*$ . Denoting the  $k + 1$  directed edge as  $c \leftarrow d$ , according to the rules in the Algorithm 1, there are only two situations that can identify the  $c \leftarrow d$  as following:

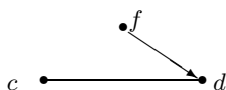


Fig. 11.  $SG_4$

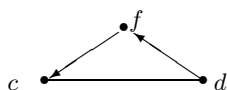


Fig. 12.  $SG_5$

In the situation as Figure 11, because  $f \rightarrow d$  be in every DAG  $G' \in G_{e(a)}^*$ , In order to voiding new v-structure,  $c \leftarrow d$  must be in every DAG  $G' \in G_{e(a)}^*$  too. So  $c \leftarrow d \in G_{e(a)}^*$ . In Figure 12, because  $d \rightarrow f$  and  $f \rightarrow c$  be in every DAG  $G' \in G_{e(a)}^*$ , In order to voiding directed circle,  $c \leftarrow d$  must be in every DAG  $G' \in G_{e(a)}^*$  too. So  $c \leftarrow d \in G_{e(a)}^*$ . Now we get that the  $k + 1$  directed edge identified by the second step of Algorithm 1 also be in  $G_{e(a)}^*$ . And all directed edges in  $H$  also be in  $G_{e(a)}^*$ . So  $G_{e(a)}^* \subseteq H$ .

Because  $H$  is a chain graph form Lemma 3, and we also have  $H \subseteq G^*$ . From the Lemma 2, for any undirect edge  $a-b$  of  $H_\tau$ , where  $\tau$  is a chain component of  $H$ , there exists  $G_1, G_2 \in G_{e(a)\tau}^*$  such that  $a \rightarrow b$  occurs in  $G_1$  and  $a \leftarrow b$  occurs in  $G_2$ . It means that  $a-b$  also occurs in  $G_{e(a)}^*$ . So  $H \subseteq G_{e(a)}^*$ , we have  $G_{e(a)}^* = H$ .  $\square$

Now, we give the proof of the Theorem 1 as follows.

From the definition of  $G_{e(v)}^*$ , we have that  $G_{e(v)}^*$  has the same skeleton as the essential graph  $G^*$  and contains all directed edges of  $G^*$ . That is, all directed edges in  $G^*$  is also directed in  $G_{e(v)}^*$ . So the second condition of Theorem 1 holds obviously. The third condition of Theorem 1 also holds because all DAGs represented by  $G_{e(v)}^*$  are Markov equivalent. From Lemmas 3 and 4, we can easily get that  $G_{e(v)}^*$  is a chain graph. So, the three conditions all hold.  $\square$





interventions are described as specific modifications of some factors in the products of (4).

Let  $V_i$  be a variable of DAG  $G = (V, E)$ . If we intervene  $V_i$  with distribution  $P'(V_i)$ , then the post-interventional joint distribution  $P_{V_i}(V)$  should be

$$P_{V_i}(v_1, v_2, \dots, v_n) = P'(v_i) \prod_{\{j|j \neq i\}} P(v_j|pa(v_j)). \tag{5}$$

Given a causal graph, the Markov condition determines a set of independence relations, which are encoded in the factorization of joint distribution. In order to discover causal structure, a further axiom other than Markov condition should be introduced, Spirtes [8] called it faithfulness condition and defined it as follow:

**Definition 1. Faithfulness Condition**

Let  $G$  be a causal graph and  $P$  a joint distribution over  $V$ . We say that  $P$  is faithful to  $G$  if and only if all the independence relations in  $P$  are entailed by the Markov condition applied to  $G$ .

If a distribution  $P$  satisfies both the Markov condition and the faithfulness condition according to a causal graph  $G$ , we have that all and only the independence relations of  $P$  are entailed by the Markov condition applied to  $G$ .

Let  $pa(V_i)$  and  $child(V_i)$  represent the parents and children of  $V_i$  respectively. From the factorization of (5), we can easily get lemma 5 to describe the independent relations of  $X_i$  in the post-interventional distribution.

**Lemma 5.** Let  $P$  be a joint distribution over  $V$  that is faithful to  $G$ . Then, for any variable  $X_i$  in  $V$ , the following two conditions are equivalent:  $(X_i \perp\!\!\!\perp pa(X_i))_{P_{X_i}(X)}$  and  $(X_i \perp\!\!\!\perp child(X_i))_{P_{X_i}(X)}$ .

**Theorem 3.** Let  $G'$  be a causal graph and  $\tau$  a set of variables. For any variable  $b \in \tau$ , the following two conditions are equivalent:  $(b \perp\!\!\!\perp c)_{P_b(X)}$  and  $(b \perp\!\!\!\perp c)_{P_b(X)}$ .

From lemma 5, for each neighbor  $c$  of  $b$ , we orient the  $c \dashv b$  to  $c \rightarrow b$  if  $(b \perp\!\!\!\perp c)_{P_b(X)}$  and to  $c \leftarrow b$  if  $(b \perp\!\!\!\perp c)_{P_b(X)}$ . So, the correction of Theorem 3 is obvious.

Now, we can give the procedure of learning causal structure based on Markov equivalence class using data from randomization interventional experiments. The main steps of this procedure are shown in following table 1.

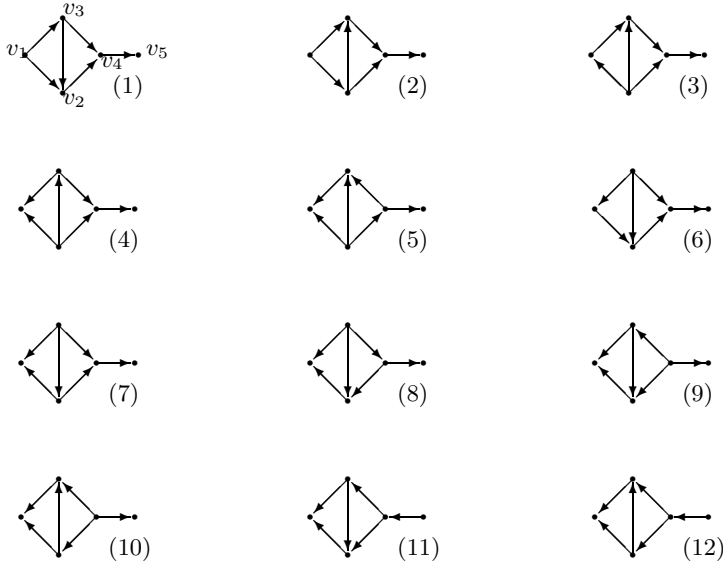
This procedure provides an approach to discover the exact causal structure from a given Markov equivalence class. We obtain the additional causal information from randomization experiments. In each iterative step, we choose one variable to do experiments on. There may be exist an optimization criterion to choose the sequence of variable to do experiments on such that we can discover exact structure with the minimum times of experiments. However, we do not intend to discuss this problem in this paper.

**Table 1.** The procedure of learning causal structure based on Markov equivalence class using data from randomization interventional experiments

---

Input: The essential graph  $G^*$ . Let  $H = G^*$   
 Output: The directed acyclic graph  $G$   
 Step 1. Choose a variable  $v$  that is in a chain component of  $H$ ;  
 Step 2. Randomization interventional experiments on  $v$ , get data set  $\mathcal{D}$ ;  
 Step 3. Test the dependent relations of the variable  $v$  and its adjacent variables denoting as  $adj(v)$ ;  
 Step 4. For any variable  $w \in adj(v)$ , if  $v \perp\!\!\!\perp w$  according to the dependent test with data  $\mathcal{D}$ , then orient  $v-w$  to  $w \rightarrow v$ , otherwise  $w \leftarrow v$ ;  
 Step 5. Carry out Algorithm 1 to identify other undirected edges in  $H$ ;  
 Step 6. Get a new chain graph  $H$ ;  
 Step 7. Return to step 1 if  $H$  is not directed acyclic graph;  
 Step 8. Let  $G = H$  and output  $G$ .

---

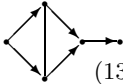
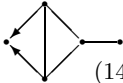


**Fig. 13.** All DAGs are equivalent to directed acyclic graph (1), There are 12 directed acyclic graphs that have the same skeleton and immortals as DAG (1), all unlabelled graph have the same labels as DAG (1)

Now, we give some examples to illuminate the results of this paper. Figure 13 shows 12 DAGs from the same Markov equivalence class. The variables are labelled only in graph (1) for concision. Other graphs have the same labels. Table 2 shows the four subclasses of the Markov equivalence class according to the directions of edges adjacent to  $v_1$ . They include graphs  $\{1,2\}, \{3\}, \{4,5,7,8,9,10,11,12\}$ ,

and  $\{6\}$  respectively. The second subset  $ii$  and the fourth subset  $iv$  have only one DAG. In the same subclass, the adjacent edges of  $v_1$  have the same directions for all DAGs. The constrained essential graphs (13) and (14) in table 2 represent two subclasses with  $E(v_1)$  equal to  $v_2 \leftarrow v_1 \rightarrow v_3$  and  $v_2 \rightarrow v_1 \leftarrow v_3$  respectively. We can find that the graphs (13) and (14) are chain graphs and also satisfy the second and third conditions in Theorem 2. A sequential partition according to sequential constraints  $E(v_1), E(v_2), E(v_3)$ , and  $E(v_4)$  is shown as Table 3.

**Table 2.** The subclasses and constrained essential graph given  $E(v_1)$ . There are four configurations of  $E(v_1)$ , each of them identify a constrained essential graph.

No of subset	$E(v_1)$	DAGs in Subclasses	<i>constrained essential graphs</i>
(i)	$v_2 \leftarrow v_1 \rightarrow v_3$	(1, 2)	 (13)
(ii)	$v_2 \rightarrow v_1 \rightarrow v_3$	(3)	
(iii)	$v_2 \rightarrow v_1 \leftarrow v_3$	(4, 5, 7 - 12)	 (14)
(iv)	$v_2 \leftarrow v_1 \leftarrow v_3$	(6)	

If the underlying causal structure is one of directed acyclic graphs in Figure 13, we can get that the constrained essential graphs in Table 2 after randomization interventional experiments on  $v_1$ . There are four possible independent relations among  $v_1, v_2$ , and  $v_3$  as shown in Table 4. Obviously, if the underlying causal structure is DAG (3) or DAG (6) in Figure 13, we can learn the exact structure only based on randomization experiments on variable  $v_1$ . If the underlying structure is other than DAG (3) and (6), experiments just on variable  $v_1$  can not provide enough information to distinguish the underlying structure from other DAGs in the same subset. A sequential procedure is needed to learn exact structure. Table 5 shows all possible dependent relations of variables according to sequential distributions of randomization experiments, where  $\perp\!\!\!\perp$  denotes the independent relation and  $\perp\!\!\!\perp$  denotes the dependent relation for the test of data.. It also presents the procedures of learning exact structure shown in Figure 13.

For example, if we obtain data satisfying  $v_1 \perp\!\!\!\perp v_2$  and  $v_1 \perp\!\!\!\perp v_3$  from randomization experiments on  $v_1$ . We can get that  $v_2 \rightarrow v_1 \leftarrow v_3$  from Theorem 3. Carrying out Algorithm 1, we find that no other undirected edges could be oriented. The hypotheses causal structure space is refined to a smaller constrained equivalence class, which is the third subset including DAGs  $\{4,5,7,8,9,10,11,12\}$

**Table 3.** A sequential partition of DAGs in Figure 13 with sequential constraints  $E(v_1), E(v_2), E(v_3), E(v_4)$ . Here  $E(v_i)$  denotes the configuration of undirected edges adjacent to  $v_i$ .

$E(v_1)$	$E(v_2)$	$E(v_3)$	$E(v_4)$	DAG in Figure 13
$v_2 \rightarrow v_1 \rightarrow v_3$	*	*	*	(3)
$v_2 \leftarrow v_1 \leftarrow v_3$	*	*	*	(6)
$v_2 \leftarrow v_1 \rightarrow v_3$	$v_2 \leftarrow v_3$	*	*	(1)
	$v_2 \rightarrow v_3$	*	*	(2)
$v_2 \rightarrow v_1 \leftarrow v_3$	$v_3 \rightarrow v_2 \rightarrow v_4$	*	*	(7)
	$v_3 \leftarrow v_2 \rightarrow v_4$	$v_3 \rightarrow v_4$	*	(4)
		$v_3 \leftarrow v_4$	*	(5)
	$v_3 \rightarrow v_2 \leftarrow v_4$	$v_3 \rightarrow v_4$	*	(8)
		$v_3 \leftarrow v_4$	$v_4 \rightarrow v_5$	(9)
	$v_4 \leftarrow v_5$		(11)	
	$v_3 \leftarrow v_2 \leftarrow v_4$	*	$v_4 \rightarrow v_5$	(10)
			$v_4 \leftarrow v_5$	(12)

**Table 4.** Independent relations of  $v_1, v_2$ , and  $v_3$  according to distribution of randomization experiments on  $v_1$

independent relations of $v_1, v_2$ , and $v_3$	No of subset
$v_1 \not\perp\!\!\!\perp v_2$ and $v_1 \not\perp\!\!\!\perp v_3$	(i)
$v_1 \perp\!\!\!\perp v_2$ and $v_1 \not\perp\!\!\!\perp v_3$	(ii)
$v_1 \perp\!\!\!\perp v_2$ and $v_1 \perp\!\!\!\perp v_3$	(iii)
$v_1 \not\perp\!\!\!\perp v_2$ and $v_1 \perp\!\!\!\perp v_3$	(iv)

**Table 5.** The dependent relations of variables according to distributions of randomization experiments on the variable in the first row. Here  $v_i$  in first row denotes the experimental variable. \* shows there is no experiments done on corresponding variable.

$v_1$	$v_2$	$v_3$	$v_4$	DAG in Figure 13
$v_1 \perp\!\!\!\perp v_2$ and $v_1 \not\perp\!\!\!\perp v_3$	*	*	*	(3)
$v_1 \not\perp\!\!\!\perp v_2$ and $v_1 \perp\!\!\!\perp v_3$	*	*	*	(6)
$v_1 \not\perp\!\!\!\perp v_2$ and $v_1 \not\perp\!\!\!\perp v_3$	$v_2 \perp\!\!\!\perp v_3$	*	*	(1)
	$v_2 \not\perp\!\!\!\perp v_3$	*	*	(2)
$v_1 \perp\!\!\!\perp v_2$ and $v_1 \perp\!\!\!\perp v_3$	$v_2 \perp\!\!\!\perp v_3$ and $v_2 \not\perp\!\!\!\perp v_4$	*	*	(7)
		$v_2 \not\perp\!\!\!\perp v_3$ and $v_2 \not\perp\!\!\!\perp v_4$	$v_3 \not\perp\!\!\!\perp v_4$	*
	$v_3 \perp\!\!\!\perp v_4$		*	(5)
	$v_3 \not\perp\!\!\!\perp v_4$		*	(8)
	$v_2 \perp\!\!\!\perp v_3$ and $v_2 \perp\!\!\!\perp v_4$	$v_3 \perp\!\!\!\perp v_4$	$v_4 \not\perp\!\!\!\perp v_5$	(9)
			$v_4 \perp\!\!\!\perp v_5$	(11)
	$v_2 \not\perp\!\!\!\perp v_3$ and $v_2 \perp\!\!\!\perp v_4$	*	$v_4 \not\perp\!\!\!\perp v_5$	(10)
			$v_4 \perp\!\!\!\perp v_5$	(12)

in Figure 13. Then random experiments on  $v_2$  generate data satisfying  $v_2 \perp\!\!\!\perp v_3$  and  $v_2 \perp\!\!\!\perp v_4$ , so we have  $v_4 \rightarrow v_2 \rightarrow v_3$  and further we can get  $v_3 \leftarrow v_4$  from Algorithm 1. The hypotheses causal structure space now contains two DAGs, (10) and (12) in Figure 13. Finally, experiments on  $v_4$  could determine the exact causal structure. If we get  $v_4 \perp\!\!\!\perp v_5$  from the dependent test using experimental data on  $v_4$ , we have  $v_4 \leftarrow v_5$ . So, the underlying causal structure is directed acyclic graph (12) in Figure 13.

## 5 Conclusion

In this paper, we have proposed an approach to learn causal structures based on Markov equivalence class with data generated by randomization experiments. We discussed the framework to learn causal structure with addition causal knowledge from a given Markov equivalence that can be represented by an essential graph. We discussed the partition problem of Markov equivalence class of directed acyclic graphs. We also showed that the Markov equivalence class of DAGs can be partitioned into constrained equivalence subclasses that can be represented by a chain graph and an iterative procedure can be used to get smaller and smaller constrained equivalence subclasses until the class contains only one causal structure. Then, we showed that randomization experiments can provide the necessary causal knowledge to distinguish the underlying structure from other structure in the same equivalence class. Those results provide us a more deeper understand about the Markov equivalence and also provide us an approach to discover the underlying causal structure.

**Acknowledgements.** The authors would like to thank the Editor and referees for their helpful comments. This research was supported by the 36th postdoctoral science fund 2004036180, NSFC and NBRP 2003CB715900.

## References

- [1] Andersson, S.A., Madigan, D., and Perlman, M.D., A characterization of Markov equivalence classes for acyclic digraphs, *Annals of Statistics*, 25, 505-541, 1998
- [2] Cooper G.F. and Yoo, C., Causal discovery from a mixture of experimental and observational data. In *Uncertainty in artificial intelligence: proceedings of the fifteenth conference*, 1999
- [3] Friedman, N., Inferring cellular networks using probabilistic graphical models. *Science* 303 (5659) (2004) 799-805
- [4] Heckerman, D. Geiger, D. and Chickering D.M., Learning Bayesian networks: The Combination of knowledge and statistical data. *Machine Learning*, 20,1995,197-243
- [5] Jansen R., Yu H. Y., Greenbaum, D.: A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science* 302(5644) (2003) 449-453
- [6] Pearl, J., *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988
- [7] Pearl, J. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000

- [8] Spirtes, P., Glymour, C., Scheines, R., Causation, Prediction, and Search, Springer-Verlag New York, Inc. 1993
- [9] Tian J. and Pearl J., Causal Discovery from Changes, in Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), 2001
- [10] Tian J. and Pearl J., Causal Discovery from Changes: a Bayesian Approach, UCLA Cognitive Systems Laboratory, Technical Report (R-285), February 2001
- [11] Volf, M. and Studeny, M., A graphical characterization of the largest chain graphs, International Journal of Approximate Reasoning, 20,1999,209-236
- [12] Verma, T. and Pearl, J. Equivalence and synthesis of causal models. In Uncertainty in artificial intelligence: proceedings of the sixth conference, 1990, 220-227

# Stochastic Complexity for Mixture of Exponential Families in Variational Bayes

Kazuho Watanabe<sup>1</sup> and Sumio Watanabe<sup>2</sup>

<sup>1</sup> Department of Computational Intelligence and Systems Science,  
Tokyo Institute of Technology, Mail Box:R2-5,  
4259 Nagatsuta, Midori-ku, Yokohama, 226-8503 Japan  
`kazuho23@pi.titech.ac.jp`

<sup>2</sup> P&I Lab, Tokyo Institute of Technology, Mail Box:R2-5,  
4259 Nagatsuta, Midori-ku, Yokohama, 226-8503 Japan  
`swatanab@pi.titech.ac.jp`

**Abstract.** The Variational Bayesian learning, proposed as an approximation of the Bayesian learning, has provided computational tractability and good generalization performance in many applications. However, little has been done to investigate its theoretical properties.

In this paper, we discuss the Variational Bayesian learning of the mixture of exponential families and derive the asymptotic form of the stochastic complexities. We show that the stochastic complexities become smaller than those of regular statistical models, which implies the advantage of the Bayesian learning still remains in the Variational Bayesian learning. Stochastic complexity, which is called the marginal likelihood or the free energy, not only becomes important in addressing the model selection problem but also enables us to discuss the accuracy of the Variational Bayesian approach as an approximation of the true Bayesian learning.

## 1 Introduction

The Variational Bayesian (VB) framework was proposed as an approximation of the Bayesian learning in the models with hidden variables[3][6]. This framework provides computationally tractable posterior distributions over the hidden variables and parameters with an iterative algorithm. The Variational Bayesian learning has been applied to various learning machines and it has performed good generalization with only modest computational costs compared to Markov chain Monte Carlo (MCMC) methods that are the major schemes of the Bayesian learning.

In spite of its tractability and its wide range of applications, little has been done to investigate the theoretical properties of the Variational Bayesian learning itself. Although the Variational Bayesian framework is an approximation, questions like how accurately it approximates the true one remained unanswered until quite recently. To address these issues, the asymptotic form of the stochastic complexity in the Variational Bayesian learning of gaussian mixture models was clarified and the accuracy of the Variational Bayesian learning was discussed[12].

In this paper, we focus on the Variational Bayesian learning of more general mixture models, namely the mixtures of exponential families which include mixtures of distributions such as gaussian, binomial and gamma. Mixture models are known to be non-regular statistical models since they have non-identifiability of the parameter caused by their hidden variables. In some recent studies, the Bayesian stochastic complexities of non-regular models have been clarified and it has been proven that they become smaller than those of regular models[13][14][15]. This indicates an advantage of the Bayesian learning when it is applied to non-regular models.

In this paper, we derive the upper and lower bounds of the stochastic complexity in the Variational Bayesian learning of the mixture of exponential families and show that the stochastic complexity becomes smaller than those of regular models. Since the derived bounds show us the accuracy of the Variational Bayesian learning as an approximation method, our result implies that the advantage of the Bayesian learning still remains in the Variational Bayesian learning. In addition, the derived bounds give us an indication of how the hyperparameter in the prior distribution influences the process of the learning. We consider the case in which the true distribution is contained in the learner model. Analyzing the stochastic complexity in this case is most valuable for comparing the Variational Bayesian learning with the true Bayesian learning. This is because the advantage of the Bayesian learning is typical in this case[13]. Furthermore, this analysis is necessary and essential for addressing the model selection and hypothesis testing problems.

The paper is organized as follows. In Section 2, we introduce the mixture of exponential family model. In Section 3, we describe the Bayesian learning. In Section 4, the Variational Bayesian framework is described and the variational posterior distribution for the mixture of exponential family model is derived. In Section 5, we present our main result. The main theorem is proved in Appendix. Discussion and conclusion follow in Section 6 and Section 7.

## 2 Mixture of Exponential Family

Denote by  $c(x|b)$  a probability density function of the input  $x \in \mathbb{R}^N$  given an  $M$ -dimensional parameter vector  $b = (b^{(1)}, b^{(2)}, \dots, b^{(M)})^T \in B$  where  $B$  is a subset of  $\mathbb{R}^M$ . The general mixture model  $p(x|\theta)$  with a parameter vector  $\theta$  is defined by

$$p(x|\theta) = \sum_{k=1}^K a_k c(x|b_k),$$

where  $K$  is the number of components and  $\{a_k | a_k \geq 0, \sum_{k=1}^K a_k = 1\}$  is the set of mixing proportions. The parameter  $\theta$  of the model is  $\theta = \{a_k, b_k\}_{k=1}^K$ .

A model  $p(x|\theta)$  is called a mixture of exponential family (MEF) model or exponential family mixture model if the probability distribution  $c(x|b)$  is given by the following form,

$$c(x|b) = \exp\{b \cdot f(x) + f_0(x) - g(b)\}, \quad (1)$$



where  $b \in B$  is called the natural parameter,  $b \cdot f(x)$  is the inner product with the vector  $f(x) = (f_1(x), \dots, f_M(x))^T$ ,  $f_0(x)$  and  $g(b)$  are real-valued functions of the input  $x$  and the parameter  $b$ , respectively[5]. Suppose functions  $f_1, \dots, f_M$  and a constant function are linearly independent and the effective number of parameters in a single component distribution  $c(x|b)$  is  $M$ .

The conjugate prior distribution  $\varphi(\theta)$  for the mixture of exponential family model is defined by the product of the following two distributions on  $\mathbf{a} = \{a_k\}_{k=1}^K$  and  $\mathbf{b} = \{b_k\}_{k=1}^K$

$$\varphi(\mathbf{a}) = \frac{\Gamma(K\phi_0)}{\Gamma(\phi_0)^k} \prod_{k=1}^K a_k^{\phi_0-1}, \tag{2}$$

$$\varphi(\mathbf{b}) = \prod_{k=1}^K \varphi(b_k) = \prod_{k=1}^K \frac{1}{C(\xi_0, \nu_0)} \exp\{\xi_0(b_k \cdot \nu_0 - g(b_k))\}, \tag{3}$$

where the function  $C(\xi, \mu)$  of  $\xi \in R$  and  $\mu \in \mathbb{R}^M$  is defined by

$$C(\xi, \mu) = \int \exp\{\xi(\mu \cdot b - g(b))\} db. \tag{4}$$

Here  $\xi_0 > 0$ ,  $\nu_0 \in \mathbb{R}^M$  and  $\phi_0 > 0$  are constants called hyperparameters.

The mixture model can be rewritten as follows by using a hidden variable  $y = (y^1, \dots, y^K) \in \{(1, 0, \dots, 0), (0, 1, \dots, 0), \dots, (0, 0, \dots, 1)\}$ ,

$$p(x, y|\theta) = \prod_{k=1}^K [a_k c(x|b_k)]^{y^k}.$$

The hidden variable  $y$  is not observed and is representing a component from which the datum  $x$  is generated. If and only if the datum  $x$  is from the  $k$ th component, then  $y^k = 1$ .

The mixture model is a non-regular statistical model, since it has non-identifiability of the parameter. More specifically, if the true distribution is realized by a model with the smaller number of components, the true parameter is not a point but an analytic set with singularities. If a model parameter is non-identifiable, the usual asymptotic theory of regular statistical models cannot be applied. Some studies have revealed that the mixture model has quite different properties from those of regular statistical models[7][14].

### 3 The Bayesian Learning

Suppose  $n$  training samples  $X^n = \{x_1, \dots, x_n\}$  are independently and identically taken from the true distribution  $p_0(x)$ . In the Bayesian learning of a model  $p(x|\theta)$  whose parameter is  $\theta$ , first, the prior distribution  $\varphi(\theta)$  on the parameter  $\theta$  is set. Then the posterior distribution  $p(\theta|X^n)$  is computed from the given dataset and the prior by

$$p(\theta|X^n) = \frac{1}{Z(X^n)} \varphi(\theta) \prod_{i=1}^n p(x_i|\theta),$$

where  $Z(X^n)$  is the normalization constant that is also known as the marginal likelihood or the evidence of the dataset  $X^n$ [8].

The Bayesian predictive distribution  $p(x|X^n)$  is given by averaging the model over the posterior distribution as follows,

$$p(x|X^n) = \int p(x|\theta)p(\theta|X^n)d\theta. \quad (5)$$

The stochastic complexity  $F(X^n)$  is defined by

$$F(X^n) = -\log Z(X^n), \quad (6)$$

which is also called the free energy and is important in most data modelling problems. Practically, it is used as a criterion by which the learner model is selected and the hyperparameters in the prior are optimized[1][11].

The Bayesian posterior can be rewritten as

$$p(\theta|X^n) = \frac{1}{Z_0(X^n)} \exp(-nH_n(\theta))\varphi(\theta), \quad (7)$$

where  $H_n(\theta)$  is the empirical Kullback information,

$$H_n(\theta) = \frac{1}{n} \sum_{i=1}^n \log \frac{p_0(x_i)}{p(x_i|\theta)}, \quad (8)$$

and  $Z_0(X^n)$  is the normalization constant. Putting  $S(X^n) = -\sum_{i=1}^n \log p_0(x_i)$ , we define the normalized stochastic complexity  $F_0(X^n)$  by

$$F_0(X^n) = F(X^n) - S(X^n). \quad (9)$$

It is noted that the empirical entropy  $S(X^n)$  does not depend on the model  $p(x|\theta)$  and its expectation value over all sets of training samples is equal to  $nS$  where  $S = -\int p_0(x) \log p_0(x)dx$  is the entropy. Therefore minimization of  $F(X^n)$  is equivalent to that of  $F_0(X^n)$ .

We define the average normalized stochastic complexity  $F_0(n)$  by

$$F_0(n) = E_{X^n} [F_0(X^n)], \quad (10)$$

where  $E_{X^n}[\cdot]$  denotes the expectation value over all sets of training samples.

Recently, it was proved that the average normalized stochastic complexity  $F_0(n)$  has the following asymptotic form[13],

$$F_0(n) \simeq \lambda \log n - (m - 1) \log \log n + O(1), \quad (11)$$

where  $\lambda$  and  $m$  are the rational number and the natural number respectively which are determined by the singularities of the set of true parameters. In regular statistical models,  $2\lambda$  is equal to the number of parameters and  $m = 1$ , whereas in non-regular models such as the mixture model,  $2\lambda$  is not larger than the number

of parameters and  $m \geq 1$ . This means an advantage of non-regular models in the case when the Bayesian learning is applied to them.

However, in order to carry out the Bayesian learning practically, one computes the stochastic complexity or the predictive distribution by integrating over the posterior distribution, which typically cannot be performed analytically. As an approximation, the Variational Bayesian framework was proposed[3][4][6].

## 4 The Variational Bayesian Learning

### 4.1 The Variational Bayesian Framework

Using the likelihood on the complete data  $\{X^n, Y^n\}$  added the corresponding hidden variables  $Y^n = \{y_1, \dots, y_n\}$ , we can rewrite the stochastic complexity eq.(6) as

$$F(X^n) = -\log \int \sum_{Y^n} p(X^n, Y^n, \theta) d\theta,$$

where  $p(X^n, Y^n, \theta) = \varphi(\theta) \prod_{i=1}^n p(x_i, y_i | \theta)$  and the sum over  $Y^n$  ranges over all possible values of all hidden variables.

In the Variational Bayesian framework, the Bayesian posterior distribution  $p(Y^n, \theta | X^n)$  of the hidden variables and the parameters is approximated by the variational posterior distribution  $q(Y^n, \theta | X^n)$ , which factorizes as

$$q(Y^n, \theta | X^n) = Q(Y^n | X^n) r(\theta | X^n), \tag{12}$$

where  $Q(Y^n | X^n)$  and  $r(\theta | X^n)$  are probability distributions on the hidden variables and the parameters respectively. The variational posterior  $q(Y^n, \theta | X^n)$  is chosen so that it minimizes the functional  $\bar{F}[q]$  defined by

$$\bar{F}[q] = \sum_{Y^n} \int q(Y^n, \theta | X^n) \log \frac{q(Y^n, \theta | X^n)}{p(X^n, Y^n, \theta)} d\theta, \tag{13}$$

$$= F(X^n) + K(q(Y^n, \theta | X^n) || p(Y^n, \theta | X^n)), \tag{14}$$

where  $K(q(Y^n, \theta | X^n) || p(Y^n, \theta | X^n))$  is the Kullback information between the true Bayesian posterior  $p(Y^n, \theta | X^n)$  and the variational posterior  $q(Y^n, \theta | X^n)$ <sup>1</sup>. This leads to the following theorem. The proof is well-known[4][10].

**Theorem 1.** *Let  $\bar{F}[q]$  be the functional defined by (13) and  $Q(Y^n | X^n)$  be a probability distribution on the hidden variables. Then the variational posterior  $q(Y^n, \theta | X^n)$  that minimizes  $\bar{F}[q]$  is given by*

$$r(\theta | X^n) = \frac{1}{C_r} \varphi(\theta) \exp \langle \log p(X^n, Y^n | \theta) \rangle_{Q(Y^n | X^n)}, \tag{15}$$

---

<sup>1</sup> Throughout this paper, we use the notation  $K(q(x) || p(x))$  for the Kullback information from a distribution  $q(x)$  to a distribution  $p(x)$ , that is,

$$K(q(x) || p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx.$$

$$Q(Y^n|X^n) = \frac{1}{C_Q} \exp \langle \log p(X^n, Y^n|\theta) \rangle_{r(\theta|X^n)}, \tag{16}$$

Hereafter, we omit the condition  $X^n$  of the variational posteriors and abbreviate them to  $q(Y^n, \theta)$ ,  $Q(Y^n)$  and  $r(\theta)$ .

Note that eq.(15) and eq.(16) give only a necessary condition that  $r(\theta)$  and  $Q(Y^n)$  minimize the functional  $\overline{F}[q]$ . The variational posteriors that satisfy eq.(15) and eq.(16) are searched by an iterative algorithm.

We define the stochastic complexity in the Variational Bayesian learning  $\overline{F}(X^n)$  by the minimum value of the functional  $\overline{F}[q]$  attained by the above optimal variational posteriors, that is ,

$$\overline{F}(X^n) = \min_{r, Q} \overline{F}[q].$$

Since  $\overline{F}(X^n)$ , the stochastic complexity in the Variational Bayesian learning, gives the upper bound of the true stochastic complexity  $F(X^n)$ ,  $\overline{F}(X^n)$  itself is an estimate of  $F(X^n)$  and is used for the model selection in the Variational Bayesian learning. Moreover, from eq.(14), it is noted that the difference between  $\overline{F}(X^n)$  and the original stochastic complexity  $F(X^n)$  is the Kullback information from the variational posterior to the true posterior, which shows us the accuracy of the Variational Bayesian approach as an approximation of the true Bayesian learning.

We define the normalized stochastic complexity  $\overline{F}_0(X^n)$  in the Variational Bayesian learning by

$$\overline{F}_0(X^n) = \overline{F}(X^n) - S(X^n). \tag{17}$$

Putting eq.(16) into eq.(13) gives the following lemma.

**Lemma 1.**

$$\overline{F}_0(X^n) = \min_{r(\theta)} \{K(r(\theta)||\varphi(\theta)) - (\log C_Q + S(X^n))\}, \tag{18}$$

$$C_Q = \sum_{Y^n} \exp \langle \log p(X^n, Y^n|\theta) \rangle_{r(\theta)}.$$

### 4.2 Variational Posterior for Mixture of Exponential Family Model

In this subsection, we derive the variational posterior  $r(\theta)$  for the mixture of exponential family model based on eq.(15) and then define the variational parameter and the variational estimator for this model.

Using the complete data  $\{X^n, Y^n\} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , we put

$$\overline{y}_i^k = \langle y_i^k \rangle_{Q(Y^n)}, \quad n_k = \sum_{i=1}^n \overline{y}_i^k, \quad \text{and} \quad \nu_k = \frac{1}{n_k} \sum_{i=1}^n \overline{y}_i^k f(x_i),$$

---

<sup>2</sup> For an arbitrary distribution  $p(x)$ ,  $\langle \cdot \rangle_{p(x)}$  denotes the expectation over  $p(x)$ .

where  $y_i^k = 1$  if the  $i$ th datum  $x_i$  is from the  $k$ th component, if otherwise,  $y_i^k = 0$ . The variable  $n_k$  is the expected number of the data that are estimated to be from the  $k$ th component. From eq.(15) and the respective prior eq.(2) and eq.(3), the variational posterior  $r(\theta) = r(\mathbf{a})r(\mathbf{b})$  is obtained as the product of the following two distributions,

$$r(\mathbf{a}) = \frac{\Gamma(n + K\phi_0)}{\prod_{k=1}^K \Gamma(n_k + \phi_0)} \prod_{k=1}^K a_k^{n_k + \phi_0 - 1}, \quad (19)$$

$$r(\mathbf{b}) = \prod_{k=1}^K r(b_k) = \prod_{k=1}^K \frac{1}{C(\gamma_k, \bar{\mu}_k)} \exp\{\gamma_k(\bar{\mu}_k \cdot b_k - g(b_k))\}, \quad (20)$$

where  $\bar{\mu}_k = \frac{n_k \nu_k + \xi_0 \nu_0}{n_k + \xi_0}$ , and  $\gamma_k = n_k + \xi_0$ .

Let

$$\bar{a}_k = \langle a_k \rangle_{r(\mathbf{a})} = \frac{n_k + \phi_0}{n + K\phi_0} \quad \left( \frac{\phi_0}{n + K\phi_0} \leq \bar{a}_k \leq 1 - \frac{(K-1)\phi_0}{n + K\phi_0} \right), \quad (21)$$

$$\bar{b}_k = \langle b_k \rangle_{r(b_k)} = \frac{1}{\gamma_k} \frac{\partial \log C(\gamma_k, \bar{\mu}_k)}{\partial \bar{\mu}_k}, \quad (22)$$

and define the variational parameter  $\bar{\theta}$  by

$$\bar{\theta} = \langle \theta \rangle_{r(\theta)} = \{\bar{a}_k, \bar{b}_k\}_{k=1}^K. \quad (23)$$

It is noted that  $\bar{b}_k$  is the expectation parameter of  $b_k$  with the variational posterior  $r(b_k)$ . It is also noted that the variational posterior  $r(\theta)$  and  $C_Q$  in eq.(16) are parameterized by the variational parameter  $\bar{\theta}$ . Therefore, we denote them as  $r(\theta|\bar{\theta})$  and  $C_Q(\bar{\theta})$  henceforth. We define the variational estimator  $\bar{\theta}_{vb}$  by the variational parameter  $\bar{\theta}$  that attains the minimum value of the normalized stochastic complexity  $\bar{F}_0(X^n)$ . Then, Lemma 1 claims that

$$\bar{\theta}_{vb} = \underset{\bar{\theta}}{\operatorname{argmin}} \{K(r(\theta|\bar{\theta})|\varphi(\theta)) - (\log C_Q(\bar{\theta}) + S(X^n))\}. \quad (24)$$

In the Variational Bayesian learning, the variational parameter  $\bar{\theta}$  is updated iteratively to find the optimal solution  $\bar{\theta}_{vb}$ . Therefore, our aim is to evaluate the minimum value of the right hand side of eq.(24) as a function of the variational parameter  $\bar{\theta}$ .

## 5 Main Result

The average normalized stochastic complexity  $\bar{F}_0(n)$  in the Variational Bayesian learning is defined by

$$\bar{F}_0(n) = E_{X^n}[\bar{F}_0(X^n)]. \quad (25)$$

We assume the following conditions.

- (i) The true distribution  $p_0(x)$  is represented by a mixture of exponential family model  $p(x|\theta_0)$  which has  $K_0$  components and the parameter  $\theta_0 = \{a_k^*, b_k^*\}_{k=1}^{K_0}$ ,

$$p(x|\theta_0) = \sum_{k=1}^{K_0} a_k^* \exp\{b_k^* \cdot f(x) + f_0(x) - g(b_k^*)\},$$

where  $b_k^* \in \mathbb{R}^M$  and  $b_k^* \neq b_j^* (k \neq j)$ . And suppose that the true distribution can be realized by the model, that is, the model  $p(x|\theta)$  has  $K$  components,

$$p(x|\theta) = \sum_{k=1}^K a_k \exp\{b_k \cdot f(x) + f_0(x) - g(b_k)\},$$

and  $K \geq K_0$  holds.

- (ii) The prior distribution of the parameters is the conjugate prior  $\varphi(\theta) = \varphi(\mathbf{a})\varphi(\mathbf{b})$  where  $\varphi(\mathbf{a})$  and  $\varphi(\mathbf{b})$  are given by eq.(2) and eq.(3).
- (iii) Regarding the distribution  $c(x|b)$  of each component, the Fisher information matrix

$$I(b) = \frac{\partial^2 g(b)}{\partial b \partial b}$$

satisfies  $0 < |I(b)| < +\infty$ , for arbitrary  $b \in B$ <sup>3</sup>. The function  $\mu \cdot b - g(b)$  has a stationary point at  $\hat{b}$  in the interior of  $B$  for each  $\mu \in \mathbb{R}^M$ .

Under these conditions, we prove the following theorem. The proof is done in Appendix.

**Theorem 2. (Main Result)**

$$\underline{\lambda} \log n + E_{X^n} [nH_n(\bar{\theta}_{vb})] + C_1 \leq \bar{F}_0(n) \leq \bar{\lambda} \log n + C_2, \tag{26}$$

$$\underline{\lambda} = \begin{cases} (K-1)\phi_0 + \frac{M}{2} & (\phi_0 \leq \frac{M+1}{2}), \\ \frac{MK+K-1}{2} & (\phi_0 > \frac{M+1}{2}), \end{cases} \tag{27}$$

$$\bar{\lambda} = \begin{cases} (K-K_0)\phi_0 + \frac{MK_0+K_0-1}{2} & (\phi_0 \leq \frac{M+1}{2}), \\ \frac{MK+K-1}{2} & (\phi_0 > \frac{M+1}{2}). \end{cases} \tag{28}$$

This theorem shows the asymptotic form of the average stochastic complexity in the Variational Bayesian learning. The coefficient  $\bar{\lambda}$  and  $\underline{\lambda}$  are identified by  $K, K_0$ , that are the numbers of components of the learner and the true distribution, the number of parameters  $M$  of each component and the hyperparameter  $\phi_0$  of the conjugate prior given by eq.(2).

In this theorem,  $nH_n(\bar{\theta}_{vb})$  is equal to  $-\sum_{i=1}^n \log p(x_i|\bar{\theta}_{vb}) - S(X^n)$ , and the term  $-\frac{1}{n} \sum_{i=1}^n \log p(x_i|\bar{\theta}_{vb})$  is a training error which is computable during

<sup>3</sup>  $\frac{\partial^2 g(b)}{\partial b \partial b}$  denotes the matrix whose  $ij$ th entry is  $\frac{\partial^2 g(b)}{\partial b^{(i)} \partial b^{(j)}}$  and  $|\cdot|$  denotes the determinant of a matrix.

the learning. If the term  $E_{X^n} [nH_n(\bar{\theta}_{vb})]$  is a bounded function of  $n$ , then it immediately follows from this theorem that

$$\underline{\lambda} \log n + O(1) \leq \bar{F}_0(n) \leq \bar{\lambda} \log n + O(1),$$

where  $O(1)$  is a bounded function of  $n$ . In certain cases, such as binomial mixtures, it is actually a bounded function of  $n$ . In the case of gaussian mixtures, if  $B = \mathbb{R}^N$ , it is conjectured that the minus likelihood ratio  $\min_{\theta} nH_n(\theta)$ , a lower bound of  $nH_n(\bar{\theta}_{vb})$ , is at most of the order of  $\log \log n$ [7]. Note that however, even if  $E_{X^n} [\min_{\theta} nH_n(\theta)]$  diverges to minus infinity, this does not necessarily mean  $E_{X^n} [nH_n(\bar{\theta}_{vb})]$  diverges in the same order.

Since the dimension of the parameter  $\theta$  is  $MK + K - 1$ , the average normalized stochastic complexity of regular statistical models, which coincides with the Bayesian information criterion (BIC)[11] and the minimum description length (MDL)[9], is given by  $\lambda_{\text{BIC}} \log n$  where

$$\lambda_{\text{BIC}} = \frac{MK + K - 1}{2}. \tag{29}$$

Theorem 2 claims that the coefficient  $\bar{\lambda}$  of  $\log n$  is smaller than  $\lambda_{\text{BIC}}$  when  $\phi_0 \leq (M + 1)/2$ . This means that the stochastic complexity  $\bar{F}_0(n)$  becomes smaller than the BIC and implies that the advantage of non-regular models in the Bayesian learning still remains in the Variational Bayesian learning.

## 6 Discussion

In this paper, we showed the upper and lower bounds of the stochastic complexity for mixtures of exponential families in the Variational Bayesian learning.

Firstly let us discuss the lower bound. The lower bound in eq.(26) can be improved to give

$$\bar{F}_0(n) \geq \bar{\lambda} \log n + E_{X^n} [nH_n(\bar{\theta}_{vb})] + C_1, \tag{30}$$

if the consistency of the variational estimator  $\bar{\theta}_{vb}$  is proven. Note that the coefficient  $\bar{\lambda}$  is the same as that of the upper bound given in Theorem 2. The consistency means that the mixing coefficient  $\bar{a}_k$  does not tend to zero for at least  $K_0$  components and they are always used to learn the  $K_0$  true components when the sample size  $n$  is sufficiently large. We conjecture that the variational estimator is consistent and the lower bound in eq.(30) is obtained for most mixture components. However, little has been known so far about the behavior of the variational estimator. Analyzing its behavior and investigating the consistency is an important undertaking.

Secondly, we compare the stochastic complexity shown in Theorem 2 with the one in the true Bayesian learning. The stochastic complexities in the true Bayesian learning of several non-regular models have been clarified in some recent studies. On the mixture models with  $M$  parameters in each component, the

following upper bound on the coefficient of the average normalized stochastic complexity  $F_0(n)$  in eq.(11) is known[14][15],

$$\lambda \leq \begin{cases} (K + K_0 - 1)/2 & (M = 1), \\ (K - K_0) + (MK_0 + K_0 - 1)/2 & (M \geq 2), \end{cases} \quad (31)$$

Under the same condition (i) about the true distribution and the model described in Section 5 and certain conditions about the prior distribution. Since these conditions about the prior are satisfied by putting  $\phi_0 = 1$  in the condition (ii) of Theorem 2, we can compare the stochastic complexity in this case. Putting  $\phi_0 = 1$  in eq.(28), we have

$$\bar{\lambda} = K - K_0 + (MK_0 + K_0 - 1)/2. \quad (32)$$

Let us compare this  $\bar{\lambda}$  of the Variational Bayesian learning to  $\lambda$  in eq.(31) of the true Bayesian learning. When  $M = 1$ , that is, each component has one parameter,  $\bar{\lambda} \geq \lambda$  holds since  $K_0 \leq K$ . This means that the more redundant components the model has, the more the Variational Bayesian learning differs from the true Bayesian learning. In this case,  $2\bar{\lambda}$  is equal to  $2K - 1$  that is the number of the parameters of the model. Hence the BIC[11] and the MDL[9] correspond to  $\bar{\lambda} \log n$  when  $M = 1$ . If  $M \geq 2$ , the upper bound of  $\lambda$  is equal to  $\bar{\lambda}$ . This implies that the variational posterior is close to the true Bayesian posterior when  $M \geq 2$ . More precise discussion about the accuracy of the approximation can be done for models on which tighter bounds or exact values of the coefficient  $\lambda$  in eq.(11) are given[12][16].

Thirdly, we point out that Theorem 2 shows how the hyperparameter  $\phi_0$  influences the process of the Variational Bayesian learning. The coefficient  $\bar{\lambda}$  in eq.(28) is divided into two cases,  $\phi_0 \leq (M + 1)/2$  or otherwise, indicating that the influence of the hyperparameter  $\phi_0$  in the prior  $\varphi(\mathbf{a})$  appears depending on the dimension  $M$  of the parameter in each component. More specifically, only when  $\phi_0 \leq (M + 1)/2$ , the prior distribution works to reduce the redundant components that the model has and otherwise it works to use all the components.

And lastly, let us give examples to show how to use the theoretical bounds in eq.(26). One can examine experimentally whether the actual iterative algorithm converges to the optimal variational posterior instead of local minima by comparing the stochastic complexity with the theoretical bounds. The theoretical bounds would also enable us to compare the accuracy of the Variational Bayesian learning with that of the Laplace approximation or the MCMC method. Furthermore, as mentioned in Section 4, the stochastic complexity  $\bar{F}(X^n)$  is used as a criterion for the model selection in the Variational Bayesian learning. Our result is important for developing effective model selection methods using  $\bar{F}(X^n)$ .

## 7 Conclusion

In this paper, we mathematically proved the lower and upper bounds of the stochastic complexity of the Variational Bayesian learning in mixtures of general exponential families. These bounds will be used for evaluation and optimization of variational learning systems.



## Acknowledgements

This work was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for JSPS Fellows 4637 and for Scientific Research 15500130, 2005.

## References

- [1] H.Akaike, "Likelihood and Bayes procedure," *Bayesian Statistics*, (Bernald J.M. eds.) University Press, Valencia, Spain, pp.143-166, 1980.
- [2] H.Alzer, "On some inequalities for the Gamma and Psi functions," *Mathematics of computation*, Vol.66, No.217, pp.373-389, 1997.
- [3] H.Attias, "Inferring parameters and structure of latent variable models by variational bayes," *Proc. of Uncertainty in Artificial Intelligence(UAI'99)*, 1999.
- [4] M.J.Beal, "Variational algorithms for approximate bayesian inference," Ph.D. Thesis, University College London, 2003.
- [5] L.D.Brown, "Fundamentals of statistical exponential families," IMS Lecture Notes-Monograph Series 9, 1986.
- [6] Z.Ghahramani, M.J.Beal, "Graphical models and variational methods," *Advanced Mean Field Methods - Theory and Practice*, eds. D. Saad and M. Opper, MIT Press, 2000.
- [7] J.A.Hartigan, "A Failure of likelihood asymptotics for normal mixtures," *Proceedings of the Berkeley Conference in Honor of J.Neyman and J.Kiefer*, Vol.2, 807-810, 1985.
- [8] D.J. Mackay, "Bayesian interpolation," *Neural Computation*, Vol.4, No.2, pp.415-447, 1992.
- [9] J.Rissanen, "Stochastic complexity and modeling" *Annals of Statistics*, Vol.14, No.3, pp.1080-1100, 1986.
- [10] M.Sato, "Online model selection based on the variational bayes," *Neural Computation*, Vol.13, No.7, pp.1649 - 1681, 2004.
- [11] G.Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, Vol.6, No.2, pp.461-464, 1978.
- [12] K.Watanabe, S.Watanabe, "Lower bounds of stochastic complexities in variational bayes learning of gaussian mixture models," *Proceedings of IEEE conference on Cybernetics and Intelligent Systems (CIS04)*, pp.99-104, 2004.
- [13] S.Watanabe, "Algebraic analysis for non-identifiable learning machines," *Neural Computation*, Vol.13, No.4, pp.899-933, 2001.
- [14] K.Yamazaki, S.Watanabe, "Singularities in mixture models and upper bounds of stochastic complexity," *International Journal of Neural Networks*, 16, pp.1029-1038, 2003.
- [15] K.Yamazaki, S.Watanabe "Stochastic complexity of bayesian networks," *Proc. of Uncertainty in Artificial Intelligence(UAI'03)*, 2003.
- [16] K.Yamazaki, S.Watanabe "Newton diagram and stochastic complexity in mixture of binomial distributions," *Proc. of Algorithmic Learning Theory(ALT2004)*, pp.350-364, 2004.

## Appendix: Proof of Theorem 2

From Lemma 1, it is noted that we can evaluate the normalized stochastic complexity  $\bar{F}_0(X^n)$  by analyzing two terms  $K(r(\theta|\bar{\theta})||\varphi(\theta))$  and  $(\log C_Q(\bar{\theta}) + S(X^n))$

respectively. First, we evaluate the first one. Since the variational posterior satisfies  $r(\theta|\bar{\theta}) = r(\mathbf{a}|\bar{\mathbf{a}})r(\mathbf{b}|\bar{\mathbf{b}})$ , we have

$$K(r(\theta|\bar{\theta})|\varphi(\theta)) = K(r(\mathbf{a}|\bar{\mathbf{a}})|\varphi(\mathbf{a})) + \sum_{k=1}^K K(r(b_k|\bar{b}_k)|\varphi(b_k)). \quad (33)$$

$K(r(b_k|\bar{b}_k)|\varphi(b_k))$  is evaluated as follows <sup>4</sup>.

**Lemma 2.**

$$K(r(b_k|\bar{b}_k)|\varphi(b_k)) = \frac{M}{2} \log(n_k + \xi_0) - \log \varphi(\bar{b}_k) + O_p(1).$$

Using the variational posterior, eq.(20), we obtain

$$K(r(b_k|\bar{b}_k)|\varphi(b_k)) = -\log \frac{C(\gamma_k, \bar{\mu}_k)}{C(\xi_0, \nu_0)} + n_k \{ \nu_k \langle b_k \rangle_{r(b_k|\bar{b}_k)} - \langle g(b_k) \rangle_{r(b_k|\bar{b}_k)} \}, \quad (34)$$

where we put  $\gamma_k = n_k + \xi_0$ . Also it follows from eq.(4) and eq.(22) that

$$\langle g(b_k) \rangle_{r(b_k|\bar{b}_k)} = \bar{b}_k \cdot \bar{\mu}_k - \frac{\partial \log C(\gamma_k, \bar{\mu}_k)}{\partial \gamma_k}. \quad (35)$$

Now let us evaluate the value of  $C(\gamma_k, \bar{\mu}_k)$  when  $\gamma_k$  is sufficiently large. From the condition (iii), using the saddle point approximation, we obtain

$$C(\gamma_k, \bar{\mu}_k) = \exp \left[ \gamma_k \{ \bar{\mu}_k \cdot \hat{b}_k - g(\hat{b}_k) \} \right] \sqrt{\frac{2\pi}{\gamma_k}}^M \sqrt{|I(\hat{b}_k)|}^{-1} \left\{ 1 + \frac{C'}{\gamma_k} + O_p\left(\frac{1}{\gamma_k^2}\right) \right\}, \quad (36)$$

where  $C'$  is a constant and  $\hat{b}_k$  is the maximizer of the function  $\bar{\mu} \cdot b_k - g(b_k)$ , that is,

$$\frac{\partial g(\hat{b}_k)}{\partial b_k} = \bar{\mu}_k.$$

Therefore,  $-\log C(\gamma_k, \bar{\mu}_k)$  is evaluated as follows,

$$-\log C(\gamma_k, \bar{\mu}_k) = \frac{M}{2} \log \frac{\gamma_k}{2\pi} + \frac{1}{2} \log |I(\hat{b}_k)| - \gamma_k (\bar{\mu}_k \cdot \hat{b}_k - g(\hat{b}_k)) + \frac{C'}{\gamma_k} + O_p\left(\frac{1}{\gamma_k^2}\right). \quad (37)$$

Consequently, we obtain

$$\frac{\partial \log C(\gamma_k, \bar{\mu}_k)}{\partial \bar{\mu}_k} = \gamma_k \cdot \hat{b}_k, \quad (38)$$

$$\frac{\partial \log C(\gamma_k, \bar{\mu}_k)}{\partial \gamma_k} = (\bar{\mu}_k \cdot \hat{b}_k - g(\hat{b}_k)) + \frac{M}{2} \frac{1}{\gamma_k} + O_p\left(\frac{1}{\gamma_k^2}\right). \quad (39)$$

It follows from eq.(22) and eq.(38),

<sup>4</sup> In this proof,  $O_p(1)$  denotes a random variable bounded in probability.

$$\hat{b}_k = \bar{b}_k, \tag{40}$$

and from eq.(35) and eq.(39),

$$\langle g(b_k) \rangle_{r(b_k|\bar{b}_k)} = g(\bar{b}_k) + \frac{M}{2} \frac{1}{\gamma_k} + O_p\left(\frac{1}{\gamma_k^2}\right). \tag{41}$$

Thus from eqs.(34),(22),(41) and (37), we obtain the lemma. □

Then, the first term on the right-hand side of eq.(18) is given in the following.

**Lemma 3.**

$$K(r(\theta|\bar{\theta})||\varphi(\theta)) = G(\bar{\mathbf{a}}) - \sum_{k=1}^K \log \varphi(\bar{b}_k) + O_p(1) \tag{42}$$

where  $G(\bar{\mathbf{a}})$ ,  $\bar{\mathbf{a}} = \{\bar{a}_k\}_{k=1}^K$ ,

$$G(\bar{\mathbf{a}}) = \frac{MK + K - 1}{2} \log n + \left\{ \frac{M}{2} - \left(\phi_0 - \frac{1}{2}\right) \right\} \sum_{k=1}^K \log \bar{a}_k. \tag{43}$$

From eq.(19) and eq.(2), we obtain

$$K(r(\mathbf{a}|\bar{\mathbf{a}})||\varphi(\mathbf{a})) = \sum_{k=1}^K h(n_k) - n\Psi(n + K\phi_0) + \log \Gamma(n + K\phi_0) + \log \frac{\Gamma(\phi_0)^K}{\Gamma(K\phi_0)}, \tag{44}$$

where  $\Psi(x) = \Gamma'(x)/\Gamma(x)$  is the di-gamma(psi) function and we used

$$\langle \log a_k \rangle_{r(\mathbf{a}|\bar{\mathbf{a}})} = \Psi(n_k + \phi_0) - \Psi(n + K\phi_0)$$

and the notation  $h(x) = x\Psi(x + \phi_0) - \log \Gamma(x + \phi_0)$ .

By using inequalities for the di-gamma function  $\Psi(x)$  and the log-gamma function  $\log \Gamma(x)$ , for  $x > 0$  and for a positive constant  $C$ ,

$$\frac{1}{2x} < \log x - \Psi(x) < \frac{1}{x}, \tag{45}$$

$$0 \leq \log \Gamma(x) - \left(x - \frac{1}{2}\right) \log x + x - \frac{1}{2} \log 2\pi \leq \frac{C}{x}, \tag{46}$$

we obtain

$$h(x) = -\left(\phi_0 - \frac{1}{2}\right) \log(x + \phi_0) + x + O(1),$$

and from eqs.(33),(44) and Lemma 2, we complete the proof. □

Let us now turn to the second term,  $\log C_Q(\bar{\theta})$ , on the right-hand side of eq.(18). It is evaluated as follows.

**Lemma 4.**

$$nH_n(\bar{\theta}) + O_p(1) \leq -(\log C_Q(\bar{\theta}) + S(X^n)) \leq n\bar{H}_n(\bar{\theta}) + O_p(1) \tag{47}$$

$$\bar{H}_n(\bar{\theta}) = \frac{1}{n} \sum_{i=1}^n \log \frac{p(x_i|\theta_0)}{\sum_{k=1}^K \bar{a}_k c(x_i|\bar{b}_k) \exp\left\{-\frac{M+2}{2(n_k + \min\{\phi_0, \xi_0\})} + O_p\left(\frac{1}{n_k^2}\right)\right\}}.$$

$$\begin{aligned} C_Q(\bar{\theta}) &= \prod_{i=1}^n \sum_{k=1}^K \exp\langle \log a_k c(x_i|b_k) \rangle_{r(\theta|\bar{\theta})} \\ &= \prod_{i=1}^n \sum_{k=1}^K \exp\{\Psi(n_k + \phi_0) - \Psi(n + K\phi_0) + \bar{b}_k \cdot f(x_i) - \langle g(b_k) \rangle_{r(\theta|\bar{\theta})} + f_0(x_i)\}. \end{aligned}$$

Using again the inequalities (45) and eq.(41), we obtain

$$\log C_Q(\bar{\theta}) \geq \sum_{i=1}^n \log \left[ \sum_{k=1}^K \bar{a}_k c(x_i|\bar{b}_k) \exp\left\{-\frac{M+2}{2(n_k + \min\{\phi_0, \xi_0\})} + O_p\left(\frac{1}{n_k^2}\right)\right\} \right] + O_p(1),$$

$$\log C_Q(\bar{\theta}) \leq \sum_{i=1}^n \log \left[ \sum_{k=1}^K \bar{a}_k c(x_i|\bar{b}_k) \right] + O_p(1),$$

which give the upper and lower bounds in eq.(47) respectively. □

From above lemmas, we show the following theorem on the upper bound in eq.(26).

**Theorem 3.**

$$\bar{F}_0(X^n) \leq \bar{\lambda} \log n + O_p(1),$$

$$\bar{\lambda} = \bar{\lambda}(\bar{\theta}, \bar{\mathbf{a}}, \bar{\mathbf{b}}).$$

From Lemma 1, Lemma 3 and Lemma 4, it follows that

$$\begin{aligned} \bar{F}_0(X^n) &\leq \min_{\bar{\theta}} \left[ G(\bar{\mathbf{a}}) - \sum_{k=1}^K \log \varphi(\bar{b}_k) + n\bar{H}_n(\bar{\theta}) \right] + O_p(1) \\ &\equiv \min_{\bar{\theta}} [T_n(\bar{\theta})] + O_p(1). \end{aligned} \tag{48}$$

From eq.(48), it is noted that the function values of  $T_n(\bar{\theta})$  at specific points of the variational parameter  $\bar{\theta}$  give upper bounds of the normalized stochastic complexity  $\bar{F}_0(X^n)$ . Hence, let us consider the following two cases where  $R_1$  and  $R_2$  are random variables of the order of  $\frac{1}{\sqrt{n}}$ .  $\bar{H}_n(\bar{\theta}) = O_p(\frac{1}{n})$  holds in both cases.

(I) :When

$$\bar{a}_k = a_k^* \quad (1 \leq k \leq K_0 - 1), \quad \bar{a}_k = a_{K_0}^*/(K - K_0 + 1) \quad (K_0 \leq k \leq K),$$

$$\bar{b}_1 = b_1^* + R_1, \quad \bar{b}_k = b_k^* \quad (2 \leq k \leq K_0 - 1), \quad \bar{b}_k = b_{K_0}^* \quad (K_0 \leq k \leq K),$$

then

$$T_n(\bar{\theta}) = \frac{MK + K - 1}{2} \log n + O_p(1).$$

(II) :When

$$\bar{a}_k = a_k^* \frac{n + K_0 \phi_0}{n + K \phi_0} \quad (1 \leq k \leq K_0), \quad \bar{a}_k = \frac{\phi_0}{n + K \phi_0} \quad (K_0 + 1 \leq k \leq K),$$

$$\bar{b}_1 = b_1^* + R_2, \quad \bar{b}_k = b_k^* \quad (2 \leq k \leq K_0), \quad \bar{b}_k = \nu_0 \quad (K_0 + 1 \leq k \leq K),$$

then

$$T_n(\bar{\theta}) = \left\{ (K - K_0) \phi_0 + \frac{MK_0 + K_0 - 1}{2} \right\} \log n + O_p(1).$$

From eq.(48), we prove the theorem.  $\square$

Next we show the following theorem on the lower bound in eq.(26).

**Theorem 4.** Let  $\bar{F}_0(X^n)$  be the empirical distribution function of  $X^n$  and  $\bar{\theta}_{vb}$  be the true parameter vector. Then

$$\bar{F}_0(X^n) \geq \lambda \log n + nH_n(\bar{\theta}_{vb}) + O_p(1), \quad (49)$$

where  $\lambda = \lambda(\phi_0)$  is a positive constant.

Since  $\log \varphi(\bar{b}_k)$  is a bounded function of  $n$ , it follows from Lemma 1, Lemma 3 and Lemma 4,

$$\bar{F}_0(X^n) \geq \min_{\bar{\mathbf{a}}} \{G(\bar{\mathbf{a}})\} + nH_n(\bar{\theta}_{vb}) + O_p(1). \quad (50)$$

If  $\phi_0 > \frac{M+1}{2}$ , then

$$G(\bar{\mathbf{a}}) \geq \frac{MK + K - 1}{2} \log n - \left( \frac{M+1}{2} - \phi_0 \right) K \log K, \quad (51)$$

since Jensen's inequality yields that  $\sum_{k=1}^K \log \bar{a}_k \leq K \log(\frac{1}{K})$ .

If  $\phi_0 \leq \frac{M+1}{2}$ , then

$$G(\bar{\mathbf{a}}) \geq \left\{ (K - 1) \phi_0 + \frac{M}{2} \right\} \log n + O_p(1), \quad (52)$$

since  $\bar{a}_k \geq \frac{\phi_0}{n + K \phi_0}$  holds for every  $k$  and the constraint  $\sum_{k=1}^K \bar{a}_k = 1$  ensures that  $\log \bar{a}_k = O_p(1)$  for at least one index  $k$ . From eqs.(50),(51) and (52), we obtain the theorem.

Let us combine these theorems and complete the proof.

**(Proof of Theorem 2)**

From Theorem 3 and Theorem 4, we have

$$\bar{\lambda} \log n + nH_n(\bar{\theta}_{vb}) + O_p(1) \leq \bar{F}_0(X^n) \leq \bar{\lambda} \log n + O_p(1).$$

Taking the expectation over all sets of training samples gives Theorem 2.  $\square$

# ACME: An Associative Classifier Based on Maximum Entropy Principle

Risi Thonangi and Vikram Pudi

Center for Data Engineering,  
International Institute of Information Technology, Hyderabad  
rishi@research.iiit.ac.in, vikram@iiit.ac.in

**Abstract.** Recent studies in classification have proposed ways of exploiting the association rule mining paradigm. These studies have performed extensive experiments to show their techniques to be both efficient and accurate. However, existing studies in this paradigm either do not provide any theoretical justification behind their approaches or assume independence between some *parameters*. In this work, we propose a new classifier based on association rule mining. Our classifier rests on the maximum entropy principle for its statistical basis and does not assume any independence not inferred from the given dataset. We use the classical generalized iterative scaling algorithm (GIS) to create our classification model. We show that GIS fails in some cases when itemsets are used as features and provide modifications to rectify this problem. We show that this modified GIS runs much faster than the original GIS. We also describe techniques to make GIS tractable for large feature spaces – we provide a new technique to divide a feature space into independent clusters each of which can be handled separately. Our experimental results show that our classifier is generally more accurate than the existing classification methods.

## 1 Introduction

*Classification* has been an age old problem. It involves labeling a query with one among a set of possible class labels by learning from available query-label pairs. Previous studies such as decision trees [20], rule learning [3], naive-bayes [8] and other statistical approaches [15] have developed heuristic/greedy search techniques for building classifiers. These techniques build a set of rules covering the given dataset and use them for prediction. Machine learning approaches like SVMs [5] do classification by learning boundaries between classes and checking on which side of the boundary the query lies.

Recent studies in classification have proposed ways to exploit the paradigm of association rule mining for the problem of classification. These methods mine high quality association rules and build classifiers based on them [16] [7] [18]. We refer to these approaches as *associative classifiers* and they have several advantages – (1) Frequent itemsets capture all the dominant relationships between items in a dataset. (2) Efficient itemset mining algorithms exist. (3) These classifiers naturally handle *missing values* and *outliers* as they only deal with *statistically significant* associations. This property translates well into the classification framework to make it robust. (4) Extensive performance studies [14] [17] have shown such classifiers to be generally more *accurate*.

However, existing studies in the associative classification paradigm either do not provide any theoretical justification behind their approaches [14] or assume independence between some *parameters* in the domain [18] [8].

In this paper we propose **ACME** - a new Associative Classifier based on Maximum Entropy. The maximum entropy principle is well-accepted in the statistics community. It states that given a collection of known facts about a probability distribution, choose a model for this distribution that is consistent with all the facts but otherwise is as uniform as possible. Hence, the chosen model does not assume any independence between its parameters that is not reflected in the given facts. In this paper, we use the *Generalized Iterative Scaling (GIS)* algorithm to compute the maximum entropy model.

Our main contributions in this work are as follows:

1. We develop ACME, a new associative classifier.
2. We show that the classical GIS algorithm fails in some cases when itemsets are used as features. In particular it fails in the presence of itemsets that are not *closed*<sup>1</sup>.
3. We provide a modification to GIS to work around the above-mentioned problem when itemsets are used as features. The modified GIS is faster than the original GIS.
4. We describe techniques to make GIS tractable for large feature spaces – we provide a new technique to efficiently divide a feature space into independent clusters each of which can be handled separately.

The rest of this paper is organized as follows: Section 2 formally introduces the classification problem and in Section 3, we describe the overall design of the proposed ACME classifier. In Section 3.1, we describe the classical GIS algorithm that computes the max-entropy distribution. Section 4 describes the drawbacks of the classical Maximum Entropy model and in Section 5, we present the new Maximum Entropy model. In Section 5.1, we adapt the GIS algorithm to converge in the presence of itemsets that are not *closed*. In Section 6, we describe techniques to make GIS tractable for large feature spaces and describe related work in Section 7. In Section 8, we experimentally evaluate the accuracy of ACME and finally conclude our study in Section 9.

## 2 Problem Definition

Let  $I = \{i_1, i_2, \dots, i_n\}$  be the set of items that can appear in a transaction and  $L = \{c_1, c_2, \dots, c_l\}$  be a set of  $l$  classes. A transaction containing items  $\{i_{k_1}, i_{k_2}, \dots, i_{k_j}\}$  is denoted by  $x_r$  where  $r$  is a *whole number* whose  $k_1, k_2, \dots, k_j$  bits are set to 1 and the remaining bits are set to 0. Hence the set  $X = \{x_0, x_1, \dots, x_{2^n-1}\}$  represents the set of all possible transactions. The training dataset  $D$  is a set of transactions, each of which is labeled with one of the  $m$  classes.

Given a transaction  $x$  to classify (i.e.  $x$  is a *query*), we label it with class  $c_i$  for which the posterior probability  $p(c_i|x)$  is maximum. *Bayes formula* allows us to compute this probability from the prior probability  $p(c_i)$  and the class-conditioned probability  $p(x|c_i)$  as follows

$$p(c_i|x) = \frac{p(x|c_i) * p(c_i)}{p(x)}$$

---

<sup>1</sup> An itemset is not closed iff it has the same frequency as one of its supersets.

Since the denominator  $p(x)$  is common for all the classes, it is ignored.  $p(c_i)$  is the relative frequency of class  $c_i$  in  $D$ , which is trivial to calculate. Hence, the classification problem is translated to the correct estimation of  $p(x|c_i)$ , given the dataset  $D$ .

We can calculate  $p(x|c_i)$  by directly measuring the frequency of  $x$  in the transactions in  $D$  which belong to the class  $c_i$ . But, because most training datasets are not large enough, this method would not be accurate if  $x$  is infrequent or non-existent in the dataset.

### 3 The ACME Classifier

The ACME classifier estimates  $p(x|c_i)$  in the following way. Let  $S = \{s_1, s_2, \dots, s_{|S|}\}$  be the union of the frequent itemsets extracted from each class  $c_i$ .

$$S = \{s_j \mid \exists c_i \text{ s.t. } P(s_j|c_i) \geq \sigma\} \quad (1)$$

$$\text{where } P(s_j|c_i) = \sum_{x \in X \wedge s_j \subset x} p(x|c_i)$$

$P(s_j|c_i)$  denotes the support of  $s_j$  in class  $c_i$  and  $\sigma$  is the user given support threshold. Each  $s_j \in S$  is called an *fset*. We use the itemsets in set  $S$  as parameters to model each class, such that each itemset  $s_j$  together with its support  $P(s_j|c_i)$  in class  $c_i$  forms a *constraint* that needs to be satisfied by the statistical model for that particular class. Thus, for each class  $c_i$  we have a set of constraints  $C_i = \{(s_j, P(s_j|c_i)) \mid s_j \in S\}$ . The probability distribution that we build for class  $c_i$  must satisfy these constraints. However, there could be multiple distributions satisfying these constraints. We follow the *maximum entropy principle* [10] and among these distributions select the one with the highest entropy. This is referred to as the Maximum Entropy model. It is unique and can be expressed in the following product form [22]:

$$p(x|c_i) = \pi \prod_{j=1}^{|C_i|} \mu_j^{f_j(x)} \quad (2)$$

$$\begin{aligned} \text{where } f_j(x) &= 1 \quad \text{if } s_j \subseteq x \\ &= 0 \quad \text{otherwise} \end{aligned}$$

In Equation 2,  $\pi$  is a normalization constant which ensures  $\sum_{x \in X} p(x|c_i) = 1$ . There are well-defined iterative algorithms to compute  $\mu_j$ 's. In this paper we use the Generalized Iterative Scaling (GIS) algorithm [6] described in Section 3.1. The model given in Equation 2 is referred to as the classical Maximum Entropy model.

Maximum entropy modeling does not differentiate between a class-variable and a normal variable. Infact, the model given in Equation 2 is called as conditional maximum entropy model as it models the domain for each class-variable. For brevity, we refer to it as "maximum entropy model". In the ensuing discussion we drop the *conditional* in  $P(s_j|c_i)$  and write it as  $P(s_j)$  and likewise wherever required. The *conditional* is implicitly assumed.



In ACME, the training phase would involve finding the set of constraints  $S$  and computing  $\mu$  values for all the classes. The computed  $\mu$  values for each class are stored and are used in the actual classification phase. The procedure to classify a given transaction  $x$  is to first extract all the itemsets in  $S$  that are subsets of  $x$ . These are the *features* of  $x$ . Then compute Equation 2 for each class and select that class which maximizes this equation.

### 3.1 Computing Parameters of the Maximum Entropy Model

The  $\mu_j$ 's in Equation 2 are estimated by a procedure called the Generalized Iterative Scaling (GIS) [22]. This is an iterative method that improves the estimation of the parameters with each iteration. The algorithm stops when there is no significant change in the  $\mu_j$  values. This solution is globally optimal as it has been proved that the search space of  $\mu_j$ 's over which we are searching for the final solution is concave leading to the conclusion that every locally optimal solution is globally optimal [6]. In this section, we present the GIS algorithm and discuss some issues about it.

The GIS algorithm runs with the constraint set  $C$  on the domain  $X$  computing a model which has the highest entropy and satisfies all the constraints. We call  $X$  and  $C$ , the parameters of GIS. The GIS algorithm first initializes all the  $\mu_j$ 's to 1, and executes the following procedure until convergence:

$$\mu_j^{(n+1)} = \mu_j^{(n)} \left[ \frac{P(s_j)}{P^{(n)}(s_j)} \right] \quad (3)$$

where

$$P^{(n)}(s_j) = \sum_{x \in X} p^{(n)}(x) f_j(x)$$

$$p^{(n)}(x) = \pi \prod_{j=1}^{|C|} (\mu_j^{(n)})^{f_j(x)}.$$

The variable  $n$  in the above system of equations denotes the iteration number.  $P^{(n)}(s_j)$  is the expected support of  $s_j$  in the  $n$ 'th iteration while  $P(s_j)$  is the actual support of  $s_j$  calculated from the training dataset. Convergence is achieved when the expected and actual supports of every  $s_j$  are nearly equal.

Note that for the GIS to converge, minor modifications to the above formalism (in Equation 3) are required. Details of this are available in [21] and are not required for the discussion in this paper.

**Time Complexity.** Every time Equation 3 needs to be executed,  $P^{(n)}(s_j)$  is to be calculated from the distribution  $p^{(n)}$ . This step takes  $O(|X|)$ , and to execute it for all  $s_j$  it takes  $O(|X| * |C|)$ . If the algorithm requires  $m$  iterations for the distribution  $p$  to converge, the time complexity of GIS can be given as  $O(m * |X| * |C|)$ . Under practical circumstances, the number of iterations  $m$  is hard-coded and the algorithm is made to stop once it reaches those many iterations without waiting for the distribution to fully converge<sup>2</sup> [22].

<sup>2</sup> In our experiments, the GIS was run for 100 iterations.

One drawback in the above approach is that the probability model expressed in Equation 2 *fails* in some cases [22]. In particular, it fails in the presence of itemsets that are not *closed*. In Section 5.1 we discuss this drawback in detail, and in Section 5 we propose a solution to this problem.

Another drawback in the above approach is that it is not tractable when the set of items  $I$  is very large. This is because the complexity of the GIS algorithm is exponential w.r.t.  $|I|$ . We describe a technique to overcome this drawback in Section 6. It involves partitioning  $I$  into independent clusters of items so that GIS can be run on each cluster separately.

## 4 Failure of Approach Due to Non-closed Itemsets

In this section, we explain why the classical Maximum Entropy model as given by the product form in Equation 2 fails in some cases. An itemset is not *closed* iff it has the same frequency as one of its supersets. The formal definition of a *non-closed* itemset follows.

**Definition 1.** An itemset  $s_u \in S$  is non-closed iff

$$P(s_u) \neq 0 \text{ and}$$

$$\exists s_v \in S \text{ s.t. } s_u \subset s_v \wedge P(s_u) = P(s_v)$$

The presence of  $s_u$  in a transaction  $x$  means that  $s_v$  will also be present in  $x$ . We denote such a pair by  $[s_u, s_v]$ , a fully-confident itemset pair.  $\square$

A major disadvantage with the Maximum Entropy model is that Equation 2 does not have a solution when the system of constraints have *non-closed* itemsets in them. We prove this formally in Theorem 1.

**Theorem 1.** Equation 2 does not have a solution when the system of constraints have non-closed itemsets.

*Proof.* Let  $s_u \in S$  be a non-closed itemset. From Definition 1:

$$P(s_u) > 0 \tag{4}$$

Since  $s_u$  is a non-closed itemset,  $\exists s_v \in S \text{ s.t. } s_u \subset s_v \wedge P(s_u) = P(s_v)$  which means probability of any transaction that contains  $s_u$  but not  $s_v$  is 0. Let  $x$  be a transaction s.t. no other items except the ones in  $s_u$  are present in it. It means:

$$i_k \in s_u \Leftrightarrow i_k \in x$$

where  $i_k$  is an item in  $I$ . Since  $s_v$  is strictly a superset of  $s_u$ ,  $s_v \not\subseteq x$ . Hence  $p(x) = 0$ . From Equation 2:

$$p(x) = \pi \prod_{s_w \subseteq s_u} \mu_w = 0 \Rightarrow \because \pi \neq 0, \exists s_w \in S \wedge s_w \subseteq s_u \text{ s.t. } \mu_w = 0$$

This means  $\forall t \in X$  with  $s_u \subseteq t$ ,  $p(t) = 0$  as  $s_w$  will be present in the product form of  $p(t)$  since  $s_w \subseteq s_u$ . From the above statement,

$$P(s_u) = \sum_{s_u \subseteq x} p(x) = 0$$

which contradicts Equation 4.  $\therefore$  Equation 2 does not have a solution when the system of constraints have *non-closed* itemsets. ■

Hence, in cases when the system of constraints have non-closed constraints, the exact solution does not exist in the form of the probability model given in Equation 2 and the model parameters will not converge under the GIS algorithm. We propose a modification to the maximum entropy model enabling it to accommodate *non-closed* itemsets in its system of constraints and give a proof of convergence for this new model.

### 5 The Modified Maximum Entropy Model

The modified product form of the Maximum Entropy model is given as:

$$p(x) = 0 \quad \text{if } \exists [s_u, s_v] \text{ s.t. } s_u \subseteq x \wedge s_v \not\subseteq x$$

$$= \pi \prod_{i=1}^{|C'|} \mu_i^{f_i(x)} \quad \text{otherwise} \tag{5}$$

$C'$  in the above equation is the set of closed constraints in  $C$ . The non-closed constraints in  $C$  are only used to determine whether the probability of the transaction is 0 or not. When we find that the transaction has positive probability, its value  $p(x)$  is calculated using the closed constraints in  $C$ .

Let  $X' \subset X$  be the set of transactions for which  $p(x) > 0$ . We call such transactions as *closed* transactions. Transactions in the set  $\{X - X'\}$  are called *non-closed* transactions.

**Theorem 2.** Equation 5 refers to the Maximum Entropy model satisfying the given constraints inclusive of the closed-itemsets.

*Proof.* Same as that of the classical GIS given in [22], except that the transaction set would be  $X'$  instead of  $X$  and constraint set is  $C'$  instead of  $C$ . ■

#### 5.1 Computing Parameters for the Modified Maximum Entropy Model

Instead of the usual *parameters*  $X$  and  $C$ , the GIS algorithm is run with  $C'$  on the domain  $X'$  to build the new Maximum Entropy model. It first initializes all the  $\mu_j$ 's in  $C'$  to 1 and executes the GIS procedure until convergence:

$$\mu_j^{(n+1)} = \mu_j^{(n)} \left[ \frac{P(s_j)}{P^{(n)}(s_j)} \right] \quad \mu_j \in C' \tag{6}$$

where

$$\begin{aligned}
 P^{(n)}(s_j) &= \sum_{x' \in X'} p^{(n)}(x') f_j(x') \\
 p^{(n)}(x') &= \pi \prod_{j=1}^{|C'|} (\mu_j^{(n)}) f_j(x') \quad \forall x' \in X'
 \end{aligned}
 \tag{7}$$

**Theorem 3.** *The modified GIS as given in Equations 6 and 7 converges to the desired Maximum Entropy model.*

*Proof.* Same as that of the classical GIS given in [6], except that the transaction set would be  $X'$  instead of  $X$  and constraint set is  $C'$  instead of  $C$ . ■

**Time Complexity.** The time complexity of the modified GIS algorithm is  $O(m * |X'| * |C'|)$ . Notice that  $|C'|$  and  $|X'|$  will be smaller than or equal to  $|C|$  and  $|X|$  respectively. In cases where some *non-closed* itemsets are removed from  $C$ , the size of  $X'$  will be significantly smaller than the decrease in the size of  $C$ . This is because of the reason that for every *non-closed* itemset removed from  $C$ , an exponential number of transactions are removed from  $X$ . Hence, this new modified GIS algorithm not only computes the correct Maximum Entropy model but also has the advantage of running much faster than the classical GIS algorithm.

## 6 Improving the Execution Time of GIS

The running time complexity of GIS is  $O(m * |C| * 2^{|I|})$ , where  $m$  is the number of iterations required by the GIS for convergence.  $|C|$  is typically exponential in  $|I|$  as there is one constraint for every frequent itemset mined: The set of frequent itemsets are *generally* exponential. Hence, the two important parameters which influence the execution time of GIS are  $|C|$  and  $|I|$ .

In this section, we discuss methods to overcome the effects of these two parameter values. In Section 6.1 we provide a technique to prune the constraint set  $C$  and in Section 6.2 we describe a procedure to split  $I$  into smaller mutually-exclusive and collectively exhaustive sub-parts, each of which can be handled separately to produce the global distribution.

### 6.1 Pruning Constraints

Below, we define the term *confidence* of an itemset  $s_j$  for a class  $c_i$ .

**Definition 2.** *We call  $P(c_i|s_j)$  as the confidence of seeing class  $c_i$  in transactions containing the itemset  $s_j$ .  $P(c_i|s_j)$  (confidence of  $s_j$  in  $c_i$ ) is computed as  $P(c_i|s_j) = \frac{P(s_j \cup \{c_i\})}{P(s_j)}$  □*

As discussed earlier, the set of constraints are exponential in  $|I|$  and pruning them to a handful of *interesting* itemsets will improve the execution time of GIS. We give an interestingness measure that can be employed for the purpose of pruning.

- The *confidence*, defined in Definition 2, is considered a measure of interestingness as the higher the confidence value the more the association between  $s_j$  and  $c_i$ . We denote this interestingness measure by  $\mathcal{I}(s_j)$ .

$$\mathcal{I}(s_j) = \max_{c_i} P(c_i|s_j) \quad (8)$$

An itemset  $s_j$  is included in the set of constraints of all the classes, if there exists atleast one class  $c_i$  such that  $s_j$  has high confidence in  $c_i$  (i.e.  $P(c_i|s_j) > \text{minconf}$ , for a given minimum confidence threshold  $\text{minconf}$ ). Notice that an itemset  $s_j$  is either included as a constraint in all the classes or is not included in any of the classes. Since  $\sum_{c_i} P(c_i|s_j) = 1$ , a large value for a particular  $P(c_i|s_j)$  would imply that  $\forall c_k \neq c_i$ ,  $P(c_k|s_j)$  will be very less. If  $P(c_i|s_j) > \text{minconf}$ , it means that  $s_j$  is a good distinguishing factor between the classes. By including  $s_j$  in all the classes, we ensure that this information remains in the system of constraints.

Note that an effect of this pruning is that the new distribution that will be computed using the GIS is not the actual distribution of the data. However, it is a good representative in that the distributions computed for all classes are equally affected by this pruning. Hence, the outcome of the classifier is not changed.

## 6.2 Decomposing the Domain $I$

Using the interestingness measure  $\mathcal{I}$  for pruning, the effects of large  $|C|$  values on the running time of GIS can be decreased. However the algorithm still has to go through the entire possible transaction set  $X$  for every iteration. The transaction set  $X$  is the power-set of  $I$ , and hence it's size can be quite large for even modest values of  $|I|$ . To decrease the effects of  $|I|$  on the running time of GIS, we divide it into *clusters*  $\{I_1, I_2, \dots, I_k\}$  which are mutually exclusive and collectively exhaustive so that the GIS algorithm can be applied to each cluster and the final global distribution can be built by combining the outputs of GIS for these clusters.

Let  $C_i''$  denote the final set of constraints for class  $c_i$  obtained after pruning  $C_i$ . The division of the set of items  $I$  into  $\{I_1, I_2, \dots, I_k\}$  is in such a way that a constraint in  $C_i''$  with itemset  $s_j$  is fully contained in only one  $I_i$ . Such a division of  $I$  ensures that  $I_i$  and  $I_j$ , where  $i \neq j$ , will not have any item in common and  $\bigcup_i I_i = I$ . The items of  $I_i$  are said to be independent of items in  $I_j$ , for every  $i$  and  $j$ , since there exist no constraints which overlap with both  $I_i$  and  $I_j$ . If there were items in  $I_i$  which had dependency relationships with items in  $I_j$ , then there should be atleast one constraint containing these items which would have stayed through the pruning process to remain in the final set of constraints. The final global distribution over  $I$  can be obtained by combining the local distributions of  $\{I_1, I_2, \dots, I_k\}$  in a naive-bayes fashion.

In the above discussion,  $I_i$  and  $I_j$  will be independent of each other if there is no constraint which has an overlap with both  $I_i$  and  $I_j$ . Two cases can exist for such a constraint's absence:

1. The constraint was pruned away based on the interestingness measure  $\mathcal{I}$ .
2. The constraint was not frequent enough in any of the classes to enter the system of constraints.

If a constraint was not frequent enough, the statistical relationships it encodes are not strongly represented in the dataset. In such a case, we assume that these relationships are not important and the classifier will not lose much by pruning them away and using only those relationships which are strongly represented in the dataset. On the other hand, if the constraint was removed based on the interestingness measure  $\mathcal{I}$ , it means that the constraint was not *distinguishing* enough between the classes. This means, although the itemset was frequent enough in atleast one of the classes, it is not important for the task of classification.

## 7 Related Work

The surprising ability of naive-bayes (NB) at classification has spawned many extensions to it, most of which try to decrease the assumptions they make about the statistical characteristics of the features. TAN (Tree Augmented Naive Bayesian classifier) [4] learns a restricted tree-structured bayesian network taking into account only the most important correlations between pairs of attributes. The Selective Bayesian Classifier [12] deals with highly correlated features by incorporating only some attributes into the final decision process. The Semi-Naive Bayesian Classifier [11] iteratively joins pairs of attributes to relax the strongest independence assumptions.

The above methods try to decrease the strong independence assumptions made by NB. Two variables  $x, y$  are conditionally independent given  $z$  if for all values of  $x, y$  and  $z$ ,  $p(x|y, z) = p(x|z)$ . The conditional independence fails, even if a few instantiations of  $x, y$  and  $z$  don't satisfy this condition. If such a case occurs, the above algorithms would consider storing all elements of the joint probability distribution  $p(x, y, z)$  to obtain more accurate estimations. This method is error prone as enough data might not exist to provide reliable probability estimations for each  $p(x, y, z)$ . [2] introduced *context-specific independence*, which describes independence relations among variables that hold in certain contexts only. *Large Bayes* [18] considers only relationships between variable instantiations. Classification in Large Bayes is done by incrementally building a product approximation of  $p(x|c_i)$  with the available subsets of  $x$ . The approximation strategy judiciously uses subsets of  $x$  available at hand to decrease the independence assumptions it is making. However, all the above methods still assume independence between some pairs of attributes eventhough they are much lesser than NB. Our classifier, does not assume any relationships between items and uses the well accepted Maximum Entropy framework to effectively handle the independences inferred from sets of features. By representing relationships with itemsets, this classifier considers only independences between variable instantiations without trying to generalize.

Our work is somewhat similar to [16] [14] in that they use association rules to capture relationships. CBA [16] picks out the most *confident* rule from the mined rules for classification. Such single rule based classifiers are prone to unreliable estimates. CMAR [14] overcomes this problem by considering multiple rules at the time of classification. This method gives a formula to compare between groups of rules of each class, but a statistical justification for this formula is not provided.

The principle of Maximum Entropy has been widely used for a variety of natural language tasks including language modeling [23], text segmentation [1], part-of-speech

tagging [22] and prepositional phrase attachment [22]. Ours is the first approach, to our knowledge, which uses it for categorical data classification with mined frequent itemsets as constraints.

The paper [13] discussed the Maximum Entropy model's failure to accommodate *non-closed* constraints and proposed solutions to this problem. It applied Good-Turing discounting to the observed constraint frequencies and ran the classical GIS algorithm on these constraints. For example, a constraint's frequency will be changed to  $f_j - \epsilon$ , instead of its usual observed frequency  $f_j$  before including it for GIS algorithm. In this approach there is no guarantee that the constraints would remain consistent with each other after they have been changed. It also reports results on a *fuzzy maximum entropy framework* in which the objective is to maximize the sum of the entropy function and a penalty function which is heuristically selected to penalize deviations from unreliable (less frequent) constraints. Our solution to this problem, fixes the Maximum Entropy model in a simple fashion, without adding any additional complexity, such that the GIS algorithm will converge and will generate the *correct* maximum entropy distribution. Further, this new solution runs much faster than the classical GIS algorithm (see the Section 8).

## 8 Experiments

In this section, we show the evaluation of ACME classifier on 11 datasets chosen from the UCI Machine-Learning Repository [19]. We compare ACME with four other state-of-the-art classifiers: (1) the widely known and used decision tree classifier C4.5 [20] (2) a recently proposed associative classifier CBA [16] (3) the naive-bayes classifier and (4) the Tree-Augmented naive-bayes (TAN) classifier [4].

Continuous variables in the dataset were first discretized to interval attributes using entropy based discretization as given in [9]. The same discretized datasets were used for all the classification methods, except for C4.5 which accepts continuous valued variables directly. In all the experiments, accuracy is measured using the 10-fold cross validation. We implemented the classification methods naive-bayes and ACME in C++ and the classifiers TAN and C4.5 were borrowed from the *weka* machine learning software [24]. The results for the CBA classifier were borrowed from [14]. Experiments were run on an Intel Celeron dual-processor machine with two 3.0GHz processors and running RedHat Linux 9.0.

Figure 1 gives the characteristics of the ten datasets used in the evaluation of ACME. The column *fsets* in this figure, gives the size of the union of all the frequent itemsets extracted from every class. As given in Equation 1, this set is denoted as  $S$ . The next column "largest cluster ( $I_c$ )" gives the size of the largest cluster in all the classes. Column "closed trans." gives the percentage of transactions in the cluster  $I_c$  which had non-zero probability (see Section 5). Only these transactions were used in the learning phase, instead of the entire domain  $X$ . Apart from the transactions which are not *closed*, we also removed transactions which satisfy a constraint whose support was 0. The  $\mu$  values of such constraints were set to 0 and they were removed from the learning phase. The last column gives the number of constraints in the largest cluster ( $I_c$ ) among all the classes. If a dataset has multiple classes which contain the largest cluster, it is

Dataset	Attrs.	Classes	size	<i>fsets</i>	largest cluster( $I_c$ )	<i>closed</i> trans.	Cons. in $I_c$
Australian	14	2	690	354	17	10.1%	263
Breast	10	2	699	189	17	8.89%	180
Cleve*	13	2	303	246	15	9.96%	204
Diabetes	8	2	768	85	15	7.42%	61
German	20	2	1000	54	20	9.66%	44
Heart*	13	2	270	115	9	55.1%	95
Hepatitis	19	2	155	32.1k	17	1.3%	153
Lymph*	18	4	148	29	9	12.1%	14
Pima	8	2	768	87	15	8.6%	55
Waveform	21	3	300	99	18	1.3%	24

**Fig. 1.** Characteristics of the Datasets

marked with a ‘\*’. For these datasets, the last two columns give the average calculated over these clusters.

Looking at Figure 1 we can see that even if the decrease in the size of the set of constraints  $C$  is modest, the decrease in the size of  $X$  is significant. The decrease in the size of  $X$  is nearly 99% for datasets *Waveform* and *Hepatitis* as the decrease in the size of their constraint set  $C$  is very large. Since the complexity of GIS is proportional to  $|X|$  (calculated in Section 3.1), the new GIS algorithm will run much faster than the classical GIS algorithm in the presence of *non-closed* constraints.

If an attribute  $a$  is discretized into  $\{a_1, a_2, \dots, a_z\}$  binary variables, utmost one of these binary variables can be present in any transaction. This fact is not represented in the set of constraints for any class. We explicitly add constraints of the form  $(\{a_i, a_j\}, 0)$ , for every  $a_i$  and  $a_j$ , to the set of constraints for every class to ensure the presence of this information. Each constraint means that the probability of seeing  $a_i$  and  $a_j$  together in a transaction is 0. We call these constraints as *zero-constraints*. This implementation detail improved ACME’s accuracy. None of the classifiers that we know, including the ones with which ACME is compared in this section, had the provision to improve their accuracy by using the *zero-constraints*.

In our experiments, the parameters for all the classifiers were set to the standard values as reported in the literature. For CBA the minimum support was set to 0.01 (i.e. 1%) and the minimum confidence is set to 50% with the pruning option enabled. The minimum support threshold for ACME was set to 0.1 and the threshold confidence is set to 0.8. TAN classifier is run with default settings as used in the *weka* toolkit.

Figure 2 gives a detailed description of the datasets employed in the experiments and the accuracies of various classifiers. The winners for each dataset are highlighted in bold font. As can be seen from this table, ACME outperforms all the other classifiers. Out of the ten datasets employed for the testing, ACME wins in five cases. NB wins in 3 datasets and the remaining algorithms perform the best on one dataset each. However, ACME has an additional advantage of handling missing values. Even for other datasets in which ACME does not win, it comes close to the winner.

An important point that needs to be noted is, for the *Hepatitis* and *Waveform* datasets, eventhough the new modified GIS algorithm is made to run on only 1.3% of the entire possible domain, it is achieving significant results. ACME was the winner for



Dataset	NB	C4.5	CBA	TAN	ACME
Australian	84.34	<b>85.50</b>	84.9	84.9	85.36
Breast	<b>97.28</b>	95.13	96.3	96.56	96.49
Cleve	<b>83.82</b>	76.23	82.8	82.5	<b>83.82</b>
Diabetes	75.78	72.39	74.5	76.30	<b>77.86</b>
German	70.0	70.9	<b>73.4</b>	73.0	71.3
Heart	<b>83.70</b>	80.0	81.87	81.85	82.96
Hepatitis	80.22	81.93	81.8	81.29	<b>82.58</b>
Lymph	83.10	77.0	77.8	<b>84.45</b>	78.4
Pima	76.17	74.34	72.9	76.30	<b>77.89</b>
Waveform	80.82	76.44	80.0	81.52	<b>83.08</b>

**Fig. 2.** Accuracies of various Classifiers on UCI ML Datasets

these two datasets. The reason behind this is that the eliminated portion of the domain is actually set to the correct probability 0. This substantiates our earlier argument that not only ACME runs faster, but also that the distribution it generates will be more closer to the distribution we are trying to mimic. Another important point to note is that the *naive-bayes* classifier is performing better than the remaining classifiers.

## 9 Conclusions

In this paper we proposed a new classifier based on the paradigm of association rules. Though recent classifiers involving this paradigm have shown their techniques to be accurate, their approaches to build a classifier either assume not-observed statistical relationships in the dataset or have no statistical basis. Our classifier ACME uses the well-known Maximum Entropy principle to build a statistical model for the purpose of classification. We show that the classical Maximum Entropy model cannot have a solution for some input cases and propose a new Maximum Entropy model which can fit in all the input cases. We gave a modification to GIS, an algorithm to compute the classical Maximum Entropy model, to compute the new model. The modified GIS, apart from computing the correct Maximum Entropy model, also runs atleast as fast as the original GIS. We also described techniques to make GIS tractable for large feature spaces – we provided a new technique to divide a feature space into independent clusters each of which can be handled separately. Finally, our experimental results show that our classifier is generally more accurate than the existing classification methods.

## References

- [1] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210, 1999.
- [2] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian-networks. In *Uncertainty in Artificial Intelligence(UAI)*, 1996.
- [3] P. Clark and T.Niblett. The **cn2** induction algorithm. *Machine Learning*, 2:261–283, 1989.
- [4] P. Clark and T.Niblett. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

- [5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.
- [6] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.
- [7] G. Dong, X. Zhang, L. Wong, and J. Li. Classification by aggregating emerging patterns. In *Discovery Science*, Dec. 1999.
- [8] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.
- [9] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Intl. Joint Conf. on Artificial Intelligence(IJCAI)*, pages 1022–1029, 1993.
- [10] I. Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *Annals of Mathematical Statistics*, 34:911–934, 1963.
- [11] I. Kononenko. Semi-naive bayesian classifier. In *European Working Session on Learnign*, pages 206–219, 1991.
- [12] P. Langley and S. Sage. Induction of selective-bayesian classifiers. In *Uncertainty in Artificial Intelligence(UAI)*, pages 399–406, 1994.
- [13] R. Lau. Adaptive statistical language modeling. Master’s thesis, Massachusetts Institute of Technology, Cambridge, MA, 1994.
- [14] W. Li, J. Han, and J. Pei. **CMAR**: Accurate and efficient classification based on multiple class-association rules. In *ICDM*, 2001.
- [15] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, Sept. 2000.
- [16] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of 4th Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, Aug. 1998.
- [17] D. Meretakis, H. Lu, and B. Wuthrich. A study on the performance of large bayes classifier. In *ECML*, pages 271–279. LNAI, 2000.
- [18] D. Meretakis and B. Wuthrich. Extending naive-bayes classifiers using long itemsets. In *KDD*, pages 165–174, 1999.
- [19] C. Merz and P. Murphy. **UCI** repository of machine learning databases, 1996. <http://cs.uci.edu/mllearn/MLRepository.html>.
- [20] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [21] A. Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. Technical Report IRCS Report 97-98, Institute for Research in Cognitive Science, University of Pennsylvania, May 1997.
- [22] A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Institute for Research in Cognitive Science, University of Pennsylvania, 1998.
- [23] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, 1994.
- [24] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition, 2005.

# Constructing Multiclass Learners from Binary Learners: A Simple Black-Box Analysis of the Generalization Errors

Jittat Fakcharoenphol<sup>1</sup> and Boonserm Kijsirikul<sup>2</sup>

<sup>1</sup> Department of Computer Engineering,  
Kasetsart University, Bangkok, Thailand  
jtf@ku.ac.th

<sup>2</sup> Department of Computer Engineering,  
Chulalongkorn University, Bangkok, Thailand  
Boonserm.K@chula.ac.th

**Abstract.** Multiclass learning is widely solved by reducing to a set of binary problems. By considering base binary classifiers as black boxes, we analyze generalization errors of various constructions, including Max-Win, Decision Directed Acyclic Graphs, Adaptive Directed Acyclic Graphs, and the unifying approach based on coding matrix with Hamming decoding of Allwein, Schapire, and Singer, using only elementary probabilistic tools. Many of these bounds are new, some are much simpler than previously known. This technique also yields a simple proof of the equivalences of the learnability and polynomial-learnability of the multiclass problem and the induced pairwise problems.

## 1 Introduction

In many supervised machine learning problems, we want to classify a given data point into one of the possible classes, usually more than two classes. However, powerful algorithms available usually work only with two classes, e.g., AdaBoost [1], support-vector machine [2, 3].

There are quite a few techniques invented to remedy this problem. For problems with  $k$  classes, the first approach, called One-vs-All, trains  $k$  binary classifiers each distinguishing one of the classes from the others. On the other hand, in the One-vs-One approach,  $\binom{k}{2}$  classifiers are trained to distinguish one class from another. By making all-pair comparisons, a simple technique based on majority vote [4, 5] (denoted as Max-Win), in practice, gives a very reliable answer. It, however, needs  $O(k^2)$  applications of binary classifiers for the problem with  $k$  classes. The popular Decision Directed Acyclic Graphs [6] or the elimination method [7] reduces the number of comparisons down to  $O(k)$ . By reducing the depth of the path to  $O(\log k)$ ,<sup>1</sup> a recent approach called Adaptive Directed Acyclic Graphs [8] tries to improve the success probability while retaining a linear number of applications of the binary classifiers. Among these methods, only

---

<sup>1</sup> If not specified otherwise, all logarithms in this paper are base 2.

DDAG has a proof of the generalization error bound, based on perceptron decision trees. However, the proof is very involved, and works only for the specific class of base binary classifiers.

Another approach based on error-correcting codes was introduced by Dietterich and Bakiri [9]. In this framework, the classes are partitioned into opposing subsets using error-correcting codes. Specifically, a codeword of length  $l$  from the set  $\{+1, -1\}^l$  is assigned to each class, and  $l$  binary classifiers, each classifier for each bit of the codes, are trained. The classifier for the  $s$ -th bit has to distinguish between classes whose  $s$ -th bit of the codeword is  $+1$  and those whose bit is  $-1$ . The framework is further analyzed by Guruswami and Sahai [10]. Allwein, Schapire, and Singer [11] gave a generalized form of this coding method, where “don’t care bits” can be used, i.e., codewords are from  $\{+1, 0, -1\}^l$ . Their framework unifies the coding approach, the One-vs-All, and the One-vs-One methods. They analyzed the training errors and for classifiers that use AdaBoost as the binary learner, generalization errors. However, for the general case of the base binary learners, the generalization performance of the construction is unknown and is left as an open problem.

In this paper, we present a novel combinatorial proof technique that relies mainly on the structure of the constructions, not the base classifiers. The main observation is that in all aforementioned construction, each base binary classifier has a specific goal, i.e., to distinguish one set of classes from another, so its generalization error can be analyzed using well-known techniques for the family of classifiers. Then, we look at base classifiers as black-boxes, and use their performance guarantees to analyze the multiclass constructions.

With this view of the problem, we are able to derive generalization errors of various constructions, including Max-Win, Decision Directed Acyclic Graphs, Adaptive Directed Acyclic Graphs, and the unifying approach based on coding matrix with Hamming decoding of Alwein et al. Many of these bounds are new, some are much simpler than previously known, and the proofs use only elementary probabilistic tools. This technique also yields a simple proof of the equivalences of the learnability and polynomial-time learnability of the multiclass problem and the induced pairwise problems.

Later in this section, we review various multiclass learning algorithms. In Section 2, we describe the models of the learning problem, and give the definition of the induced binary concepts. We analyze learning algorithms in the later two sections: Section 3 gives the proofs for constructions based on pairwise learners; and Section 4 presents the analysis of the generalization errors for algorithms that use coding matrix. We discuss the equivalence of the pairwise (polynomial-time) learnability and multiclass (polynomial-time) learnability in Section 5. We conclude with open problems in Section 6.

## 1.1 Review of Multiclass Learning Algorithms

There are many methods for solving the multiclass learning problem. In this paper we focus on methods that construct multiclass classifiers using a set of binary classifiers, each of which distinguishes between classes. We start with

pairwise constructions, i.e., constructions that use a set of binary classifiers, each distinguishing one class from another class. Then, we review the error-correcting approach together with the coding matrix approach. There are other approaches, some also uses binary classifiers, (e.g., learning with constraints classification [12], and learning with equivalence constraints[13]), but they do not fit into our framework.

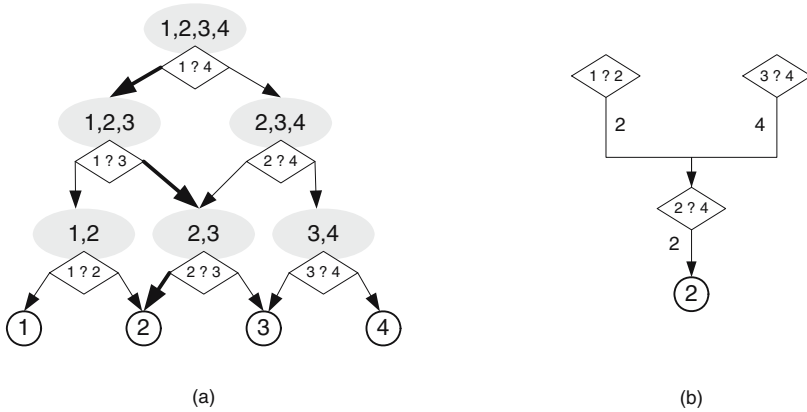
**Pairwise Constructions.** Base binary classifiers for this type of constructions are those that distinguish one class from another class. Formally, for each pair  $\{i, j\} \in [k] \times [k]$ , where  $i \neq j$ , we have a binary classifier  $A_{i,j} : X \rightarrow \{i, j\}$  that answers whether data point  $x \in X$  belongs to  $i$  or  $j$ . Ideally, if  $x$  belongs to class  $i$ ,  $A_{i,j}(x)$  should return  $i$ . We do not consider what  $A_{i,j}(x)$  would return when  $x$  does not belong in either  $i$  or  $j$ .

The algorithm  $\dots$  or  $\dots$  performs all  $\binom{k}{2}$  one-against-one classifications. For each class  $i$ , it picks all other  $k - 1$  classes, runs binary classifiers, and keeps track of the score  $s_i$  that  $i$  wins. It then outputs that  $x$  belongs to the highest score class. To classify a given data point,  $O(k^2)$  calls to base classifiers are needed.

Platt, Cristianini, and Shawe-Taylor [6] introduced a new learning architecture, called the  $\dots$  (or  $\dots$ ), which reduces the number of calls to binary classifiers down to  $O(k)$ . It starts with a set of candidate classes  $C = \{1, 2, \dots, k\}$ ; then, it proceeds in rounds, each eliminating one of the classes. For each round, let  $c_l = \min\{i : i \in C\}$  and  $c_r = \max\{i : i \in C\}$ . DDAG then calls  $A_{c_l, c_r}$  and eliminates the losing class from  $C$ . Therefore, to get the classification result, one needs  $k - 1$  rounds. DDAG is illustrated in Figure 1 (a). Fakcharoenphol [14] considered a randomized version of DDAG, where two candidate classes are chosen randomly for each round, and analyzed the performance using a simple model where errors occur independently.

Kijsirikul, Ussivakul, and Meknavin [8] observed that the structure of the elimination in DDAG introduces a long path of cascading calls to binary classifiers. They proposed a new structure called the  $\dots$  (or  $\dots$ ), whose number of cascading calls is reduced to only logarithmic. The algorithm works in  $\lceil \log k \rceil$  rounds, each eliminating half of the candidate classes. It starts with the same initial candidate set  $C = \{1, \dots, k\}$ . For each round, the algorithm pairs up  $|C|$  classes, and calls  $\lfloor |C|/2 \rfloor$  binary classifiers to eliminate losing classes. Kijsirikul et al. conducted experiments that show that ADAG performs better than DDAG. They also gave an analysis using a simple probabilistic model. Figure 1 (b) shows one possible execution of ADAG.

**Algorithms That Use Coding Matrix.** In this framework introduced by Dietterich and Bakiri[9], the classes are partitioned into opposing subsets using error-correcting codes. We discuss the more general version by Allwein, Schapire, and Singer [11]. Let  $l$  denote the length of the codeword. Each class  $i$  is assigned a codeword  $w_i$  from  $\{+1, 0, -1\}^l$ . The list of all of codewords forms the coding matrix  $\mathbf{M} \in \{+1, 0, -1\}^{k \times l}$ , with  $k$  rows and  $l$  columns. Let  $M(i, s)$  denote the  $s$ -th column of the  $i$ -th row, the  $s$ -th bit of the codeword for class  $i$ . For each



**Fig. 1.** DDAG and ADAG: (a) DDAG for 4 classes; bold arrows show one possible execution that predicts class 2; (b) One possible execution of ADAG that predicts class 2

column  $s$ , a binary classifier  $A_s$  is trained to distinguish classes whose  $s$ -th bit are  $+1$  and classes whose  $s$ -th bit are  $-1$ . Other classes where  $s$ -th bit are  $0$  are ignored. The trained classifier  $A_s$  outputs  $+1$  or  $-1$ .

Given a data point  $x$ , it is evaluated by  $A_s$  for each column  $s$ , producing an  $l$ -bit word  $w'$  in  $\{+1, -1\}^l$ . To determine which class  $x$  belongs to, a distance measure between  $w'$  and each codeword is used. The simplest one counts the number of different bits, but treating  $0$  as half the bit. Allwein et al. call this decoding  $\dots$ . Another, more refined, decoding method, called  $\dots$ , uses the confidence level returned by the classifiers. We focus only the Hamming decoding.

## 2 Learning Model

Given a set  $X$  of possible objects and a finite set  $Y = \{1, 2, \dots, k\}$  of labels, the  $\dots$  can be defined as follows. Given a set  $S$  of  $m$  examples drawn independently from an unknown distribution  $\mathcal{D}$  over  $X \times Y$ , find a hypothesis  $h : X \rightarrow Y$  from the hypothesis space  $\mathcal{H}$  that predicts the label for an unknown data  $x$ , drawn from the same distribution. There are two types of errors, a training error and a generalization error. The training error can be calculated as

$$\frac{1}{m} |\{(x, y) \in S : h(x) \neq y\}|,$$

while the generalization error is defined as

$$\Pr_{(x,y) \sim \mathcal{D}} [h(x) \neq y].$$

Note that the unknown distribution  $\mathcal{D}$  can be seen as an unknown multiclass concept; the marginal distribution of labels conditioned on the object  $x$  forms the labels of  $x$  in the concept.

### 2.1 Pairwise Concepts

Given the multiclass learning problem with  $k$  classes, one can define an induced pairwise concept for classes  $i$  and  $j$  as follows. Given a set  $S$  of  $m$  examples drawn independently from  $\mathcal{D}$  satisfying the condition that all the labels are either  $i$  or  $j$ , find a binary hypothesis  $h_{ij} : X \rightarrow \{i, j\}$  that distinguishes between objects from class  $i$  and class  $j$ . We want  $h_{ij}(x)$  to output  $i$  if it predicts that  $x$  is from class  $i$  and  $j$  if the data is from class  $j$ . The generalization error  $\epsilon_{ij}$  for  $h_{ij}$  is thus,

$$\Pr_{(x,y) \sim \mathcal{D}} [h_{ij}(x) \neq y \mid y \in \{i, j\}]$$

Again, one can see that, given an unknown multiclass concept  $\mathcal{D}$ , one can define induced pairwise concepts.

We call a set of  $\binom{k}{2}$  induced pairwise concepts the all-pair concept. We say that the all-pair concept is learnable (or, efficiently learnable), iff each of the  $\binom{k}{2}$  binary concept is learnable (or, efficiently learnable).

### 2.2 Binary Concepts from Coding Matrix

In the same way as for the induced pairwise concepts, given the multiclass learning problem with  $k$  classes and a coding matrix  $\mathbf{M}$ , with  $k$  rows and  $l$  columns, one can define an induced binary concept for column  $s$  as follows. Let  $C^+(s)$  be a set of classes  $i$  where  $M(i, s) = +1$ , and  $C^-(s)$  be a set of classes  $i$  where  $M(i, s) = -1$ . We call classes in  $C^+(s)$  positive classes, and classes in  $C^-(s)$  negative classes. Given a set  $S$  of  $m$  examples drawn independently from  $\mathcal{D}$  satisfying the condition that all the labels are either in  $C^+(s)$  or  $C^-(s)$ , find a binary hypothesis  $h_s : X \rightarrow \{+1, -1\}$  that distinguishes between objects from positive classes and negative classes. The generalization error  $\epsilon_s$  for  $h_s$  is thus,

$$\Pr_{(x,y) \sim \mathcal{D}} [h_s(x) \neq M(y, s) \mid y \in C^+(s) \cup C^-(s)].$$

## 3 Generalization Errors for Pair-Wise Constructions

All constructions we consider here give correct classification if all binary classifiers invoked during the process return correct predictions. We note one simple fact here. Since we know the generalization performance of each hypothesis, we can find the upperbound of the probability that none of the hypotheses make wrong predictions, i.e., it is at most the sum of the generalization errors of the hypotheses. This is clearly an application of union bound, which works for any set of random events.

Assume that we are given a set of hypotheses  $h_{ij}$  for  $i, j \in [k]$ , with generalization errors  $\epsilon_{ij}$ . In the following sections, we analyze the generalization error of the constructions which use pair-wise binary classifiers. To help understanding the bounds, we also consider the uniform error case where every binary hypothesis has generalization error of  $\bar{\epsilon}$ .

In what follows, for each algorithm, we mainly analyze the error  $\delta_i$  when the query point is from class  $i$ ; the generalization error is thus the error of the worst  $i$ . Frequently, we consider the other classes in order of their generalization errors. We shall use the following notation. For each  $i \in Y$  and  $1 \leq j < k$ , let  $r_i(j)$  denote the class having the  $j$ -th largest generalization error when compared with class  $i$ , i.e.,  $\epsilon_{i,r_i(1)} \geq \epsilon_{i,r_i(2)} \geq \dots \epsilon_{i,r_i(k-1)}$ . We break ties arbitrary.

### 3.1 Max-Win

Using the black-box approach, the worst-case bound on the generalization error of Max-Win can be proved using union bound.

**Theorem 1.**

$$\max_{i \in [k]} \sum_{j \neq i} \epsilon_{i,j} \leq (k-1)\bar{\epsilon}$$

Suppose that the input  $x$  is from class  $i$ . If all classifiers  $h_{ij}(x)$ , for all possible  $j$ , answer  $i$ , Max-Win would definitely return  $i$  as a prediction, because  $i$  would get  $k - 1$  votes while all other classes would get at most  $k - 2$  votes. The probability that  $h_{ij}(x)$  for a fixed  $j$  returns a wrong prediction is at most  $\epsilon_{ij}$ . Therefore, using union bound, the probability that some  $h_{ij}$  makes a wrong prediction is at most  $\sum_{j:j \neq i} \epsilon_{ij}$ . The theorem follows because we consider the worst case  $i$ .

We note that this bound is very loose. However, we believe that given only black-box information of the generalization error of the base classifiers, this bound is best possible. In practice, Max-Win performs very well. The reason might be from the gap between the score of the winning class and the second runner up. We note that if one looks into each binary classifier, thus ignoring the black-box approach, the margin analysis of the voting method in Schapire, Freund, Barlett, and Lee [15] directly applies to Max-Win. Also, following the same research direction, the result of Paugam-Moisy, Elisseeff, and Guermeur [16] uses the margin analysis to show the generalization error of the One-vs-All approach.

### 3.2 DDAG

For DDAG, we have  $\binom{k}{2}$  binary classifiers. If they all give correct predictions, the final prediction must be correct. This gives a bound of  $\sum_{j \neq i} \epsilon_{i,j}$  for the generalization error. However, we can do a lot better by noting that if the query is of class  $i$ , relevant classifiers are those concerning class  $i$ . Furthermore, these classifiers will not be called unless  $i$  is the candidate with minimum class id or the maximum id, i.e., when all the smaller-id classes or the larger-id classes get eliminated. In the worst case, this happens when either only larger-id classes are eliminated, or only smaller ones. Therefore, we get the following theorem.

**Theorem 2.**

$$\delta_i \stackrel{def}{=} \max\{\sum_{j < i} \epsilon_{i,j}, \sum_{j > i} \epsilon_{i,j}\} \leq \max_{i \in [k]} \delta_i \leq (k-1)\bar{\epsilon}$$



**Comparison to Previous Bounds.** Platt, Cristianini, and Shawe-Taylor [6] consider the Proceptron DDAG, i.e., DDAG with a perceptron at each node. Let  $\gamma_i$  denote the margin observed at node  $i$ . Using the technique of Bennett, Cristianini, Shawe-Taylor, and Wu [17], which is also based on that of [18], they show that with probability at least  $1 - \delta$  the generalization error of DDAG is at most

$$\frac{130R^2}{m} \left( D' \log(4em) \log(4m) + \log \frac{2(2m)^K}{\delta} \right),$$

where  $K$  is the number of nodes in DDAG,  $D' = \sum_{i=1}^K \frac{1}{\gamma_i^2}$ ,  $m$  is the number of samples used for training, and  $R$  is radius of the ball containing the distribution's support, for DDAG that correctly classifies  $m$  examples. They also consider the generalization error for class  $j$ . Let  $j$ -nodes denote the set of perceptron nodes involving class  $j$ . They get the bound of

$$\frac{130R^2}{m} \left( D' \log(4em) \log(4m) + \log \frac{2(2m)^{k-1}}{\delta} \right),$$

where  $D' = \sum_{i \in j\text{-nodes}} \frac{1}{\gamma_i^2}$ .

With previous analysis, we also get the same bound. For a perceptron which correctly classifies  $m$  examples with margin  $\gamma$  on the distribution whose support contains in a ball in  $\mathbb{R}^n$  centered at the origin of radius  $R$ , Shawe-Taylor, Barlett, Williamson, and Anthony [18] gives a bound of

$$\frac{2}{m} \left( l \log \left( \frac{8em}{l} \right) \log(32m) + \log \frac{8m}{\delta} \right),$$

where  $l = \lfloor 577R^2/\gamma^2 \rfloor$ . We plug this into Theorem 2. Recall that the bound in Theorem 2 uses only subsets of  $j$ -nodes; thus, it is at most

$$\begin{aligned} & \sum_{i \in j\text{-nodes}} \left[ \frac{2}{m} \left( \left( \frac{577R^2}{\gamma_i^2} \right) \log \left( \frac{8em}{(577R^2)/\gamma_i^2} \right) \log(32m) + \log \frac{8m}{\delta} \right) \right] \\ & \leq \left( \frac{1154R^2}{m} \right) \left( \sum_{i \in j\text{-nodes}} \left( \frac{1}{\gamma_i^2} \right) \right) \log(8em) \log(32m) + \frac{2}{m} \left( \sum_{i \in j\text{-nodes}} \log \frac{8m}{\delta} \right), \\ & \leq \left( \frac{1154R^2}{m} \right) [D' \log(8em) \log(32m)] + \left( \frac{2}{m} \right) (k-1) \log \frac{8m}{\delta}, \\ & = \left( \frac{1154R^2}{m} \right) [D' \log(8em) \log(32m)] + \left( \frac{2}{m} \right) \log \frac{(8m)^{k-1}}{\delta}, \end{aligned}$$

if  $1154R^2 \geq 2$ , and  $R \geq \gamma_i$  for all  $i$ . This bound is similar to the bound in [6] up to a small constant factor.

Our analysis can also be extended to include the case where there are training errors for perceptrons as well.

### 3.3 Improvements to DDAG: ADAG, Randomized-DDAG

**ADAG.** The solution of ADAG depends crucially on the order of elimination. However, since the path is short, we need only  $l = \lceil \log k \rceil$  applications of the binary classifiers. For query point in class  $i$ , we can assume that these  $l$  classifications are done with the worst  $l$  classes.

**Theorem 3.**  $\delta_i \stackrel{def}{=} \sum_{j=1}^l \epsilon_{i, r_i(j)} \max_i \delta_i = O(\log k) \bar{\epsilon}$

We note that the generalization error of ADAG seems to depend on the order of elimination. However, we are not sure if the worst-case behavior can be improved with randomization.

**Randomized-DDAG.** For DDAG, the following theorem shows that randomization does help.

**Theorem 4.**

$$\delta_i \stackrel{def}{=} \sum_{l=1}^{k-1} \left( \frac{2\epsilon_{i, r_i(k-l)}}{k-l+1} \right).$$

$\max_i \delta_i = O(\log k) \bar{\epsilon}$

We consider the process of the randomized DDAG in rounds; there are  $n - 1$  rounds. In each round, two candidate classes are chosen and a binary classifier that distinguishes between such classes is called. The winner stays on to the next round; the loser are eliminated. The first round is round 1. We analyze the probability that the correct class, say class  $i$ , gets eliminated during the process. Suppose that  $i$  survives after  $l$  rounds. On the one hand, if  $i$  is not chosen, it remains to round  $l + 1$  automatically; this happens with probability  $1 - \frac{2}{k-l}$ . On the other hand, if it is chosen, it remains in the next round only if the classifier predicts correctly. Let  $D_l$  denote the event that class  $i$  gets eliminated in round  $l$ , given that it survives up to round  $l - 1$ . Randomized-DDAG predicts incorrectly if one of these events occurs; thus, the probability that it gives incorrect prediction is  $\Pr[\bigcup_{l=1}^{k-1} D_l]$ .

For the analysis, we consider a different procedure. Suppose that  $i$  is known. In round  $l$ , we describe an equivalent procedure for choosing a pair of classes. If  $i$  has been eliminated in previous rounds, we pick a pair of classes randomly. Now if  $i$  remains one of the candidates, we decide whether class  $i$  will be evaluated by flipping a coin with  $2/(k - l + 1)$  head probability. If the coin comes up tail, we pick a pair of classes beside  $i$ , and in this case  $i$  remains to the next round with probability 1, and some class  $j$  get eliminated. If the coin comes up head, we choose another class randomly. Suppose that  $j$  is picked, the probability that

the classifier gives wrong answer is thus  $\epsilon_{ij}$ . For both cases, we call class  $j$  the pairing class of class  $i$ . The notion of pairing class is important to our analysis, and note that, by definition,  $i$  cannot be a pairing class.

To finish the analysis, we consider the choices for the pairing class to be the worst case. Suppose that the sequence  $\mathcal{S}$  of pairing classes over the execution of the algorithm is  $c_1, c_2, \dots, c_{k-1}$ . Thus, the probability that  $i$  gets eliminated in round  $l$  is  $\Pr[D_l | \mathcal{S}] = \frac{2\epsilon_{i, c_l}}{k-l+1}$ . Using the union bound, we have that the probability that randomized DDAG fails, given  $\mathcal{S}$ , is at most

$$\sum_{l=1}^{k-1} \frac{2\epsilon_{i, c_l}}{k-l+1} = \left( \frac{2\epsilon_{i, c_1}}{k} + \frac{2\epsilon_{i, c_2}}{k-1} + \dots + \frac{2\epsilon_{i, c_{k-2}}}{2} + \frac{2\epsilon_{i, c_{k-1}}}{1} \right)$$

This value is maximum when classes  $j$  in  $\mathcal{S}$  are in increasing order of  $\epsilon_{ij}$ . Thus, the randomized DDAG gives wrong prediction with probability at most

$$\delta_i = \sum_{l=1}^{k-1} \frac{2\epsilon_{i, r_i(k-l)}}{k-l+1}.$$

The theorem follows.

## 4 Generalization Errors for the Output Coding Approaches

We use the black-box approach to analyze the generalization error of the output coding method. The proof is very similar to that of [10], but instead of training error, we use generalization of the base classifier. In Subsection 4.1, we consider the simpler case where the coding matrix  $\mathbf{M}$  contains only +1 and -1. In the next subsection, we prove the theorem for the general matrix as in Allwein et al.

### 4.1 Coding Matrix Without Don't Care Bits

We note that the argument of Guruswami and Sahai [10], that bounds the training error, applies to the generalization error as well if we replace the averaging argument with Markov's inequality. We state and reprove their result here for completeness.

**Theorem 5 ([10]).** Let  $\mathbf{M} \in \{+1, -1\}^{k \times l}$  be a coding matrix with  $k$  columns and  $l$  rows. Let  $\epsilon_s$  be the fraction of rows that are wrong on column  $s$ . Then, the fraction of rows that are wrong on at least one column is at most  $\frac{2}{\Delta} \sum_{s=1}^l \epsilon_s$ .

For each column  $s$ , let a random variable  $S_s$  be 1 if the hypothesis for that column makes a wrong prediction, and 0 otherwise. Clearly  $\mathbf{E}[S_s] = \epsilon_s$ . Let random variable  $S$  be the number of columns having wrong predictions. By linearity of expectation,  $\mathbf{E}[S] = \mathbf{E}[\sum_{s=1}^l S_s] = \sum_{s=1}^l \mathbf{E}[S_s] = \sum_{s=1}^l \epsilon_s$ . Since

the coding matrix with Hamming distance  $\Delta$  can correct up to  $d \stackrel{def}{=} \lfloor (\Delta - 1)/2 \rfloor$  errors, if  $S \leq d$ , the algorithm would return the correct class. The case where the algorithm makes a wrong mistake is when  $S > d$ , which occurs with probability  $\Pr[S > d] \leq \Pr[S \geq \Delta/2] \leq \mathbf{E}[S]/(\Delta/2) = \frac{2}{\Delta} \sum_{s=1}^l \epsilon_s$ , by Markov's Inequality, as claimed.

### 4.2 General Matrix

We now consider the general matrix. As in Allwein et al., we assign each class  $r \in Y$  a row in the coding matrix  $\mathbf{M} \in \{-1, 0, +1\}^{k \times l}$ . However, the proof from the previous section does not follow unless we change the distance function  $\Delta(u, v)$  between two rows  $u, v$  of  $\mathbf{M}$ . Previously, the distance between any bit with “don't care” bit is  $1/2$ . This is too optimistic, when dealing with generalization error. Therefore, we ignore that “distance,” i.e., we define  $\Delta(u, v)$  to be

$$\Delta(u, v) = \sum_{s=1}^l \begin{cases} 1 & \text{if } u_s \neq v_s, u_s \neq 0, \text{ and } v_s \neq 0. \\ 0 & \text{otherwise} \end{cases}$$

We let  $\Delta \stackrel{def}{=} \Delta(M)$  be  $\min_{u, v \in M} \Delta(u, v)$ , the minimum distance between any pair of rows in  $M$ . Also, let  $I_i$  denote the set of indices  $s$  such that  $M(i, s) \neq 0$ . The following theorem states the generalization error bound for the combined classifier.

**Theorem 6.**  $\frac{2}{\Delta} \max_i \sum_{s \in I_i} \epsilon_s$

For each  $s \in I$ , let an indicator random variable  $E_s$  be one if  $C_s(x) \neq M(i, s)$ , and zero otherwise. Therefore,  $\Pr[E_s = 1] = \epsilon_s$ , by the definition of  $\epsilon_s$ . Let  $E$  be the number of error bits, i.e.,  $E = \sum_{s \in I} E_s$ . By linearity of expectation, we have  $\mathbf{E}[E] = \sum_{s \in I} \epsilon_s$ . Since the minimum hamming distance between any two rows is  $\Delta$ , if  $E < \Delta/2$ , the classifier would always predict class  $i$ , i.e., it makes no mistake. Using Markov's Inequality, we have

$$\Pr[\text{predict incorrectly}] \leq \Pr[E \geq \Delta/2] \leq \mathbf{E}[E]/(\Delta/2) = \frac{2}{\Delta} \sum_{s \in I} \epsilon_s;$$

thus, the theorem is proved.

We note that the generalization error of Max-Win (Section 3.1) also follows from this theorem, since in that case  $\Delta = 1$ .

## 5 Learnability

In this section, we discuss the equivalence of the learnability, and efficient learnability of the multiclass concepts and the induced all-pair concepts. One direction

of the equivalence, from multiclass to pair-wise, is simple. We focus on the other direction.

The equivalence of the learnability is implicit in the work of Ben-David, Cesa-Bianchi, Haussler, and Long [19], which proves various relation between many notions of dimensions, including the Natarajan dimension [20]. Consider a particular pairwise concept  $\mathcal{C}_{ij}$  induced by any classes  $i$  and  $j$ . If this concept is learnable, the VC-dimension [21] of  $\mathcal{C}_{ij}$  is finite [22]. Since this is true for any  $i$  and  $j$ , this means that the uniform Natarajan dimension of the concept class is finite; hence the Natarajan dimension is also finite, by Theorem 5 in [19]. This shows that the multiclass concept is learnable.

For efficient learnability, we note that if an all-pair concept is efficiently learnable, one can use Max-Win (or any other constructions discussed in this paper) to construct a polynomial-time learner for the multiclass concept. Specifically, if one needs a multiclass classifier with error probability  $\epsilon$ . We first find a binary classifier for each pair of classes with generalization error  $\epsilon/k^2$ . Theorem 1 ensures that the combined classifier has the generalization within the required bound. The running time for constructing hypothesis only increases by a polynomial factor of  $k$ . Thus, we get the desired reduction.

## 6 Discussions and Open Problems

By considering the base binary classifiers as black-boxes, we are able to analyze various algorithms for the multiclass classification problem. The analysis focuses mainly on the combinatorial structures of the constructions. The result in this paper contradicts previous beliefs that the order of the class evaluation in DDAG has no significant effect, and also it gives supportive argument for a new algorithm such as ADAG.

We list a few interesting open problems here.

1. Our analysis of the generalization error of Max-Win is very loose. We believe that the observed gap between the winning class and the second runner-up (as in [15]) can be used to give better bounds. However, the margin seems to be a property of the construction; therefore, we do not know how to put it into our framework. This might give an evidence of the limitation of the black-box analysis.
2. There are a few other algorithms which perform well in practice, but have no proof of the generalization performance, notably, an improvement to ADAG, called Reordering-ADAG by Phetkaew, Kijirikul, and Rivepiboon [23], that uses the error information to minimize the overall prediction error.
3. For the coding approach, we consider only Hamming decoding. It might be possible to extend our approach for the loss-based decoding.
4. It is interesting to see if one can combine an improvement to the proof of Allwein et al. by Klautau, Jevtić, and Orlitsky [24] with our technique.
5. One of the biggest open problems in multiclass learning is the question that asks if these new techniques devised to improve the simplest One-vs-All do really help [25]. The technique in this paper, however, offers no insight into this problem.

## Acknowledgement

We would like to thank Aharon Bar-Hillel and Danupon Nanongkai for thoughtful discussions. We also thank the anonymous referees for helpful comments.

## References

- [1] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **55** (1997) 119–139
- [2] Vapnik, V.: *Statistical Learning Theory*. Wiley (1998)
- [3] Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20** (1995) 273–297
- [4] Friedman, J.H.: Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University (1996)
- [5] Hastie, T., Tibshirani, R.: Classification by pairwise coupling. In: *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, Cambridge, MA, USA, MIT Press (1998) 507–513
- [6] Platt, J., Cristianini, N., Shawe-Taylor, J.: Large margin DAGs for multiclass classification. In: *Advance in Neural Information Processing System*. Volume 12., MIT Press (2000)
- [7] Kreßel, U.H.G.: Pairwise classification and support vector machines. In: *Advances in kernel methods: support vector learning*. MIT Press, Cambridge, MA, USA (1999) 255–268
- [8] Kijisirikul, B., Ussivakul, N., Meknavin, S.: Adaptive directed acyclic graphs for multiclass classification. In: *PRICAI 2002*. (2002) 158–168
- [9] Dietterich, T.G., Bakiri, G.: Error-correcting output codes: a general method for improving multiclass inductive learning programs. In Dean, T.L., McKeown, K., eds.: *Proceedings of the Ninth AAAI National Conference on Artificial Intelligence*, Menlo Park, CA, AAAI Press (1991) 572–577
- [10] Guruswami, V., Sahai, A.: Multiclass learning, boosting, and error-correcting codes. In: *Computational Learning Theory*. (1999) 145–155
- [11] Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Mach. Learn. Res.* **1** (2001) 113–141
- [12] Har-Peled, S., Roth, D., Zimak, D.: Constraint classification: A new approach to multiclass classification and ranking. In: *NIPS*. (2003)
- [13] Bar-Hillel, A., Weinshall, D.: Learning with equivalence constraints, and the relation to multiclass learning. In: *COLT*. (2003)
- [14] Fakcharoenphol, J.: A note on random DDAG. Manuscript (2003)
- [15] Schapire, R.E., Freund, Y., Bartlett, P.L., Lee, W.S.: Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics* **26** (1998) 1651–1686
- [16] Paugam-Moisy, H., Elisseeff, A., Guermeur, Y.: Generalization performance of multiclass discriminant models. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN 2000, Neural Computing: New Challenges and Perspectives for the New Millennium*, Como, Italy, July 24–27, 2000, Volume 4, IEEE (2000)
- [17] Bennett, K.P., Cristianini, N., Shawe-Taylor, J., Wu, D.: Enlarging the margins in perceptron decision trees. *Mach. Learn.* **41** (2000) 295–313
- [18] Shawe-Taylor, J., Bartlett, P.L., Williamson, R.C., Anthony, M.: A framework for structural risk minimisation. In: *COLT '96: Proceedings of the ninth annual conference on Computational learning theory*, ACM Press (1996) 68–76

- [19] Ben-David, S., Cesa-Bianchi, N., Haussler, D., Long, P.M.: Characterizations of learnability for classes of  $\{0, \dots, n\}$ -valued functions. *J. Comput. Syst. Sci.* **50** (1995) 74–86
- [20] Natarajan, B.K.: On learning sets and functions. *Mach. Learn.* **4** (1989) 67–97
- [21] Vapnik, V.N., Chervonenkis, A.Y.: On the uniform convergence of relative frequencies of events to their probabilities. *Theoret. Probi. and its Appl.* **16** (1971) 264–280
- [22] Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the vapnik-chervonenkis dimension. *J. ACM* **36** (1989) 929–965
- [23] Phetkaew, T., Kijirikul, B., Rivepiboon, W.: Reordering adaptive directed acyclic graphs for multiclass support vector machines. In: *Proceedings of the Third International Conference on Intelligent Technologies (InTech 2002)*. (2002)
- [24] Klautau, A., Jevtić, N., Orlitsky, A.: On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines. *J. Mach. Learn. Res.* **4** (2003) 1–15
- [25] Rifkin, R., Klautau, A.: In defense of one-vs-all classification. *J. Mach. Learn. Res.* **5** (2004) 101–141

# On Computability of Pattern Recognition Problems<sup>\*</sup>

Daniil Ryabko

IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland  
daniil@ryabko.net

**Abstract.** In statistical setting of the pattern recognition problem the number of examples required to approximate an unknown labelling function is linear in the VC dimension of the target learning class. In this work we consider the question whether such bounds exist if consider only computable pattern recognition methods, assuming that the unknown labelling function is also computable. We find that in this case the number of examples required for a computable method to approximate the labelling function not only is not linear, but grows faster (in the VC dimension of the class) than any computable function. No time or space constraints are put on the predictors or target functions; the only resource we consider is the training examples.

The task of pattern recognition is considered in conjunction with another learning problem — data compression. An impossibility result for the task of data compression allows us to estimate the sample complexity for pattern recognition.

## 1 Introduction

The task of pattern recognition consists in predicting an unknown label of some observation (or object). For instance, the object can be an image of a handwritten letter, in which case the label is the actual letter represented by this image. Other examples include DNA sequence identification, recognition of an illness based on a set of symptoms, speech recognition, and many others.

More formally, the objects are drawn independently from the object space  $X$  (usually  $X = [0, 1]^d$  or  $\mathbb{R}^d$ ) according to some unknown but fixed probability distribution  $P$  on  $X$ , and labels are defined according to some function  $\eta : X \rightarrow Y$ , where  $Y$  is a finite set (often  $Y = \{0, 1\}$ ). The task is to construct a function  $\varphi : \{0, 1\}^* \rightarrow Y$  which approximates  $\eta$ , i.e. for which  $P\{x : \eta(x) \neq \varphi(x)\}$  is small, where  $P$  and  $\eta$  are unknown but examples  $x_1, y_1, \dots, x_n, y_n$  are given;  $y_i := \eta(x_i)$ . In the framework of statistical learning theory [7],[8] it is assumed that the function  $\eta$  belongs to some known class of functions  $\mathcal{C}$ . Good error estimated can be obtained if the class  $\mathcal{C}$  is small enough. More formally, the number of examples required to obtain a certain level of accuracy (or the  $\epsilon$ -sample complexity of  $\mathcal{C}$ ) is linear in the VC-dimension of  $\mathcal{C}$ .

---

<sup>\*</sup> This work was supported by SNF grant 200020-107616.



In this work we investigate the question whether such bounds can be obtained if we consider only computable (on some Turing machine) pattern recognition methods. To make the problem more realistic, we also assume that the target function  $\eta$  is also computable. Both the predictors and the target functions are of the form  $\{0, 1\}^\infty \rightarrow \{0, 1\}$ .

We show that there are classes  $\mathcal{C}_k$  of functions for which the number of examples needed to approximate the pattern recognition problem to a certain accuracy grows faster in the VC dimension of the class than any computable function (rather than being linear as in the statistical setting). In particular this holds if  $\mathcal{C}_k$  is the class of all computable functions of length not greater than  $k$ .

Importantly, the same negative result holds even if we allow the data to be generated “actively”, e.g. by some algorithm, rather than just by some fixed probability distribution.

To obtain this negative result we consider the task of data compression: an impossibility result for the task of data compression allows us to estimate the sample complexity for pattern recognition. We also analyze how tight is the negative result, and show that for some simple computable rule (based on the nearest neighbour estimate) the sample complexity is finite in  $k$ , under different definitions of computational patterning recognition task.

In comparison to the vast literature on pattern recognition and related learning problems relatively little attention had been paid to the “computable” version of the task; at least this concerns the task of approximating any computable function. There is a track of research in which different concepts of computable learnability of functions on countable domains are studied, see [2]. A link between this framework and statistical learning theory is proposed in [5], where it is argued that for a uniform learnability finite VC dimension is required.

Another approach is to consider pattern recognition methods as functions computable in polynomial time, or under other resource constraints. This approach leads to many interesting results, but it usually considers more specified settings of a learning problem, such as learning DNFs, finite automata, etc. See [3] for an introduction to this theory and for references.

## 2 Preliminaries

A string  $x = x_1 \dots x_n$  is a member of the set  $\{0, 1\}^* = \cup_{i=0}^\infty \{0, 1\}^i$ . The length of a string  $x$  will be denoted by  $|x|$ , while  $x^i$  is the  $i$ th element of  $x$ ,  $1 \leq i \leq |x|$ . For a set  $A$  the symbol  $|A|$  is used for the number of elements in  $A$ . We will assume the lexicographical order on the set of strings, and when necessary will identify  $\{0, 1\}^*$  and  $\mathbb{N}$  via this ordering. Let  $\mathbb{N}$  be the sets of natural numbers. The symbol  $\log$  is used for  $\log_2$ . For a real number  $\alpha$  the symbol  $\lceil \alpha \rceil$  is the least natural number not smaller than  $\alpha$ .

In pattern recognition a labelling function is usually a function from the interval  $[0, 1]$  or  $[0, 1]^d$  (sometimes more general spaces are considered) to a finite space  $Y := \{0, 1\}$ . As we are interested in computable functions, we consider instead the functions of the form  $\{0, 1\}^\infty \rightarrow \{0, 1\}$ . Moreover, we call a partial

recursive function (or program)  $\eta$  accepts all strings from  $X_t := \{0, 1\}^t$  and only such strings <sup>1</sup>. For an introduction to the computability theory see for example [6].

It can be argued that this definition of a labelling function is too restrictive to approximate well the notion of a real function. However, as we are after negative results (for the class of all labelling functions), it is not a disadvantage. Other possible definitions are discussed in Section 4, where we are concerned with tightness of our negative results.

All computable function can be encoded (in a canonical way) and thus the set of computable functions can be effectively enumerated. Define the encoding of  $\eta$  as  $l(\eta) := |n|$  where  $n$  is the number of  $\eta$  in such enumeration.

Define the task of computational pattern recognition as follows. An (unknown) labelling function  $\eta$  is fixed. The objects  $x_1, \dots, x_n \in X$  are drawn according to some distribution  $P$  on  $X_{t(\eta)}$ . The labels  $y_i$  are defined according to  $\eta$ , that is  $y_i := \eta(x_i)$ .

A family of functions (indexed by  $n$ )

$$\varphi_n(x_1, y_1, \dots, x_n, y_n, x),$$

taking values in  $Y$ , such that for any  $n$  and any  $t \in \mathbb{N}$ , if  $x_i \in X_t$  for each  $i$ ,  $1 \leq i \leq n$ , then the marginal  $\varphi(x)$  is a total recursive function on  $X_t$  (that is,  $\varphi_n(x)$  accepts any  $x \in X_t$ ). We will often identify  $\varphi_n$  with its marginal  $\varphi_n(x)$  when the values of other variables are clear.

Thus, given a family of labelled objects of the same size  $t$ , a predictor produces a computable function; this function is supposed to approximate the labelling function  $\eta$  on  $X_t$ .

A family of predictors is a predictor which for any  $t \in \mathbb{N}$  and any  $n \in \mathbb{N}$  is a total recursive function on  $X_t \times Y \times \dots \times X_t \times Y \times X_t$

### 3 Main Results

We are interested in what size sample is required to approximate a labelling function  $\eta$ . Moreover, for a (computable) predictor  $\varphi$ , a labelling function  $\eta$  and  $0 < \varepsilon \in \mathbb{R}$  define

$$\delta_n(\varphi, \eta, \varepsilon) := \sup_{P_t} P_t \left\{ x_1, \dots, x_n \in X_t : P_t \{ x \in X_t : \varphi_n(x_1, y_1, \dots, x_n, y_n, x) \neq \eta(x) \} > \varepsilon \right\},$$

where  $t = t(\eta)$  and  $P_t$  ranges over all distributions on  $X_t$ . For  $\delta \in \mathbb{R}$ ,  $\delta > 0$  define the complexity of  $\eta$  with respect to  $\varphi$  as

$$N(\varphi, \eta, \delta, \varepsilon) := \min \{ n \in \mathbb{N} : \delta_n(\varphi, \eta, \varepsilon) \leq \delta \}.$$

---

<sup>1</sup> It is not essential for this definition that  $\eta$  is not a total function. An equivalent (for our purposes) definition would be as follows. A labelling function is any total function which outputs the string 00 on all inputs except on the strings of some length  $t =: t(\eta)$ , on each of which it outputs either 0 or 1.

The number  $N(\varphi, \eta, \delta, \varepsilon)$  is the minimal sample size required for a predictor  $\varphi$  to achieve  $\varepsilon$ -accuracy with probability  $1 - \delta$  when the (unknown) labelling function is  $\eta$ .

We can use statistical learning theory [7] to derive the following statement

**Proposition 1.**

$$N(\varphi, \eta, \delta, \varepsilon) \leq \max \left( l(\eta) \frac{8}{\varepsilon} \log \frac{13}{\varepsilon}, \frac{4}{\varepsilon} \log \frac{2}{\delta} \right)$$

where  $l(\eta)$  is the length of the shortest program  $\eta$  such that  $\eta(x_i) = y_i$  for all  $i \leq n$ ,  $\varepsilon, \delta > 0$

Observe that the bound is linear in the length of  $\eta$ .

In what follows the proof of this simple statement, we investigate the question of whether any such bounds exist if we restrict our attention to computable predictors.

The predictor  $\varphi$  is defined as follows. For each sample  $x_1, y_1, \dots, x_n, y_n$  it finds a shortest program  $\bar{\eta}$  such that  $\bar{\eta}(x_i) = y_i$  for all  $i \leq n$ . Clearly,  $l(\bar{\eta}) \leq l(\eta)$ . Observe that the VC-dimension of the class of all functions of length not greater than  $l(\eta)$  is bounded from above by  $l(\eta)$ , as there are not more than  $2^{l(\eta)}$  such functions. Moreover,  $\varphi$  minimises empirical risk over this class of functions. It remains to use the following bound (see e.g. [1], Corollary 12.4)

$$\sup_{\eta \in \mathcal{C}} N(\varphi, \eta, \delta, \varepsilon) \leq \max \left( V(\mathcal{C}) \frac{8}{\varepsilon} \log \frac{13}{\varepsilon}, \frac{4}{\varepsilon} \log \frac{2}{\delta} \right)$$

where  $V(\mathcal{C})$  is the VC-dimension of the class  $\mathcal{C}$ .

The main result of this work is that for any computable predictor  $\varphi$  there is no computable upper bound in terms of  $l(\eta)$  on the sample complexity of the function  $\eta$  with respect to  $\varphi$ :

**Theorem 1.**

Let  $\beta : \mathbb{N} \rightarrow \mathbb{N}$  be any computable function. Then for any computable predictor  $\varphi$  and any function  $\eta$  such that  $n > \beta(l(\eta))$ ,

$$P\{x \in X_{t(\eta)} : \varphi(x_1, y_1, \dots, x_n, y_n, x) \neq \eta(x)\} > 0.05,$$

where  $X_{t(\eta)} = \{x_1, \dots, x_n \in X_{t(\eta)} : y_i = \eta(x_i)\}$  and  $P$  is the probability measure over  $X_{t(\eta)}$ .

For example, we can take  $\beta(n) = 2^n$ , or  $2^{2^n}$ .

**Corollary 1.**

Let  $\beta : \mathbb{N} \rightarrow \mathbb{N}$  be any computable function. Then for any computable predictor  $\varphi$  and any function  $\eta$  such that  $\delta < 1$

$$\sup_{\eta: l(\eta) \leq k} N(\varphi, \eta, \delta, 0.05) > \beta(k)$$

for all  $k \in \mathbb{N}$ .

Observe that there is no  $\delta$  in the formulation of Theorem 1. Moreover, it is not important how the objects  $(x_1, \dots, x_n)$  are generated — it can be any individual sample. In fact, we can assume that the sample is chosen in any manner, for example by some algorithm. This means that no computable upper bound on sample complexity exists even for

It appears that the task of pattern recognition is closely related to another learning task — data compression. Moreover, to prove Theorem 1 we need a similar negative result for this task. Thus before proceeding with the proof of the theorem, we introduce the task of data compression and derive some negative results for it. We call a total recursive function  $\psi : \{0, 1\}^* \rightarrow \{0, 1\}^*$  an *injection* if it is an injection (i.e.  $x_1 \neq x_2$  implies  $\psi(x_1) \neq \psi(x_2)$ ). We say that a data compressor  $\zeta$  compresses the string  $x$  if  $|\psi(x)| < |x|$ . Clearly, for any natural  $n$  any data compressor compresses not more than a half of strings of size not greater than  $n$ .

We will now present a definition of Kolmogorov complexity; for fine details see [4], [9]. A *machine* is any total computable function. The complexity of a string  $x \in \{0, 1\}^*$  with respect to a machine  $\zeta$  is defined as

$$C_\zeta(x) = \min_p \{l(p) : \zeta(p) = x\},$$

where  $p$  ranges over all partial functions (minimum over empty set is defined as  $\infty$ ). There exists such a machine  $\zeta$  that  $C_\zeta(x) \leq C_{\zeta'}(x) + c_{\zeta'}$  for any  $x$  and any machine  $\zeta'$  (the constant  $c_{\zeta'}$  depends on  $\zeta'$  but not on  $x$ ). Fix any such  $\zeta$  and define the *complexity* of a string  $x \in \{0, 1\}^*$  as

$$C(x) := C_\zeta(x).$$

Clearly,  $C(x) \leq |x| + b$  for any  $x$  and for some  $b$  depending only on  $\zeta$ . A string is called *c-incompressible* if  $C(x) \geq |x| - c$ . Obviously, any data compressor can not compress many *c-incompressible* strings, for any  $c$ . However, highly compressible strings (that is, strings with Kolmogorov complexity low relatively to their length) might be expected to be compressed well by some sensible data compressor. The following lemma shows that it can not be always the case, no matter what we mean by “relatively low”.

The proof of this lemma is followed by the proof of Theorem 1.

**Lemma 1.** *There exists a machine  $\psi : \mathbb{N} \rightarrow \mathbb{N}$  such that for any function  $\gamma : \mathbb{N} \rightarrow \mathbb{N}$  monotonically increasing to infinity  $C(x) \leq \gamma(|x|) \implies |\psi(x)| \geq |x|$ .*

Suppose the contrary, i.e. that there exist an data compressor  $\psi$  and some function  $\gamma : \mathbb{N} \rightarrow \mathbb{N}$  monotonically increasing to infinity such that for any string  $x$  if  $C(x) \leq \gamma(|x|)$  then  $|\psi(x)| < |x|$ . Let  $T$  be the set of all strings which are not compressed by  $\psi$

$$T := \{x : |\psi(x)| \geq |x|\}.$$

Define the function  $\tau$  on the set  $T$  as follows:  $\tau(x)$  is the number of the element  $x$  in  $T$

$$\tau(x) := \#\{x' \in T : x' \leq x\}$$

for each  $x \in T$ . Obviously, the set  $T$  is infinite. Moreover,  $\tau(x) \leq x$  for any  $x \in T$  (recall that we identify  $\{0, 1\}^*$  and  $\mathbb{N}$  via lexicographical ordering). Observe that  $\tau$  is a total recursive function on  $T$  and onto  $\mathbb{N}$ . Thus  $\tau^{-1} : \mathbb{N} \rightarrow \{0, 1\}^*$  is a total recursive function on  $\mathbb{N}$ . Thus, for any  $x \in T$ ,

$$C(\tau(x)) \geq C(\tau^{-1}(\tau(x))) - c = C(x) - c > \gamma(|x|) - c, \tag{1}$$

for constant  $c$  depending only on  $\tau$ , where the first inequality follows from computability of  $\tau^{-1}$  and the last from the definition of  $T$ .

It is a well-known result (see e.g. [4], Theorem 2.3.1) that for any partial function  $\delta$  that goes monotonically to infinity there is  $x \in \{0, 1\}^*$  such that  $C(x) \leq \delta(|x|)$ . In particular, allowing  $\delta(|x|) = \gamma(|x|) - 2c$ , we conclude that there exist such  $x \in T$  that

$$C(\tau(x)) \leq \gamma(|\tau(x)|) - 2c \leq \gamma(|x|) - 2c,$$

which contradicts (1).

Suppose the contrary, that is that there exists such a computable predictor  $\varphi$  and a total function  $\beta : \mathbb{N} \rightarrow \mathbb{N}$  such that for any labelling function  $\eta$ , and any  $n > \beta(l(\eta))$  we have

$$P\{x : \varphi(x_1, y_1, \dots, x_n, y_n, x) \neq \eta(x)\} \leq 0.05,$$

for some  $x_i \in X_{t(\eta)}$ ,  $y_i = \eta(x_i)$ ,  $i \in \mathbb{N}$ , where  $P$  is the uniform distribution on  $X_{t(\eta)}$ .

Not restricting generality we can assume that  $\beta$  is strictly increasing. Define the (total) function  $\beta^{-1}(n) := \max\{m \in \mathbb{N} : \beta(m) \leq n\}$ . Define  $\varepsilon := 0.05$ . Construct the data compressor  $\psi$  as follows. For each  $y \in \{0, 1\}^*$  define  $m := |y|$ ,  $t := \lceil \log m \rceil$ . Generate (lexicographically) first  $m$  strings of length  $t$  and denote them by  $x_i$ ,  $1 \leq i \leq m$ . Define the labelling function  $\eta_y$  as follows:  $t(\eta_y) = t$  and  $\eta_y(x_i) = y^i$ ,  $1 \leq i \leq m$ . Clearly,  $C(\eta_y) \leq C(y) + c$ , where  $c$  is some universal constant capturing the above description.

Let  $n := \sqrt{m}$ . Next we run the predictor  $\varphi$  on all possible tuples  $\mathbf{x} = (x_1, \dots, x_n) \in X_t^n$  and each time count errors that  $\varphi$  makes on all elements of  $X_t$ :

$$E(\mathbf{x}) := \{x \in X_t : \varphi(x_1, y^1, \dots, x_n, y^n, x) \neq \eta_y(x)\}.$$

If  $|E(\mathbf{x})| > \varepsilon m$  for each  $\mathbf{x} \in X_t$  then  $\psi(y) := 0y$ .

Otherwise proceed as follows. Fix some tuple  $\mathbf{x} = (x'_1, \dots, x'_n)$  such that  $|E(\mathbf{x})| \leq \varepsilon m$ , and let  $H := \{x'_1, \dots, x'_n\}$  be the unordered tuple  $\mathbf{x}$ . Define

$$\kappa^i := \begin{cases} e_0 & x_i \in E(\mathbf{x}) \setminus H, y^i = 0 \\ e_1 & x_i \in E(\mathbf{x}) \setminus H, y^i = 1 \\ c_0 & x_i \in H, y^i = 0 \\ c_1 & x_i \in H, y^i = 1 \\ * & \text{otherwise} \end{cases}$$

for  $1 \leq i \leq m$ . Thus, each  $\kappa^i$  is a member of a five-letter alphabet (a five-element set)  $\{e_0, e_1, c_0, c_1, *\}$ . Denote the string  $\kappa^1 \dots \kappa^m$  by  $K$ .

Observe that the string  $K$ , the predictor  $\varphi$  and the order of  $(x'_1, \dots, x'_n)$  (which is not contained in  $K$ ) are sufficient to restore the string  $y$ . Furthermore, the  $n$ -tuple  $(x'_1, \dots, x'_n)$  can be obtained from  $H$  (the un-ordered tuple) by the appropriate permutation; let  $r$  be the number of this permutation in some fixed ordering of all  $n!$  such permutations. Using Stirling's formula, we have  $|r| \leq 2n \log n$ ; moreover, to encode  $r$  with some self-delimiting code we need not more than  $4n \log n$  symbols (for  $n > 3$ ). Denote such encoding of  $r$  by  $\rho$ .

Next, as there are  $(1 - \varepsilon - \frac{1}{\sqrt{m}})m$  symbols  $*$  in the  $m$ -element string  $K$ , it can be encoded by some simple binary code  $\sigma$  in such a way that

$$|\sigma(K)| \leq \frac{1}{2}m + 7(\varepsilon m + n). \tag{2}$$

Indeed, construct  $\sigma$  as follows. First replace all occurrences of the string  $**$  with 0. Encode the rest of the symbols with any fixed 4-bit encoding such that the code of each letter starts with 1. Clearly,  $\sigma(K)$  is uniquely decodable. Moreover, it is easy to check that (2) is satisfied, as there are not less than  $\frac{1}{2}(m - 2(\varepsilon m + n))$  occurrences of the string  $**$ . We also need to write  $m$  in a self-delimiting way (denote it by  $s$ ); clearly,  $|s| \leq 2 \log m$ .

Finally,  $\psi(\bar{y}) = 1\rho s\sigma(K)$  and  $|\psi(y)| \leq |\bar{y}|$ , for  $m > 2^{10}$ . Thus,  $\psi$  compresses any  $\bar{y}$  such that  $n > \beta(C(\eta_y))$ ; i.e. such that  $\sqrt{m} > \beta(C(\eta_y)) \geq \beta(C(y) + c)$ . This contradicts Lemma 1 with  $\gamma(k) := \beta^{-1}(\sqrt{k} - c)$ .  $\square$

### 4 On Tightness of the Negative Results

In this section we discuss how tight are the conditions of the statements and to what extent they depend on the definitions.

Let us consider a question whether there exist any (not necessarily computable) sample-complexity function

$$\mathcal{N}_\varphi(k, \delta, \varepsilon) := \sup_{\eta: l(\eta) \leq k} N(\varphi, \eta, \delta, \varepsilon),$$

at least for some predictor  $\varphi$ , or it is always infinity from some  $k$  on.

**Proposition 2.** *For any predictor  $\varphi$ ,  $\mathcal{N}_\varphi(k, \delta, \varepsilon) < \infty$ ,  $\varepsilon, \delta > 0$ ,  $k \in \mathbb{N}$*

Clearly,  $C(\eta) \geq C(t_\eta)$ . Moreover,  $\liminf_{t \rightarrow \infty} C(t) = \infty$  so that

$$\max\{t_\eta : l(\eta) \leq k\} < \infty$$

for any  $k$ . It follows that the ‘‘pointwise’’ predictor

$$\varphi(x_1, y_1, \dots, x_n, y_n, x) = \begin{cases} y_i & \text{if } x = x_i, 1 \leq i \leq n \\ 0 & x \notin \{x_1, \dots, x_n\} \end{cases} \tag{3}$$

satisfies the conditions of the proposition.

It can be argued that probably this statement is due to our definition of a labelling function. Next we will discuss some other variants of this definition.

First, observe that if we define a labelling function as any total function on  $\{0, 1\}^*$  then some labelling functions will not approximate any real function; for example such is the function  $\eta_+$  which counts bitwise sum of its input:  $\eta_+(x) := \sum_{i=1}^{|x|} x_i \pmod 2$ . That is why we require a labelling function to be defined only on  $X_t$  for some  $t$ .

Another way to define a labelling function (which perhaps makes labelling functions most close to real functions) is as a function which accepts any binary string. Let us call an any total recursive function  $\eta : \{0, 1\}^\infty \rightarrow \{0, 1\}$ . That is,  $\eta$  is computable on a Turing machine with an input tape on which one way infinite input is written, an output tape and possibly some working tapes. The program  $\eta$  is required to halt on any input. The next proposition shows that even if we consider such definition the situation does not change. The definition of a labelling function  $\eta$  in which it accepts only finite strings is chosen in order to stay within conventional computability theory.

**Lemma 2.**  $\eta$  is a labelling function if and only if  $n_\eta \in \mathbb{N}$  and  $\eta(x) = \eta(x')$  for any  $x, x' \in \{0, 1\}^*$  such that  $x_i = x'_i, i \leq n_\eta$ .

For any  $x \in \{0, 1\}^*$  the program  $\eta$  does not scan its tape further some position  $n(x)$  (otherwise  $\eta$  does not halt on  $x$ ). For any  $\chi \in \{0, 1\}^\infty$  denote by  $n_\eta(\chi)$  the maximal  $n \in \mathbb{N}$  such that  $\eta$  scans the input tape up to the position  $n$  on the input  $\chi$ .

Suppose that  $\sup_{\chi \in \{0, 1\}^\infty} n_\eta(\chi) = \infty$ , i.e. that the proposition is false. Define  $x^0$  to be the empty string. Furthermore, let

$$x^i = \begin{cases} 0 & \sup_{\chi \in \{0, 1\}^\infty} n_\eta(x^1, \dots, x^{i-1}\chi) = \infty \\ 1 & \text{otherwise} \end{cases}$$

By our assumption,  $x_i$  is defined for each  $i \in \mathbb{N}$ . Moreover, it easy to check that  $\eta$  never stops on the input string  $x_1x_2\dots$ .

Besides, it is easy to check that the number  $n_\eta$  is computable.

Finally, it can be easily verified that Proposition 2 holds true if we consider i-labelling functions instead of labelling functions, constructing the required predictor based on the nearest neighbour predictor.

**Proposition 3.**  $\sup_{k \in \mathbb{N}} i\mathcal{N}_\varphi(k, \delta, \varepsilon) < \infty$  for any  $\varepsilon, \delta > 0$ .

Indeed, it suffices to replace the “pointwise” predictor in the proof of Proposition 2 by the following predictor  $\varphi$ , which assigns to the object  $x$  the label of that object among  $x_1, \dots, x_n$  with whom  $x$  has longest mutual prefix:  $\varphi(x_1, y_1, \dots, x_n, y_n, x) := y_k$ , where

$$k := \operatorname{argmax}_{1 \leq m \leq n} \{\max\{i \in \mathbb{N} : x^1 \dots x^i = x_m^1 \dots x_m^i\}\};$$

to avoid infinite run in case of ties,  $\varphi$  considers only first (say)  $n$  digits of  $x_i$  and break ties in favour of the lowest index.

## References

- [1] L. Devroye, L. Györfi, G. Lugosi, A probabilistic theory of pattern recognition. New York: Springer, 1996.
- [2] S. Jain, D. Osherson, J. Royer, and A. Sharma. Systems That Learn: An Introduction to Learning Theory, 2nd edition. The MIT Press, Cambridge, MA, 1999.
- [3] M. Kearns and U. Vazirani An Introduction to Computational Learning Theory The MIT Press, Cambridge, Massachusetts, 1994.
- [4] M. Li, P. Vitányi. An introduction to Kolmogorov complexity and its applications. Second edition, Springer, 1997.
- [5] W. Menzel, F. Stephan. Inductive versus approximative learning. In: Perspectives of Adaptivity and learning, edited by R. Kuehn et al., pp. 187–209, Springer, 2003.
- [6] H. Rogers. Theory of recursive functions and effective computability, McGraw-Hill Book Company, 1967.
- [7] V. Vapnik, and A. Chervonenkis. Theory of Pattern Recognition. Nauka, Moscow, 1974
- [8] V. Vapnik, Statistical Learning Theory: New York etc.: John Wiley & Sons, Inc. 1998
- [9] A.K. Zvonkin and L.A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. Russian Math. Surveys, 25(6), pp 83–124, 1970.



# PAC-Learnability of Probabilistic Deterministic Finite State Automata in Terms of Variation Distance\*

Nick Palmer and Paul W. Goldberg

Dept. of Computer Science, University of Warwick,  
Coventry CV4 7AL, U.K.

{npalmer, pwg}@dcs.warwick.ac.uk

<http://www.dcs.warwick.ac.uk/research/acrg>

**Abstract.** We consider the problem of PAC-learning distributions over strings, represented by probabilistic deterministic finite automata (PDFAs). PDFAs are a probabilistic model for the generation of strings of symbols, that have been used in the context of speech and handwriting recognition, and bioinformatics. Recent work on learning PDFAs from random examples has used the KL-divergence as the error measure; here we use the variation distance. We build on recent work by Clark and Thollard, and show that the use of the variation distance allows simplifications to be made to the algorithms, and also a strengthening of the results; in particular that using the variation distance, we obtain polynomial sample size bounds that are independent of the expected length of strings.

## 1 Introduction

A probabilistic deterministic finite automaton (PDFA) is a deterministic finite automaton that has, for each state, a probability distribution over the transitions going out from that state. Thus, a PDFA defines a probability distribution over the set of strings over its alphabet. The topic of PAC-learning of PDFAs was introduced by Ron et al. [10], where they show how to PAC-learn PDFAs, and apply the algorithm to speech and handwriting recognition. Recently Clark and Thollard [3] presented an algorithm that PAC-learns general PDFAs, using the Kullback-Leibler divergence (see Cover and Thomas [4]) as the error measure (the distance between the true distribution defined by the target PDFA, and the hypothesis returned by the algorithm). The algorithm is polynomial in three parameters: the number of states, the “distinguishability” of states, and the expected length of strings generated from any state of the target PDFA.

---

\* This work was supported by EPSRC Grant GR/R86188/01. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

In this paper we study the same problem, using variation distance instead of Kullback-Leibler divergence. The general message of this paper is that this modification allows some strengthening and simplifications of the resulting algorithms. The main one is that – as conjectured in [3] – a polynomial bound on the sample-size requirement is obtained that does not depend on the length of strings generated by the automaton. We also have no need for a distinguished “final symbol” that must terminate all data strings, or a “ground state” in the automaton constructed by the algorithm.

The variation distance between probability distributions  $D$  and  $D'$  is the  $L_1$  distance; for a discrete domain  $X$ , it is  $L_1(D, D') = \sum_{x \in X} |D(x) - D'(x)|$ . KL divergence is in a strong sense a more “sensitive” measure than variation distance; this was pointed out in Kearns et al. [8], which introduced the general topic of PAC-learning probability distributions. In Cryan et al. [5] a smoothing technique is given for distributions over the boolean domain — an algorithm that PAC learns distributions using the variation distance can be converted to an algorithm that PAC learns using the KL-divergence. (Abe et al. [1] give a similar result in the context of learning p-concepts.) Over the domain  $\Sigma^*$  (strings of unrestricted length over alphabet  $\Sigma$ ) that technique does not apply, which is why we might expect stronger results as a result of switching to the variation distance.

In the context of pattern classification, the variation distance is useful in the following sense. Suppose that we seek to classify labelled data by fitting distributions to each label class, and using the Bayes classifier on the hypothesis distributions. (See [6] for a discussion of the motivation for this general approach.) We show in [9] that PAC learnability using the variation distance implies agnostic PAC classification. The corresponding result for KL-divergence is that the expected negative log-likelihood cost is close to optimum.

Our approach follows [3], in that we divide the algorithm into two parts. The first (Algorithm 1 of Figure 1) finds a DFA that represents the deterministic structure of the hypothesis, and the second (Algorithm 2 of Figure 2) finds estimates of the transition probabilities. Algorithm 1 constructs (with high probability) a DFA whose states and transitions are a subset of those of the target. Algorithm 2 learns the transition probabilities by following the paths of random strings through the DFA constructed by Algorithm 1. We take advantage of the fact that commonly-used transitions can be estimated more precisely.

## 2 Terms and Definitions

A probabilistic deterministic finite state automaton (PDFSA) stochastically generates strings of symbols. The automaton has a finite set of states - one of which is denoted as the initial state. The automaton generates a string by making transitions between states (starting at the initial state), each occurring with a constant probability specifically associated with that transition, and a symbol is output as a function of the transition. The automaton halts when the initial state is reached.

**Definition 1.** A Probabilistic Deterministic Finite State Automaton (PDFSA) is a tuple  $(Q, \Sigma, q_0, q_f, \tau, \gamma)$  where

- $Q$  is a finite set of states
- $\Sigma$  is a finite set of characters
- $q_0 \in Q$  is the initial state
- $q_f \notin Q$  is the final state
- $\tau : Q \times \Sigma \rightarrow Q \cup \{q_f\}$  is the transition function
- $\gamma : Q \times \Sigma \rightarrow [0, 1]$  is the output probability function

Where appropriate, we extend the use of  $\tau$  and  $\gamma$  to strings:

$$\begin{aligned} \tau(q, \sigma_1\sigma_2\dots\sigma_k) &= \tau(\tau(q, \sigma_1), \sigma_2\dots\sigma_k) \\ \gamma(q, \sigma_1\sigma_2\dots\sigma_k) &= \gamma(q, \sigma_1) \cdot \gamma(\tau(q, \sigma_1), \sigma_2\dots\sigma_k) \end{aligned}$$

We use the pair  $(q, \sigma)$  to denote the transition from state  $q \in Q$  labeled with character  $\sigma \in \Sigma$ . Note that  $\gamma(q, \sigma) = 0$  when  $\tau(q, \sigma)$  is undefined. It should also be noted that the output probabilities from each state sum to one:

$$\forall q \in Q : \sum_{\sigma \in \Sigma} \gamma(q, \sigma) = 1.$$

We assume that the final state can be reached from any state of the automaton, that is,  $\forall q \in Q, \exists s \in \Sigma^* : \tau(q, s) = q_f \wedge \gamma(q, s) > 0$ . It follows that the PDFSA  $A$  defines a probability distribution over all strings in  $\Sigma^*$ . Let  $D_A(s)$  denote the probability that  $A$  generates  $s \in \Sigma^*$ , so we have

$$D_A(s) = \gamma(q_0, s) \text{ for } s \text{ such that } \tau_A(q_0, s) = q_f.$$

We define  $D_A(q)$  to be the probability that a random string generated by  $A$  uses state  $q \in Q$ . Thus  $D_A(q)$  is the probability that  $s \sim D_A$  (i.e.  $s$  sampled from distribution  $D_A$ ) has a prefix  $p$  with  $\tau(q_0, p) = q$ . In a similar way,  $D_A(q, \sigma)$  is the probability that a random string generated by  $A$  uses transition  $(q, \sigma)$  — the probability that a random string  $s \sim D_A$  has a prefix  $p\sigma$  with  $\tau(q_0, p) = q$ .

Suppose  $D$  and  $D'$  are probability distributions over  $\Sigma^*$ . The variation ( $L_1$ ) distance between  $D$  and  $D'$  is  $L_1(D, D') = \sum_{s \in \Sigma^*} |D(s) - D'(s)|$ . A class  $\mathcal{C}$  of probability distributions is PAC-learnable by algorithm  $\mathcal{A}$  with respect to the variation distance if the following holds. Given parameters  $\epsilon > 0$ ,  $\delta > 0$ , and access to samples from  $D_A \in \mathcal{C}$ , using runtime and sample size polynomial in  $\epsilon^{-1}$  and  $\delta^{-1}$ ,  $\mathcal{A}$  should, with probability  $1 - \delta$ , output a distribution  $D_H$  with  $L_1(D_A, D_H) < \epsilon$ . If  $\mathcal{C}$  is described in terms of additional parameters that represent the complexity of  $D_A$ , then we require  $\mathcal{A}$  to be polynomial in these parameters as well as  $\epsilon^{-1}$  and  $\delta^{-1}$ .

### 3 Constructing the PDFSA

The algorithm is shown in Figure 1. We have the following parameters (in addition to the PAC parameters  $\epsilon$  and  $\delta$ ):

- $|\Sigma|$ : the alphabet size,
- $n$ : an upper bound on the number of states of the target automaton,
- $\mu$ : a lower bound on distinguishability, defined below.

In the context of learning using the KL-divergence, a simple class of PDFAs (see [3]) can be constructed to show that the parameters above are insufficient for PAC learnability in terms of just those parameters. In [3], parameter  $L$  is also used, denoting the expected length of strings.

We construct a digraph  $G = \langle V, E \rangle$  with labelled edges ( $V$  is a set of vertices and  $E \subseteq V \times \Sigma \times V$  is a set of edges). Each edge is labelled with a letter  $\sigma \in \Sigma$ . Note that due to the deterministic nature of the automaton, there can be at most one vertex  $v_q$  such that  $(v_p, \sigma, v_q) \in E$  for any  $v_p \in V$  and  $\sigma \in \Sigma$ .

From the target automaton  $A$  we generate a hypothesis automaton  $H$  using a variation on the method described by Clark and Thollard [3] utilising  $\hat{q}_{u, \sigma}$ , where the  $L_\infty$  norm between the suffix distributions of states is used to distinguish between them (as studied also in [7, 10]). We define a Candidate Node in the same way as [3]. Suppose  $G$  is a graph whose vertices correspond to a subset of the states of  $A$ . Initially  $G$  will have a single vertex corresponding to the initial state;  $G$  is then constructed in a greedy incremental fashion.

**Definition 2.** Let  $G = \langle V, E \rangle$  be a digraph with labelled edges. For a vertex  $u \in V$  and a letter  $\sigma \in \Sigma$ , let  $\hat{q}_{u, \sigma}$  denote the distribution over strings generated using state  $u$  as the initial state and  $\sigma$  as the first letter of the string. Let  $S_{u, \sigma}$  denote a multiset of strings sampled from  $\hat{q}_{u, \sigma}$ .

The  $L_\infty$ -norm is a measure of distance between a pair of distributions, defined as follows.

**Definition 3.**  $L_\infty(D, D') = \max_{s \in \Sigma^*} |D(s) - D'(s)|$

Let  $D_q(s)$  denote the distribution over strings generated using state  $q$  as the initial state, so that

$$D_q(s) = \gamma(q, s) \text{ for } s \text{ such that } \tau(q, s) = q_f.$$

As in [10, 3], we say that a pair of nodes  $(q_1, q_2)$  are  $\mu$ -distinguishable if  $L_\infty(D_{q_1}, D_{q_2}) = \max_{s \in \Sigma^*} |D_{q_1}(s) - D_{q_2}(s)| \geq \mu$ . We define as follows the  $\hat{L}_\infty$ -norm (an empirical version of the  $L_\infty$ -norm) with respect to multisets of strings  $S_{q_1}$  and  $S_{q_2}$ , where  $S_{q_1}$  and  $S_{q_2}$  have been respectively sampled from  $D_{q_1}$  and  $D_{q_2}$ .

**Definition 4.** Let  $q_1, q_2$  be nodes in a digraph  $G$ . Let  $S_{q_1}$  and  $S_{q_2}$  be multisets of strings sampled from  $D_{q_1}$  and  $D_{q_2}$ .

$$\hat{L}_\infty(D_{q_1}, D_{q_2}) = \max_{s \in \Sigma^*} \left( \left| \frac{|s \in S_{q_1}|}{|S_{q_1}|} - \frac{|s \in S_{q_2}|}{|S_{q_2}|} \right| \right)$$

where  $|S_{q_i}|$  is the number of strings in  $S_{q_i}$  and  $|s \in S_{q_i}|$  is the number of strings in  $S_{q_i}$  that are equal to  $s$ .

**Algorithm 1** *Construct Automaton.*

```

Hypothesis Graph  $G = \langle V, E \rangle = \langle \{q_0\}, \emptyset \rangle$ 
 $m_0 = \frac{n|\Sigma|}{\mu^2\delta'}$ 
 $N = \max\left(\frac{8n^2|\Sigma|^2}{\epsilon^2} \ln\left(\frac{2n^2|\Sigma|^2}{\delta'}\right), \frac{4m_0n|\Sigma|}{\epsilon}\right)$ 
complete = false
repeat
  % create candidate node for each undefined transition from each vertex in G
  for each vertex  $v \in V$ 
    for each symbol  $\sigma \in \Sigma$ , where  $\tau_G(v, \sigma)$  is undefined
      create a candidate node  $\hat{q}_{v, \sigma}$  with associated multiset  $S_{v, \sigma} = \emptyset$ 
    generate a sample  $S$  of  $N$  strings iid from  $D_A$ 
    for each string  $s \in S$ , where  $s = r\sigma't$  and  $\hat{q}_{\tau_G(q_0, r), \sigma'}$  is a candidate node
       $S_{\tau_G(q_0, r), \sigma'} \leftarrow S_{\tau_G(q_0, r), \sigma'} \cup \{t\}$ 
    identify candidate node  $\hat{q}_{u, \sigma''}$  with the largest multiset,  $S_{u, \sigma''}$ 
    if  $(|S_{u, \sigma''}| \geq m_0)$ 
      replace multiset  $S_{u, \sigma''}$  with  $m_0$  suffixes chosen iid from  $S_{u, \sigma''}$ 
      if  $(\exists v \in V : \hat{L}_\infty(D_{\hat{q}_{u, \sigma''}}, D_v) \leq \frac{\mu}{2})$  % candidate "looks like" existing node
        add edge  $(u, \sigma'', v)$  to  $E$ 
      else
        add node  $\hat{q}_{u, \sigma''}$  to  $V$ , with multiset  $S_{u, \sigma''}$ 
        add edge  $(u, \sigma'', \hat{q}_{u, \sigma''})$  to  $E$ 
    else
      complete = true
      delete all candidate nodes  $q \notin V$ 
until(complete)
return G

```

**Fig. 1.** Constructing the underlying graph

The algorithm uses two quantities,  $m_0$  and  $N$ .  $m_0$  is the number of suffixes required in the multiset of a candidate node for the node to be added as a state (or as a transition) to the hypothesis. It will be shown that  $m_0$  is a sufficiently large number to allow us to establish that the distribution over suffixes in the multiset that begin at state  $q$  is likely to approximate the true distribution  $D_q$  over suffixes at that state.  $N$  is the number of strings generated iid during each iteration of the algorithm. Polynomial expressions for  $m_0$  and  $N$  are given in Algorithm 1.

We show that the probability of Algorithm 1 failing to adequately learn the structure of the automaton is upper bounded by  $\delta'$ . In Section 5 we show that the transition probabilities are learnt by Algorithm 2 with a failure probability of at most  $\delta''$ . Overall, the probability of the algorithms failing to learn the target PDFa within a variation distance of  $\epsilon$  is at most  $\delta$ , for  $\delta = \delta' + \delta''$ .

Algorithm 1 differs from [3] as follows. We do not introduce a node to catch any undefined transitions in the hypothesis graph so as to give a probability greater than zero to the generation of any string. Instead, any state  $q$ , for which  $D_A(q) < \frac{\epsilon}{2n|\Sigma|}$  can be discarded - no corresponding node is formed in our hypothesis graph. There is only a small probability that a string is generated such that our hypothesis automaton rejects it (there is no corresponding path through the graph), which means that the contribution to the overall variation distance is very small.

### 4 Analysis of PDFA Construction Algorithm

**Theorem 1.** Let  $(q_1, q_2) \in A$  and  $(\hat{q}_1, \hat{q}_2) \in H$  such that  $L_\infty(D_{q_1}, D_{q_2}) > \mu$  and  $\hat{L}_\infty(D_{\hat{q}_1}, D_{\hat{q}_2}) > \frac{\mu}{2}$ . Let  $m_0 \geq \frac{n|\Sigma|}{\mu^2 \delta'}$ .

States  $q_1$  and  $q_2$  are distinguished if there exists a string  $s'$  such that:

$$\left| \frac{|s' \in S_{q_1}|}{|S_{q_1}|} - \frac{|s' \in S_{q_2}|}{|S_{q_2}|} \right| \geq \frac{\mu}{2}$$

Due to the assumption that  $L_\infty(D_{q_1}, D_{q_2}) > \mu$ , a string  $s''$  exists such that:

$$|D_{q_1}(s'') - D_{q_2}(s'')| > \mu$$

We give a sufficient sample size, such that the proportion of each string occurring in the sample is within  $\frac{\mu}{4}$  of the expected proportion (with probability at least  $1 - \frac{\delta'}{2n^2|\Sigma|^2 m_0}$ ). From Hoeffding's Inequality (see for example [2]) we obtain that, for state  $q$  and string  $s$ :

$$\Pr \left( \left| \left( \frac{|s \in \hat{S}_q|}{|\hat{S}_q|} \right) - D_q(s) \right| \geq \frac{\mu}{4} \right) \leq e^{-2m_0 \left(\frac{\mu}{4}\right)^2} \tag{1}$$

A value of  $m_0$  is chosen such that for sufficiently large  $n$ :

$$e^{-\frac{m_0 \mu^2}{8}} \leq \frac{\delta'}{2n^2|\Sigma|^2 m_0} \tag{2}$$

It can be verified that this is satisfied if we choose  $m_0 \geq \frac{n^2|\Sigma|^2}{\mu^2 \delta'}$ .

From Equation 2 it can be seen that (for some string  $s$  and some state  $q$ ):

$$\Pr \left( \left| \left( \frac{|s \in \hat{S}_q|}{|\hat{S}_q|} \right) - D_q(s) \right| \geq \frac{\mu}{4} \right) \leq e^{-2m_0 \left(\frac{\mu}{4}\right)^2} \leq \frac{\delta'}{2n^2|\Sigma|^2 m_0} \tag{3}$$

---

<sup>1</sup> At every iteration of the algorithm, a bijection  $\Phi$  exists between the states of  $H$  and candidate states, and a subset of the states of  $A$ , such that  $\tau_A(u, \sigma) = v \Leftrightarrow \tau_H(\Phi(u), \sigma) = \Phi(v)$ .

For state  $q$ , a multiset  $S_q$  is said to be  $\frac{\mu}{4}$ -representative of the true distribution with respect to  $q$ , if  $\forall s \in S_q : \left| \left( \frac{|s \in S_q|}{|S_q|} \right) - D_q(s) \right| \leq \frac{\mu}{4}$ . If two states  $q_1$  and  $q_2$  have representative multisets, then given that  $L_\infty(D_{q_1}, D_{q_2}) > \mu$ , it must be the case that for some string  $s''$ :

$$\left| \frac{|s'' \in S_{q_1}|}{|S_{q_1}|} - \frac{|s'' \in S_{q_2}|}{|S_{q_2}|} \right| \geq \frac{\mu}{2}$$

Each multiset is representative (given that it contains  $m_0$  suffixes) with probability at least  $1 - \frac{\delta'}{2n^2|\Sigma|^2}$ , due to a union bound. There are at most  $n|\Sigma|$  candidate nodes in total and candidate nodes are re-generated (as are their multisets) in each iteration of the algorithm (of which there are at most  $n|\Sigma|$ ). Therefore, the probability that a candidate node has a representative multiset at the point when it is converted to a node in the hypothesis graph (or found to be indistinct from another node in the graph) is at least  $1 - \left( \frac{\delta'}{2n^2|\Sigma|^2} \cdot n^2|\Sigma|^2 \right) = 1 - \frac{\delta'}{2}$ .  $\square$

**Proposition 1.** Let  $A$  be a DFA with  $q_0$  the start state. Let  $A'$  be a DFA with  $q_1$  the start state. Let  $S$  be a multiset of suffixes of length  $m_0$ . Let  $D_A$  and  $D_{A'}$  be the distributions of suffixes of length  $m_0$  in  $A$  and  $A'$  respectively. Let  $\tau(q, \sigma)$  be the state reached from  $q$  on input  $\sigma$ . Let  $s_1 \sigma s_2$  be a suffix of length  $m_0$  in  $S$ . Let  $\tau(q_0, s_1) = q_1$ .

$$\Pr \left( \left| \left( \frac{|S_{q,\sigma}(A')|}{|S|} \right) - E \left[ \frac{|S_{q,\sigma}(A')|}{|S|} \right] \right| \geq \frac{\epsilon}{8n|\Sigma|} \right) \leq \frac{\delta'}{2n^2|\Sigma|^2}.$$

From Hoeffding's Inequality it can be seen that

$$\Pr \left( \left| \left( \frac{|S_{q,\sigma}(A')|}{|S|} \right) - E \left[ \frac{|S_{q,\sigma}(A')|}{|S|} \right] \right| \geq \frac{\epsilon}{8n|\Sigma|} \right) \leq e^{-2|S| \left( \frac{\epsilon}{4n|\Sigma|} \right)^2} \tag{4}$$

We need  $|S|$  to satisfy  $e^{-\frac{|S|\epsilon^2}{8n^2|\Sigma|^2}} \leq \frac{\delta'}{2n^2|\Sigma|^2}$ . Equivalently,

$$\frac{8n^2|\Sigma|^2}{\epsilon^2} \ln \left( \frac{2n^2|\Sigma|^2}{\delta'} \right) \leq |S|.$$

So the sample size identified in the statement is indeed sufficiently large.

**Theorem 2.** Let  $T'$  be a set of states. Let  $A$  be a DFA with  $q_0$  the start state. Let  $Q'$  be a set of states. Let  $D_A(q, \sigma)$  be the distribution of suffixes of length  $m_0$  in  $A$  starting from state  $q$  on input  $\sigma$ . Let  $D_A(q)$  be the distribution of suffixes of length  $m_0$  in  $A$  starting from state  $q$ . Let  $H$  be the set of states  $q \notin Q'$  such that  $D_A(q) \geq \frac{\epsilon}{4n|\Sigma|}$ .

Theorem 1 shows that if a candidate node has a multiset containing at least  $m_0$  suffixes, then there is a probability of at least  $1 - \frac{\delta'}{2n^2|\Sigma|^2}$  that the

multiset is representative (as defined in the proof of Theorem 1). Furthermore, it shows that the probability of all candidate nodes having representative multisets (if the multisets contain at least  $m_0$  suffixes) is at least  $1 - \frac{\delta'}{2}$ , from which we can deduce that all candidate nodes can be correctly distinguished from any nodes<sup>2</sup> in the hypothesis automaton.

Proposition 1 shows that with a probability of at least  $1 - \frac{\delta'}{2n^2|\Sigma|^2}$ , the proportion of strings in a sample  $S$  (generated iid over  $D_A$ , and for  $|S| \geq \frac{8n^2|\Sigma|^2}{\epsilon^2} \ln\left(\frac{2n^2|\Sigma|^2}{\delta'}\right)$ ) reaching candidate node  $\hat{q}$  is within  $\frac{\epsilon}{8n|\Sigma|}$  of the expected proportion  $D_A(\hat{q})$ . This holds for each of the candidate nodes (of which there are at most  $n|\Sigma|$ ), in each iteration of the algorithm (of which there are at most  $n|\Sigma|$ ), with a probability of at least  $1 - \frac{\delta'}{2}$ .

If a candidate node (or a potential candidate node<sup>3</sup>)  $\hat{q}$ , for which  $D_A(\hat{q}) \geq \frac{\epsilon}{2n|\Sigma|}$ , is not included in  $H$ , then from the facts above it follows that at least  $\frac{\epsilon N}{4n|\Sigma|}$  strings in the sample are not accepted by the hypothesis graph. For each string not accepted by  $H$ , a suffix is added to the multiset of a candidate node, and there are at most  $n|\Sigma|$  such candidate nodes. From this it can be seen that some candidate node has a multiset containing at least  $\frac{\epsilon N}{4}$  suffixes. From the definition of  $N$ ,  $N \geq \frac{4m_0n|\Sigma|}{\epsilon}$ . Therefore, some multiset contains at least  $m_0n|\Sigma|$  suffixes, which must be at least as great as  $m_0$ . This means that as long as there exists some significant transition or state that has not been added to the hypothesis, some multiset must contain at least  $m_0$  suffixes, so the associated candidate node will be added to  $H$ , and the algorithm will not halt.

Therefore it has been shown that all candidate nodes which are significant enough to be required in the hypothesis automaton (at least a fraction  $\frac{\epsilon}{2n|\Sigma|}$  of the strings generated reach the node) are present with a probability of at least  $1 - \frac{\delta'}{2}$ , and that since all multisets contain  $m_0$  suffixes, the candidate nodes and hypothesis graph nodes are all correctly distinguished from each other (or combined as appropriate) with a probability of at least  $1 - \frac{\delta'}{2}$ . We conclude that with a probability of at least  $1 - \delta'$ , every transition  $(q, \sigma) \notin T'$  in target automaton  $A$  for which  $D_A(q, \sigma) \geq \frac{\epsilon}{2n|\Sigma|}$  and every state  $q \notin Q'$  in target automaton  $A$  for which  $D_A(q) \geq \frac{\epsilon}{2n|\Sigma|}$ , has a corresponding transition or state in hypothesis automaton  $H$ . □

## 5 Finding Transition Probabilities

The algorithm is shown in Figure 2. We can assume that we have at this stage found DFA  $H$ , whose graph is a subgraph of the graph of target PDFA  $A$ . Algorithm 2 finds estimates of the probabilities  $\gamma(q, \sigma)$  for each state  $q$  in  $H$ ,  $\sigma \in \Sigma$ .

<sup>2</sup> Note that due to the deterministic nature of the automaton, distinguishability of transitions is not an issue.

<sup>3</sup> A potential candidate node is any state or transition in the target automaton which has not yet been added to  $H$ , and is not currently represented by a candidate node.



If we generate a sample  $S$  from  $D_A$ , we can trace each  $s \in S$  through  $H$ , and each visit to a state  $q_H \in H$  provides an observation of the distribution over the transitions that leave the corresponding state  $q_A$  in  $A$ . For string  $s = \sigma_1\sigma_2 \dots \sigma_\ell$ , let  $q_i$  be the state reached by the prefix  $\sigma_1 \dots \sigma_{i-1}$ . The probability of  $s$  is  $D_A(s) = \prod_{i=0}^{\ell-1} \gamma(q_i, \sigma_{i+1})$ . Let  $n_{q,\sigma}(s)$  denote the number of times that string  $s$  uses transition  $(q, \sigma)$ , then

$$D_A(s) = \prod_{q,\sigma} \gamma(q, \sigma)^{n_{q,\sigma}(s)} \tag{5}$$

Let  $\hat{\gamma}(q, \sigma)$  denote the estimated probability that is given to transition  $(q, \sigma)$  in  $H$ . Provided  $H$  accepts  $s$ , the estimated probability of string  $s$  is given by

$$D_H(s) = \prod_{q,\sigma} \hat{\gamma}(q, \sigma)^{n_{q,\sigma}(s)} \tag{6}$$

We aim to ensure that with high probability, for  $s \sim D_A$ , if  $H$  accepts  $s$  then the ratio  $D_H(s)/D_A(s)$  is close to 1. This is motivated by the following.

**Observation 3.** For any  $\epsilon \in [0, 1]$ , if  $L_1(D_A, D_H) \leq \frac{1}{2}\epsilon$ , then for  $s \sim D_A$ ,  $D_H(s)/D_A(s) \in [1 - \frac{1}{4}\epsilon, 1 + \frac{1}{4}\epsilon]$ .

$$L_1(D_A, D_H) = \sum_{s \in \Sigma^*} |D_A(s) - D_H(s)|$$

Let  $X = \{s \in \Sigma^* : D_H(s)/D_A(s) \in [1 - \frac{1}{4}\epsilon, 1 + \frac{1}{4}\epsilon]\}$ . Then

$$L_1(D_A, D_H) = \sum_{s \in X} |D_A(s) - D_H(s)| + \sum_{s \in \Sigma^* \setminus X} |D_A(s) - D_H(s)|$$

The first term of the right-hand side is  $\sum_{s \in X} D_A(s)(1 - D_H(s)/D_A(s)) \leq \sum_{s \in X} D_A(s) \cdot (\frac{1}{4}\epsilon) \leq \frac{1}{4}\epsilon$ .

$D_A(X) \geq 1 - \frac{1}{4}\epsilon$  and  $D_H(X) \geq D_A(X) - \frac{1}{4}\epsilon$ , hence the second term in the right-hand side is at most  $\frac{1}{4}\epsilon$ .  $\square$

We have so far allowed the possibility that  $H$  may fail to accept up to a fraction  $\frac{1}{4}\epsilon$  of strings generated by  $D_A$ . Of the strings  $s$  that are accepted by  $H$ , we want to ensure that with high probability  $D_H(s)/D_A(s)$  is close to 1, to allow Observation 3 to be used.

Suppose that  $n_{q,\sigma}(s)$  is large, so that  $s$  uses transition  $(q, \sigma)$  a large number of times. In that case, errors in the estimate of transition probability  $\gamma(q, \sigma)$  can have a disproportionately large influence on the ratio  $D_H(s)/D_A(s)$ . What we show is that with high probability for random  $s \sim D_A$ , regardless of how many times transition  $(q, \sigma)$  typically gets used, the training sample contains a large enough subset of strings that use that transition more times than  $s$  does, so that  $\gamma(q, \sigma)$  is nevertheless known to a sufficiently high precision.

We say that  $s \in \Sigma^*$  is  $(q, \sigma)$ -good for some transition  $(q, \sigma)$ , if  $s$  satisfies:

$$\Pr_{s' \sim D_A} (n_{q,\sigma}(s') > n_{q,\sigma}(s)) \leq \frac{\epsilon}{4n|\Sigma|}$$

A  $(q, \sigma)$ -good string is one that is more useful than most in providing an estimate of  $\gamma(q, \sigma)$ .

**Observation 4.** For any transition  $(q, \sigma)$  and any sample  $S$  of size  $m \geq 1$  generated at random over  $D_A$ , if  $|S| \geq m \left( \frac{32n|\Sigma|}{\epsilon} \right) \ln \left( \frac{2n|\Sigma|}{\delta''} \right)$ , then with probability at least  $1 - \frac{\delta''}{2n|\Sigma|}$ , the number of  $(q, \sigma)$ -good strings in  $S$  is at least  $\frac{\epsilon}{8n|\Sigma|} |S|$ .

From the definition of  $(q, \sigma)$ -good, the probability that a string generated at random over  $D_A$  is  $(q, \sigma)$ -good for transition  $(q, \sigma)$ , is at least  $\frac{\epsilon}{4n|\Sigma|}$ .

Applying a standard Chernoff Bound (see e.g. [2], p360), for any transition  $(q, \sigma)$ , given sample  $S$ , with high probability the observed number of  $(q, \sigma)$ -good strings in  $S$  is at least half the expected number:

$$\Pr \left( |\{s \in S : s \text{ is } (q, \sigma)\text{-good}\}| < \frac{1}{2} \cdot \frac{\epsilon}{4n|\Sigma|} |S| \right) \leq \exp \left( -\frac{\frac{1}{4} \left( \frac{\epsilon}{4n|\Sigma|} \right) |S|}{2} \right) \tag{7}$$

We wish to bound this probability to be at most  $\frac{\delta''}{2n|\Sigma|}$ , so from Equation (7),

$$\exp \left( -\frac{\frac{1}{4} \left( \frac{\epsilon}{4n|\Sigma|} \right) |S|}{2} \right) \leq \frac{\delta''}{2n|\Sigma|}$$

$$|S| \geq \left( \frac{32n|\Sigma|}{\epsilon} \right) \ln \left( \frac{2n|\Sigma|}{\delta''} \right)$$

□

**Notation.** Suppose  $S_{q,\sigma}$  is as defined in Algorithm 2. Let  $M_{q,\sigma}$  be the largest number with the property that at least a fraction  $\frac{\epsilon}{8n|\Sigma|}$  of strings in  $S_{q,\sigma}$  use  $(q, \sigma)$  at least  $M_{q,\sigma}$  times.

**Observation 5.** For any transition  $(q, \sigma)$  and any sample  $S$  of size  $m = \left( \frac{2n|\Sigma|}{\delta''} \right) \left( \frac{32n|\Sigma|^2}{\epsilon} \right)^2$ , then with probability at least  $1 - \frac{\delta''}{2n|\Sigma|}$ , the number of strings in  $S$  that use  $(q, \sigma)$  at least  $M_{q,\sigma}$  times is at least  $\frac{\epsilon}{8n|\Sigma|} m$ .

$$\Pr_{s \sim D_A} (n_{q,\sigma}(s) > M_{q,\sigma}) \leq \frac{\epsilon}{4n|\Sigma|} \tag{8}$$

**Theorem 6.** Let  $H$  be a hypothesis and  $A$  be a target function. Let  $D_A$  and  $D_H$  be distributions over  $\Sigma^*$ . Let  $\hat{\gamma}(q, \sigma)$  be the estimated value of  $\gamma(q, \sigma)$  from a sample  $S$  of size  $m = \frac{1}{2\epsilon} \left( \frac{32n|\Sigma|}{\delta''} \right)^2 \ln \left( \frac{2n|\Sigma|}{\delta''} \right)$ . Then  $L_1(D_A, D_H) < \epsilon$  implies  $\frac{1}{2\epsilon} \left( \frac{32n|\Sigma|}{\delta''} \right)^2 \ln \left( \frac{2n|\Sigma|}{\delta''} \right) \hat{\gamma}(q, \sigma) < \frac{1}{2\epsilon} \left( \frac{32n|\Sigma|}{\delta''} \right)^2 \ln \left( \frac{2n|\Sigma|}{\delta''} \right) \gamma(q, \sigma) + \delta''$ .

**Algorithm 2** *Finding Transition Probabilities.*

Input: DFA  $H$ , a subgraph of  $A$ .

For each state  $q \in H$ ,  $\sigma \in \Sigma$ :

generate sample  $S_{q,\sigma}$  from  $D_A$ ;  $|S_{q,\sigma}| = \binom{2n|\Sigma|}{\delta''} \left(\frac{32n|\Sigma|^2}{\epsilon}\right)^2 \binom{32n|\Sigma|}{\epsilon} \ln\left(\frac{2n|\Sigma|}{\delta''}\right)$ ;  
repeat

for strings  $s \in S_{q,\sigma}$ , trace paths through  $H$ ;

Let  $N_{q,-\sigma}$  be random variable: number of observations of state  $q$  before we observe transition  $(q, \sigma)$  (include observations of  $q$  and  $(q, \sigma)$  in rejected strings).

until(all strings in  $S_{q,\sigma}$  have been traced)

Let  $\hat{N}_{q,-\sigma}$  be the mean of the observations of  $N_{q,-\sigma}$ ;

Let  $\hat{\gamma}(q, \sigma) = 1/\hat{N}_{q,-\sigma}$ .

Let  $n_{q,\sigma}$  be number of observations of  $(q, \sigma)$ ;

for all  $q$  let  $\sigma_{\min}(q) = \arg \min_{\sigma} (n_{q,\sigma})$ .

Adjust  $\hat{\gamma}(q, \sigma_{\min}(q))$  such that  $\sum_{\sigma} \hat{\gamma}(q, \sigma) = 1$ .

**Fig. 2.** Finding Transition Probabilities

**Comment.** It can be seen that the sample size used by Algorithm 2 is polynomial in the parameters of the problem. It is linear in  $\frac{1}{\delta''}$ ; we believe that a more refined analysis would yield a logarithmic bound, alternatively one could modify the algorithm to obtain a logarithmic bound. The general idea would be to make  $O(\log(\frac{1}{\delta''}))$  independent empirical estimates of each  $N_{q,-\sigma}$ , and take their median.

Recall Observation 5, that with probability  $1 - \frac{\delta''}{2n|\Sigma|}$ ,

$$\Pr_{s \sim D_A} (n_{q,\sigma}(s) > M_{q,\sigma}) \leq \frac{\epsilon}{4n|\Sigma|}$$

Using Observation 4, the sets  $S_{q,\sigma}$  are large enough to ensure that with probability  $1 - \frac{\delta''}{2n|\Sigma|}$ , there are  $M_{q,\sigma} \binom{2n|\Sigma|}{\delta''} \left(\frac{32n|\Sigma|^2}{\epsilon}\right)^2$  uses of transition  $(q, \sigma)$ . This is because at least  $\binom{2n|\Sigma|}{\delta''} \left(\frac{32n|\Sigma|^2}{\epsilon}\right)^2$  members of  $S_{q,\sigma}$  use  $(q, \sigma)$  at least  $M_{q,\sigma}$  times.

Consequently, (again with probability  $1 - \frac{\delta''}{2n|\Sigma|}$  over random choice of  $S_{q,\sigma}$ ) the set  $S_{q,\sigma}$  generates a sequence of independent observations of state  $q$ , which continues until  $M_{q,\sigma} \binom{2n|\Sigma|}{\delta''} \left(\frac{32n|\Sigma|^2}{\epsilon}\right)^2$  of them resulted in transition  $(q, \sigma)$ .

Let  $N_{q,-\sigma}$  denote the random variable which is the number of times  $q$  is observed before transition  $(q, \sigma)$  is taken. Each time state  $q$  is visited, the selection of the next transition is independent of previous history, so we obtain a sequence of independent observations of  $N_{q,-\sigma}$ . So, with probability  $1 - \frac{\delta''}{2n|\Sigma|}$ , the number of observations of  $N_{q,-\sigma}$  is at least  $M_{q,\sigma} \binom{2n|\Sigma|}{\delta''} \left(\frac{32n|\Sigma|^2}{\epsilon}\right)^2$ .

Recall Chebyshev’s inequality, that for random variable  $X$  with mean  $\mu$  and variance  $\sigma^2$ , for positive  $k$ ,

$$\Pr(|X - \mu| > k) \leq \frac{\sigma^2}{k^2}.$$

$N_{q,-\sigma}$  has a discrete exponential distribution with mean  $\gamma(q, \sigma)^{-1}$  and variance  $\leq \gamma(q, \sigma)^{-2}$ . Hence the empirical mean  $\hat{N}_{q,-\sigma}$  is a random variable with mean  $\gamma(q, \sigma)^{-1}$  and variance at most  $\gamma(q, \sigma)^{-2} (M_{q,\sigma})^{-1} (\frac{2n|\Sigma|}{\delta''})^{-1} (\frac{32n|\Sigma|^2}{\epsilon})^{-2}$ . Applying Chebyshev’s inequality with  $\hat{N}_{q,-\sigma}$  for  $X$ , and  $k = \gamma(q, \sigma)^{-1} (\frac{\epsilon}{32n|\Sigma|^2 \sqrt{M_{q,\sigma}}})$ , we have

$$\Pr\left(|\hat{N}_{q,-\sigma} - \gamma(q, \sigma)^{-1}| > \gamma(q, \sigma)^{-1} \left(\frac{\epsilon}{32n|\Sigma|^2 \sqrt{M_{q,\sigma}}}\right)\right) \leq \frac{\delta''}{2n|\Sigma|}.$$

Since  $\gamma(q, \sigma) = 1/E[N_{q,-\sigma}]$  and  $\hat{\gamma}(q, \sigma) = 1/\hat{N}_{q,-\sigma}$ ,

$$\Pr\left(|\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)| > 2\gamma(q, \sigma) \left(\frac{\epsilon}{32n|\Sigma|^2 \sqrt{M_{q,\sigma}}}\right)\right) \leq \frac{\delta''}{2n|\Sigma|}.$$

The rescaling at the end of Algorithm 2 loses a factor of  $|\Sigma|$  from the upper bound on  $|\gamma(q, \sigma) - \hat{\gamma}(q, \sigma)|$ . Overall, with high probability  $1 - \frac{\delta''}{2n|\Sigma|}$ ,

$$|\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)| \leq \left(\frac{\epsilon\gamma(q, \sigma)}{16n|\Sigma| \sqrt{M_{q,\sigma}}}\right) \tag{9}$$

For  $s \in \Sigma^*$  let  $n_q(s)$  denote the number of times the path of  $s$  passes through state  $q$ . By definition of  $M_{q,\sigma}$ , with high probability  $1 - \frac{\epsilon}{4n|\Sigma|}$  for  $s \sim D_A$ ,

$$M_{q,\sigma} > n_q(s) \cdot \gamma(q, \sigma). \tag{10}$$

For  $s \sim D_A$  we upper bound the expected log-likelihood ratio,

$$\log\left(\frac{D_H(s)}{D_A(s)}\right) = \sum_{i=1}^{|s|} \frac{\hat{\gamma}(q_i, \sigma_i)}{\gamma(q_i, \sigma_i)}$$

where  $\sigma_i$  is the  $i$ -th character of  $s$  and  $q_i$  is the state reached by the prefix of length  $i - 1$ .

Suppose  $A$  generates a prefix of  $s$  and reaches state  $q$ . Let random variable  $X_q$  be the contribution to  $\log(\frac{D_H(s)}{D_A(s)})$  when  $A$  generates the next character.

$$\begin{aligned} E[X_q] &= \sum_{\sigma} \gamma(q, \sigma) \log\left(\frac{\hat{\gamma}(q, \sigma)}{\gamma(q, \sigma)}\right) \\ &= \sum_{\sigma} \gamma(q, \sigma) [\log(\hat{\gamma}(q, \sigma)) - \log(\gamma(q, \sigma))] \end{aligned}$$

We claim that (with high probability  $1 - \frac{\delta''}{2n|\Sigma|}$ )

$$\log(\hat{\gamma}(q, \sigma)) - \log(\gamma(q, \sigma)) \leq |\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)| \frac{1}{\gamma(q, \sigma)} \cdot A_{q, \sigma} \quad (11)$$

for some  $A_{q, \sigma} \in [1 - \frac{\epsilon}{8n|\Sigma|\sqrt{M_{q, \sigma}}}, 1 + \frac{\epsilon}{8n|\Sigma|\sqrt{M_{q, \sigma}}}]$ . The claim follows from (9) and the inequality, for  $|\xi| < x$ , that  $\log(x + \xi) - \log(x) \leq \xi \cdot \frac{1}{x} (1 + \frac{2\xi}{x})$  (plug in  $\gamma(q, \sigma)$  for  $x$ ). Consequently,

$$\begin{aligned} E[X_q] &\leq \sum_{\sigma} \gamma(q, \sigma) \left( \frac{1}{\gamma(q, \sigma)} \right) A_{q, \sigma} [\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)] \\ &= \sum_{\sigma} A_{q, \sigma} [\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)] \\ &= \sum_{\sigma} [\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)] + \sum_{\sigma} B_{q, \sigma} [\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)] \end{aligned}$$

for some  $B_{q, \sigma} \in [-\frac{\epsilon}{8n|\Sigma|\sqrt{M_{q, \sigma}}}, \frac{\epsilon}{8n|\Sigma|\sqrt{M_{q, \sigma}}}]$ . The first term vanishes, so we have

$$\begin{aligned} E[X_q] &\leq \sum_{\sigma} B_{q, \sigma} [\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)] \\ &= \frac{\epsilon}{8n|\Sigma|} \sum_{\sigma} \left( \frac{1}{\sqrt{M_{q, \sigma}}} \right) [\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)] \\ &\leq \frac{\epsilon}{8n|\Sigma|} \sum_{\sigma} \frac{\gamma(q, \sigma)}{M_{q, \sigma}} \end{aligned}$$

using (9). For  $s \sim D_A$ , given values  $n_q(s)$ , the expected contribution to  $\log(\frac{D_A(s)}{D_A(s)})$  from all  $n_q(s)$  usages of state  $q$  is, using (10), at most

$$n_q(s) \frac{\epsilon}{8n|\Sigma|} \sum_{\sigma} \frac{1}{n_q(s)} = \frac{\epsilon n_q(s)}{8n|\Sigma|} |\Sigma| \frac{1}{n_q(s)} = \frac{\epsilon}{8n}$$

The total contribution from all  $n$  states  $q$ , each being used  $n_q(s)$  times is

$$\sum_q \frac{\epsilon}{8n} = \frac{\epsilon}{8}. \quad (12)$$

So the expected difference between the likelihood of string  $s$  using the  $\hat{\gamma}(q, \sigma)$  values in place of the  $\gamma(q, \sigma)$  values, is small. Using (11),

$$\begin{aligned} \text{Var}[X_q] &\leq \sum_{\sigma} \frac{A_{q, \sigma}^2}{\gamma(q, \sigma)} [\hat{\gamma}(q, \sigma) - \gamma(q, \sigma)]^2 \\ &\leq \sum_{\sigma} \frac{A_{q, \sigma}^2}{\gamma(q, \sigma)} \frac{\gamma(q, \sigma)^2}{M_{q, \sigma}} \left( \frac{\epsilon}{8n|\Sigma|} \right)^2 \end{aligned}$$

$$\begin{aligned} &\leq \left(\frac{\epsilon}{8n|\Sigma|}\right)^2 \sum_{\sigma} \frac{A_{q,\sigma}^2 \gamma(q, \sigma)}{M_{q,\sigma}} \\ &\leq \left(\frac{\epsilon}{8n|\Sigma|}\right)^2 \sum_{\sigma} \frac{2}{n_q(s)} \end{aligned}$$

Hence the variance of the total contribution to the error  $\log\left(\frac{D_H(s)}{D_A(s)}\right)$  from all  $n_q(s)$  uses of state  $q$ , is at most  $\left(\frac{\epsilon}{8n|\Sigma|}\right)^2$ . Using (12), with high probability for  $s \sim D_A$ , all the states contribute at most  $\frac{1}{8}\epsilon$  to  $\log\left(\frac{D_H(s)}{D_A(s)}\right)$ .

Finally, to use Observation 3, note that  $\frac{D_H(s)}{D_A(s)} \in [1 - \frac{1}{4}\epsilon, 1 + \frac{1}{4}\epsilon]$  follows from  $\log\left(\frac{D_H(s)}{D_A(s)}\right) \in [-\frac{1}{8}\epsilon, \frac{1}{8}\epsilon]$ . □

## References

- [1] N. Abe, J. Takeuchi and M. Warmuth. Polynomial Learnability of Stochastic Rules with respect to the KL-divergence and Quadratic Distance. *IEICE Trans. Inf. and Syst., Vol E84-D(3)* pp. 299-315 (2001).
- [2] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press (1999)
- [3] A. Clark and F. Thollard. PAC-learnability of Probabilistic Deterministic Finite State Automata. *Journal of Machine Learning Research* 5 pp. 473-497 (2004)
- [4] T.M. Cover and J.A. Thomas. *Elements of Information Theory* Wiley Series in Telecommunications. John Wiley & Sons (1991).
- [5] M. Cryan and L. A. Goldberg and P. W. Goldberg. Evolutionary Trees can be Learnt in Polynomial Time in the Two-State General Markov Model. *SIAM Journal on Computing* 31(2) pp. 375-397 (2001)
- [6] P.W. Goldberg When Can Two Unsupervised Learners Achieve PAC Separation? *Procs. of COLT/EUROCOLT, LNAI 2111*, pp. 303-319 (2001)
- [7] C. de la Higuera and J. Oncina. Learning Probabilistic Finite Automata. *tech. rept. EURISE, Université de Saint-Etienne and Departamento de Lenguajes y Sistemas Informaticos* (2002)
- [8] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire and L. Sellie. On the Learnability of Discrete Distributions. *Procs. of STOC*, pp. 273-282 (1994).
- [9] Nick Palmer and Paul. W. Goldberg. PAC Classification via PAC Estimates of Label Class Distributions. *Tech rept. 411, Dept. of Computer Science, University of Warwick* (2004)
- [10] D. Ron, Y. Singer and N. Tishby. On the Learnability and Usage of Acyclic Probabilistic Finite Automata. *Journal of Computer and System Sciences*, 56(2), pp. 133-152 (1998).

# Learnability of Probabilistic Automata via Oracles

Omri Guttman, S.V.N. Vishwanathan, and Robert C. Williamson

Statistical Machine Learning Program,  
National ICT Australia, RSISE, Australian National University,  
Canberra, ACT, Australia  
{Omri.Guttman, SVN.Vishwanathan, Bob.Williamson}@nicta.com.au

**Abstract.** Efficient learnability using the state merging algorithm is known for a subclass of probabilistic automata termed  $\mu$ -distinguishable. In this paper, we prove that state merging algorithms can be extended to efficiently learn a larger class of automata. In particular, we show learnability of a subclass which we call  $\mu_2$ -distinguishable. Using an analog of the Myhill-Nerode theorem for probabilistic automata, we analyze  $\mu$ -distinguishability and generalize it to  $\mu_p$ -distinguishability. By combining new results from property testing with the state merging algorithm we obtain KL-PAC learnability of the new automata class.

## 1 Introduction

In this paper we investigate the following question: given a finite set of samples drawn from a target distribution<sup>1</sup> generated by a probabilistic automaton such that the suffix distribution from any two states of the automata can be distinguished efficiently by an oracle; can we learn this distribution efficiently? The definition of an oracle as well as our notion of efficiency is defined rigorously in the following sections.

Probabilistic deterministic finite automata (or PDFAs) are stochastic extensions of deterministic finite automata (DFA), a well studied class of formal models (see e.g. Hopcroft and Ullman, 1979). The PDFFA class finds uses in a variety of areas in pattern recognition and machine learning including computational linguistics, time series analysis, computational biology, speech recognition, and network intrusion detection (see e.g. Vidal et al., 2005a,b). The problem of learning PDFFA efficiently from sample data is therefore of significant practical importance.

Many researchers have investigated the KL-PAC learnability of PDFFA. There are indications that the general problem of PDFFA learning is hard. For instance, Kearns et al. (1994) showed that KL-PAC learnability of PDFFA implies the computability of the  $\text{KL-PAC learnability of PDFFA}$ , thus violating the  $\text{KL-PAC learnability of PDFFA}$ , widely believed to be true in the cryptography community (see e.g. Kearns,

---

<sup>1</sup> Note that we assume the samples are drawn from the target distribution, and therefore our framework differs from the distribution-free setting.

1993). This is demonstrated by showing how by KL-PAC learning a specific family of (acyclic) PDFA, one can evaluate the noisy parity function<sup>2</sup>.

On the other hand, Ron et al. (1995) showed that a subclass of acyclic PDFA, which they call  $\mu$ -distinguishable, can be efficiently learned using the state merging algorithm. Clark and Thollard (2004) extended this result to PDFA with bounded expected suffix length from every state. Roughly speaking, for an automaton in this subclass, given any two states there exists at least one string whose suffix probability from the two states differs by at least  $\mu$ . In other words, the suffix distribution of any two states of the PDFA are at least  $\mu$  apart in the  $L_\infty$  distance. The state merging algorithm uses an efficient test to distinguish between suffix distributions.

However, the  $L_\infty$  distance between two distributions can often be misleading. Consider two extreme cases: let,  $D_1$  and  $D_2$  be distributions over  $\{1, \dots, 2n\}$ . Furthermore, let  $D_1(i) = 1/n$  for  $i = 1 \dots n$  and 0 otherwise, while  $D_2(i) = 1/n$  for  $i = n + 1 \dots 2n$  and 0 otherwise. Clearly,  $D_1$  and  $D_2$  have disjoint support, but  $\|D_1 - D_2\|_\infty$  is only  $1/n$ . On the other hand, even if  $D_1$  and  $D_2$  agree on all but two elements,  $\|D_1 - D_2\|_\infty$  could potentially be large. The class of  $\mu$ -distinguishable automata may therefore be unsatisfactory, and we would like to guarantee learnability of a more general family.

In Carrasco and Oncina (1999) it was shown that corresponding to every distribution induced by a PDFA there exists a canonical PDFA with the minimal number of states which induces the same distribution. Furthermore, the suffix distributions of the states of the canonical PDFA are unique. Therefore, if we are given an oracle which can distinguish between two suffix distributions, we can learn the PDFA. In this paper we show how the state merging algorithm can use an oracle to learn  $\mu_p$ -distinguishable PDFA. Our definition of  $\mu_p$  distinguishability is a generalization of  $\mu$ -distinguishability. The suffix distributions of any two states of a  $\mu_p$ -distinguishable PDFA are at least  $\mu$  apart in the  $L_p$  distance for some  $1 \leq p \leq \infty$ .

In order to distinguish between suffix distributions we use property testing algorithms. Given the ability to perform local queries, property testing algorithms aim to determine if an object satisfies a particular property or is far from satisfying it. These algorithms are typically sub-linear because they perform the task by inspecting only a fraction of samples drawn from the global object, and typically provide bounds on the probability of failure (see e.g. Ron, 2001).

For our purposes, property testing algorithms act as an oracle in the following sense: Given a norm  $\|\cdot\|$  and two distributions  $D_1$  and  $D_2$  over  $\Sigma^*$ , the algorithm outputs "yes" with probability at least  $1 - \delta$  whenever  $\|D_1 - D_2\| < \varepsilon$  and it outputs "no" with probability at least  $1 - \delta$  whenever  $\|D_1 - D_2\| > \varepsilon'$ , ( $\varepsilon' > \varepsilon$ ). In particular, using a property testing algorithm described in

---

<sup>2</sup> We mention in passing that even in the weaker,  $L_1$ -PAC learning model, we have shown, and will publish elsewhere, that general PDFA learnability still violates the noisy parity assumption. This implies that the difficulty is inherent to PDFA learning, and not merely an artifact of the KL-divergence.



Batu et al. (2000) we present a modified state merging algorithm for learning  $\mu_2$ -distinguishable PDFA efficiently.

Characterization of the family of PDFA which can be learned efficiently is a deep question for which only partial answers are known so far. For instance, it is well known that the noisy parity counterexample construction of Kearns et al. (1994) is not  $\mu$ -distinguishable. In fact, for every  $p > 1$ , the automaton's  $\mu_p$ -distinguishability decays exponentially with the number of states. Yet it is readily shown (Murphy, 1996) that this family of automata is  $\mu_1$ -distinguishable whilst remaining hard to learn. The lower bounds for  $L_1$  distance testing proven by Batu et al. (2000) also imply that learning  $\mu_1$ -distinguishable PDFA (using state merging) is hard. We seek to understand the criterion which implies efficient PDFA learning and this paper is a step in that direction.

We begin by formalizing our notation in Sect. 2 and review the Myhill-Nerode theorem and its analog for PDFA in Sect. 3. Section 4 is devoted to a description of  $\mu_p$ -distinguishability. In Sect. 5 we describe our modified state merging algorithm. In Sect. 5.1 we demonstrate a particular instance of our algorithm for learning  $\mu_2$ -distinguishable automata, and establish its polynomial sample and computational complexity. We conclude with a discussion of possible extensions and interesting research directions.

## 2 Notation

For concepts related to PDFA learning, we adopt the notation of Clark and Thollard (2004) where applicable. We also use the notation of Batu et al. (2000) when we discuss property testing over distributions.

We use  $\Sigma$  to denote a discrete alphabet consisting of  $|\Sigma|$  elements, and by  $\Sigma^*$  we denote the set of finite (but unbounded) length sequences over  $\Sigma$ . A subset  $\mathcal{L} \subseteq \Sigma^*$  is called a language. A distribution  $D$  over  $\Sigma^*$  is a function which assigns a probability  $D(s)$  to all sequences  $s \in \Sigma^*$ , such that  $0 \leq D(s) \leq 1$ , and  $\sum_{s \in \Sigma^*} D(s) = 1$ . The distribution  $D$  is also called a probability mass function.

A PDFA, denoted by  $A$ , is a tuple  $(Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$ , where  $Q$  is a finite set of states,  $q_0 \in Q$  is the initial state,  $q_f \notin Q$  is the final state,  $\zeta \notin \Sigma$  is the termination symbol,  $\tau : Q \times \Sigma \cup \{\zeta\} \rightarrow Q \cup \{q_f\}$  is the transition function, and  $\gamma : Q \times \Sigma \cup \{\zeta\} \rightarrow [0, 1]$  is the next symbol probability function.  $\gamma(q, \sigma) = 0$  for any  $\sigma \in \Sigma$  when  $\tau(q, \sigma)$  is not defined. From each state  $q \in Q$ , the sum of output probabilities must be one:  $\sum_{\sigma \in \Sigma \cup \{\zeta\}} \gamma(q, \sigma) = 1$ .

A labelled graph  $G = (V, E)$  is defined by a set of nodes  $V$  and a set of labelled edges  $E \subseteq V \times \Sigma \times V$ . Every PDFA  $A$  has an associated underlying labelled graph  $G_A$ . When it is clear from context we will use  $G$  and  $G_A$  interchangeably.

All transitions which emit the  $\zeta$  symbol point to the final state. The functions  $\gamma$  and  $\tau$  can be recursively extended to strings:  $\gamma(q, \sigma_1 \sigma_2 \dots \sigma_k) = \gamma(q, \sigma_1) \cdot \gamma(\tau(q, \sigma_1), \sigma_2 \dots \sigma_k)$ , and when  $\gamma(q, \sigma_1 \sigma_2 \dots \sigma_k) > 0$ , we have  $\tau(q, \sigma_1 \sigma_2 \dots \sigma_k) = \tau(\tau(q, \sigma_1), \sigma_2 \dots \sigma_k)$ .

A PDFA thus induces a distribution, or in other words, a probability mass function over  $\Sigma^*$ , with  $P^A(s) = \gamma(q_0, s\zeta)$ , the probability of generating the string  $s$ . We define  $P_q^A(s) = \gamma(q, s\zeta)$  to be the probability of generating the string  $s$  from the state  $q$ .

Given two distributions  $D_1$  and  $D_2$  over  $\Sigma^*$ , we will use the following notions of distance:

- For  $1 \leq p < \infty$ , the  $L_p$  distance between  $D_1$  and  $D_2$  is defined as:

$$\|D_1 - D_2\|_p = \sum_{s \in \Sigma^*} \left[ |D_1(s) - D_2(s)|^p \right]^{1/p}.$$

- In the  $p \rightarrow \infty$  limit, the resulting  $L_\infty$  distance, is defined as:

$$\|D_1 - D_2\|_\infty = \max_{s \in \Sigma^*} |D_1(s) - D_2(s)|.$$

- The KL-divergence between  $D_1$  and  $D_2$  is defined as:

$$\text{KL}(D_1 \| D_2) = \sum_{s \in \Sigma^*} D_1(s) \log \left( \frac{D_1(s)}{D_2(s)} \right).$$

Note that the KL divergence is not a metric. As shown in Cover and Thomas (1991), for any pair of distributions  $D_1$  and  $D_2$ , it holds that  $\text{KL}(D_1 \| D_2) \geq \frac{1}{2 \ln 2} \|D_1 - D_2\|_1^2$ , which in turn upper-bounds all  $L_p$ -distances. Where appropriate, we will use the notation  $\text{KL}(A \| \hat{A})$  to denote  $\text{KL}(P^A \| P^{\hat{A}})$ .

We will use the following notion of learnability, due to Clark and Thollard (2004):

**Definition 1 (KL-PAC).** Let  $\mathcal{D}$  be a distribution over  $\Sigma^*$ . A language  $L$  is  $\epsilon$ -KL-PAC learnable if there exists a learning algorithm  $\hat{D}$  such that for all  $\delta > 0$  and  $m > q(1/\epsilon, 1/\delta, |D|)$ ,  $\Pr \left[ \text{KL}(D \| \hat{D}) > \epsilon \right] < \delta$ .

The definition of  $L_p$ -PAC learnability is analogous, with the corresponding change in distance measure.

### 3 Characterization Theorem for Suffix Distributions

In the theory of formal languages, the well known Myhill-Nerode theorem provides a necessary and sufficient condition for a language to be regular, and thus accepted by a DFA (see e.g. Hopcroft and Ullman, 1979). The theorem states that a language is regular if and only if it is the union of some equivalence classes of a partition of  $\Sigma^*$  of finite index. Definitions of the key concepts in the theorem are:

- For a given pair of strings  $x, y \in \Sigma^*$  and a formal language  $\mathcal{L}$ , a relation  $\sim$  is defined by:

$$x \sim y \iff (xz \in \mathcal{L} \iff yz \in \mathcal{L}) \quad \forall z \in \Sigma^*.$$

- The index of a language is defined as the number of its equivalence classes.

A analogous theorem for PDFA is due to Carrasco and Oncina (1999). The concept of a right equivalence relation for a stochastic language is as follows:

- For a given pair of strings  $x, y \in \Sigma^*$ , and a stochastic language  $\mathcal{L}$  generated by a PDFA  $A$ , a “right equivalence relation”  $\sim$  is defined by:

$$x \sim y \iff P^A(xz) = P^A(yz) \quad \forall z \in \Sigma^*.$$

The modified theorem now states that a stochastic language is generated by a PDFA if and only if it is the union of some equivalence classes of a stochastic right invariant equivalence relation of finite index.

The canonical generator is defined to be the PDFA with the minimal number of states generating a given stochastic regular language  $\mathcal{L}$  (Carrasco and Oncina, 1999). Using the definition of the right invariant equivalence relation and the minimality of the canonical generator, it can be shown that the suffix distribution of every state  $q$  of the canonical generator is distinct. In other words, given an oracle which can distinguish between suffix distributions, we can learn the distribution induced by a PDFA by identifying the right invariant equivalence classes. In Sect. 4 we show that the state merging algorithm for learning  $\mu$ -distinguishable automata implicitly uses such an oracle which exploits the  $L_\infty$  distance between suffix distributions. We then extend the algorithm to use other oracles, guaranteeing learnability of a larger class of PDFA.

### 4 $\mu_p$ -Distinguishability

The general PDFA learning problem is hard, and therefore additional conditions are required before attempting to provide an efficient learning algorithm. The concept of  $\mu$ -distinguishability was first introduced by Ron et al. (1995), where it was shown to be sufficient for KL-PAC learning of PDFA.

**Definition 2 ( $\mu$ -distinguishability).** Let  $A = (Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$  be a PDFA.  $A$  is  $\mu$ -distinguishable if for every  $q, q' \in Q$  and  $s \in \Sigma^*$  such that  $|\gamma(q, s\zeta) - \gamma(q', s\zeta)| \geq \mu$ .

Recently, Clark and Thollard (2004) extended the result to general PDFA learning, while imposing an additional condition, namely an upper bound on the expected suffix length from all states.

We seek to understand which criteria will in fact enable efficient testing for discrimination between distributions. For instance,  $\mu$ -distinguishability is equivalent to demanding that  $\|P_q^A - P_{q'}^A\|_\infty \geq \mu$  for every  $q, q' \in Q$ . This provides for an efficient and accurate test for deciding whether or not two distributions over strings are similar (i.e. drawn from the same suffix distribution). We argue in Sect. 5 that such efficient and accurate testing can in fact be achieved using relaxed conditions, which in turn also guarantee KL-PAC learnability.

4.1 Definition of  $\mu_p$ -Distinguishability

**Definition 3** ( $\mu_p$ -distinguishability). Let  $A = (Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$  be a PDFDA,  $q, q' \in Q$  and  $\mu_p$  be a distribution over  $L_p$ . We say that  $A$  is  $\mu_p$ -distinguishable if

$$\|P_q^A - P_{q'}^A\|_p \geq \mu.$$

Note that for  $p = \infty$  we recover the original  $\mu$ -distinguishability condition. Moreover, for any distribution  $D$  over  $\Sigma^*$ , if  $1 \leq p_1 < p_2 \leq \infty$  then we have  $\|D\|_{p_1} \geq \|D\|_{p_2}$ ; hence the  $\mu_{p_1}$ -distinguishable class contains the  $\mu_{p_2}$ -distinguishable class.

5 State Merging with Oracles

In order to describe how state merging algorithms can use oracles to learn PDFDA distributions, we first provide a modular analysis of the proof due to Clark and Thollard (2004), and then extend it to deal with oracles. In particular, we show that the state merging algorithm may be decoupled into two parts:

- A construction algorithm which iteratively builds the PDFDA graph and sets the transition probabilities.
- An oracle, which enables efficient and accurate testing for whether or not two sample sets were drawn from two distinct suffix distributions.

Given such an oracle, the state merging algorithm will induce a PDFDA such that with high probability, the KL-divergence between target and induced distributions can be made arbitrarily small.

The learning algorithm is given the following parameters as input: an alphabet  $\Sigma$ , an upper bound on the expected length of strings generated from a state of the target  $L$ , an upper bound on the number of states of the target  $n$ , a confidence parameter  $\delta$  and a precision parameter  $\epsilon$ . Pseudocode for the state merging algorithm is given in Algorithm A (Appendix A), while a full description can be found in Sect. 3 of Clark and Thollard (2004).

For our purposes, an oracle is a black-box which can distinguish between suffix distributions. More formally:

**Definition 4 (Oracle).** Let  $\mathcal{H}$  be a set of PDFDAs. An oracle  $\mathcal{O}_{\mathcal{H}}$  is a  $(\delta, m)$ -match between  $\mathcal{H}$  and  $\mathcal{H}$ . Let  $A \in \mathcal{H}$  and  $q, q' \in A$ . Then  $\mathcal{O}_{\mathcal{H}}$  is a  $(\delta, m)$ -match between  $q$  and  $q'$  if  $\|P_q^A - P_{q'}^A\|_1 \geq 1 - \delta$ .

The learning algorithm we use is analogous to the state merging algorithm described in Clark and Thollard (2004), with the oracle  $\mathcal{O}_{\mathcal{H}}$  testing whether to merge a hypothesized state (see Definition 5) with an existing one, or to construct a new state. This leads to our main result:

**Theorem 1.** Let  $\mathcal{H}$  be a class of distributions over a set  $\Sigma$ . For  $\epsilon > 0$ ,  $\delta > 0$ ,  $L$ ,  $n$ ,  $\delta_1, \delta_2, \epsilon_1, m_2$  and  $(1)$ ,  $(2)$  as in (1), (2) above, let  $\mathcal{O}_{\mathcal{H}} = (\delta_1, m_1)$  be an oracle. For  $n$  large enough,  $A \in \mathcal{H}$ ,  $L$  and  $\hat{A}$  be the output of the algorithm. Then  $\text{KL}(A \| \hat{A}) < \epsilon$  with probability  $1 - \delta$ .

Our proof closely follows Clark and Thollard (2004), with a number of key changes:

- In the original proof, distinguishability between two states’ suffix distributions (with high probability) is inherent to the sample size bounds. In our case, the oracle provides the distinguishing test, so the size of the multiset drawn at each step of the algorithm is reformulated to reflect this (see (1), Appendix A).
- In our case, two sources of randomness are present: the randomly drawn sample sets and the oracle. To bound the probability of error, both sources need to be accounted for.

With the exceptions noted above, the original proof is directly transferable to our setting. The proof sketch can be found in Appendix B.

### 5.1 Learning $\mu_2$ Efficiently

We use a result on testing distribution proximity in  $L_2$  due to Batu et al. (2000) to show learnability of the  $\mu_2$ -distinguishable class. The result applies to distributions over discrete sets, and the computational complexity does not depend on the size of the set. Specifically, for Algorithm  $L_2$  described in Appendix C the following is proven:

**Theorem 2 (Batu et al. (2000)).** Let  $D_1, D_2$  be two distributions over a set of size  $m = O(\frac{1}{\epsilon^4})$ . Let  $\|D_1 - D_2\|_2 \leq \epsilon/2$ . Then Algorithm  $L_2$  will ACCEPT with probability  $(1 - \delta)$  and REJECT with probability  $1 - \delta$  if  $\|D_1 - D_2\|_2 \geq \epsilon$ . The error probability is  $O(\epsilon^{-4} \log \frac{1}{\delta})$ .

Note that the running time of the test is independent of  $n$ . As a consequence of Theorem 2, the  $L_2$ -Distance-Test algorithm can serve as a  $(\delta, \frac{C_2}{\epsilon^4})$ -matching oracle for the  $\mu_2$ -distinguishable PDFA class, where  $C_2$  is a constant hidden in the asymptotic notation. Thus, as a direct consequence of Theorem 1 and the  $L_2$  matching oracle, we have the following lemma:

**Lemma 1.** Let  $\mu_2$  be a class of distributions over a set  $\Sigma$ . For  $\epsilon > 0$ ,  $\delta > 0$ ,  $L$ ,  $n$ ,  $\delta_1, \delta_2, \epsilon_1, m_2$  and  $(1)$ ,  $(2)$  as in (1), (2) above, let  $\mathcal{O}_{\mu_2} = (\delta_1, m_1)$  be an oracle. For  $n$  large enough,  $A \in \mu_2$ ,  $L$  and  $\hat{A}$  be the output of the algorithm. Then  $\text{KL}(A \| \hat{A}) < \epsilon$  with probability  $1 - \delta$ .

## 6 Discussion and Conclusion

We introduced  $\mu_p$ -distinguishable automata, which generalize the concept of  $\mu$ -distinguishable automata. Using a new modularized analysis we extended the

proof of KL-PAC learnability of  $\mu$ -distinguishable automata via the state merging algorithm. This new insight allows us to extend state merging to use oracles. We use an existing  $L_2$  property testing algorithm to learn  $\mu_2$ -distinguishable automata efficiently via state merging.

In the noisy parity PDFA family, the  $\mu_1$ -distinguishability is a constant, while the  $\mu_2$ -distinguishability is  $O(2^{-n/2})$  and the  $\mu_\infty$ -distinguishability is  $O(2^{-n})$  (where  $n$  denotes the number of states). By setting  $n = \alpha \log t$  we obtain for this family a  $\mu_2$ -distinguishability of  $O(t^{-\alpha/2})$ , and a  $\mu_\infty$ -distinguishability of  $O(t^{-\alpha})$ . Thus, we have exhibited a class for which our modified state merging algorithm outperforms the original by an arbitrary polynomial factor.

A natural problem for further research regards the efficient learnability of  $\mu_p$ -distinguishable PDFA classes for the different values of  $p$ . We conjecture that for  $p > 1$ , these classes are indeed efficiently learnable.

## Acknowledgements

We thank Alex Smola for insightful discussions. National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program.

## References

- T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *41<sup>st</sup> Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE Computer Society, 2000.
- R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *Journal of Computer and System Sciences*, 33(1): 1–20, 1999.
- A. Clark and F. Thollard. PAC-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497, 2004.
- T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.
- J. E. Hopcroft and J. D. Ullman. *Introduction to the Theory of Computation*. Addison-Wesley, Reading, Massachusetts, first edition, 1979.
- M. Kearns. Efficient noise-tolerant learning from statistical queries. In *25<sup>th</sup> Annual Symposium on Foundations of Computer Science*, pages 392–401. ACM Press, New York, NY, 1993.
- M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire, and L. Sellie. On the learnability of discrete distributions. In *26<sup>th</sup> Annual Symposium on Foundations of Computer Science*, pages 273–282, 1994.
- K. Murphy. Passively learning finite automata. Technical report, Santa Fe Institute, 1996.
- D. Ron. Property testing. In S. Rajasekaran, P. Pardalos, J. Reif, and J. Rolim, editors, *Handbook of Property Testing*, volume II, pages 597–649. Kluwer Academic, 2001.

- D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *Proceedings of the 8th Conference on Artificial Intelligence and Statistics*, pages 31–40. ACM Press, New York, NY, 1995.
- E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines – Part I. *Journal of Machine Learning Research*, 2005a. to appear.
- E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines – Part II. *Journal of Machine Learning Research*, 2005b. to appear.

## A Pseudocode for the State Merging Algorithm

Pseudocode for the state merging algorithm is provided in Algorithm A below. The algorithm is guaranteed to learn (with high probability) a PDFa class  $\mathcal{H}$ , for which a matching oracle is provided.

The number of oracle queries performed at each step of the state merging algorithm is upper bounded by  $n^2|\Sigma|$ , as there are at most  $n$  nodes in the graph at any time and at most  $n|\Sigma|$  candidate nodes. When the algorithm runs correctly there are at most  $n|\Sigma| + 2$  steps. Therefore, over a complete run of the algorithm the number of oracle calls is at most  $n^2|\Sigma|(n|\Sigma| + 2)$ .

## B Proof Sketch of Theorem 1

We present a proof sketch using the notation of Clark and Thollard (2004), and provide exact references to the original proof where appropriate. We begin by decomposing our proof into the following:

- (i) Given sufficiently many samples, a sample set is likely to be "good". Namely, a string will appear with an empirical probability that is close to its actual probability.
- (ii) Assuming an oracle which matches the PDFa family under consideration, at each step the hypothesis graph will be isomorphic to a subgraph of the target with high probability.
- (iii) With high probability, when the algorithm stops drawing samples, there will be in the hypothesis graph a state representing each frequent state in the target. In addition, all frequent transitions will also have a representative edge in the graph.
- (iv) After the algorithm terminates, (again with high probability) all transition probability estimates will be close to their correct target values.
- (v) A KL-PAC result between target and hypothesis is derived using the previous modules.

The following definitions in Clark and Thollard (2004) are redefined for the purpose of our proof:

The definition of a  $\delta$ -good sample set is replaced (or rather generalized) by the following:

---

**Algorithm 1.** State merging with oracle

---

**Input:**  $\Sigma$  (input alphabet),  $L$  (upper bound on the expected length of strings generated from any state),  $n$  (upper bound on the number of states),  $\mathcal{O}_{\mathcal{H}}$  (a  $(\delta_1, m_1)$ -matching oracle for  $\mathcal{H}$ ),  $\delta$  (confidence parameter),  $\varepsilon$  (precision parameter). The algorithm is also supplied with a random source of strings generated independently by  $A$ , the target PDFa.

**Output:**  $\hat{A}$ , a hypothesis PDFa such that  $\text{KL}(A||\hat{A}) < \varepsilon$  with probability at least  $1 - \delta$ .

**Data:** The algorithm maintains a graph  $G = (V, E)$  with labelled edges (i.e.  $E \subseteq V \times \Sigma \times V$ ), which holds the current hypothesis about the structure of the target automaton.

**repeat**

    Draw  $N$  strings from  $A$

**foreach**  $u \in V$  and  $\sigma \in \Sigma$  which does not yet label an edge out of  $u$  **do**

        Hypothesize a candidate node, referred to as  $(u, \sigma)$

        Compute  $S_u$  (multiset of suffixes of this node)

**if**  $|S_u| \geq m_0$  **then**

**foreach**  $v \in V$  **do**

                Query  $\mathcal{O}_{\mathcal{H}}$  to compare  $S_{u,\sigma}$  with  $S_v$

**if**  $\mathcal{O}_{\mathcal{H}}$  returns *ACCEPT* **then**

                    Add arc labelled with  $\sigma$  from  $u$  to  $v$ .

**end**

**end**

**if**  $\mathcal{O}_{\mathcal{H}}$  returns *REJECT* on all comparisons **then**

            Create new node to graph  $G$

            Add an edge labelled with  $\sigma$  from  $u$  to the new node

**end**

**end**

**until** no candidate node has a sufficiently large suffix multiset

Complete  $G$  by adding a *ground node* which represents low frequency states

Add a final state  $\hat{q}_f$  and transitions labelled with  $\zeta$  from each state to  $\hat{q}_f$

---

**Definition 5 (Candidate node).**

A candidate node  $(u, \sigma)$  is a node  $u$  in graph  $G$  with a suffix multiset  $S_{u,\sigma}$  for  $\sigma \in \Sigma$ . The node  $(u, \sigma)$  is similar to node  $v$  in graph  $G$  if  $\mathcal{O}_{\mathcal{H}}$  returns *ACCEPT* for  $(S_{u,\sigma}, S_v)$ .

Definition 7 of a *ground node* is now relaxed, and the original condition  $L_{\infty}(\hat{S}, P_q^A) < \mu/4$  (which guaranteed distinguishability) is now dropped:

**Definition 6 (good multiset).**

A multiset  $S$  is good if  $\forall \sigma \in \Sigma \cup \{\zeta\}, |(S(\sigma)/|S|) - \gamma(q, \sigma)| < \varepsilon_1$ .

The threshold  $m_0$  on the minimal multiset size for testing distribution proximity now becomes  $m_0 = \max(m_1, m_2)$ , where  $m_1$  is the number of samples required by the matching oracle  $\mathcal{O}_{\mathcal{H}}$  to guarantee an error probability of at most



$\delta_1$ , and  $m_2$  is the multiset size shown in Clark and Thollard (2004) to guarantee a good sample (according to Definition 6 above) with probability at least  $1 - \delta_2$ :

$$m_2 = \frac{1}{2\varepsilon_1^2} \log \left( \frac{24n|\Sigma|(|\Sigma| + 1)(n|\Sigma| + 2)}{\delta_2} \right), \quad \text{with} \quad (1)$$

$$\varepsilon_1 = \frac{\varepsilon^2}{16(|\Sigma| + 1)(L + 1)^2}.$$

The number of samples drawn at each iteration of the algorithm now becomes:

$$N = \frac{4n|\Sigma|L^2(L + 1)^3}{\varepsilon_3^2} \max \left( 2n|\Sigma|m_0, 4 \log \frac{2(n|\Sigma| + 2)}{\delta} \right), \quad \text{with}$$

$$\varepsilon_3 = \frac{\varepsilon}{2(n + 1) \log (4(L + 1)(|\Sigma| + 1)/\varepsilon)}.$$

Rigorous statements and proofs of modules (i), (iii) and (iv) are given in Clark and Thollard (2004). References to the relevant sections follow.

The “good sample” concept of module (i) is defined in Definitions 7, 8, 10 and 11 of Sect. 4.2. Note that the original definition of a good multiset is now relaxed to Definition 6 above. The probability of obtaining a good sample is lower-bounded in Sect(s). 6.1 and 6.2. Specifically, using Chernoff bounds it is shown that for a sample size of  $m_2$  the condition of Definition 6 is assured with probability of error less than  $e^{-2m_2\varepsilon_1^2}$ , which equals  $\frac{\delta_2}{24n|\Sigma|(|\Sigma|+1)(n|\Sigma|+2)}$ .

Module (iii) is proven in Lemmas 12 and 13 of Sect(s). 4.2 and 4.4 respectively, and our proof requires no changes. Module (iv) is discussed in Sect. 4.3, where again no changes are necessary. The modifications to module (ii) are addressed in the following lemma:

**Lemma 2.** *Let  $\mathcal{H} = (\delta_1, m_1)$  be a matching oracle,  $\mathcal{O}_{\mathcal{H}}$  be a matching oracle, and  $\varepsilon > 0$ . Then, for any  $\mu$ -distinguishable class  $\mathcal{C}$ , the algorithm queries  $\mathcal{O}_{\mathcal{H}}$  at most  $n^2|\Sigma|$  times at each step.*

The algorithm queries  $\mathcal{O}_{\mathcal{H}}$  at most  $n^2|\Sigma|$  times at each step. By applying the union bound and using the definition of a matching oracle we obtain the lemma. □

Finally, we prove module (v) and derive a KL-PAC bound. For the original  $\mu$ -distinguishable class, a KL-PAC bound is proved by Clark and Thollard (2004). The authors show that assuming a good sample had been drawn, the KL-PAC bound follows. In our framework, an additional degree of randomness is present due to the probabilistic nature of the oracle. However, if this probability of error is managed, the same KL-divergence bound between target and hypothesis PDFAs (namely  $\varepsilon$ ) follows.

By setting

$$\delta_1 = \frac{\delta}{2n^2|\Sigma|(n|\Sigma| + 2)}, \quad (2a)$$

$$\delta_2 = \delta/2, \quad (2b)$$

using multisets of size  $m_0 = \max(m_1, m_2)$ , and applying the union bound, the probability of error obtained is not greater than  $\delta$ , and we retain the  $\varepsilon$  accuracy. The proof of Theorem 1 follows.

## C Pseudocode for the $L_2$ Distance Test

Pseudocode for the  $L_2$  distance test is provided in Algorithm C below.  $r_D$  denotes the number of self-collisions in the set  $F_D$ , namely the count of  $i < j$  such that the  $i^{\text{th}}$  sample in  $F_D$  is same as the  $j^{\text{th}}$  sample in  $F_D$ . Similarly,  $c_{D_1 D_2}$ , the number of collisions between  $D_1$  and  $D_2$  is the count of  $(i, j)$  such that the  $i^{\text{th}}$  sample in  $D_1$  is same as the  $j^{\text{th}}$  sample in  $D_2$ .

---

### Algorithm 2. $L_2$ -Distance-Test

---

**Input:**  $D_1, D_2, m, \varepsilon, \delta$

**Result:** ACCEPT or REJECT

**repeat**

    Let  $F_{D_1}$  = a set of  $m$  samples from  $D_1$

    Let  $F_{D_2}$  = a set of  $m$  samples from  $D_2$

    Let  $r_{D_1} = |F_{D_1} \cap F_{D_1}|$  (the number of self-collisions in  $F_{D_1}$ )

    Let  $r_{D_2} = |F_{D_2} \cap F_{D_2}|$

    Let  $Q_{D_1}$  = a set of  $m$  samples from  $D_1$

    Let  $Q_{D_2}$  = a set of  $m$  samples from  $D_2$

    Let  $c_{D_1 D_2} = |Q_{D_1} \cap Q_{D_2}|$

    Let  $r = \frac{2m}{m-1}(r_{D_1} + r_{D_2})$

    Let  $s = 2c_{D_1 D_2}$

**if**  $r - s > m^2 \varepsilon^2 / 2$  **then** REJECT **else** ACCEPT

**until**  $O(\log(\frac{1}{\delta}))$  iterations

REJECT if the majority of iterations reject

ACCEPT otherwise

---

# Learning Attribute-Efficiently with Corrupt Oracles

Rotem Bennet and Nader H. Bshouty

Department of Computer Science, Technion, Haifa, Israel  
{rotemt, bshouty}@cs.technion.ac.il

**Abstract.** We study learning in a modified EXACT model, where the oracles are corrupt and only few of the presented attributes are relevant. Both modifications were already studied in the literature, and efficient solutions were found to most of their variants. Nonetheless, their reasonable combination is yet to be studied, and combining the existing solutions either fails or works with complexity that can be significantly improved. In this paper we prove equivalence of EXACT learning attribute-efficiently with and without corrupt oracles. For each of the possible scenarios we describe a generic scheme that enables learning in these cases using modifications of the standard learning algorithms. We also generalize and improve previous non attribute-efficient algorithms for learning with corrupt oracles.

## 1 Introduction

In this paper we prove the equivalence of learning attribute-efficiently with and without corrupt oracles (limited or malicious). A membership oracle is “limited” if it might answer “I don’t know” on some chosen subset of the instance space, and an oracle is “malicious” if it flips the classifications of the target function, for some chosen subset of the instance space, and answers accordingly. An “attribute-efficient” algorithm is defined to be one whose query complexity has only sub-linear dependency on the total number of attributes (variables). Angluin et al. [4] have presented, for several concept classes, learning algorithms which are efficient despite the use of corrupt oracles. However, the more general question, of whether EXACT learning with corruptions is reducible to standard EXACT learning, remained an open question at that time. Only recently, have Bisht, Bshouty and Khoury [8] resolved this open question, by presenting efficient reductions from standard EXACT learning to learning with various types of corruptions. Nonetheless, their algorithms, which transform standard learning algorithms to ones for learning with corrupt oracles, multiply the complexity of the standard algorithms by the total number of variables (and the number of corruptions), and thus the resulting algorithms are non-attribute-efficient, regardless of the possible attribute-efficiency of the original algorithms. This has motivated us to study the question of attribute-efficient learning with corrupt oracles. We prove the attribute-efficient (with corruptions) analogs of the results

of Bisht, Bshouty and Khoury [8], and improve some of their algorithms also for the non-attribute-efficient case.

In the full paper we also present an example of utilizing the properties of a specific concept class, CDNF (which contains DT), for a drastically improved learning of it with corrupt oracles (improved relative to our more generic algorithms).

Our discussed learning models require the learner to learn the target function “strictly” as presented to him by the most accurate oracle he has, meaning that if he has access to only malicious oracles, then he should learn the target function along with its corruptions. In the case of a limited membership oracle, we also discuss “non-strict” learning of the target function without the instances on which the oracle answers “I don’t know” (i.e. on these instances the learner is not obliged to be accurate). Angluin et al. [4] have already proved that efficient non-strict learnability implies efficient strict learnability. In the full paper we notably improve their transformation scheme between the two models, and show that our scheme is optimal.

### 1.1 Preliminaries

**Boolean Concept Classes and Functions.** We define a  $F$  to be  $\{F_n\}_{n>0}$ , where each  $F_n$  is a set of boolean formulae defined over a set of boolean variables  $V_n = \{x_1, \dots, x_n\}$ . We define the corresponding  $C$  to be  $\{C_n\}_{n>0}$ , where each  $C_n$  is the set of boolean functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  defined by their representations in  $F_n$ . The size of a function  $f \in C_n$ , denoted by  $size_F(f)$ , is the size (number of bits) of its minimal representation in  $F_n$  (we may write just  $size(f)$ , when  $F$  is understood from the context). We will sometimes refer to the boolean function  $f$  as the subset of  $\{0, 1\}^n$  which satisfies the function, i.e.  $\{x \in \{0, 1\}^n \mid f(x) = 1\}$ . For a set  $S$ , we denote by  $P(S)$  the power-set (set of all possible subsets) of  $S$ . For a boolean assignment  $a \in \{0, 1\}^n$ , we denote by  $a_i \in \{0, 1\}$  ( $i \in \{1, \dots, n\}$ ) the  $i$ 'th bit of  $a$ . For a boolean assignment  $a \in \{0, 1\}^n$  and a bit value  $\sigma \in \{0, 1\}$ , we denote by  $a|_{a_i \leftarrow \sigma}$  the assignment  $b \in \{0, 1\}^n$  for which  $b_j = a_j$  for all  $j \neq i$  and  $b_i = \sigma$ .

For a given boolean function  $f \in C_n$ , a variable  $x_i \in V_n$  is *relevant* if there exists an assignment  $a \in \{0, 1\}^n$  for which  $f(a|_{a_i \leftarrow 0}) \neq f(a|_{a_i \leftarrow 1})$ . We denote by  $C_n^r$  the class of boolean functions  $f \in C_n$  with at most  $r$  relevant variables. We say that two instances  $a, b \in \{0, 1\}^n$  are *equivalent*, with respect to a boolean function  $f$ , if  $a$  and  $b$  have equal values in all the relevant variables. Two instances from different equivalence classes would be called *non-equivalent*.

A *restriction* of a set  $S \subseteq \{0, 1\}^n$  to a boolean function  $\rho$ , is the set of all instances in  $S$  which satisfy  $\rho$ ; that is,  $\{a \in S \mid \rho(a) = 1\}$ . For a boolean function  $f$ , the *restriction* of  $f$  to  $\rho$  is defined by the function  $f \wedge \rho$ . A *term* is  $T^\pi = (\pi_1, \dots, \pi_n)$ , where  $\pi_i \in \{0, 1, x_i\}$ . Its *support* is the conjunction of all the variables  $x_i$  for which  $\pi_i \in \{0, 1\}$ , where  $x_i$  is negated if  $\pi_i = 0$  (it can also be thought of as the maximal-size term which is satisfied by  $\pi$ ). The *restriction* of  $\pi$  is the set  $\{x_i \mid \pi_i \in \{0, 1\}\} \subseteq V_n$ , of the

variables whose value is  $\{0, 1\}$ , i.e. 0 or 1. We will sometimes refer to a variable that is not in the defined set, as a *free* or *unassigned* variable. The *restriction* of an assignment  $a \in \{0, 1\}^n$  to a partial assignment  $\pi$ , denoted by  $\pi(a)$ , is the assignment  $b \in \{0, 1\}^n$  for which  $b_i = p_i$  if  $p_i \in \{0, 1\}$  and  $b_i = a_i$  otherwise.

The projection of a set  $A \subseteq \{0, 1\}^n$  to  $\pi$  is naturally defined as  $\pi(A) \stackrel{\text{def}}{=} \bigcup_{a \in A} \pi(a)$ . The projection of a boolean function  $f(a)$  to  $\pi$  is the function  $\pi(f)(a) \stackrel{\text{def}}{=} f(\pi(a))$ . Equivalently, if  $f$  is represented by a formula  $\phi$  over the set of variables  $V_n = \{x_1, \dots, x_n\}$ , then  $\pi(f)$  is represented by the formula  $\pi(\phi)$  generated from  $\phi$  by replacing each  $x_i$  with  $\pi_i$ . We will denote the projected concept class  $\{\pi(f) \mid f \in C\}$  by  $\pi(C)$ . A concept class  $C$  is *closed under projection* if for any partial assignment  $\pi$ ,  $\pi(C) \subseteq C$ . Note that Valiant [17] uses the terms “restriction” and “projection” interchangeably, whereas we reserve the later for our above distinctive definition.

Let  $f$  be a boolean function, and let  $E$  be a set of labelled examples  $\{(x, \sigma) \in \{0, 1\}^n \times \{0, 1\}\}$ . The function which is equivalent to  $f$  but has  $E$  as a set of “hard coded” values, is denoted by  $[f, E]$ , and defined for all  $x \in \{0, 1\}^n$  as:

$$[f, E](x) \stackrel{\text{def}}{=} \begin{cases} \sigma & (x, \sigma) \in E \\ f(x) & \text{otherwise} \end{cases}$$

We will sometimes describe  $[f, E]$  as the *restriction* of  $E$  to  $f$ . Note that some of the labelled instances in  $E$  might actually agree with their classification by  $f$ , and thus  $E$  should not necessarily be considered as a set of “exceptions” to  $f$ .

**Attribute Efficient Learning.** Let  $I : \mathbb{N} \rightarrow \mathbb{N}$ , be such that  $I(n) = o(n)$ . A learning algorithm is *attribute efficient* (or A.E.) if its **query** complexity is bounded by  $O(I(n)p(r, \text{size}(f)))$ , for some polynomial  $p$ . We will sometimes discuss *attribute efficient* learning without mentioning the exact  $I(n)$  dependency on  $n$ , which means that our statements in these cases are true for all  $I(n) = o(n)$ . Note that the **time complexity** must be at least  $\Omega(n)$ , as the algorithm must eventually see all the  $n$  bits in the given examples in order to get some information about the identity of the relevant variables.

**Malicious and Limited Oracles.** Sometimes the given membership or equivalence oracle errs on some (adversarial) chosen subset of the instance space  $\{0, 1\}^n$ . In this case the oracle is called *malicious* and denoted by  $\text{MMQ}_f$  and  $\text{MEQ}_f$  respectively. In a weaker version, the membership oracle is never mistaken, yet might answer “I don’t know” (denoted by  $\perp$ ) on some subset of the instance space. In this case the oracle is called *limited* and denoted by  $\text{LMQ}_f$ . We shall use the common name *malicious* to anything (e.g. oracles, labelled instances) that behaves inconsistently with the true target function, and the name *malicious* to anything that is not corrupt. We denote by  $\text{CMQ}$  (resp.  $\text{CEQ}$ ) **any** type of corrupt membership (resp. equivalence) oracle (results in terms of this oracle obviously imply also results for a standard, non-corrupt, oracle). Note that in our study the corrupt oracles are required to be persistent; that is, each

oracle answers the same answer when asked the same queries (in other models it is not necessarily so, e.g. Sakakibara [14]). We denote the set of maliciously or limitedly classified instances by  $E \subseteq \{0, 1\}^n \times \{0, 1, \perp\}$ , and denote  $L \stackrel{\text{def}}{=} |E|$ . This means that for all  $(x, \sigma) \in E$ , we have  $f(x) \neq [f, E](x) = \sigma$ , where  $\sigma \in \{0, 1, \perp\}$  (the definition of  $[f, E]$  is slightly generalized for this purpose). In accordance with the above definitions of the oracles, any corrupt membership oracle classifies queried instances  $x \in \{0, 1\}^n$  according to  $[f, E](x)$ , and the malicious equivalence oracle  $\text{MEQ}_f$  might return counterexamples  $(x, \sigma) \in E$  for which the learner's hypothesis actually satisfies  $f(x) = h(x) \neq \sigma$ .

The definition of  $I(n)$ -attribute-efficient learnability is modified when the oracles are corrupt. It requires query complexities of  $O(I(n)\text{poly}(r, \text{size}(f), L))$ . That is, we allow the learner to use additional computational resources, which are polynomial in the size of the corrupt set, in order to overcome the oracle's inconsistencies with the target function.

Note that, when discussing  $I(n)$ -attribute-efficiency (for some  $I(n) = o(n)$ ), the number of corrupt examples  $L$  is bounded by  $o(I(n)\text{poly}(r, \text{size}(f)))$ , since otherwise, any  $I(n)$ -attribute-efficient algorithm might get **only** random or  $\perp$  answers (chosen maliciously so), and hence receive **no information** about the learned target function.

**Strict and Non-strict Learning.** In the model of learning, we require the learner to output a hypothesis  $h$  for which  $[h, E] \equiv [f, E]$ , meaning that the hypothesis  $h$  is obliged to be consistent only with the non-corrupt classifications of the oracles. This relaxed model is used only when learning with a **limited membership** oracle, either with or without an additional equivalence oracle (without an equivalence oracle this is the only possible model). Thus, when learning non-strictly, the standard equivalence oracle would answer accordingly 'Yes' for all queries on such a hypothesis  $h$ , without the ability to return any counterexample from the limited set  $E$ .

In the model of learning, the learner should output a hypothesis  $h$  equivalent to the target function as presented to him by the oracles. When at least one of the oracles is standard, then we require the learner to output  $h$  such that  $h \equiv f$ , since the learner has some source of information about the true (non-corrupt)  $f$ . The requirements from the learner in each scenario are summarized in Table 1.

**Table 1.** The requirements from the output hypothesis in each combination of available membership and equivalence oracles

Query Type	MQ	MMQ	LMQ	Without MQ
EQ	$h \equiv f$	$h \equiv f$	$h \equiv f$ or $[h, E] \equiv [f, E]$	$h \equiv f$
MEQ	$h \equiv f$	$h \equiv [f, E]$	$h \equiv [f, E]$	$h \equiv [f, E]$
Without EQ	$h \equiv f$	$h \equiv [f, E]$	$[h, E] \equiv [f, E]$	–

## 2 Learning with Corrupt Oracles

We shall now present the main theorem, which states an equivalence between attribute-efficient learning with and without corruptions. Bisht, Bshouty and Khoury [8] have already shown how to overcome errors in queries, when there is no attribute efficiency requirement, and here we generalize this result. The theorem would then be proved by a sequence of algorithms, each handles another type of an error.

**Theorem 1.** *Let  $\mathcal{C}_n$  be a concept class over a domain  $X$  of size  $n$ . Let  $\mathcal{A}$  be an EXACT learning algorithm for  $\mathcal{C}_n$  with time, equivalence query and hypotheses-size complexities of  $\psi_t(n, r)$ ,  $\psi_e(n, r)$  and  $\psi_s(n, r)$ , respectively, for some functions  $\psi_t, \psi_e, \psi_s : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ . Then, there exists an attribute-efficient learning algorithm for  $\mathcal{C}_n$  with access to  $\mathcal{A}$  and  $\psi_s$  corrupt oracles.*

### 2.1 Learning with MEQ

We study the case of learning a concept class  $C_n^r$  attribute-efficiently in the EXACT model (equivalence queries only), where at most  $L$  examples are maliciously corrupt. First we present a brute-force solution which uses few queries but each with a relatively large hypothesis, and then we improve our solution using a "divide & conquer" algorithm which uses more queries (though still attribute-efficient) yet each of the queries and the output hypothesis are fairly small. These algorithms may be used also in order to overcome corrupt oracles for classes that are not attribute-efficiently learnable (and then the learning remains not attribute-efficient). Thus, the algorithms in this subsection are described as using **any** basic learning algorithm. Let  $\mathbf{ALG}_{\mathbf{EQ}}$  be such an EXACT learning algorithm for the concept class  $C_n^r$  (possibly  $r = n$ ), with time, equivalence query and hypotheses-size complexities of  $\psi_t(n, r)$ ,  $\psi_e(n, r)$  and  $\psi_s(n, r)$ , respectively, for some functions  $\psi_t, \psi_e, \psi_s : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ .

**Deterministic Brute-Force Algorithm.** When letting  $\mathbf{ALG}_{\mathbf{EQ}}$  run with access to a **malicious** equivalence oracle, we can be certain that if the algorithm runs more than  $\psi_t(n, r)$  steps, or asks more than  $\psi_e(n, r)$  queries (or gets stuck), then it was misled by the oracle in at least one of the received counterexamples along the run. If we could have discriminated between the corrupt and non-corrupt counterexamples, then we would run the same algorithm again, but this time add to any hypothesis on which we ask an equivalence query, a "table" with all the corrupt counterexamples from the previous run as "exceptions" to the hypothesis, thus preventing the oracle from lying again on these examples. After repeating this procedure at most  $L$  times, the oracle cannot lie any more, since all the corrupt counterexamples are classified by all the hypotheses during the  $L + 1$  run according to their corrupt value in  $E$ , and thus we succeed to obtain the correct hypothesis. Since we do not have a way to locate the erroneous examples among all the seen examples, then we attach to our queried hypotheses a table with **all** the past examples. In this way, any additional counterexample is inevitably a new counterexample. The algorithm  $\mathbf{BruteForce}_{\mathbf{MEQ}}$  is detailed in Algorithm 1.

---

**Algorithm 1.** Algorithm **BruteForce<sub>MEQ</sub>** - Deterministic learning with **malicious equivalence** queries

---

```

1:  $T \leftarrow \emptyset$  // The set of all past counterexamples.
2: For  $i = 1, \dots, L + 1$  do
3:   Run ALGEQ, and whenever it asks  $\text{EQ}_f(h)$  ask  $\text{MEQ}_f([h, T])$  instead.
4:   If ALGEQ has found a hypothesis  $h$  for which  $\text{MEQ}_f([h, T]) = \text{“Yes”}$  then
5:     Return  $[h, T]$ .
6:   else
7:     Let  $T' \subseteq \{0, 1\}^n \times \{0, 1\}$  be the set of all received counterexamples.
8:      $T \leftarrow T \cup T'$ 
9:   end If
10: end For

```

---

**Lemma 1.**  $C_n^r$   $O(\psi_t(n, r)L)$   $O(\psi_e(n, r)L)$   $O(\psi_s(n, r) + \psi_e(n, r)L)$

**Corollary 1.**  $I(n)$   $C_n^r$   $O(\psi_t(n, r))$   $O(\psi_e(n, r))$   $O(\psi_s(n, r))$   $O(I(n)L)$   $C_n^r$   $O(\psi_t(n, r)L)$   $O(\psi_e(n, r)L)$  **malicious equivalence**  $O(\psi_s(n, r) + \psi_e(n, r)L)$

**Probabilistic Divide & Conquer Algorithm.** The algorithm consists of two nested algorithms: **Divide<sub>MEQ</sub>** and **Conquer<sub>MEQ</sub>**. It partitions the instance space to several disjoint subsets (“Divide”), defined by a set of boolean restrictions  $\rho_1, \rho_2, \dots, \rho_m$ , and learns the target function  $f$  by learning separately each of  $f$ ’s restrictions  $f_1, f_2, \dots, f_m$  to the subsets (“Conquer”). The final hypothesis classifies each instance using the restriction to which it belongs, and thus is defined as  $h \stackrel{\text{def}}{=} (\rho_1 \wedge f_1) \vee (\rho_2 \wedge f_2) \vee \dots \vee (\rho_m \wedge f_m)$  (a “conditioned” disjunction). This method proves particularly useful when it is easier to learn each of the separate restrictions of  $f$  than to learn the complete  $f$ . In our case, we would partition the instance space in a way that, with high probability, each subset would include no more than a single corrupt instance, and then learn the separate restrictions of  $f$  to each of these subsets, using a simpler algorithm for learning with at most one corrupt example (i.e.  $L = 1$ ).

$L = 1$ , **Conquer<sub>MEQ</sub>** When the number of corrupt examples is at most one, we may run **ALG<sub>EQ</sub>** until it gets the answer “Yes” or until it exceeds its complexity, which then implies that necessarily one of the oracle’s counterexamples was maliciously incorrect. In the later case we rerun the algorithm, but this time assume that the first counterexample was corrupt and ask all the equivalence queries on the same hypotheses as **ALG<sub>EQ</sub>** would ask, but with this counterexample attached, which prevents the oracle from repeating this lie again. In this way, the rest of the algorithm’s run is assumed to be corruptions free, and thus if the first example was indeed the corrupt one, then the algorithm would find the target function in this run with at most  $O(\psi_e(n, r))$  queries and  $O(\psi_t(n, r))$  time. If also this run exceeds its supposed complexities,



then we know that it wasn't the first example that was corrupt, and we continue searching for the false counterexample until **ALG<sub>EQ</sub>** finally succeeds. The conquering algorithm is detailed in Algorithm 2.

---

**Algorithm 2.** Algorithm **Conquer<sub>MEQ</sub>** - Learning with **malicious equivalence** queries, when at most one counterexample is malicious

---

```

1: Run ALGEQ.
2: If ALGEQ has found a hypothesis  $h$  for which  $\text{MEQ}_f(h) = \text{"Yes"}$  then
3:   Return  $h$ .
4: else
5:   Let  $\{e_1 \stackrel{\text{def}}{=} (x_1, f(x_1)), \dots, e_q \stackrel{\text{def}}{=} (x_q, f(x_q))\}$  be the set of all received counterexamples.
6:   For  $i = 1, \dots, q$  do
7:     Run ALGEQ, and whenever it asks  $\text{EQ}_f(h)$  ask  $\text{MEQ}_f([h, \{e_i\}])$  instead.
8:     If ALGEQ has found a hypothesis  $h$  for which  $\text{MEQ}_f([h, \{e_i\}]) = \text{"Yes"}$  then
9:       Return  $[h, \{e_i\}]$ 
10:    end If
11:  end For
12: end If

```

---

**Lemma 2.**  $L \leq 1$   $\text{Conquer}_{\text{MEQ}} \leq C_n^r \cdot O(\psi_t(n, r)\psi_e(n, r)) \leq O(\psi_e(n, r)^2) \leq O(\psi_s(n, r))$

$L > 1$  **Divide<sub>MEQ</sub>** When running **Conquer<sub>MEQ</sub>**, if we finish tracing all the branches of the tree and in each branch we exceeded the (time or query) complexity bound, then we may deduce that the oracle has lied at least twice in at least one branch: the branch for which we “corrected” a truly corrupt counterexample and nevertheless haven’t got a good hypothesis. Thus we would like to divide the instance space arbitrarily to two equal-sized subsets, such that with high probability, the (at least) two corrupt examples would be separated by this division, and as a consequence, the bound on the allowed number of corruptions in each of  $f$ ’s restrictions would be reduced. The division is done by defining a boolean function  $\rho$ , and using it to partition the instance space to two complementary sets:  $\{a \in \{0, 1\}^n \mid \rho(a) = 1\}$ ,  $\{a \in \{0, 1\}^n \mid \bar{\rho}(a) = 1\}$ . Following this division we rerun **Conquer<sub>MEQ</sub>** on each of the two complementary restrictions in parallel, ask equivalence queries that are the conditioned disjunctions of the two restrictions’ hypotheses, i.e.  $\text{MEQ}((\rho \wedge h_\rho) \vee (\bar{\rho} \wedge h_{\bar{\rho}}))$ , and return the counterexample of each such query to one of the two parallel runs of **Conquer<sub>MEQ</sub>**, according to the restriction to which the counterexample belongs. Any restriction in which we fail is redivided, and **Conquer<sub>MEQ</sub>** is rerun in each of the subdivisions, until we get disjoint restrictions with no more than one corrupt example in each.

How do we partition each restriction to two such subsets? We would like it to be partitioned in a way that no adversary would be able to put all the

corrupt examples in only one of the partitions. In order to avoid an adversarial behavior which maliciously fits the choice of corrupt examples to the algorithm’s partitioning method, we should partition the set randomly. The simplest way to arbitrarily bisect any instance set is to define the partitioning function as the parity of some randomly chosen subset of the instances’ bits. The division policy of the algorithm (detailed in Algorithm 3) leads to an expected number of  $O(L)$  divisions, which implies the following lemma (proved in the full paper):

---

**Algorithm 3.** Algorithm  $\text{Divide}_{\text{MEQ}}$  - Probabilistic “Divide & Conquer” learning with **malicious equivalence queries**

---

- 1:  $\rho_0 \leftarrow 1, \Gamma \leftarrow \{\rho_0\}$
  - 2: Run  $\text{Conquer}_{\text{MEQ}}^{\rho_0}$  until it asks  $\text{MEQ}_f(h_{\rho_0})$
  - 3: Let  $a \in \{0, 1\}^n \cup \{\text{“Yes”}\}$  be the received answer.
  - 4: **while**  $a \neq \text{“Yes”}$  **do**
  - 5:   Let  $\rho \in \Gamma$  be the one for which  $\rho(a) = 1$ .
  - 6:   Return  $a$  as a counterexample to  $\text{Conquer}_{\text{MEQ}}^\rho$ , and continue running until it asks  $\text{MEQ}_f(h_{\rho_i})$ .
  - 7:   **If**  $\text{Conquer}_{\text{MEQ}}^\rho$  failed **then**
  - 8:     – Choose randomly  $S \subseteq \{1, \dots, n\}$
  - 8:     – Define a new restriction:  $\rho^* \leftarrow \bigoplus_{i \in S} x_i$
  - 8:     –  $\rho_1 \leftarrow \rho \wedge \rho^*, \quad \rho_2 \leftarrow \rho \wedge \overline{\rho^*}$
  - 8:     –  $\Gamma \leftarrow \Gamma \setminus \{\rho\} \cup \{\rho_1, \rho_2\}$
  - 8:     – Run algorithms:  $\text{Conquer}_{\text{MEQ}}^{\rho_1}$  and  $\text{Conquer}_{\text{MEQ}}^{\rho_2}$ , until they ask  $\text{MEQ}_f(h_{\rho_1})$  and  $\text{MEQ}_f(h_{\rho_2})$  respectively.
  - 9:   **end If**
  - 10:   Ask  $\text{MEQ}_f(\bigvee_{\rho \in \Gamma} (\rho \wedge h_\rho))$ ; Let  $a \in \{0, 1\}^n \cup \{\text{“Yes”}\}$  be the received answer.
  - 11: **end while**
  - 12: Return  $\bigvee_{\rho \in \Gamma} (\rho \wedge h_\rho)$
- 

**Lemma 3.** *Let  $\text{Divide}_{\text{MEQ}}$  be the algorithm  $\text{Divide}_{\text{MEQ}}$  with  $C_n^r$  hypotheses. Then its size complexity is  $O(\psi_t(n, r)\psi_e(n, r)L)$  and its query complexity is  $O(\psi_e(n, r)^2L)$ . Similarly,  $\text{Divide}_{\text{MEQ}}$  with  $C_n^r$  hypotheses has size complexity  $O(\psi_s(n, r)L)$ .*

The decision as to which of the above two algorithms to use is implied from the following lower bound on the queries’ size complexity, when learning attribute-efficiently with equivalence queries only:

**Theorem 2.** *Let  $\tilde{\text{A}}_{\text{EQ}}$  be some  $I(n)$ -attribute-efficient algorithm. Then its size complexity is  $\Omega(\frac{n}{I(n)\text{poly}(r, \text{size}(f))})$ .*

Let  $\tilde{\text{A}}_{\text{EQ}}$  be some  $I(n)$ -attribute-efficient algorithm, and let  $\tilde{\varphi}_s(n, r)$  be its size complexity. First observe that each of the  $n$  variables must appear in some of the queried hypotheses, since otherwise the algorithm might not include in its hypotheses some relevant variable, which implies that it cannot succeed. Since the query complexity, for any  $I(n)$ -attribute-efficient algorithm, is bounded by  $O(I(n)\text{poly}(r, \text{size}(f)))$ , then we have

$$O(I(n)poly(r, size(f))) \cdot \tilde{\varphi}_s(n, r) \geq n$$

which implies the required lower-bound for  $\tilde{\varphi}_s(n, r)$ .

**Corollary 2.**  $O(\sqrt{n}) \cdot \tilde{\varphi}_s(n, r) \geq \tilde{\varphi}_e(n, r) \geq \tilde{\varphi}_s(n, r)$

Which means that for learning classes, which are  $O(\sqrt{n})$ -attribute-efficient learnable, we shall prefer the **BruteForce**<sub>MEQ</sub> algorithm. For other classes, our decision depends on the priority we give to each of the relevant complexity parameters.

### 2.2 Learning with CMQ

Note that there is an easy way to overcome the problem of the corrupt oracles as defined above, when having access to a membership oracle over an  $n$ -dimensional instance space with few relevant variables ( $o(n)$ ). For any two conceptually equivalent instances  $x \equiv_f y$ , we have  $f(x) = f(y)$ , and since the size of each equivalence class is as large as  $2^{n-r}$ , this gives us a huge amount of redundancy in the instance space, and we may use this redundancy to find the true value of  $f$  on any instance. If the learner asks a membership query  $MMQ_f(x)$  (or  $LMQ_f(x)$ ), then in order to know the true value  $f(x)$ , he may ask additional  $2r + 2L$  membership queries on instances which differ from  $x$  in a single bit each, and take the majority of all the  $2r + 2L + 1$  answers. That is, if we denote by  $x^i$ , the instance  $x$  with its  $i$ 'th bit flipped, then the set of additionally queried instances consists of the  $x^i$ 's, for each index  $i$  from an arbitrarily chosen subset of  $\{1, \dots, n\}$ , of size  $2r + 2L$ . Since the oracle may return only  $L$  maliciously incorrect answers, and the true value of the answer may differ from  $f(x)$  in at most  $r$  queries (where we accidentally flipped a relevant bit), then necessarily at least  $r + L + 1$  answers will be equal to  $f(x)$  and thus the majority of the answers equals  $f(x)$ , as desired. The correctness of counterexamples received from a MEQ oracle can also be verified using membership queries in a similar way, and if some counterexample is found to be incorrect then we simply attach it to all subsequent equivalence queries.

Using this simple technique we get the following corollaries:

**Corollary 3.**  $I(n) \cdot O(\tilde{\varphi}_m(n, r)) \geq C_n^r \cdot O(\tilde{\varphi}_t(n, r)) \geq O(\tilde{\varphi}_e(n, r)) \geq O(\tilde{\varphi}_s(n, r))$   
 $O(\tilde{\varphi}_s(n, r)) \geq O(I(n)L) \cdot O(\tilde{\varphi}_t(n, r) + (r + L)\tilde{\varphi}_m(n, r)) \geq O((r + L)\tilde{\varphi}_m(n, r))$  **corrupt membership**  
 $O(\tilde{\varphi}_e(n, r))$  **standard equivalence**  
 $O(\tilde{\varphi}_s(n, r))$

**Corollary 4.**  $I(n) \cdot O(\tilde{\varphi}_m(n, r)) \geq C_n^r \cdot O(\tilde{\varphi}_t(n, r)) \geq O(\tilde{\varphi}_e(n, r)) \geq O(\tilde{\varphi}_s(n, r))$   
 $O(\tilde{\varphi}_s(n, r)) \geq O(I(n)L) \cdot O(\tilde{\varphi}_t(n, r) + (r + L)(\tilde{\varphi}_m(n, r) + \tilde{\varphi}_e(n, r))) \geq O((r + L)(\tilde{\varphi}_m(n, r) + \tilde{\varphi}_e(n, r)))$  **corrupt membership**  
 $O(\tilde{\varphi}_e(n, r))$  **corrupt equivalence**  
 $O(\tilde{\varphi}_s(n, r) + L)$

### 3 Learning with *Conceptually* Corrupt Oracles

As we defined the corrupt oracles thus far, an oracle might be inconsistent on instances that are actually conceptually equivalent. Such an oracle can be thought of as one that is **carelessly** mistaken, rather than one that does not know the concept well enough. As we have seen, such an oracle can be used for learning by simply asking enough queries to reveal his true knowledge about the concept. We would like to define oracles that are corrupt in the same sense that was studied in the literature for the case where all the variables are relevant (without redundancy); that is, oracles that are “conceptually wrong” by actually **not knowing** how the learned concept classifies some instances. Thus we define a *conceptually consistent* (or *conceptually consistent*) oracle to be such that is necessarily consistent on all conceptually equivalent instances.

In our new definition, for a given conceptually corrupt oracle, we would denote the number of conceptually different corrupt instances by  $l$ , and an  $I(n)$ -attribute-efficient learning algorithm is bounded to using query complexity of  $O(I(n)poly(r, size(f), l))$  ( $L$  is replaced by  $l$ ). Note that now the total number of corrupt instances is  $L = l \cdot 2^{n-r}$ , which is by far more than the number of corruptions that we can efficiently handle using the previously described “redundancy technique”. In fact, in this new setting, the technique cannot be practically used, even if we disregard computational issues, since the oracles are conceptually consistent, meaning there is no redundancy in the instance space.

We shall denote the conceptually limited and conceptually malicious membership oracles by CLMQ and CMMQ respectively. Consistent with the notations of *CEQ* and *CMQ*, where the ‘*C*’ stood for “Corrupt”, we denote the conceptually corrupt oracles by *CCEQ* and *CCMQ*, respectively. With the above new definition of the problem, we may now state a stronger theorem for the case of having access to a membership oracle as well:

**Theorem 3.** *Let  $f$  be a conceptually consistent,  $O(\log n)$ -attribute-efficient, conceptually corrupt oracles*

We will first show how this theorem can be proved with a simple combination of known algorithms, and then we present a novel algorithm which improves the complexity for the case of learning with a conceptually limited membership oracle (and a standard equivalence oracle). Note that the results in this paper that handle learning with membership queries are constrained to classes that are closed under projection. Nonetheless, almost all the classes considered in the literature are actually closed under projection, and thus the constraint can be practically disregarded.

#### 3.1 Learning with *CCEQ* and *CCMQ*

Two known algorithms would be combined (nested) to achieve attribute-efficient learning with conceptually corrupt oracles. The main algorithm we shall use is **FindRelevantEQ+MQ**, described below, to find the relevant variables. As

its sub-algorithm, it uses a “divide & conquer” learning algorithm, originally presented in [8], for non-attribute-efficient learning with corrupt oracles (denoted here by  $\mathbf{D\&C}_{\mathbf{CEQ+CMQ}}$ ).

An algorithm quite similar to  $\mathbf{FindRelevant}_{\mathbf{EQ+MQ}}$  was initially presented by Angluin, Hellerstein and Karpinski [2] for finding the signs of variables in a read-once formula. It would be used here for finding the  $r$  relevant variables, with respect to some unknown function  $f$ , among the total of  $n$  variables. This paves the way to using a **non-attribute-efficient** algorithm for learning the projection of the concept class to a partial assignment whose defined set consists of the irrelevant variables, and by this learning with an attribute-efficient complexity. This modification of the technique was already done by Blum, Hellerstein and Littlestone [9] for ONLINE learning, and similar principles are used here for the EXACT model.

The algorithm uses an existing non-attribute-efficient algorithm  $\mathbf{A}_{\mathbf{EQ+MQ}}$  (received as its parameter), to iteratively learn the class  $C_n^r$ , which is projected in each iteration on a partial assignment  $\pi$  with a decreasing defined set (since each newly found relevant variable is “freed” from the projection). Whenever algorithm  $\mathbf{A}_{\mathbf{EQ+MQ}}$  asks a membership query  $\text{MQ}_f(y)$ ,  $\mathbf{FindRelevant}_{\mathbf{EQ+MQ}}$  asks  $\text{MQ}_f(\pi(y))$  instead. For each asked equivalence query  $\text{EQ}_f(h)$ , if the received counterexample is  $a$ , then  $\mathbf{FindRelevant}_{\mathbf{EQ+MQ}}$  first asks a membership query  $\text{MQ}_f(\pi(a))$ . If  $f(\pi(a)) \neq h(a)$ , then since  $\pi(f)(a) = f(\pi(a))$ ,  $a$  is returned as a counterexample to  $\mathbf{A}_{\mathbf{EQ+MQ}}$ . Otherwise,  $f(\pi(a)) = h(a) \neq f(a)$ , and we have found the desired two instances  $a$  and  $\pi(a)$ , which differ on some **relevant** variables from the defined set of  $\pi$ , and whose classifications by  $f$  are opposite. At this point  $\mathbf{FindRelevant}_{\mathbf{EQ+MQ}}$  performs a simple binary-search (with membership queries), in order to find a single variable whose flipping also flips the received classification. The found variable is added to the set of relevant variables, removed from the defined set of  $\pi$ , and the learning restarts, only now the learned class is projected on the modified partial assignment  $\pi$ .

The algorithm is detailed in Algorithm 4, and has the following property [9]:

**Lemma 4.**  $\mathbf{A}_{\mathbf{EQ+MQ}}$  learns  $C_n$  using  $\varphi_t(n), \varphi_e(n), \varphi_m(n), \varphi_s(n) : \mathbb{N} \rightarrow \mathbb{N}$  queries.  $\mathbf{FindRelevant}_{\mathbf{EQ+MQ}}(\mathbf{A}_{\mathbf{EQ+MQ}})$  learns  $C_n^r$  using  $O(r\varphi_t(r)), O(r\varphi_m(r) + r \log n), O(r\varphi_e(r)), O(\varphi_s(r))$  queries.

The algorithm works also when its iterated algorithm learns using corrupt oracles. It should only be noted that when using CMMQ (or MMQ), the iterated algorithm should be able to handle also MEQ, since some of its counterexamples are from  $\mathbf{FindRelevant}_{\mathbf{EQ+CMMQ}}$  and rely on (possibly malicious) answers received from CMMQ (which might cause erroneous counterexamples).

Bisht, Bshouty and Houry [8] have proved that EXACT(MQ) learnability implies EXACT(MQ) learnability with corrupt oracles. Their algorithms (denoted here by  $\mathbf{D\&C}_{\mathbf{CEQ+CMQ}}$ ) achieve the result in the following Theorem (In their paper they actually need an additional assumption, regarding the knowl-

---

**Algorithm 4.** Algorithm **FindRelevant**<sub>EQ+MQ</sub>(**A**<sub>EQ+MQ</sub>) (based on [2] and [9]) - Learning attribute-efficiently while finding the relevant variables of the target function

---

- 1:  $R \leftarrow \emptyset$
  - 2: Arbitrarily choose an assignment  $\pi \in \{0, 1\}^n$ .
  - 3: **loop**
  - 4: Run **A**<sub>EQ+MQ</sub> on the projected class  $\pi(C_n^r)$ , with the following changes:
    - If it asks  $\text{MQ}_f(y)$ , ask  $\text{MQ}_f(\pi(y))$  instead.
    - If it asks  $\text{EQ}_f(h)$ , let  $a \in \{0, 1\}^n \cup \{\text{“Yes”}\}$  be the received answer:
      - If  $a = \text{“Yes”}$ , return  $h$ .
      - Otherwise, ask  $\text{MQ}_f(\pi(a))$  and let  $\sigma$  be the received answer:
        - \* If  $\sigma = \overline{h(a)}$ , return  $a$  as a counterexample to **A**<sub>EQ+MQ</sub>.
        - \* Otherwise, call **BinarySearch**<sub>MQ</sub>(( $a, \overline{h(a)}$ ), ( $\pi(a), \sigma$ )), and let  $\{x_r\}$  be the returned variable.  $R \leftarrow R \cup \{x_r\}$ ,  $\pi_r = x_r$ .
  - 5: **end loop**
- 

edge of some parameter’s value, in order to achieve this result, but in the full paper we show how to derive the same result without this extra assumption):

**Theorem 4.** Let  $C_n$  be a class of functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  with complexities  $O(\varphi_t(n))$ ,  $O(\varphi_m(n))$ ,  $O(\varphi_e(n))$ ,  $O(\varphi_s(n))$ ,  $O(nl\varphi_t(n))$ ,  $O(nl\varphi_m(n))$ ,  $O(nl\varphi_e(n))$ ,  $O(nl\varphi_s(n))$ ,  $O(l\varphi_s(n))$  *corrupt membership*,  $O(nl\varphi_e(n))$  *corrupt equivalence*, and  $O(l\varphi_s(n))$ .

And thus, using **FindRelevant**<sub>CEQ+CCMQ</sub>(**D&C**<sub>CEQ+CMQ</sub>), we get an immediate corollary from the above theorem and Lemma 4, which proves our main statement (Theorem 3):

**Corollary 5.** Let  $C_n^r$  be a class of functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  with complexities  $O(\varphi_t(n))$ ,  $O(\varphi_m(n))$ ,  $O(\varphi_e(n))$ ,  $O(\varphi_s(n))$ ,  $O(\log n)$ -*attribute-efficient*,  $O(r^2l\varphi_t(r))$ ,  $O(r^2l\varphi_m(r) + r \log n)$ ,  $O(r^2l\varphi_e(r))$ ,  $O(r^2l\varphi_s(r))$ ,  $O(l\varphi_s(r))$  *conceptually corrupt membership*,  $O(r^2l\varphi_e(r))$  *conceptually corrupt equivalence*, and  $O(l\varphi_s(r))$ .

### 3.2 Improved Learning with CLMQ

We shall now see how to improve the above general complexity for the specific case of learning with a **conceptually limited membership** oracle (and a standard equivalence oracle). First we present an algorithm for handling LMQ when all the bits are relevant, and then we run it “inside” **FindRelevant**<sub>EQ+CLMQ</sub> in order to achieve an attribute-efficient learning with CLMQ. Let **A**<sub>EQ+MQ</sub> be an algorithm that learns  $C_n$  (non attribute-efficiently), with complexities  $\varphi_t(n)$ ,  $\varphi_e(n)$ ,  $\varphi_m(n)$  and  $\varphi_s(n)$ , for some functions  $\varphi_t, \varphi_e, \varphi_m, \varphi_s : \mathbb{N} \rightarrow \mathbb{N}$ .

### Deterministic Divide & Conquer LMQ Algorithm

$L = 1$  **Conquer**<sub>EQ+LMQ</sub> The algorithm runs **A**<sub>EQ+MQ</sub> until it gets  $\perp$  as the answer of some membership query  $\text{CLMQ}_f(y)$ , and then it splits to two parallel runs **A**<sub>EQ+MQ</sub><sup>0</sup> and **A**<sub>EQ+MQ</sub><sup>1</sup>. Each of the algorithm's copies **A**<sub>EQ+MQ</sub> <sup>$\sigma$</sup>  assumes that the answer to  $\text{CLMQ}_f(y)$  is  $\sigma$ . Since we assume that  $l = 1$  (and that there are no irrelevant variables, which means that also  $L = 1$ ), from that point, all the membership queries other than  $\text{CLMQ}_f(y)$  will return a definite answer - either 0 or 1. This means that at least one of the two runs must succeed. Note that, when learning in the strict setting, either one of the algorithms would fail, or both algorithms would output equivalent hypotheses, since the final hypothesis must satisfy  $h \equiv f$ .

**Lemma 5.**  $L \leq 1$  **Conquer**<sub>EQ+LMQ</sub>  $C_n$   $O(\varphi_t(n))$ ,  $O(\varphi_m(n))$ ,  $O(\varphi_e(n))$ ,  $O(\varphi_s(n))$

$L > 1$  **Divide**<sub>EQ+LMQ</sub> The algorithm begins by trying to run **Conquer**<sub>EQ+LMQ</sub> on the given concept class  $C_n^r$ . If it receives two  $\perp$  answers, say for  $\text{CLMQ}_f(y)$  and  $\text{CLMQ}_f(z)$ , then let  $x_i \in V_n$  be any variable such that  $y_i \neq z_i$ . **Divide**<sub>EQ+LMQ</sub> divides the instance space according to the value of this  $i$ 'th bit, and reruns two parallel copies of **Conquer**<sub>EQ+LMQ</sub> on each of the corresponding projections of  $C_n^r$ . The division of the instance space is done by projecting it on  $x_i = 0$  and  $x_i = 1$ . If any of these runs receives two  $\perp$  answers, then we halt it, and split the corresponding projection again in the same manner. Note that by partitioning the instance space according to a relevant variable that separates two  $\perp$  answers, we can be sure that each of the partitions contains at least one corruption less than before the partition.

If some copy of algorithm **Conquer**<sub>EQ+LMQ</sub> <sup>$\lambda$</sup> , which learns over a partition of  $\{0, 1\}^n$  defined by the partial assignment  $\lambda$ , asks  $\text{CLMQ}_f(y)$ , then we ask  $\text{CLMQ}_f(\lambda(y))$  instead. If it asks  $\text{EQ}_f(h_\lambda)$ , we wait until all the copies of the algorithm ask their equivalence queries  $\text{EQ}_f(h_{\lambda_1}), \dots, \text{EQ}_f(h_{\lambda_m})$ , and then we ask  $\text{EQ}_f((T^{\lambda_1} \wedge h_{\lambda_1}) \vee \dots \vee (T^{\lambda_m} \wedge h_{\lambda_m}))$  and return the received counterexample  $a$  to **Conquer**<sub>EQ+LMQ</sub> <sup>$\lambda_i$</sup>  for which  $T^{\lambda_i}(a) = 1$  (similar to the way it was done in the divide & conquer algorithm for MEQ).

**Lemma 6.** **Divide**<sub>EQ+LMQ</sub>  $C_n$   $O(L\varphi_t(n))$ ,  $O(L\varphi_m(n))$ ,  $O(L\varphi_e(n))$ ,  $O(L\varphi_s(n))$

**Corollary 6.**  $C_n$   $O(\varphi_t(n))$ ,  $O(\varphi_m(n))$ ,  $O(\varphi_e(n))$ ,  $O(\varphi_s(n))$ ,  $C_n$   $O(L\varphi_t(n))$ ,  $O(L\varphi_m(n))$  **limited membership**,  $O(L\varphi_e(n))$ ,  $O(L\varphi_s(n))$

In the full paper we improve this result dramatically (to only an **additive** factor of  $nL$  membership queries) for the specific case of learning CDNF functions (which contains also the concept class of decision trees), using a modification of the "monotone theory" of Bshouty [10].

Observe that using **FindRelevant**<sub>EQ+CLMQ</sub> with **Divide**<sub>EQ+LMQ</sub> as its iterated learning algorithm, we get an improved attribute-efficient learning algorithm with a **conceptually limited** membership oracle. From Lemmas 4 and 6 we get the complexity of the combined algorithm and then a general corollary (which significantly improves the more generic result of Corollary 5):

**Lemma 7.**  $C_n^r$   $O(\varphi_t(n))$   $O(\varphi_m(n))$   $O(\varphi_e(n))$   $O(\varphi_s(n))$   $O(\log n)$ -**attribute-efficient**  $O(r\varphi_t(r))$   $O(r\varphi_m(r) + r \log n)$   $O(r\varphi_e(r))$   $O(l\varphi_s(r))$

**Corollary 7.**  $C_n$   $O(\varphi_t(n))$   $O(\varphi_m(n))$   $O(\varphi_e(n))$   $O(\varphi_s(n))$   $O(\log n)$ -**attribute-efficient**  $C_n^r$   $O(r\varphi_t(r))$   $O(r\varphi_m(r) + r \log n)$  **conceptually limited membership**  $O(r\varphi_e(r))$   $O(l\varphi_s(r))$

## 4 Conclusions

The major contribution of this paper is the presentation of a completely generic adaptation of attribute-efficient algorithms to learning with corrupt oracles as well. The main question that has still remained open is related to the case of  $\varphi$ -learning; that is, learning with hypotheses that are constrained to be from the learned concept class. Our study uses "improperness" extensively, both in the "attachment technique", where we attach a table of previous counterexamples to the asked hypothesis, and both in the "divide & conquer" technique, by using hypotheses that are disjunctions of distinct hypotheses for the divided instance space. In both of these methods we allow ourselves to use hypotheses that are not necessarily from the learned concept class, which arises the question whether our results could also be achieved for the proper case.

## References

- [1] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, 1987.
- [2] Dana Angluin, Lisa Hellerstein, and Marek Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.
- [3] Dana Angluin and Mārtiņš Krikis. Learning with malicious membership queries and exceptions (extended abstract). In *COLT '94: Proceedings of the seventh annual conference on Computational learning theory*, pages 57–56. ACM Press, 1994.
- [4] Dana Angluin, Mārtiņš Krikis, Robert H. Sloan, and György Turán". Malicious omissions and errors in answers to membership queries. *Machine Learning*, 28:211–255, 1997.



- [5] Dana Angluin and Philip Laird. Learning From Noisy Examples. *Machine Learning*, 2(4):343–370, 1988.
- [6] Peter Auer and Nicolò Cesa-Bianchi. On-Line Learning with Malicious Noise and the Closure Algorithm. *Ann. Math. Artif. Intell.*, 23(1-2):83–99, 1998.
- [7] Peter Auer and Philip M. Long. Simulating Access to Hidden Information while Learning. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(67), 2000.
- [8] Laurence Bisht, Nader H. Bshouty, and Lawrance Houry. Learning with Errors in Answers to Membership Queries (Extracted Abstract). In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 611–620. IEEE Computer Society, 2004.
- [9] Blum, Hellerstein, and Littlestone. Learning in the Presence of Finitely or Infinitely Many Irrelevant Attributes (Extended Abstract). In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1991.
- [10] Nader H. Bshouty. Exact learning Boolean functions via the monotone theory. *Inf. Comput.*, 123(1):146–153, 1995.
- [11] Nader H. Bshouty. Simple learning algorithms using divide and conquer. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 447–453. ACM Press, 1995.
- [12] Nader H. Bshouty and Lisa Hellerstein. Attribute-Efficient Learning in Query and Mistake-Bound Models. *journal = "Journal of Computer and System Sciences*, 56(3):310–319, 1998.
- [13] Nick Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning*, 2(4):285–318, 1988.
- [14] Yasubumi Sakakibara. On learning from queries and counterexamples in the presence of noise. *Inf. Process. Lett.*, 37(5):279–284, 1991.
- [15] Robert H. Sloan and Gyorgy Turan. Learning with Queries but Incomplete Information (Extended Abstract). In *Computational Learning Theory*, pages 237–245, 1994.
- [16] Leslie G. Valiant. A Theory of the Learnable. *Commun. ACM*, 27(11):1134–1142, 1984.
- [17] Leslie G. Valiant. Projection learning. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 278–293. ACM Press, 1998.

# Learning DNF by Statistical and Proper Distance Queries Under the Uniform Distribution

Wolfgang Lindner

Fakultät für Informatik, Universität Ulm

**Abstract.** We show that  $s$ -term DNF formulas can be learned under the uniform distribution in quasi-polynomial time with statistical queries of tolerance  $\Omega(\varepsilon/s)$ . The tolerance improves on the known tolerance  $\Omega(\varepsilon^2/s)$  and is optimal with respect to its dependence on the error parameter  $\varepsilon$ . We further consider the related model of learning with *proper* distance queries and show that DNF formulas can be learned under the uniform distribution with quasi-polynomial queries, where the hypotheses are DNF formulas of polynomial size. Finally we consider the class of majorities over DNF formulas and provide polynomially related upper and lower bounds for the number of distance queries required to learn this class.

## 1 Introduction

The learnability of DNF formulas in Valiant's PAC learning model [17] is one of the main open questions in algorithmic learning theory. There are, however, various algorithms for learning DNF formulas, including Jackson's Harmonic Sieve [10] for learning DNF formulas in polynomial time, which, however, works only with respect to the uniform distribution and with the additional help of membership queries, and the recent algorithm of Klivans and Servedio [13], which is an algorithm in Angluin's model of exact learning with equivalence queries [1], but requires sub-exponential time.

In the original PAC model, it is assumed that the classification of the random samples are correctly delivered to the learning algorithm. In practice, however, this might be difficult to achieve, and therefore one can find several learning models in the literature for learning with noise. A particularly attractive model in this setting is the model of learning by statistical queries [12]. Here, the learning algorithm may ask for the values of statistics on the distribution of labeled examples, and receives the requested statistics to within an additive error specified by the algorithm. The additive error is also called the  $\varepsilon$ -error of the algorithm.

Concerning the learnability of DNF formulas, it is known that  $s$ -term DNF formulas can be learned by statistical queries in time  $n^{O(\log(s/\varepsilon))}$  with tolerance  $\Omega(\varepsilon^2/s)$  [14]. The algorithm is based on a well-known fact concerning the Fourier spectrum of DNF formulas [5], which immediately leads to a simple weak learning

algorithm for DNF. Applying Freund's boosting algorithm [7], which has been analysed in the model of learning by statistical queries by Aslam and Decatur [2], yields a learning algorithm for arbitrary error bounds  $\varepsilon$ . As pointed out in [14], the query complexity of the algorithm is close to the lower bound  $n^{\Omega(\log(s))}$ , which holds whenever the tolerance is at least  $1/n^{\Omega(\log(s))}$ , and  $s \leq 2\sqrt{n}$ .

In this paper we show that the tolerance can be improved to  $\Omega(\varepsilon/s)$ . It is not hard to see that any algorithm for learning the class of  $s$ -term DNF formulas requires tolerance  $O(\varepsilon)$ , even when the number of terms  $s$  is constant. Thus, the tolerance  $\Omega(\varepsilon/s)$  is optimal with respect to its dependence on  $\varepsilon$ . To achieve this improvement we give a detailed analysis of a simplified variant of Servedio's smooth boosting algorithm [16] in the model of learning by statistical queries.

Learning by statistical queries is closely related to the model of learning by distances [4]. In fact, when restricted to distances of the form  $d(f, g) = \Pr_D[f(x) \neq g(x)]$ , for a distribution  $D$ , then both models are (essentially) equivalent. Thus, DNF formulas are learnable by  $n^{O(\log(s/\varepsilon))}$  distance queries with tolerance  $\Omega(\varepsilon/s)$ . In the context of exact learning by queries, it is an open question whether DNF formulas can be learned by a polynomial number of proper equivalence and membership queries, even when there is no restriction on the computational resources of the algorithm. It is therefore an interesting question whether DNF formulas can be learned by proper distance queries, i.e., distance queries with DNF formulas of related size as hypotheses.

Unfortunately, we are not able to transform the queries used by the algorithm above into proper distance queries. While it is easy to see that the weak learning step requires only distance queries with DNF formulas with polynomial terms, this property is destroyed by the subsequent boosting step. Here we show that we can avoid the boosting step so that  $s$ -term DNF formulas can be learned by proper distance queries, where the number of queries is still  $n^{O(\log(s/\varepsilon))}$ . When the error  $\varepsilon$  is set to a constant, the hypotheses of the queries are DNF formulas with polynomial in  $s$  terms. Unfortunately, the required tolerance is no longer polynomial in  $\varepsilon/s$ . It is, however, still large enough so that the lower bound  $n^{\Omega(\log(s))}$  on the number of queries applies. The algorithm is based on known bounds on the Fourier spectrum of DNF formulas with terms of bounded size due to Mansour [15]. This enables us to achieve strong learning without the subsequent boosting step.

We also consider the setting when the learning algorithm may ask queries of tolerance zero. In this setting, we can show that DNF can be exactly learned by a polynomial number of proper distance queries of polynomial size. For this we use an algorithm for learning DNF with projective equivalence queries due to Balcázar et al. [3].

Finally we consider majorities over DNF formulas with terms of bounded size. We show that boosting can be used to learn this class with proper distance queries. If  $\varepsilon$  is constant, the number of DNF formulas of the majority is polynomial in  $n$ , and the size of the terms is logarithmic in  $n$ , then both the number of queries and the tolerance is quasi-polynomial and inverse quasi-polynomial in  $n$ , respectively. The hypotheses are majorities over a quasi-polynomial num-

ber of DNF formulas with terms of poly-logarithmic size. We also provide a corresponding lower bound on the number of queries required to learn this class.

## 2 Preliminaries

In this paper we consider only boolean concept classes  $C$ . Throughout this paper, we think of boolean functions as mappings from  $\{0, 1\}^n$  to  $\{-1, 1\}$ .

In the [learning model](#) [12] the learning may ask queries of the form  $(\chi, \tau)$ , where  $\chi$  is a mapping from  $\{0, 1\}^n \times \{-1, 1\}$  to  $\{-1, 1\}$ , and  $\tau$  is a tolerance parameter of the query with  $0 \leq \tau \leq 1$ . The answer provided by the learning oracle to the query  $(\chi, \tau)$  is an estimate of the probability  $\Pr_D[\chi(x, f(x)) = 1]$  within additive error  $\tau$ , where  $f$  is the target and  $x$  is randomly chosen according to the distribution  $D$ .

A set of boolean functions  $C$  is [learnable](#) by  $d$  statistical queries of tolerance  $\tau$  with respect to a distribution  $D$  if there is an learning algorithm  $A$  such that for all  $f \in C$  and for any error parameter  $\varepsilon$ ,  $0 \leq \varepsilon \leq 1$ , after at most  $d$  statistical queries  $(\chi, \tau')$  with  $\tau' \geq \tau$ , which are answered by the learning oracle with respect to  $f$  and  $D$ , the learning algorithm  $A$  outputs a hypothesis  $h$  with [error](#) at most  $\varepsilon$ , i.e.,  $h$  satisfies  $\Pr_D[h(x) \neq f(x)] \leq \varepsilon$ . If the hypothesis  $h$  produced by  $A$  only satisfies  $\Pr_D[h(x) \neq f(x)] \leq 1/2 - \gamma$ , for some  $\gamma$  with  $0 \leq \gamma \leq 1/2$ , then we say that  $C$  is [weakly learnable](#) with [margin](#)  $\gamma$ .

In the [distance query model](#) [4], we think of concepts as points in a metric space  $(H, d)$ . Then a distance query is a pair  $(h, \tau)$ , where  $h$  is some function from  $H$  which we refer to as the [hypothesis](#), and  $\tau$  is a tolerance parameter as above. As the response to this query the learning oracle provides an estimate of the distance  $d(h, f)$  within additive error  $\tau$ , where  $f$  is a target from a concept class  $C \subseteq H$ . In this paper, we consider only boolean functions as concepts, and distances of the form  $d(f, g) = \Pr_D[f(x) \neq g(x)]$ , where  $x$  is randomly chosen according to some distribution  $D$ . Learnability and weak learnability by distance queries are defined as above for learning with statistical queries.

Obviously, in our particular setting, a distance query with hypothesis  $h$  is equivalent to a statistical query with  $\chi$  defined by  $\chi(x, b) = 1$  if and only if  $h(x) \neq b$ . On the other hand, any statistical query can be decomposed into two distance queries [5, 6]. In particular, it can be shown that for every statistical query  $\chi$  and target function  $f$  it holds that

$$\Pr_D[\chi(x, f(x)) = 1] = \frac{1}{2} (\Pr_D[\chi(x, -1) \neq f(x)] - \Pr_D[\chi(x, 1) \neq f(x)] + \Pr_D[\chi(x, -1) = 1] + \Pr_D[\chi(x, 1) = 1]) .$$

This means that any statistical query can be simulated by two distance queries with hypotheses of the form  $\chi(x, -1)$  and  $\chi(x, 1)$ . The probabilities  $\Pr_D[\chi(x, -1) = 1]$  and  $\Pr_D[\chi(x, 1) = 1]$  do not depend on the target and can either be determined exactly, in which case it is sufficient to use the same tolerance for the distance queries as for the simulated statistical query. More efficiently, however,

is to estimate the latter two probabilities by random sampling, in which case we need to ask the two distance queries with a slightly better tolerance. The analysis for this case is standard, and therefore we ignore this issue throughout the paper. Note, however, that for this reason we use probabilistic learning algorithms rather than deterministic ones in the learning by distances model.

In order to describe the Fourier transform of a boolean function  $f$  we write  $[n]$  to denote the set  $\{1, \dots, n\}$ . Then, for any set  $A \subseteq [n]$ , the parity function  $\chi_A$  is defined as  $\chi_A(x) = (-1)^{\sum_{i \in A} x_i}$ . Every real-valued function  $f$  over  $\{0, 1\}^n$  can be uniquely expressed as a linear combination  $f(x) = \sum_{A \subseteq [n]} \hat{f}(A) \chi_A(x)$ , where  $\hat{f}(A) = E[f(x) \chi_A(x)]$  for a uniformly distributed  $x$ . The coefficients  $\hat{f}(A)$  are also known as the *Fourier coefficients* of  $f$ .

The  $L_\infty$ -norm of a real-valued function  $f$  over  $\{0, 1\}^n$  is defined by  $L_\infty(f) = \max_{x \in \{0,1\}^n} |f(x)|$ .

### 3 Learning DNF by Statistical Queries via Hypotheses Boosting

In this section we show how to learn DNF by statistical queries with tolerance  $\Omega(\varepsilon/s)$ . For this we analyse a simplified variant of Servedio’s smooth boosting algorithm [16] in the model of learning by statistical queries. We start by providing some background on boosting.

Let  $A$  be a weak learning algorithm with advantage  $\gamma$ , query complexity  $N$ , and required tolerance  $\tau$  with  $0 < \tau < 1$ . Furthermore, let  $D$  be some fixed distribution,  $f$  be the target, and  $\varepsilon$  be some error bound with  $0 < \varepsilon < 1$ . A *boosting algorithm* [11] is a boosting algorithm  $B$  which proceeds in stages. In the first stage,  $B$  runs the weak learning algorithm  $A$  to produce a hypothesis  $h_1$  with error rate at most  $1/2 - \gamma$  with respect to the fixed distribution  $D$ . In each proceeding stage  $i + 1$ ,  $B$  runs  $A$  to produce a hypothesis  $h_{i+1}$  with error rate at most  $1/2 - \gamma$  with respect to some distribution  $D_{i+1}$ , where  $D_{i+1}$  depends in some way on the hypotheses  $h_1, \dots, h_i$  produced before stage  $i + 1$ . The boosting algorithm  $B$  has only access to the learning oracle with respect to the fixed distribution  $D$ . Hence, to run  $A$  with respect to the modified distribution  $D_{i+1}$  amounts to simulate the learning oracle with respect to  $D_{i+1}$  by queries to the learning oracle with respect to  $D$ . After  $B$  meets a certain abort condition in some stage  $i + 1$ ,  $B$  combines the hypotheses  $h_1, \dots, h_i$  in some way to obtain a final hypothesis  $h$  of error rate at most  $\varepsilon$  with respect to the target distribution  $D$ . In this paper we consider only boosting algorithms whose final hypothesis is the majority over the hypotheses  $h_1, \dots, h_i$  produced thus far.

Here we consider distributions  $D_{i+1}$  which are based on weighting schemes  $M_i$  defined as follows. Let  $N_i(x) = \sum_{j=1}^i h_j(x) f(x)$ . Then  $M_i$  is defined as

$$M_i(x) = \begin{cases} 1 & \text{if } N_i(x) < 0 \\ (1 - \gamma)^{N_i(x)} & \text{if } N_i(x) \geq 0 \end{cases},$$

and the distribution  $D_{i+1}$  induced by  $M_i$  is

$$D_{i+1}(x) = \frac{M_i(x)D(x)}{\sum_y M_i(y)D(y)} .$$

In the following we will refer to the denominator  $\mu_i = \sum_y M_i(y)D(y)$  as the relative size of  $M_i$ . For the first stage we let  $N_0(x) = 0$  and, consequently,  $M_0(x) = 1$  and  $D_1 = D$ .

Before  $B$  runs the weak learning algorithm  $A$  in stage  $i+1$ ,  $B$  uses the learning oracle to estimate the relative size  $\mu_i$  within the additive error  $\eta = \varepsilon\tau/(3 + 2\tau)$ . Note that  $\eta = \Omega(\varepsilon\tau)$ . If for the obtained estimate it holds that  $\tilde{\mu}_i \leq \varepsilon - \eta$ , then  $\mu_i \leq \varepsilon$ , and in this case the algorithm  $B$  stops and outputs its final hypothesis  $h = \text{maj}(h_1, \dots, h_i)$ . Otherwise,  $B$  proceeds with the simulation of  $A$ .

The final hypothesis  $h = \text{maj}(h_1, \dots, h_i)$  errs on  $x$  only if  $N_i(x) < 0$ , and by the definition of  $M_i$ , this implies  $M_i(x) = 1$ . It follows that the error of  $h$  with respect to  $D$  is bounded by  $\sum_{M_i(x)=1} D(x) \leq \sum_x M_i(x)D(x) = \mu_i$ . Since the algorithm stops only when  $\mu_i \leq \varepsilon$ , it follows that the final hypothesis  $h$  has error rate at most  $\varepsilon$ , as desired.

Concerning the  $L_\infty$ -norm of the distributions  $D_{i+1}$ , note that since  $\tau < 1$  it holds that  $\eta < \varepsilon/3$ . Thus,  $B$  simulates the weak learning algorithm  $A$  only when  $\mu_i > \varepsilon - 2\eta > \varepsilon/3$ . This implies that  $A$  is only run with respect to distributions  $D_{i+1}$  satisfying  $L_\infty(D_{i+1}) = O(L_\infty(D)/\varepsilon)$ .

Now we bound the number of stages required by the boosting algorithm. The analysis is based on the ‘‘elevator’’ argument of [9] and can be found in the appendix.

**Lemma 1.** *Let  $B$  be a weak learning algorithm with error  $O(1/\gamma^2\varepsilon)$ .*

Next we show how to estimate the relative size  $\mu_i$  by using statistical queries with respect to the fixed distribution  $D$ . Note that since  $N_i(y) \leq i$  we have

$$\begin{aligned} \mu_i &= \sum_y M_i(y)D(y) \\ &= \sum_{N_i(y)<0} M_i(y)D(y) + \sum_{k=1}^i \sum_{N_i(y)=k} M_i(y)D(y) \\ &= \sum_{N_i(y)<0} D(y) + \sum_{k=1}^i (1 - \gamma)^k \sum_{N_i(y)=k} D(y) \\ &= \sum_{k=0}^i (1 - \gamma)^k p_k^i , \end{aligned}$$

where  $p_0^i = \Pr_D[N_i(x) < 0]$  and  $p_k^i = \Pr_D[N_i(x) = k]$  for  $k > 0$ . Now suppose that we are given estimates  $\tilde{p}_k^i$  of  $p_k^i$ , each within additive error  $\gamma\eta$ . Since  $\sum_{k=0}^i (1 - \gamma)^k \leq 1/\gamma$  it holds that

$$\sum_{k=0}^i (1 - \gamma)^k \tilde{p}_k^i \leq \sum_{k=0}^i (1 - \gamma)^k p_k^i + \gamma\eta \sum_{k=0}^i (1 - \gamma)^k \leq \mu_i + \eta$$

and, similarly,  $\sum_{k=0}^i (1 - \gamma)^k \tilde{p}_k^i \geq \mu_i - \eta$ . This means that for an estimate of  $\mu_i$  within additive error  $\eta$  it is sufficient to obtain estimates of each  $p_k^i$  within additive error  $\gamma\eta$ . The latter can be obtained by using  $i + 1$  statistical queries with respect to  $D$ . Thus,  $\mu_i$  can be estimated within additive error  $\eta$  by using  $O(i)$  statistical queries with respect to  $D$ , each of tolerance  $\gamma\eta = \Omega(\varepsilon\gamma\tau)$ .

It remains to show how to simulate the queries of  $A$  with respect to  $D_{i+1}$  by queries with respect to  $D$ . For this we may assume that we are given an estimate  $\tilde{\mu}_i$  of  $\mu_i$  within additive error  $\eta$ . Since that weak learning algorithm is only run when  $\tilde{\mu}_i \geq \varepsilon - \eta$ , we may further assume that  $\tilde{\mu}_i$  satisfies  $\tilde{\mu}_i \geq \varepsilon - \eta$ . A statistical query with hypothesis  $\chi$  and tolerance  $\tau$  with respect to  $D_{i+1}$  is a request for an estimate of the probability  $p = \Pr_{D_{i+1}}[\chi(x, f(x)) = 1]$  within additive error  $\tau$ . Is not hard to verify that the probability  $p$  can be expressed as the fraction  $\sum_{k=0}^i (1 - \gamma)^k q_k^i / \mu_i$ , where  $q_0^i = \Pr_D[\chi(x, f(x)) = 1 \wedge N_i(x) < 0]$  and  $q_k^i = \Pr_D[\chi(x, f(x)) = 1 \wedge N_i(x) = k]$ , for  $k > 0$ . We use the following lemma.

**Lemma 2 ([2]).**  $0 \leq a, b, c, \tau \leq 1$ ,  $a = b/c$ . If  $a \pm \tau \leq b \pm c\tau/3$

By Lemma 2, for an estimate of  $p$  within additive error  $\tau$  it is sufficient to obtain estimates of the sum  $\sum_{k=0}^i (1 - \gamma)^k q_k^i$  and  $\mu_i$  both within additive error  $\mu_i\tau/3$ . By assumption,  $\mu_i \geq \varepsilon - 2\eta$ , and by the choice of  $\eta = \varepsilon\tau/(3 + 2\tau)$  it holds that  $\eta = (\varepsilon - 2\eta)\tau/3 \leq \mu_i\tau/3$ . Hence, the given estimate  $\tilde{\mu}_i$  is already sufficiently accurate, and for the estimate of the sum  $\sum_{k=0}^i (1 - \gamma)^k q_k^i$  it is sufficient to be within additive error  $\eta$ . Since  $\sum_{k=0}^i (1 - \gamma)^k \leq 1/\gamma$ , it is sufficient to obtain estimates of each  $q_k^i$  within additive error  $\gamma\eta$ . Hence, the desired estimate of the probability  $p$  can be obtained by using  $O(i)$  queries of tolerance  $\Omega(\varepsilon\gamma\tau)$  with respect to  $D$ . Thus, we have shown the following theorem.

**Theorem 1.** Let  $B$  be a weak learning algorithm that runs for  $B$  stages,  $0 < \varepsilon, \tau < 1$ ,  $0 < \gamma < 1/2$ . Let  $D$  be a distribution over  $X$ . Let  $A$  be a weak learning algorithm that runs for  $A$  stages,  $\gamma$  advantage,  $N$  queries,  $\tau$  tolerance. Let  $D_i$  be a distribution over  $X$ . Let  $L_\infty(D_i) = O(L_\infty(D)/\varepsilon)$ . Let  $B = O(N/\gamma^4\varepsilon^2)$ . Then  $A$  runs for  $O(N/\gamma^4\varepsilon^2)$  stages,  $\Omega(\varepsilon\gamma\tau)$  tolerance.

In the model of learning by statistical queries, Freund's boosting algorithm runs for  $O((1/\gamma^2) \log(1/\varepsilon))$  stages, simulates the weak learning algorithm only on distributions  $D_i$  with  $L_\infty(D_i) = O(L_\infty(D)/\varepsilon^2)$ , and in each stage it uses at most  $O((1/\gamma^2) \log(1/\varepsilon)N)$  queries of tolerance  $\Omega(\varepsilon^2\tau)$  [2]. By using this boosting algorithm to boost from an arbitrary advantage  $\gamma$  to some constant advantage, say  $1/4$ , and then using the boosting algorithm of Theorem 1 to boost from the advantage  $1/4$  to  $\varepsilon$  we can further increase the tolerance from  $\Omega(\varepsilon\gamma\tau)$  to  $\Omega(\varepsilon\tau)$ .

**Corollary 1.** Let  $B$  be a weak learning algorithm with advantage  $\gamma < 1/2$  and tolerance  $\tau$  ( $0 < \varepsilon, \tau < 1$ ). Let  $D$  be a distribution on  $\{0, 1\}^d$  with  $L_\infty(D) = O(L_\infty(D)/\varepsilon)$ . Let  $A$  be the boosting algorithm of Corollary 1 with respect to the uniform distribution. Then  $A$  uses  $N = n^{O(\log(s/\varepsilon))}$  queries to  $D$  and outputs a DNF formula  $f$  with  $L_\infty(f) = O(N/\gamma^4\varepsilon^2)$  and tolerance  $\Omega(\varepsilon\tau)$ .

The algorithm for DNF formulas in [14] is based on a weak learning algorithm  $A$  which, for arbitrary distributions  $D$ , runs in time  $n^{O(k)}$  where  $k = \log(s2^n L_\infty(D))$ , and both the advantage and the tolerance is  $\Omega(1/s)$ . If we apply the boosting algorithm of Corollary 1 with respect to the uniform distribution, then  $A$  is only run with distributions  $D_i$  satisfying  $2^n L_\infty(D_i) = O(1/\varepsilon)$ . Hence, the number of queries used by  $A$  in each stage is  $N = n^{O(\log(s/\varepsilon))}$  and thus the total number of queries is  $O(s^4(1/\varepsilon^2)n^{O(\log(s/\varepsilon))}) = n^{O(\log(s/\varepsilon))}$ .

**Corollary 2.** Let  $s$  be a positive integer and  $n$  a positive integer. Let  $A$  be a weak learning algorithm with advantage  $\gamma < 1/2$  and tolerance  $\tau$  ( $0 < \varepsilon, \tau < 1$ ). Let  $D$  be a distribution on  $\{0, 1\}^d$  with  $L_\infty(D) = O(L_\infty(D)/\varepsilon)$ . Let  $A$  be the boosting algorithm of Corollary 1 with respect to the uniform distribution. Then  $A$  uses  $N = n^{O(\log(s/\varepsilon))}$  queries to  $D$  and outputs a DNF formula  $f$  with  $L_\infty(f) = O(N/\gamma^4\varepsilon^2)$  and tolerance  $\Omega(\varepsilon/s)$ .

It is not hard to see that any statistical query learning algorithm for the class of monomials requires tolerance  $O(\varepsilon)$ . Thus, the tolerance of Corollary 2 is optimal with respect to its dependence on  $\varepsilon$ .

### 4 Learning DNF by Proper Distance Queries

In this section we consider the learnability of DNF formulas by proper distance queries. For this we first provide an algorithm for DNF formulas with terms of bounded size. The learning algorithm is then applied to the class of  $s$ -term DNF where we ignore the terms of size larger than  $\Omega(\log(s/\varepsilon))$ . The algorithm is based on the following known bounds on the Fourier spectrum of these formulas due to Mansour [15].

**Lemma 3 ([15]).** Let  $f$  be a DNF formula with  $d$  variables.

$$\sum_{|A|>k} \hat{f}^2(A) \leq 2 \cdot 2^{-k/20d}.$$

**Lemma 4 ([15]).** Let  $f$  be a DNF formula with  $d$  variables.

$$\sum_{|A|\leq k} |\hat{f}(A)| \leq 4(20d)^k.$$

**Theorem 2.** Let  $d$  be a positive integer. Let  $A$  be a weak learning algorithm with advantage  $\gamma < 1/2$  and tolerance  $\tau$  ( $0 < \varepsilon, \tau < 1$ ). Let  $D$  be a distribution on  $\{0, 1\}^d$  with  $L_\infty(D) = O(L_\infty(D)/\varepsilon)$ . Let  $A$  be the boosting algorithm of Corollary 1 with respect to the uniform distribution. Then  $A$  uses  $N = n^{O(d \log(1/\varepsilon))}$  queries to  $D$  and outputs a DNF formula  $f$  with  $L_\infty(f) = O(N/\gamma^4\varepsilon^2)$  and tolerance  $\Omega(d \log(1/\varepsilon))$ .



Let  $f$  be the target, let  $k = 20d \log(2/\varepsilon)$ , and let  $L = 4(20d)^k$ . Consider the real-valued approximation  $g = \sum_{A \in G} a_A \chi_A$  of  $f$  where each  $a_A$  is an estimate of  $\hat{f}(A)$  within additive error  $\tau = \varepsilon^{3/2}/L$  and  $G = \{A \mid |A| \leq k \text{ and } |a_A| \geq \varepsilon/L + \tau\}$ . Clearly,  $g$  can be obtained by asking  $\binom{n}{k} = n^{O(d \log(1/\varepsilon))}$  distance queries with tolerance  $\tau = \varepsilon^{3/2}/d^{O(d \log(1/\varepsilon))} = 1/d^{O(d \log(1/\varepsilon))}$ . The hypothesis used in each query is a parity  $\chi_A$  which depends on at most  $k$  variables and hence can be represented as a DNF formula with terms of size  $k = O(d \log(1/\varepsilon))$ .

To bound the error of the approximation  $g$ , first note that  $\hat{g}(A) = 0$  for all  $A \notin G$ . By Parseval's identity, the expected error square is therefore

$$E[(f - g)^2] = \sum_{|A| > k} \hat{f}^2(A) + \sum_{A \in G'} \hat{f}^2(A) + \sum_{A \in G} (\hat{f}(A) - \hat{g}(A))^2$$

where  $G' = \{A \mid |A| \leq k \text{ and } |a_A| < \varepsilon/L + \tau\}$ . By Lemma 3, the first sum is at most  $\varepsilon$ . For the second sum note that  $A \in G'$  implies  $|\hat{f}(A)| < \varepsilon/L + 2\tau \leq 3\varepsilon/L$ . Together with Lemma 4 we get  $\sum_{A \in G'} \hat{f}^2(A) \leq \max_{A \in G'} |\hat{f}(A)| \sum_{|A| \leq k} |\hat{f}(A)| \leq 3\varepsilon$ . For the last sum we note that  $A \in G$  implies  $|\hat{f}(A)| \geq \varepsilon/L$ . Since  $f$  is  $\pm 1$ -valued, the number of coefficients  $\hat{f}(A)$  with  $|\hat{f}(A)| \geq \theta$  is at most  $1/\theta^2$ . Hence,  $|G| \leq (L/\varepsilon)^2$ , and by the choice of  $\tau$  we get  $\sum_{A \in G} (\hat{f}(A) - \hat{g}(A))^2 \leq (L/\varepsilon)^2 \tau^2 = \varepsilon$ . Thus, the expected error square is at most  $5\varepsilon$ .

The error bound of the sign of  $g$  is at most the expected error square of  $g$  and hence at most  $5\varepsilon$ . The only obstacle which prevents us to use  $\text{sign}(g)$  as our final hypothesis is that  $\text{sign}(g)$  might not be representable by a DNF formula with terms of the required size. For this reason, we finally search for a DNF formula  $h$  with terms of size  $d$  whose distance from  $\text{sign}(g)$  is at most  $5\varepsilon$ . Since the distance between  $\text{sign}(g)$  and the target  $f$  is  $5\varepsilon$ , and since  $f$  is a DNF formula with terms of size  $d$ , we are guaranteed to find such a DNF formula  $h$ , and by the triangle inequality, the distance of  $h$  from the target  $f$  is at most  $10\varepsilon$ . The search for  $h$  does not require any further queries. Replacing  $\varepsilon$  by  $\varepsilon/10$  proves the theorem.  $\square$

In Theorem 2, the size of the terms used in the hypotheses is only increased by a factor of  $O(\log(1/\varepsilon))$ . An interesting setting is when  $\varepsilon$  is constant and  $d = O(\log n / \log \log n)$ . Then the tolerance is inverse polynomial in  $n$ .

It is possible to show that the number of distance queries necessary to learn the class of DNF formulas with terms of size  $d$  is at least  $n^{\Omega(d)}$  if the tolerance  $\tau$  is  $1/n^{\Omega(d)}$  and  $d \leq \sqrt{n}$  (see also Theorem 5 below). Hence, for constant error  $\varepsilon$ , the number of distance queries both necessary and sufficient to learn the class of DNF formulas with terms of size  $d$  is  $n^{\Theta(d)}$ , as long as  $d \leq \sqrt{n}$  and the tolerance is in the range between  $1/n^{\Omega(d)}$  and  $1/d^{\Omega(d)}$ .

When applying the algorithm of Theorem 2 to the class of  $s$ -term DNF formulas, we can ignore terms of size larger than  $\Omega(\log(s/\varepsilon))$ . This increases the expected error square by at most  $O(\varepsilon)$ , and the corresponding Fourier coefficients remain within an additive error  $O(\varepsilon)$ . The queries, however, are still parities which depend on at most  $k = O(\log(s/\varepsilon) \log(1/\varepsilon))$  variables and hence can be

represented as DNF formulas with at most  $2^k = (s/\varepsilon)^{O(\log(1/\varepsilon))}$  terms. This yields the following corollary.

**Corollary 3.** *Let  $D$  be a distribution over  $\{0, 1\}^n$  and  $f$  a function over  $\{0, 1\}^n$ . Let  $s$  and  $\varepsilon$  be constants. Then there exists an algorithm that, given  $D$  and  $f$ , outputs a DNF formula  $g$  such that  $|E_D[f\chi_A] - E_D[g\chi_A]| \leq \varepsilon$  for all  $A \subseteq [n]$  with  $|A| \leq k$ , where  $k = 20d \log(8s^2 2^n L_\infty(D))$  and  $d = O(\log(s/\varepsilon) \log(1/\varepsilon))$ . The algorithm uses  $(s/\varepsilon)^{O(\log(1/\varepsilon))}$  queries and has a tolerance of  $1/n^{O(\log \log n)}$ .*

Note that for constant error bound  $\varepsilon$ , the number of terms of hypotheses used by the algorithm of Corollary 3 is polynomial in the number of terms of the target. If, furthermore,  $s$  is polynomial in  $n$ , then the number of queries is  $n^{O(\log n)}$  and the tolerance is  $1/n^{O(\log \log n)}$ . Hence, in this setting, the number of queries is polynomial in the lower bound  $n^{\Omega(\log n)}$ , which holds whenever the tolerance is  $1/n^{\Omega(\log n)}$ .

We conclude this section by considering the setting when the algorithm may ask queries with tolerance zero. In [3] it is shown that the class of  $s$ -term DNF formulas is exactly learnable by  $O(sn)$  projective equivalence queries with  $2s$ -term DNF formulas as hypotheses. By simulating the projective equivalence queries with distance queries of tolerance zero, we are able to prove the following theorem.

**Theorem 3.** *Let  $D$  be a distribution over  $\{0, 1\}^n$  and  $f$  a function over  $\{0, 1\}^n$ . Let  $s$  and  $\tau$  be constants. Then there exists an algorithm that, given  $D$  and  $f$ , outputs a DNF formula  $g$  such that  $|E_D[f\chi_A] - E_D[g\chi_A]| \leq \tau$  for all  $A \subseteq [n]$  with  $|A| \leq k$ , where  $k = O(sn^2)$  and the algorithm uses  $O(sn)$  queries.*

### 5 Learning Majorities of DNF by Proper Distance Queries

In this final section we consider the class of majorities over DNF formulas with terms of bounded size. Our learning algorithm is based on the following lemma.

**Lemma 5.** *Let  $D$  be a distribution over  $\{0, 1\}^n$  and  $f$  a function over  $\{0, 1\}^n$ . Let  $s$  and  $d$  be constants. Let  $A \subseteq [n]$  with  $|A| \leq k$ . If  $|E_D[f\chi_A]| \geq 1/8s(20d)^k$  and  $k = 20d \log(8s^2 2^n L_\infty(D))$ , then there exists a DNF formula  $h$  such that  $|E_D[h\chi_A]| \geq 1/s$ .*

Without loss of generality we assume that  $s$  is odd, and hence  $h_1(x) + \dots + h_s(x) \neq 0$  for all  $x$ . Then, by the Discriminator Lemma of Hajnal et al. [8], one of the DNF formulas  $h = h_i$  satisfies  $|E_D[fh]| \geq 1/s$ . Consider  $g = \sum_{|A| \leq k} \hat{h}(A)\chi_A$ . By Parseval's identity and Lemma 3 we get

$$E[(g - h)^2] = \sum_{|A| > k} \hat{h}^2(A) \leq \frac{1}{4s^2 2^n L_\infty(D)}.$$

Since  $D(x) \leq L_\infty(D)$  for all  $x$ , it follows that  $E_D[(g - h)^2] \leq 2^n L_\infty(D) E[(g - h)^2] \leq 1/4s^2$ . For any  $\pm 1$ -valued function  $f$ , real-valued functions  $h$  and  $g$  and

for any distribution  $D$  it holds that  $|E_D[fg]| \geq |E_D[fh]| - (E_D[(g-h)^2])^{1/2}$  (cf. [11]). Hence,  $|E_D[fg]| \geq 1/s - 1/2s = 1/2s$ . On the other hand,  $E_D[fg] = \sum_{|A| \leq k} \hat{h}(A) E_D[f\chi_A]$ . By Lemma 4, it follows that

$$|E_D[fg]| \leq \max_{|A| \leq k} |E_D[f\chi_A]| \sum_{|A| \leq k} |\hat{h}(A)| \leq \max_{|A| \leq k} |E_D[f\chi_A]| \cdot 4(20d)^k.$$

Thus, we get  $\max_{|A| \leq k} |E_D[f\chi_A]| \geq 1/8s(20d)^k$ . □

**Theorem 4.** *Let  $s, d$  be integers,  $s \geq 20d \log(8s^2 2^n L_\infty(D))$ . Let  $D$  be a distribution on  $\{0, 1\}^d$  with  $L_\infty(D) = O(2^{-n}/\varepsilon)$ . Then there exists a weak learning algorithm  $W$  which searches for a parity  $\chi_A$  with  $|A| \leq k$  such that  $|E_D[f\chi_A]|$  is close to  $1/8s(20d)^k$  or above. The output is either  $\chi_A$  or its negation, depending on whether  $E_D[f\chi_A]$  is positive or negative. If  $L_\infty(D) = O(2^{-n}/\varepsilon)$ , then  $k = O(d \log(s/\varepsilon))$ , the number of queries is  $N = \binom{n}{k} = n^{O(k)}$  and both the advantage  $\gamma$  and the tolerance  $\tau$  is  $1/sd^{O(k)} = 1/d^{O(k)}$ .*

Let  $D$  be some distribution and let  $k = 20d \log(8s^2 2^n L_\infty(D))$ . Lemma 5 immediately leads to a simple weak learning algorithm  $W$  which searches for a parity  $\chi_A$  with  $|A| \leq k$  such that  $|E_D[f\chi_A]|$  is close to  $1/8s(20d)^k$  or above. The output is either  $\chi_A$  or its negation, depending on whether  $E_D[f\chi_A]$  is positive or negative. If  $L_\infty(D) = O(2^{-n}/\varepsilon)$ , then  $k = O(d \log(s/\varepsilon))$ , the number of queries is  $N = \binom{n}{k} = n^{O(k)}$  and both the advantage  $\gamma$  and the tolerance  $\tau$  is  $1/sd^{O(k)} = 1/d^{O(k)}$ .

We combine the weak learning algorithm  $W$  with the boosting algorithm of Theorem 1 with respect to the uniform distribution. Then  $W$  is only run with distributions  $D_i$  satisfying  $L_\infty(D_i) = O(2^{-n}/\varepsilon)$ . Hence the number of queries is  $O(N/\gamma^4 \varepsilon^2) = n^{O(d \log(s/\varepsilon))}$  and the tolerance is  $\Omega(\varepsilon \gamma \tau) = 1/d^{O(d \log(s/\varepsilon))}$ . Further note that the number of stages is  $O(1/\gamma^2 \varepsilon) = d^{O(d \log(s/\varepsilon))}$ . It remains to show how to simulate the statistical queries used by the boosting algorithm by distance queries of the required form.

Recall from section 3, that for an estimate of the relative size  $\mu_i$  in stage  $i+1$ , the boosting algorithm asks for estimates of the probabilities  $p_0^i = \Pr[N_i(x) < 0]$  and  $p_r^i = \Pr[N_i(x) = r]$ , where  $N_i(x) = \sum_{j=1}^i h_j(x)f(x)$  and  $h_j$  is the output of  $W$  in stage  $j \leq i$ . Obviously, these estimates can be obtained by asking statistical queries of the form  $\chi_r^i(x, b) = \llbracket \sum_{j=1}^i h_j(x)b \leq r \rrbracket$ , where we have to decrease the tolerance only by a factor of 2. Each statistical query  $\chi_r^i(x, b)$  can be simulated by two distance queries with hypotheses  $\chi_r^i(x, 1)$  and  $\chi_r^i(x, -1)$ . Since  $\chi_r^i(x, 1) = \llbracket \sum_{j=1}^i h_j(x) \leq r \rrbracket$  and  $\chi_r^i(x, -1) = \llbracket \sum_{j=1}^i -h_j(x) \leq r \rrbracket$ , and since each  $h_j$  is a parity which depends on at most  $O(d \log(s/\varepsilon))$  variables, it follows that each statistical query  $\chi_r^i(x, b)$  can be simulated by two distance queries with majorities over DNF formulas with terms of size  $O(d \log(s/\varepsilon))$ , where the number of DNF formulas is bounded by the number of stages  $d^{O(d \log(s/\varepsilon))} = (s/\varepsilon)^{O(d \log d)}$ .

Similarly, to simulate the distance queries used by  $W$  with respect to  $D_i$ , it is sufficient to ask statistical queries of the form  $\chi_r^i(x, b) = \llbracket h(x) \neq f(x) \wedge \sum_{j=1}^i h_j(x)b \leq r \rrbracket$ , where  $h$  is the hypotheses of a distance query of  $W$ . The

statistical query  $\chi_r^i(x, b)$  can be simulated by the two distance queries with hypotheses  $\chi_r^i(x, 1) = \llbracket h(x) = -1 \wedge \sum_{j=1}^i h_j(x) \leq r \rrbracket$  and  $\chi_r^i(x, -1) = \llbracket h(x) = 1 \wedge \sum_{j=1}^i -h_j(x) \leq r \rrbracket$ . For the hypothesis  $\chi_r^i(x, 1)$  we observe that  $h(x) = 1 \wedge \sum_{j=1}^i h_j(x) \leq r$  is true if and only if  $\sum_{j=0}^i \max(h, h_j(x)) \leq r - 1$  is true, where we use the additional constant  $h_0 = 0$  to account for the possibility  $r = i$ . Since both  $h$  and the  $h_j$ 's are parities which depend on at most  $O(d \log(s/\varepsilon))$  variables, and since the maximum corresponds to a disjunction, it follows that  $\chi_r^i(x, 1)$  can again be expressed as a majority over  $(s/\varepsilon)^{O(d \log d)}$  DNF-formulas with terms of size  $O(d \log(s/\varepsilon))$ . A similar argument can be applied to the hypothesis  $\chi_r^i(x, -1) = \llbracket h(x) = 1 \wedge \sum_{j=1}^i -h_j(x) \leq r \rrbracket$ . This proves the theorem.  $\square$

If  $\varepsilon$  is constant,  $s = n^{O(1)}$  and  $d = O(\log n)$ , then the running time of the algorithm is  $n^{O(\log^2 n)}$ , the tolerance is  $1/n^{O(\log n \log \log n)}$ , and the hypotheses are majorities over  $n^{O(\log n \log \log n)}$  DNF formulas with terms of size  $O(\log^2 n)$ .

We now show the following lower bound.

**Theorem 5.** For every class  $C$  of functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  such that  $d \log s \leq \sqrt{n}$  and  $0 < \varepsilon < 1/2$ , the number of distance queries required to learn  $C$  is at least  $\tau n^{(d \log s)/4}$  for  $\tau = 1/n^{(d \log s)/8}$ .

For every error bound  $0 < \varepsilon < 1/2$ , the number of distance queries required to learn a class  $C$  is at least  $k\tau^2$ , where  $k$  is the number of parities  $\chi_A$  with  $A \subseteq [n]$  in  $C$  [6].

Suppose we want to compute the parity of  $d \log s$  input bits. Think of the input bits as divided into  $\log s$  blocks of length  $d$ , and let  $\oplus_i$  denote the parity of each block. Then the desired result is the parity of the parities  $\oplus_1, \dots, \oplus_{\log s}$ . Each boolean function of  $r$  input bits can be computed by a CNF formula with  $2^r$  clauses or a DNF formula with terms of size  $r$ . Hence, the desired parity can be computed by a conjunction of  $s$  clauses, where each clause consists of  $\log s$  DNF formulas with terms of size  $d$ , and where each DNF formula computes one of the parities  $\oplus_i$  or its negation. It follows that the resulting formula is a conjunction over  $s$  DNF formulas with terms of size  $d$ . By replacing the conjunction with a majority (and ignoring the additional constant input bits), we get that any parity of  $d \log s$  input bits can be computed by a majority of  $s$  DNF formulas with terms of size  $d$ .

Thus, the number of parities in  $C$  is at least  $\binom{n}{d \log s} \geq (\frac{n}{d \log s})^{d \log s}$ , and since  $d \log s \leq \sqrt{n}$ , this number is at least  $n^{(d \log s)/2}$ . It follows that if  $\tau \geq 1/n^{(d \log s)/8}$ , then the number of distance queries required to learn  $C$  is at least  $n^{(d \log s)/2} \tau^2 \geq n^{(d \log s)/4}$ .  $\square$

It follows that for constant error  $\varepsilon$ , the number of distance queries both necessary and sufficient to learn the class of majorities over  $s$  DNF formulas with terms of size  $d$  is  $n^{\Theta(d \log s)}$ , as long as  $d \log s \leq \sqrt{n}$  and the tolerance is in the range between  $1/n^{\Omega(d \log s)}$  and  $1/d^{O(d \log s)}$ .

## References

- [1] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
- [2] J. Aslam and S. Decatur. General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. *Information and Computation*, 141(2):85–118, 1998.
- [3] Jose Luis Balcázar, Jorge Castro, and David Guijarro. A new abstract combinatorial dimension for exact learning via queries. *Journal of Computer and System Sciences*, 64:2–21, 2002.
- [4] Shai Ben-David, Alon Itai, and Eyal Kushilevitz. Learning by distances. *Information and Computation*, 117(2):240–250, 1995.
- [5] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proc. 26th ACM Symposium on Theory of Computing*, pages 253–262. ACM Press, 1994.
- [6] J. H. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.
- [7] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.
- [8] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turan. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46:129–154, 1993.
- [9] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proc. 36th IEEE Symposium on the Foundations of Computer Science*, pages 538–545, 1995.
- [10] J. C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.
- [11] Jeffrey C. Jackson, Adam R. Klivans, and R.A. Servedio. Learnability beyond  $AC^0$ . In *Proc. 34th ACM Symposium on Theory of Computing*, pages 776–784, 2002.
- [12] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.
- [13] A.R. Klivans and R.A. Servedio. Learning DNF in time  $2^{\tilde{O}(n^{1/3})}$ . *Journal of Computer and System Sciences*, 68(2):303–318, 2004.
- [14] Johannes Köbler and Wolfgang Lindner. A general dimension for approximately learning boolean functions. In *Algorithmic Learning Theory, 13th International Conference, ALT 2002, Lübeck, Germany November 2002, Proceedings*, pages 139–148, 2002.
- [15] Y. Mansour. An  $O(n^{\log \log n})$  learning algorithm for DNF under the uniform distribution. In *Proc. 5th Annual ACM Conference on Computational Learning Theory*, pages 53–61, 1992.
- [16] Rocco A. Servedio. Smooth boosting and learning with malicious noise. *The Journal of Machine Learning Research*, 4:633–648, 2003.
- [17] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

## A Proof of Lemma 1

We consider the sum  $A_i(x) = \sum_{j=1}^i M_{j-1}(x)h_j(x)f(x)$ . First, we give an upper bound on  $A_i$  by using the “elevator” argument of [9].

**Lemma 6.**  $A_i(x) \leq 1/\gamma + \gamma \sum_{j=1}^i M_{j-1}(x)$

For each  $k > 0$ , we match all stages  $a$  where  $N_{a-1}(x) = k - 1$  rises to  $N_a(x) = k$  with those stages  $b$  where  $N_{b-1}(x) = k$  drops to  $N_b(x) = k - 1$ , with possibly one stage  $c$  left out where  $N_{c-1}(x) = k - 1$  rises to  $N_c(x) = k$ . By definition,  $N_a(x) = N_{a-1}(x) + h_a(x)f(x)$  and, hence,  $h_a(x)f(x) = 1$ . Similarly, we get that  $h_b(x)f(x) = -1$ . Since both  $N_{a-1}(x)$  and  $N_{b-1}(x)$  are nonnegative, we further get  $M_{a-1}(x) = (1 - \gamma)^{N_{a-1}(x)} = (1 - \gamma)^{k-1}$  and  $M_{b-1}(x) = (1 - \gamma)^{N_{b-1}(x)} = (1 - \gamma)^k$ . Thus,  $M_{b-1}(x) = (1 - \gamma)M_{a-1}(x)$ . The contribution of each matched pair to the sum  $A_i(x)$  is therefore  $M_{a-1}(x)h_a(x)f(x) + M_{b-1}(x)h_b(x)f(x) = M_{a-1}(x) - (1 - \gamma)M_{a-1}(x) = \gamma M_{a-1}(x)$ . It follows that the contribution of all matches pairs is at most  $\gamma \sum_{j=1}^i M_j(x)$ .

For each unmatched stage  $c$  with  $N_{c-1}(x) = k - 1$  and  $N_c(x) = k$  we have  $h_c(x)f(x) = 1$ , and hence  $M_{c-1}(x)h_c(x)f(x) = M_{c-1}(x) = (1 - \gamma)^{k-1}$ . The contribution of all unmatched stages is therefore at most

$$\sum_{k=1}^i (1 - \gamma)^{k-1} = \frac{1 - (1 - \gamma)^i}{\gamma} \leq 1/\gamma.$$

For each  $k \leq 0$ , we do the analogues matching for all stages  $a$  where  $N_{a-1}(x) = k$  drops to  $N_a(x) = k - 1$  with those stages  $b$  where  $N_{b-1}(x) = k - 1$  rises to  $N_b(x) = k$ , with possibly one stage  $c$  left out where  $N_{c-1}(x) = k - 1$  drops to  $N_c(x) = k$ . Here we have  $h_a(x)f(x) = -1$ ,  $h_b(x)f(x) = 1$  and  $M_{a-1}(x) = M_{b-1}(x) = 1$  for the matched pairs of stages, so the contribution of each pair is  $M_{a-1}(x)h_a(x)f(x) + M_{b-1}(x)h_b(x)f(x) = 0$ . For each unmatched stage  $c$  with  $N_{c-1}(x) = k - 1$  and  $N_c(x) = k$  we have  $h_c(x)f(x) = -1$  and  $M_{c-1}(x) = 1$ , so the contribution  $M_{c-1}(x)h_c(x)f(x) = -1$  can be ignored as well.  $\square$

By Lemma 6 we have the following upper on the expectation of  $A_i(x)$ , for a randomly chosen  $x$  according to  $D$ ,

$$\sum_x D(x)A_i(x) \leq 1/\gamma + \gamma \sum_{j=1}^i \mu_{j-1}.$$

For a lower bound first recall that  $D_j(x) = M_{j-1}D(x)/\mu_{j-1}$ . It follows that

$$\sum_x D(x)A_i(x) = \sum_{j=1}^i \mu_{j-1} \sum_x D_j(x)h_j(x)f(x).$$

The hypothesis  $h_j$  produced in stage  $j$  has error rate at most  $1/2 - \gamma$  with respect to  $D_j$ . It follows that the inner sum  $\sum_x D_j(x)h_j(x)f(x)$  is at least  $2\gamma$ . Hence,

$$\sum_x D(x)A_i(x) \geq 2\gamma \sum_{j=1}^i \mu_{j-1}.$$

Combining the upper and lower bound we get that  $\sum_{j=1}^i \mu_{j-1} \leq 1/\gamma^2$ . In each stage  $i$  except for the last stage it holds that  $\mu_{j-1} > \varepsilon/3$  in all preceding stages  $j \leq i$ . Therefore,  $\sum_{j=1}^i \mu_{j-1} > i\varepsilon/3$ , and it follows that  $i < 3/(\gamma^2\varepsilon)$ . Thus, the boosting algorithm runs for at most  $3/(\gamma^2\varepsilon)$  stages. This proves Lemma 1.

# Learning of Elementary Formal Systems with Two Clauses Using Queries

Hiroataka Kato<sup>1</sup>, Satoshi Matsumoto<sup>1</sup>, and Tetsuhiro Miyahara<sup>2</sup>

<sup>1</sup> Department of Mathematical Sciences, Tokai University, Hiratsuka 259-1292, Japan  
`{hirotaka, matumoto}@ss.u-tokai.ac.jp`

<sup>2</sup> Faculty of Information Sciences, Hiroshima City University,  
Hiroshima 731-3194, Japan  
`miyahara@its.hiroshima-cu.ac.jp`

**Abstract.** An elementary formal system, EFS for short, is a kind of logic program over strings, and regarded as a set of rules to generate a language. For an EFS  $\Gamma$ , the language  $L(\Gamma)$  denotes the set of all strings generated by  $\Gamma$ . Many researchers studied the learnability of EFSs in various learning models. In this paper, we introduce a subclass of EFSs, denoted by  $r\mathcal{EFS}$ , and study the learnability of  $r\mathcal{EFS}$  in the exact learning model. The class  $r\mathcal{EFS}$  contains the class of regular patterns, which is extensively studied in Learning Theory.

Let  $\Gamma_*$  be a target EFS of learning in  $r\mathcal{EFS}$ . In the exact learning model, an oracle for superset queries answers “yes” for an input EFS  $\Gamma$  in  $r\mathcal{EFS}$  if  $L(\Gamma)$  is a superset of  $L(\Gamma_*)$ , and outputs a string in  $L(\Gamma_*) - L(\Gamma)$ , otherwise. An oracle for membership queries answers “yes” for an input string  $w$  if  $w$  is included in  $L(\Gamma_*)$ , and answers “no”, otherwise.

We show that any EFS in  $r\mathcal{EFS}$  is exactly identifiable in polynomial time using membership and superset queries. Moreover, for other types of queries, we show that there exists no polynomial time learning algorithm for  $r\mathcal{EFS}$  by using the queries. This result indicates the hardness of learning the class  $r\mathcal{EFS}$  in the exact learning model, in general.

## 1 Introduction

An elementary formal system, EFS for short, is a kind of logic program which directly manipulates strings, and is regarded as a set of rules to generate a language. A  $\langle \Sigma, \mathcal{V} \rangle$  is a nonempty finite string of constant symbols and variables. In EFSs, patterns are used as terms in a logic program. A rule (or definite clause) in EFSs is a clause of the form  $A \leftarrow B_1, \dots, B_m$  ( $m \geq 0$ ), where  $A, B_1, \dots, B_m$  are atoms. Learning of rules from string data is important in machine learning [1] and it can be applied to learning of rules from HTML files since HTML files are considered to be string data. Learning of EFSs has been long studied in Algorithmic/Computational Learning Theory [4, 8, 10, 11]. The purpose of this work is to give a new learnability of EFSs.

Consider examples of EFSs defined as follows. Let  $p$  be a unary predicate symbol,  $a$  and  $b$  constant symbols,  $x$  and  $y$  variables.  $p(ab) \leftarrow$  and  $p(axb) \leftarrow p(x)$

are examples of rules.  $\Gamma_1 = \{p(ab) \leftarrow, p(axb) \leftarrow p(x)\}$  is an example of EFS consisting of the above two rules. EFSs  $\Gamma_2 = \{p(axb) \leftarrow, p(ayb) \leftarrow p(y)\}$  and  $\Gamma_3 = \{p(axb) \leftarrow, p(aybzc) \leftarrow p(y), p(z)\}$  are defined similarly. The language  $L(\Gamma)$  generated by an EFS  $\Gamma$  is the set of all constant strings by substituting non-empty constant symbols for variables and applying Modus Ponens to rules in  $\Gamma$ . Let  $\Sigma = \{a, b, c\}$  be a finite alphabet. In the above examples,  $L(\Gamma_1) = \{a^n b^n \mid n \geq 1\}$ ,  $L(\Gamma_2) = \{a^n w b^n \mid w \in \Sigma^+, n \geq 1\}$ , and  $L(\Gamma_3) = \{aab, abb, acb, aabbaabc, aabbaabc, \dots\}$ .

In this paper, we give a polynomial time learning algorithm for a subclass of EFSs in the exact learning model. The framework of EFSs for studying formal language theory was established by [3] and the unifying framework of language learning using EFSs was originated by [4]. A pattern is *regular* if each variable appears in the pattern at most once. The target class of learning *rEFS* in this paper is defined as the set of EFSs  $\Gamma$  which satisfy the following two conditions. (1) All patterns in the heads of all definite clauses in  $\Gamma$  are regular. (2)  $\Gamma$  consists of one or two definite clauses of the form  $p(\pi) \leftarrow$ , or exactly two definite clauses of the forms  $p(\pi') \leftarrow$  and  $p(\tau) \leftarrow p(x_1), \dots, p(x_n)$ , where  $p$  is a unary predicate symbol,  $x_1, \dots, x_n$  ( $n \geq 1$ ) are all of the variables appearing in  $\tau$ , and  $\pi'$  contains at least one variable. By the definition, the classes of regular patterns and unions of two regular patterns are included in *rEFS*. In the above examples,  $\Gamma_1$  is not in *rEFS* since the pattern  $ab$  contains no variable.  $\Gamma_2$  and  $\Gamma_3$  are in *rEFS*.

Let  $\Gamma_*$  be an EFS in *rEFS* to be identified by a learning algorithm, and we say that the EFS  $\Gamma_*$  is a *target*. We introduce the exact learning model via queries due to Angluin [2]. In this model, learning algorithms can access to  $\Gamma_*$  that answer specific kinds of queries about the unknown language  $L(\Gamma_*)$ . We mainly consider the following two oracles in this paper. (1) *Superset query*: The input is an EFS  $\Gamma$  in *rEFS*. If  $L(\Gamma) \supseteq L(\Gamma_*)$ , then the output is  $\Gamma$ . Otherwise, it returns a string  $t \in L(\Gamma) - L(\Gamma_*)$ . The query is called a *superset query*. (2) *Membership query*: The input is a string  $t$  in  $\Sigma^+$ . The output is  $\Gamma_*$  if  $t \in L(\Gamma_*)$ , and  $\perp$  otherwise. The query is called a *membership query*. A learning algorithm  $\mathcal{A}$  collects information about  $L(\Gamma_*)$  by using queries and outputs an EFS  $\Gamma$  in *rEFS*. We say that a learning algorithm  $\mathcal{A}$  *learns* a target  $\Gamma_*$  in polynomial time using a certain type of queries if  $\mathcal{A}$  halts in polynomial time and outputs an EFS  $\Gamma \in \text{rEFS}$  such that  $L(\Gamma) = L(\Gamma_*)$  using queries of the specified type.

We discuss the related works of this work. A pattern  $\pi$  is regarded as a very restricted form of EFS  $\{p(\pi) \leftarrow\}$  in *rEFS*. Angluin [1] originated the research of pattern learning under another learning model of *exact learning*, which is an infinite process of learning. Angluin also showed that patterns are exactly learnable in polynomial time using restricted superset queries [2]. We showed that regular patterns are exactly learnable in polynomial time using membership queries and a positive example [5]. We showed that finite unions of subsequences are exactly learnable in polynomial time using membership and equivalence queries [6]. Moreover, we showed that finite unions of tree patterns are exactly learnable in polynomial time using restricted subset and equivalence queries [7].



The paper [10] deals with a class of restricted EFSs (called primitive EFSs), which is similar but incomparable to  $r\mathcal{EFS}$ , under the learning model of inductive inference of positive examples without allowing empty string to be substituted for variables. The paper [11] extended this learnability by allowing empty string to be substituted for variables. The work [8] deals with a class of EFSs under the exact learning model using equivalence and extensions of membership queries. The work [8] is known so far about the learnability of EFSs under exact learning model.

This paper is organized as follows. In Section 2, we explain EFSs and their languages and give  $r\mathcal{EFS}$  which is our target class of learning. In Section 3, we give our exact learning model. In Section 4, we show that the class  $r\mathcal{EFS}$  is exactly identifiable in polynomial time using membership and superset queries. In Section 5, we give the hardness of learning  $r\mathcal{EFS}$  in the exact learning model.

## 2 Preliminaries

Let  $S$  be a finite set. We denote by  $|S|$  the number of elements in  $S$ . Let  $\Sigma$  be a finite set of symbols,  $X$  a countable set of symbols, and  $\Pi$  a set of symbols. We assume that  $|\Sigma| \geq 2$  and these sets  $\Sigma$ ,  $X$  and  $\Pi$  are mutually distinct. Each predicate symbol is associated with a positive integer called its arity. Let  $w$  be a string. We denote by  $|w|$  the length of  $w$ . We denote by  $w[i]$  the  $i$ -th symbol in string  $w$ , and by  $w[i : j]$  the substring  $w[i] \cdots w[j]$  of  $w$ . We define  $w[i : j] = \varepsilon$  (empty string) if  $i > j$ . For convenience, a prefix  $w[1 : i]$  is abbreviated as  $w[: i]$ , and a suffix  $w[i : |w|]$  as  $w[i : ]$ , where  $1 \leq i \leq |w|$ . For a nonempty set  $\Delta$ , let  $\Delta^+$  denote the set of all nonempty strings.

A string  $a$  is a nonempty string over  $\Sigma \cup X$ . In particular, we say that a pattern  $\pi$  is *simple* if each variable in  $\pi$  appears at most once. An expression is an expression of the form  $p(\pi_1, \dots, \pi_n)$ , where  $p$  is a predicate symbol with arity  $n$  and  $\pi_1, \dots, \pi_n$  are patterns. A clause is a clause of the form  $A \leftarrow B_1, \dots, B_m$  ( $m \geq 0$ ), where  $A, B_1, \dots, B_m$  are atoms. The atom  $A$  is called the *head* and the part  $B_1, \dots, B_m$  the *body* of the definite clause.

**Definition 1.** An *Elementary Formal System* (EFS for short), is a finite set of definite clauses. For an EFS  $\Gamma$ , each definite clause in  $\Gamma$  is called an *instance* of  $\Gamma$ .

A substitution  $\theta$  is a homomorphism from patterns to patterns such that  $\theta(a) = a$  for each  $a \in \Sigma$  and each variable is replaced with any patterns. By  $\pi\theta$ , we denote the image of a pattern  $\pi$  by a substitution  $\theta$ . For an atom  $A = p(\pi_1, \dots, \pi_n)$  and a clause  $C = A \leftarrow B_1, \dots, B_m$ , we define  $A\theta = p(\pi_1\theta, \dots, \pi_n\theta)$  and  $C\theta = A\theta \leftarrow B_1\theta, \dots, B_m\theta$ .

For patterns  $\pi$  and  $\tau$ , we introduce binary relations  $\preceq$  and  $\equiv$  as follows:  $\pi \preceq \tau$  if  $\pi = \tau\theta$  for some substitution  $\theta$ , and  $\pi \equiv \tau$  if  $\pi \preceq \tau$  and  $\tau \preceq \pi$ . If  $\tau \preceq \pi$  and  $\tau \not\equiv \pi$ , then we write  $\tau < \pi$ .

Let  $\pi$  be a pattern,  $i$  ( $1 \leq i \leq |\pi|$ ) a positive integer, and  $\alpha$  a symbol in  $\Sigma$ . We denote by  $\pi_{i,\alpha}$  the string obtained from  $\pi$  by replacing  $\pi[i]$  with  $\alpha$ , that is,  $\pi_{i,\alpha} = \pi[: i - 1]\alpha\pi[i + 1 : ]$ .

For a pattern  $\pi$ , we denote by  $S_1(\pi)$  the set of all strings which are obtained from  $\pi$  by replacing all variables with a string of length 1. For a nonempty set  $P$  of patterns, we define  $S_1(P) = \cup_{\pi \in P} S_1(\pi)$ . Let  $T, T'$  be nonempty sets of patterns. We write  $T \sqsubseteq T'$  if for any pattern  $\pi \in T$ , there is a pattern  $\pi' \in T'$  such that  $\pi \preceq \pi'$ . If  $T \sqsubseteq T'$  and  $T \not\sqsupseteq T'$ , then we write  $T \sqsubset T'$ .

Let  $\Sigma = \{a, b\}$  be a finite alphabet,  $\pi = axbya$  a regular pattern. Then  $\pi_{1,b} = bxya$ ,  $\pi_{3,a} = axaya$  and  $S_1(\pi) = \{aaba, abba, abba, abba\}$ .

A definite clause  $C$  is an EFS  $\Gamma$ , denoted by  $\Gamma \vdash C$ , if  $C$  is obtained by finitely many applications of substitutions and Modus Ponens as in the way of usual logic programming. We define the language  $L(\Gamma, p) = \{w \in \Sigma^+ \mid \Gamma \vdash p(w)\}$ , where  $p$  is a unary predicate symbol.

**Definition 2.** We denote by  $r\mathcal{EFS}$  the set of EFSs  $\Gamma$  which satisfy the following conditions:

1. All patterns in the heads of all clauses in  $\Gamma$  are regular.
2.  $\Gamma$  consists of one or two clauses of the form  $p(\pi) \leftarrow$ , or exactly two clauses of the forms  $p(\pi') \leftarrow$  and  $p(\tau) \leftarrow p(x_1), \dots, p(x_n)$ , where  $p$  is a unary predicate symbol,  $x_1, \dots, x_n$  ( $n \geq 1$ ) are all of the variables appearing in  $\tau$ , and  $\pi'$  contains at least one variable.

By the definition, the class of regular patterns and unions of two regular patterns are included in  $r\mathcal{EFS}$ . We define the *size* of  $\Gamma$ , denoted by  $|\Gamma|$ , as follows: (1).  $|\Gamma| = |\pi|$  if  $\Gamma = \{p(\pi) \leftarrow\}$ , (2).  $|\Gamma| = |\pi_1| + |\pi_2|$  if  $\Gamma = \{p(\pi_1) \leftarrow, p(\pi_2) \leftarrow\}$ , (3).  $|\Gamma| = |\pi| + |\tau|$  if  $\Gamma = \{p(\pi) \leftarrow, p(\tau) \leftarrow p(x_1), \dots, p(x_n)\}$ .

A language  $L$  is an *regular pattern language* if  $L = L(\Gamma, p)$  for some EFS  $\Gamma$  and some unary predicate symbol  $p$ . In particular, a language  $L$  is a *regular pattern language* if  $L = L(\Gamma, p)$  for some EFS  $\Gamma = \{p(\pi) \leftarrow\}$ , where  $\pi$  is a regular pattern.

Let  $\Sigma = \{a, b, c\}$  and  $X = \{x, y, z, \dots\}$ . Let  $\Gamma_1 = \{p(ab) \leftarrow, p(axb) \leftarrow p(x)\}$ ,  $\Gamma_2 = \{p(axb) \leftarrow, p(ayb) \leftarrow p(y)\}$ ,  $\Gamma_3 = \{p(axb) \leftarrow, p(aybzc) \leftarrow p(y), p(z)\}$ ,  $\Gamma_4 = \{p(axb) \leftarrow\}$  and  $\Gamma_5 = \{p(axy) \leftarrow, p(aza) \leftarrow\}$  be EFSs. Since  $ab$  has no variable,  $\Gamma_1$  is not in  $r\mathcal{EFS}$ .  $\Gamma_2, \Gamma_3, \Gamma_4$  and  $\Gamma_5$  are EFSs in  $r\mathcal{EFS}$ .

Languages generated by  $\Gamma_2, \Gamma_3, \Gamma_4$  and  $\Gamma_5$  are as follows:  $L(\Gamma_2, p) = \{a^n w b^n \mid w \in \Sigma^+, n \geq 1\}$ ,  $L(\Gamma_3, p) = \{aab, abb, acb, aaabbaabc, aabbaabc, \dots\}$ ,  $L(\Gamma_4, p) = \{awb \mid w \in \Sigma^+\}$  and  $L(\Gamma_5, p) = \{aw_1 w_2 \mid w_1, w_2 \in \Sigma^+\} \cup \{awa \mid w \in \Sigma^+\}$ . In particular,  $L(\Gamma_4, p)$  is a regular pattern language.

**Definition 3.** Let  $\Gamma$  be an EFS in  $r\mathcal{EFS}$  and  $p$  a unary predicate symbol appearing in  $\Gamma$ .  $\Gamma$  is *reduced* if  $L(\Gamma', p) \subsetneq L(\Gamma, p)$  for any  $\Gamma' \subsetneq \Gamma$ .

Let  $\Sigma = \{a, b, c\}$  and  $X = \{x, y, z, \dots\}$ . Let  $\Gamma_6 = \{p(axa) \leftarrow, p(aya) \leftarrow p(y)\}$ ,  $\Gamma_7 = \{p(axb) \leftarrow, p(byb) \leftarrow p(y)\}$ ,  $\Gamma_8 = \{p(bxya) \leftarrow, p(bazba) \leftarrow p(z)\}$  and  $\Gamma_9 = \{p(bxya) \leftarrow, p(bazbb) \leftarrow p(z)\}$  be EFSs in  $r\mathcal{EFS}$ . Since  $bacbb \in L(\Gamma_7, p)$  and  $bacbb \notin L(\Gamma_7 - \{p(byb) \leftarrow p(y)\}, p)$ ,  $\Gamma_7$  is reduced. Since  $babccabb \in L(\Gamma_9, p)$  and  $babccabb \notin L(\Gamma_9 - \{p(bazbb) \leftarrow p(z)\}, p)$ ,  $\Gamma_9$  is reduced. Since  $L(\Gamma_6 - \{p(aya) \leftarrow p(y)\}, p) = L(\Gamma_6, p)$  and  $L(\Gamma_8 - \{p(bazba) \leftarrow p(z)\}, p) = L(\Gamma_8, p)$ ,  $\Gamma_6$  and  $\Gamma_8$  are not reduced.

In this paper, since we deal with the class  $r\mathcal{EFS}$ , we fix a unary predicate symbol, say  $p$ , and denote  $L(\Gamma, p)$  by  $L(\Gamma)$  simply. We denote by  $\Gamma = (\pi, \tau)$  (resp.,  $\Gamma = \{\pi\}$ ,  $\Gamma = \{\pi, \tau\}$ ) an EFS  $\Gamma = \{p(\pi) \leftarrow, p(\tau) \leftarrow p(x_1), \dots, p(x_n)\}$  (resp.,  $\Gamma = \{p(\pi) \leftarrow\}$ ,  $\Gamma = \{p(\pi) \leftarrow, p(\tau) \leftarrow\}$ ). Moreover, by  $L((\pi, \tau))$  (resp.,  $L(\{\pi\})$ ,  $L(\{\pi, \tau\})$ ) we denote  $L(\{p(\pi) \leftarrow, p(\tau) \leftarrow p(x_1), \dots, p(x_n)\})$  (resp.,  $L(\{p(\pi) \leftarrow\})$ ,  $L(\{p(\pi) \leftarrow, p(\tau) \leftarrow\})$ ).

For EFSs  $\Gamma_2 = \{p(axb) \leftarrow, p(ayb) \leftarrow p(y)\}$ ,  $\Gamma_3 = \{p(axb) \leftarrow, p(aybzc) \leftarrow p(y), p(z)\}$ ,  $\Gamma_4 = \{p(axb) \leftarrow\}$  and  $\Gamma_5 = \{p(axy) \leftarrow, p(aza) \leftarrow\}$ , we write  $\Gamma_2 = (axb, ayb)$ ,  $\Gamma_3 = (axb, aybzc)$ ,  $\Gamma_4 = \{axb\}$  and  $\Gamma_5 = \{axy, aza\}$  simply.

For  $\Gamma = (\pi, \tau)$ , we define the following particular pattern  $\tau_\pi = \tau\{x := \pi \mid x \text{ appears in } \tau\}$ , where all variables substituted to the variables in  $\tau$  are taken to be distinct, so  $\tau_\pi$  is always regular. It is clear that  $|\pi| \leq |\tau_\pi| \leq |\pi||\tau|$ .  $\tau_{\tau_\pi}$  is defined in a similar way.

Let  $\Gamma_{10} = (abxb, aybzc)$  be an EFS in  $r\mathcal{EFS}$ ,  $\pi = abxb$  and  $\tau = aybzc$ . We have  $\tau_\pi = aabx_1bbabx_2bc$ . Note that  $\tau_\pi$  is a regular pattern.

Let  $\Gamma = (\pi, \tau)$  be an EFS in  $r\mathcal{EFS}$ ,  $x_1, \dots, x_n$  all of the variables appearing in  $\tau$ .  $L_{[t]}$  is recursively defined as follows:  $L_{[1]} = \{\tau_\pi\}$  and for any positive integer  $t \geq 2$ ,  $L_{[t]} = L_{[t-1]} \cup \{\tau\{x_1 := \zeta_1, \dots, x_n := \zeta_n\} \mid \zeta_i \in L_{[t-1]} \cup \{\pi\}, i = 1, \dots, n\}$ . We define  $L_\tau = \cup_{t \geq 1} L_{[t]}$ . Note that  $\pi$  is not included in  $L_\tau$ . Thus,  $L(\Gamma_\tau) \subsetneq L((\pi, \tau))$ .

**Definition 1.** EFS  $\Gamma$ , a PFS for short, is defined in [11] as follows:

1. All patterns in heads of all clauses in  $\Gamma$  are regular.
2.  $\Gamma$  consists of  $p(\pi) \leftarrow$  and  $p(\tau) \leftarrow p(x_1), \dots, p(x_n)$ , where  $p$  is a unary predicate symbol, and  $x_1, \dots, x_n$  are all of the variables appearing in  $\tau$ .

An EFS in  $r\mathcal{EFS}$  is different from a PFS. In case of erasing patterns, Uemura et al. showed the following theorem in [11]. The theorem holds for any EFS  $\Gamma = (\pi, \tau)$  in  $r\mathcal{EFS}$  in case of nonerasing patterns.

**Theorem 1.** [11] Let  $\Gamma = (\pi, \tau)$  be a PFS. The following statements are equivalent: (i)  $\Gamma$  is reduced. (ii)  $L(\pi) \cap L(\Gamma_\tau) = \emptyset$ , where  $L(\Gamma_\tau) = \cup_{\zeta \in \Gamma_\tau} L(\zeta)$ .

### 3 Learning Model

In this paper, let  $\Gamma_*$  be an EFS in  $r\mathcal{EFS}$  to be identified, and we say that the EFS  $\Gamma_*$  is a  $\Gamma_*$ . Non-reduced EFSs have redundant axioms. Even if we consider only reduced EFSs, the expressive power of EFSs is same. So we assume that target EFSs are reduced.

We introduce the exact learning model via queries due to Angluin [2]. In this model, learning algorithms can access to  $\Gamma_*$  that answer specific kinds of

queries about the unknown language  $L(\Gamma_*)$ . We consider the following oracles.

- (1).  $Sup_{\Gamma_*}$ : The input is an EFS  $\Gamma$  in  $r\mathcal{EFS}$ . If  $L(\Gamma) \supseteq L(\Gamma_*)$ , then the output is  $\Gamma$ . Otherwise, it returns a string  $t \in L(\Gamma_*) - L(\Gamma)$ . The query is called a *superset query*.
- (2).  $Sub_{\Gamma_*}$ : The input is an EFS  $\Gamma$  in  $r\mathcal{EFS}$ . If  $L(\Gamma) \subseteq L(\Gamma_*)$ , then the output is  $\Gamma$ . Otherwise, it returns a string  $t \in L(\Gamma) - L(\Gamma_*)$ . The query is called a *subset query*.
- (3).  $Mem_{\Gamma_*}$ : The input is a string  $t$  in  $\Sigma^+$ . The output is  $t$  if  $t \in L(\Gamma_*)$ , and  $\perp$  otherwise. The query is called a *membership query*.
- (4).  $Equiv_{\Gamma_*}$ : The input is an EFS  $\Gamma$  in  $r\mathcal{EFS}$ . The output is  $\Gamma$  if  $L(\Gamma) = L(\Gamma_*)$ . Otherwise, it returns a string  $t \in (L(\Gamma) - L(\Gamma_*)) \cup (L(\Gamma_*) - L(\Gamma))$ . The query is called an *equivalence query*.

A learning algorithm  $\mathcal{A}$  collects information about  $L(\Gamma_*)$  by using queries and output an EFS  $\Gamma$  in  $r\mathcal{EFS}$ . We say that a learning algorithm  $\mathcal{A}$  learns a target  $\Gamma_*$  in polynomial time using a certain type of queries if  $\mathcal{A}$  halts in polynomial time with respect to  $|\Gamma_*|$ , and outputs an EFS  $\Gamma \in r\mathcal{EFS}$  such that  $L(\Gamma) = L(\Gamma_*)$  using queries of the specified type.

### 4 Learning of Restricted EFSs Using Queries

Let  $\Gamma_*$  be a target EFS in  $r\mathcal{EFS}$ . Then we consider the following cases: (i).  $\Gamma_* = \{\pi_*\}$ . (ii).  $\Gamma_* = \{\pi_*, \tau_*\}$  and the length of  $\pi_*$  is the same as  $\tau_*$ , that is,  $|\pi_*| = |\tau_*|$ . (iii).  $\Gamma_* = \{\pi_*, \tau_*\}$  and the length of  $\pi_*$  is not the same as  $\tau_*$ , that is,  $|\pi_*| \neq |\tau_*|$ . Without loss of generality, we assume  $|\pi_*| < |\tau_*|$ . (iv).  $\Gamma_* = (\pi_*, \tau_*)$ .

When  $\Gamma_*$  is in the cases (i), (ii) or (iii), we can regard  $\Gamma_*$  as a set of at most two regular patterns. Since we use Theorem 2 for some lemmas and theorems, we assume  $|\Sigma| \geq 5$  in this paper.

**Theorem 2.** [9] Suppose  $|\Sigma| \geq 2k + 1$ . Let  $P$  be a nonempty finite set of regular patterns,  $Q$  a set of at most  $k$  regular patterns. Then the following three statements are equivalent: (1)  $P \subseteq Q$ , (2)  $L(P) \subseteq L(Q)$ , (3)  $S_1(P) \subseteq L(Q)$ .

Let  $\pi$  be a regular pattern. We define the following condition, called **Condition A**, as follows:

- A-1**  $\pi$  satisfies  $|\pi| = |\pi_*|$  and  $L(\Gamma_*) \subseteq L(\{\pi, x_1x_2 \cdots x_{|\pi|+1}\})$ , and
- A-2** There is no regular pattern  $\pi'$  such that  $|\pi'| = |\pi|$ ,  $\pi' \prec \pi$  and  $L(\Gamma_*) \subseteq L(\{\pi', x_1x_2 \cdots x_{|\pi|+1}\})$  for  $\pi$ .

**Lemma 1.** Let  $\pi$  be a regular pattern satisfying Condition A. If  $\Gamma_*$  is not in the case (ii), then  $\pi \equiv \pi_*$ .

**Lemma 2.** Let  $\pi$  be a regular pattern satisfying Condition A. Then, the following statements hold. (1) If  $L(\Gamma_*) \subseteq L(\{\pi\})$ , then  $\Gamma_*$  is in the case (i) or (ii). (2) If  $L(\Gamma_*) \not\subseteq L(\{\pi\})$ , then  $\Gamma_*$  is in the case (iii) or (iv).

For a regular pattern  $\pi$ , we define the following condition, called **Condition B**, as follows:

**B-1**  $\pi$  satisfies Condition A and  $L(\Gamma_*) \subseteq L(\{\pi\})$ , and

**B-2** There are regular patterns  $\pi'$  and  $\tau'$  such that  $|\pi'| = |\tau'| = |\pi|$ ,  $\pi' \prec \pi$ ,  $\tau' \prec \pi$  and  $L(\Gamma_*) \subseteq L(\{\pi', \tau'\})$  for  $\pi$ .

**Lemma 3.** Let  $\pi$  be a regular pattern satisfying Condition A and  $L(\Gamma_*) \subseteq L(\{\pi\})$ . Then, the following statements hold. (1) If there are no regular patterns satisfying Condition B-2 for  $\pi$ , then  $\Gamma_*$  is in the case (i). (2) If there are regular patterns satisfying Condition B-2 for  $\pi$ , then  $\Gamma_*$  is in the case (ii).

Proof. By Lemma 2,  $\Gamma_*$  is in the case (i) or (ii). We show each case.

1. We assume that  $\Gamma_*$  is in the case (ii). By Condition A, we have  $|\pi_*| = |\tau_*| = |\pi|$ . By Theorem 2, since  $L(\Gamma_*) \subseteq L(\{\pi\})$ ,  $\pi_* \preceq \pi$  and  $\tau_* \preceq \pi$ . We assume  $\pi_* \equiv \pi$ . It implies  $L(\Gamma_*) \subseteq L(\{\pi_*\})$ . This contradicts that  $\Gamma_*$  is reduced. Thus, we have  $\pi_* \prec \pi$ . We can get  $\tau_* \prec \pi$  in a similar way.  $\pi_*$  and  $\tau_*$  satisfy Condition B-2 for  $\pi$ . This is a contradiction. Therefore,  $\Gamma_*$  is in the case (i).
2. We assume that  $\Gamma_*$  is in the case (i). By Condition B-2, there are regular patterns  $\pi'$  and  $\tau'$  such that  $|\pi'| = |\tau'| = |\pi|$ ,  $\pi' \prec \pi$ ,  $\tau' \prec \pi$  and  $L(\Gamma_*) \subseteq L(\{\pi', \tau'\})$  for  $\pi$ . By Theorem 2, we have  $\pi_* \preceq \pi'$  or  $\pi_* \preceq \tau'$ . We assume  $\pi_* \preceq \pi'$ .  $\pi'$  satisfies  $|\pi'| = |\pi|$ ,  $\pi_* \preceq \pi' \prec \pi$  and  $L(\{\pi_*\}) = L(\Gamma_*) \subseteq L(\{\pi', x_1x_2 \cdots x_{|\pi|+1}\})$ . This contradicts with Condition A-2 for  $\pi$ . Thus, we have  $\pi_* \not\preceq \pi'$ . We can show  $\pi_* \not\preceq \tau'$  in a similar way. Therefore,  $\Gamma_*$  is in the case (ii).  $\square$

**Lemma 4.** Let  $\pi$  be a regular pattern satisfying Condition B, and  $\pi', \tau'$  regular patterns satisfying Condition B-2 for  $\pi$ . If there are no regular patterns  $\pi''$  and  $\tau''$  satisfying the following two statements, then  $\pi' \equiv \pi_*$  and  $\tau' \equiv \tau_*$ , or  $\pi' \equiv \tau_*$  and  $\tau' \equiv \pi_*$ . (1)  $\pi''$  satisfies  $|\pi''| = |\pi'|$ ,  $\pi'' \prec \pi'$  and  $L(\Gamma_*) \subseteq L(\{\pi'', \tau'\})$ . (2)  $\tau''$  satisfies  $|\tau''| = |\tau'|$ ,  $\tau'' \prec \tau'$  and  $L(\Gamma_*) \subseteq L(\{\pi', \tau''\})$ .

For a regular pattern  $\pi$ , we define the following condition, called  $\mathfrak{C}$ , as follows:

**C-1**  $\pi$  satisfies Condition A and  $L(\Gamma_*) \not\subseteq L(\{\pi\})$ , and

**C-2** There is a regular pattern  $\tau$  satisfying the following conditions for  $\pi$ :

**C-2-1** There is a shortest string  $w \in L(\Gamma_*) - L(\{\pi\})$  such that  $|w| = |\tau|$  and  $w \preceq \tau$ .

**C-2-2**  $L(\{\tau\}) \subseteq L(\Gamma_*)$ .

**C-2-3** There is no regular pattern  $\tau'$  such that  $|\tau'| = |\tau|$ ,  $\tau \prec \tau'$  and  $L(\{\tau'\}) \subseteq L(\Gamma_*)$ .

**Lemma 5.** Let  $\pi$  be a regular pattern satisfying Condition A and  $L(\Gamma_*) \not\subseteq L(\{\pi\})$ . Let  $w$  be a shortest string in  $L(\Gamma_*) - L(\{\pi\})$ . Then, the following statements hold. (1) If  $\Gamma_*$  is in the case (iii), then  $|w| = |\tau_*|$ . (2) If  $\Gamma_*$  is in the case (iv), then  $|w| = |\tau_{*\pi_*}|$ .

Proof. By Lemma 2,  $\Gamma_*$  is in the case (iii) or (iv). By Lemma 1,  $\pi \equiv \pi_*$ . We show each case.

1. For a shortest string  $w \in L(\Gamma_*) - L(\{\pi\})$ ,  $|\tau_*| \leq |w|$ . We assume  $|\tau_*| < |w|$ . It implies  $S_1(\tau_*) \subseteq L(\pi)$ . By Theorem 2,  $\tau_* \preceq \pi$ . Since  $\pi \equiv \pi_*$ , we have  $\tau_* \preceq \pi_*$ . This contradicts that  $\Gamma_*$  is reduced. Thus, we have  $|\tau_*| = |w|$ .
2. By Theorem 1, since  $\Gamma_*$  is reduced,  $L(\pi_*) \cap L(\Gamma_{*\tau_*}) = \emptyset$ . Since  $L(\pi_*) \cap L(\Gamma_{*\tau_*}) = \emptyset$  and  $\pi \equiv \pi_*$ , the length of shortest strings in  $L(\Gamma_*) - L(\{\pi\})$  is  $|\tau_{*\pi_*}|$ . □

By using the above lemma, we can show Lemma 6.

**Lemma 6.** Let  $\pi$  be a regular pattern satisfying Condition C, and  $\tau$  a regular pattern satisfying Condition C-2 for  $\pi$ . Then, the following statements hold. (1) If  $\Gamma_*$  is in the case (iii), then  $\tau \equiv \tau_*$ . (2) If  $\Gamma_*$  is in the case (iv), then  $\tau \equiv \tau_{*\pi_*}$ .

By Lemma 2,  $\Gamma_*$  is in the case (iii) or (iv). Moreover, by Lemma 1,  $\pi \equiv \pi_*$ . We show each case.

1. We assume  $\tau \not\equiv \tau_*$ . By Condition C-2-1 and Lemma 5,  $|\tau| = |\tau_*| = |w|$ . From Condition C-2-2 and Theorem 2,  $\tau \preceq \pi_*$  or  $\tau \preceq \tau_*$ . We assume  $\tau \preceq \pi_*$ . Since  $\pi \equiv \pi_*$ , we have  $\tau \preceq \pi$ . Let  $w$  be a shortest string in  $L(\Gamma_*) - L(\{\pi\})$  satisfying Condition C-2-1 for  $\tau$ . It implies  $w \in L(\{\tau\}) \subseteq L(\{\pi\})$ . This contradicts with  $w \notin L(\{\pi\})$ . Thus, we have  $\tau \preceq \tau_*$ . Since  $\tau \not\equiv \tau_*$  and  $\tau \preceq \tau_*$ , we have  $\tau \prec \tau_*$ . Then,  $\tau_*$  satisfies  $|\tau| = |\tau_*|$ ,  $\tau \prec \tau_*$  and  $L(\{\tau_*\}) \subseteq L(\Gamma_*)$ . This contradicts with Condition C-2-3. Therefore, we have  $\tau \equiv \tau_*$ .
2. We assume  $\tau \not\equiv \tau_{*\pi_*}$ . By Condition C-2-1 and Lemma 5,  $|\tau| = |\tau_{*\pi_*}| = |w|$ . From Condition C-2-2 and  $|\tau| = |\tau_{*\pi_*}|$ ,  $S_1(\tau) \subseteq L(\{\pi_*, \tau_{*\pi_*}\})$ . By Theorem 2,  $\tau \preceq \pi_*$  or  $\tau \preceq \tau_{*\pi_*}$ . We assume  $\tau \preceq \pi_*$ . Let  $w$  be a shortest string in  $L(\Gamma_*) - L(\{\pi\})$  satisfying Condition C-2-1 for  $\tau$ . It implies  $w \in L(\tau) \subseteq L(\pi_*)$ . This contradicts with  $w \notin L(\pi_*)$ . Thus, we have  $\tau \preceq \tau_{*\pi_*}$ . Since  $\tau \preceq \tau_{*\pi_*}$  and  $\tau \not\equiv \tau_{*\pi_*}$ ,  $\tau \prec \tau_{*\pi_*}$ . Then,  $\tau_{*\pi_*}$  satisfies  $|\tau| = |\tau_{*\pi_*}|$ ,  $\tau \prec \tau_{*\pi_*}$  and  $L(\{\tau_{*\pi_*}\}) \subseteq L(\Gamma_*)$ . This contradict with Condition C-2-3. Therefore,  $\tau \equiv \tau_{*\pi_*}$ . □

**Lemma 7.** Let  $\pi$  be a regular pattern satisfying Condition C, and  $\tau$  a regular pattern satisfying Condition C-2 for  $\pi$ . Then, the following statements hold. (1) If  $L(\Gamma_*) \subseteq L(\{\pi, \tau\})$ , then  $\Gamma_*$  is in the case (iii). (2) If  $L(\Gamma_*) \not\subseteq L(\{\pi, \tau\})$ , then  $\Gamma_*$  is in the case (iv).

By Lemma 2,  $\Gamma_*$  is the case (iii) or (iv). Moreover, by Lemma 1,  $\pi \equiv \pi_*$ . We show each case.

1. We assume that  $\Gamma_*$  is in the case (iv). By Lemma 6,  $\tau \equiv \tau_{*\pi_*}$ . From  $\pi \equiv \pi_*$  and  $\tau \equiv \tau_{*\pi_*}$ ,  $L(\Gamma_*) \subseteq L(\{\pi_*, \tau_{*\pi_*}\})$ . Since  $\Gamma_*$  is reduced, we have  $L(\pi_*) \cap L(\Gamma_{*\tau_*}) = \emptyset$ . Thus, we have  $L(\Gamma_{*\tau_*}) \subseteq L(\tau_{*\pi_*})$ . Otherwise, since  $\Gamma_*$  is reduced,  $L(\tau_{*\pi_*}) \not\subseteq L(\Gamma_{*\tau_*})$ . This contradicts with  $L(\tau_{*\pi_*}) \subseteq L(\Gamma_{*\tau_*}) \subseteq L(\tau_{*\pi_*})$ .
2. We assume that  $\Gamma_*$  is in the case (iii). By Lemma 6,  $\tau \equiv \tau_*$ . Since  $\pi \equiv \pi_*$  and  $\tau \equiv \tau_*$ ,  $L(\Gamma_*) \subseteq L(\{\pi, \tau\})$ . This is a contradiction. □

**Procedure LENGTH1**

*Given:* An oracle  $Sup_{\Gamma_*}$  for the target  $\Gamma_*$ ;  
*Input:* A regular pattern  $\pi$  satisfying  
**begin**  
 $\ell := 1$ ;  
 //  $x_1 \cdots x_{\ell+1}$  is a regular pattern  
**while**  $Sup_{\Gamma_*}(\{x_1 \cdots x_{\ell+1}\}) = \text{"yes"}$  **do**  
 $\ell := \ell + 1$ ;  
**output**  $\ell$ ;  
**end.**

**Procedure LEARN\_PI( $\ell$ )**

*Input:* A positive integer  $\ell$  with  $\ell = |\pi_*|$ ;  
*Given:* An oracle  $Sup_{\Gamma_*}$  for the target  $\Gamma_*$ ;  
**begin**  
 //  $x_1 x_2 \cdots x_\ell$  is a regular pattern  
 $\pi := x_1 x_2 \cdots x_\ell$ ;  
**for**  $i := 1$  **to**  $\ell$  **do begin**  
**foreach**  $\alpha \in \Sigma$  **do begin**  
 $\pi' := \pi_{i,\alpha}$ ;  
**if**  $Sup_{\Gamma_*}(\{\pi', x_1 \cdots x_{\ell+1}\}) = \text{"yes"}$   
**then begin**  
 $\pi := \pi'$ ; **break**;  
**end**;  
**end**;  
**end**;  
**output**  $\pi$ ;  
**end.**

**Procedure LEARN\_PLTAU1( $\pi$ )**

*Input:* A regular pattern  $\pi$  satisfying  
 Condition A and  $L(\Gamma_*) \subseteq L(\{\pi\})$ ;  
*Given:* An oracle  $Sup_{\Gamma_*}$  for the target  $\Gamma_*$ ;  
**begin**  
 $i_\pi := 1$ ;  
**while**  $((i_\pi \leq |\pi|) \text{ and } (\pi[i_\pi] \in \Sigma))$  **do**  
 $i_\pi := i_\pi + 1$ ;  
**while**  $i_\pi \leq |\pi|$  **do begin**  
**foreach**  $\alpha \in \Sigma$  **do begin**  
 $i_\tau := i_\pi$ ;  
**while**  $i_\tau \leq |\pi|$  **do begin**  
**foreach**  $\beta \in \Sigma$  **do begin**  
 $\pi' := \pi_{i_\tau,\alpha}$ ;  
 $\pi'' := \pi_{i_\tau,\beta}$ ;  
**if**  $Sup_{\Gamma_*}(\{\pi', \pi''\}) = \text{"yes"}$  **then**  
**begin**  
**output**  $\pi', \pi''$ ; **halt**;  
**end**;  
**end**;  
 $i_\tau := i_\tau + 1$ ;  
**while**  $((i_\tau \leq |\pi|) \text{ and } (\pi[i_\tau] \in \Sigma))$  **do**  
 $i_\tau := i_\tau + 1$ ;  
**end**;  
**end**;  
 $i_\pi := i_\pi + 1$ ;  
**while**  $((i_\pi \leq |\pi|) \text{ and } (\pi[i_\pi] \in \Sigma))$  **do**  
 $i_\pi := i_\pi + 1$ ;  
**end**;  
**output**  $\text{"no"}$ ;  
**end.**

**Fig. 1. Procedure LENGTH1, LEARN\_PI and LEARN\_PLTAU1**

The procedure *LENGTH1* of Fig. 1 outputs a positive integer  $\ell$  with  $\ell = \min\{|w| \mid w \in L(\Gamma_*)\}$ , that is,  $\ell = |\pi_*|$ . The procedure uses  $O(|\pi_*|)$  superset queries, and runs in  $O(|\pi_*|^2)$  time.

The procedure *LEARN\_PI* of Fig. 1 takes a positive integer  $\ell$  with  $\ell = |\pi_*|$  as input, and outputs a regular pattern  $\pi$  such that  $|\pi| = |\pi_*| = \ell$  and  $L(\Gamma_*) \subseteq L(\{\pi, x_1 x_2 \cdots x_{\ell+1}\})$ .

**Lemma 8.** Let  $\ell$  be an input positive integer and  $\pi$  an output regular pattern by the procedure *LEARN\_PI*. Then, there is no regular pattern  $\pi'$  such that  $|\pi'| = |\pi|$ ,  $\pi' \prec \pi$  and  $L(\Gamma_*) \subseteq L(\{\pi', x_1 x_2 \cdots x_{\ell+1}\})$ .

By Lemma 8, the procedure *LEARN\_PI* outputs a regular pattern  $\pi$  satisfying Condition A. The procedure uses  $O(|\pi_*|)$  superset queries, and runs in  $O(|\pi_*|^2)$  time.

The procedure *LEARN\_P1\_TAU1* of Fig. 1 takes a regular pattern  $\pi$  satisfying Condition A and  $L(\Gamma_*) \subseteq L(\{\pi\})$ . If there are regular patterns  $\pi'$  and  $\tau'$  satisfying Condition B-2 for  $\pi$ , the procedure outputs such regular patterns. Otherwise, the procedure outputs  $\perp$ . The procedure uses  $O(|\pi_*|^2)$  superset queries, and runs in  $O(|\pi_*|^3)$  time.

The procedure *LEARN\_P1\_TAU2* of Fig. 2 takes regular patterns  $\pi'$  and  $\tau'$  satisfying Condition B-2 for  $\pi$  as input, where  $\pi$  satisfies Condition B. The procedure outputs regular patterns  $\pi''$  and  $\tau''$  with  $\pi'' \preceq \pi'$  and  $\tau'' \preceq \tau'$ .

**Lemma 9.** Let  $\pi''$  and  $\tau''$  be regular patterns output by *LEARN\_P1\_TAU2*. There are no regular patterns  $\pi'''$  and  $\tau'''$  such that  $|\pi'''| = |\pi''|$ ,  $\pi''' \prec \pi''$ ,  $L(\Gamma_*) \subseteq L(\{\pi''', \tau'''\})$ ,  $|\tau'''| = |\tau''|$ ,  $\tau''' \prec \tau''$  and  $L(\Gamma_*) \subseteq L(\pi'', \tau''')$ .

By Lemma 4 and Lemma 9, regular patterns  $\pi''$  and  $\tau''$  output by the procedure *LEARN\_P1\_TAU2* satisfy  $\pi_* \equiv \pi''$  and  $\tau_* \equiv \tau''$ , or  $\pi_* \equiv \tau''$  and  $\tau_* \equiv \pi''$ . The procedure uses  $O(|\pi_*| + |\tau_*|)$  superset queries, and runs in  $O(|\pi_*|^2 + |\tau_*|^2)$  time. Since  $|\pi_*| = |\tau_*|$ , it uses  $O(|\pi_*|)$  superset queries, and runs in  $O(|\pi_*|^2)$  time.

The procedure *LENGTH2* of Fig. 2 takes a regular pattern  $\pi$  satisfying Condition A and  $L(\Gamma_*) \not\subseteq L(\{\pi\})$  as input, and outputs a shortest string  $w \in L(\Gamma_*) - L(\{\pi\})$ , where  $w$  is a counterexample output by *Sup $_{\Gamma_*}$* . In the case (iv),  $|w| = |\tau_{*\pi_*}| \leq |\pi_*| |\tau_*|$ . Thus, the procedure uses  $O(|\pi_*| |\tau_*|)$  superset queries, and runs in  $O(|\pi_*|^2 |\tau_*|^2)$  time.

The procedure *LEARN\_TAU1* of Fig. 2 takes a regular pattern  $\pi$  satisfying Condition A and  $L(\Gamma_*) \not\subseteq L(\{\pi\})$ , and a shortest string  $w \in L(\Gamma_*) - L(\{\pi\})$  as input. The procedure outputs a regular pattern  $\tau$  with  $|\tau| = |w|$  and  $w \prec \tau$ .

**Lemma 10.** Let  $\Gamma_* = (\pi_*, \tau_*)$  be a reduced EFS in *rEFS* and  $w$  a shortest string in  $L(\Gamma_*) - L(\{\pi_*\})$ . Then, for a positive integer  $i$  ( $1 \leq i \leq |w|$ ), the following statements are equivalent: (1) There is a symbol  $\alpha \in \Sigma$  such that  $\alpha \neq w[i]$  and  $w_{i,\alpha} \in L(\Gamma_*) - L(\{\pi_*\})$ . (2)  $\tau_{*\pi_*}[i] \in X$ .

Since  $w$  is a shortest string in  $L(\Gamma_*) - L(\{\pi_*\})$ ,  $|w| \geq |\tau_{*\pi_*}|$ . We assume  $|w| > |\tau_{*\pi_*}|$ . It implies  $S_1(\tau_{*\pi_*}) \subseteq L(\{\pi_*\})$ . Thus,  $L(\pi_*) \cap L(\Gamma_{*\tau_*}) \neq \emptyset$ . This contradicts that  $\Gamma_*$  is reduced. Therefore,  $|w| = |\tau_{*\pi_*}|$ .

((1)  $\Rightarrow$  (2)) From  $w \in L(\Gamma_*) - L(\{\pi_*\})$ ,  $w \in L(\Gamma_{*\tau_*})$ . Since  $w \in L(\Gamma_{*\tau_*})$  and  $|w| = |\tau_{*\pi_*}|$ ,  $w \preceq \tau_{*\pi_*}$ . We can show  $w_{i,\alpha} \preceq \tau_{*\pi_*}$  in a similar way. Since  $|w| = |\tau_{*\pi_*}|$ ,  $w \preceq \tau_{*\pi_*}$  and  $w_{i,\alpha} \preceq \tau_{*\pi_*}$ , we have  $\tau_{*\pi_*}[i] \in X$ .

((2)  $\Rightarrow$  (1)) From  $\tau_{*\pi_*}[i] \in X$ ,  $w_{i,\alpha} \in L(\Gamma_{*\tau_*})$  for any symbol  $\alpha \in \Sigma$ . It implies that  $w_{i,\alpha} \in L(\Gamma_*)$ . Since  $\Gamma_*$  is reduced, we have  $L(\pi_*) \cap L(\Gamma_{*\tau_*}) = \phi$ . Thus, there is a string  $w_{i,\alpha} \in L(\Gamma_*) - L(\{\pi_*\})$ .  $\square$

To show Lemma 12, we need the following lemma. Note that we assume  $|\Sigma| \geq 5$  in this paper.

**Lemma 11.** [9] Let  $|\Sigma| \geq 3$ ,  $\pi$  and  $\tau$  regular patterns, and  $x$  a variable in  $\pi$ . If  $\pi\{x := a\} \preceq \tau$ ,  $\pi\{x := b\} \preceq \tau$  and  $\pi\{x := c\} \preceq \tau$  for symbols  $a, b$  and  $c$  in  $\Sigma$  which are mutually distinct, then  $\pi \preceq \tau$ .



**Procedure** *LEARN\_PL-TAU2*( $\pi', \tau'$ )

*Input:* Regular patterns  $\pi'$  and  $\tau'$   
satisfying Condition B-2 for  $\pi$ ,  
where  $\pi$  satisfies Condition B;  
*Given:* An oracle  $Sup_{\Gamma_*}$  for the target  $\Gamma_*$ ;

**begin** $\pi'' := \pi'; i := 1;$ 

**while** ( $i \leq |\pi''|$ ) **and** ( $\pi''[i] \in \Sigma$ ) **do**  
 $i := i + 1;$

**while**  $i \leq |\pi''|$  **do begin****foreach**  $\alpha \in \Sigma$  **do begin**

**if**  $Sup_{\Gamma_*}(\{\pi''_{i,\alpha}, \tau'\}) = \text{"yes"}$  **then**  
**begin**

 $\pi'' := \pi''_{i,\alpha};$  **break;****end;****end;** $i := i + 1;$ 

**while** ( $i \leq |\pi''|$ ) **and** ( $\pi''[i] \in \Sigma$ ) **do**  
 $i := i + 1;$

**end;** $\tau'' := \tau'; i := 1;$ 

**while** ( $i \leq |\tau''|$ ) **and** ( $\tau''[i] \in \Sigma$ ) **do**  
 $i := i + 1;$

**while**  $i \leq |\tau''|$  **do begin****foreach**  $\alpha \in \Sigma$  **do begin**

**if**  $Sup_{\Gamma_*}(\{\pi', \tau''_{i,\alpha}\}) = \text{"yes"}$  **then**  
**begin**

 $\tau'' := \tau''_{i,\alpha};$  **break;****end;****end;** $i := i + 1;$ 

**while** ( $i \leq |\tau''|$ ) **and** ( $\tau''[i] \in \Sigma$ ) **do**  
 $i := i + 1;$

**end;****output**  $\pi'', \tau'';$ **end.****Procedure** *LENGTH2*( $\pi$ )

*Input:* A regular pattern  $\pi$  satisfying  
Condition A and  $L(\Gamma_*) \not\subseteq L(\pi)$ ;  
*Given:* An oracle  $Sup_{\Gamma_*}$  for the target  $\Gamma_*$

**begin** $\ell := |\pi| + 1;$ //  $x_1 \cdots x_{\ell+1}$  is a regular pattern.

**while**  $Sup_{\Gamma_*}(\{\pi, x_1 \cdots x_{\ell+1}\}) = \text{"yes"}$  **do**  
 $\ell := \ell + 1;$

Let  $w$  be a counterexample obtained  
by the oracle  $Sup_{\Gamma_*}$ ;

**output**  $w;$ **end.****Procedure** *LEARN-TAU1*( $\pi, w$ )

*Input:* A regular pattern  $\pi$  satisfying  
Condition A and  $L(\Gamma_*) \not\subseteq L(\{\pi\})$ , and  
a shortest string  $w \in L(\Gamma_*) - L(\{\pi\})$ ;  
*Given:* An oracle  $Mem_{\Gamma_*}$  for the target  $\Gamma_*$ ;

**begin** $\tau := w;$ **for**  $i := 1$  **to**  $|w|$  **do****foreach**  $\alpha \in \Sigma$  **do****if**  $w[i] \neq \alpha$  **then**

**if**  $Mem_{\Gamma_*}(w_{i,\alpha}) = \text{"yes"}$  **then**  
**if**  $w_{i,\alpha} \notin L(\pi)$  **then begin**

 $\tau[i] := x_i;$  **break;****end;****output**  $\tau;$ **end.****Fig. 2.** Procedure *LEARN\_PL-TAU2*, *LENGTH2* and *LEARN-TAU1*

**Lemma 12.** Let  $\Gamma_* = \{\pi_*, \tau_*\}$  be a reduced EFS in  $r\mathcal{EFS}$  with  $|\pi_*| < |\tau_*|$ , and  $w$  a shortest string in  $L(\Gamma_*) - L(\{\pi_*\})$ . Then, for a positive integer  $i$  ( $1 \leq i \leq |w|$ ), the following statements are equivalent: (1) There is a symbol  $\alpha \in \Sigma$  with  $\alpha \neq w[i]$  and  $w_{i,\alpha} \in L(\Gamma_*) - L(\{\pi_*\})$ . (2)  $\tau_*[i] \in X$ .

Since  $w \in L(\Gamma_*) - L(\{\pi_*\})$ ,  $|w| \geq |\tau_*|$ . We assume  $|w| > |\tau_*|$ . It implies  $S_1(\tau_*) \subseteq L(\{\pi_*\})$ . By Lemma 2,  $\tau_* \preceq \pi_*$ . This contradicts that  $\Gamma_*$  is reduced. Thus,  $|w| = |\tau_*|$ .

(1)  $\Rightarrow$  (2) We can show in a similar way as Lemma 10.

((2)  $\Rightarrow$  (1)) From  $\tau_*[i] \in X$ ,  $w_{i,\alpha} \in L(\tau_*)$  for any symbol  $\alpha \in \Sigma$ . It implies  $w_{i,\alpha} \in L(\Gamma_*)$ . We assume  $w_{i,\alpha} \in L(\pi_*)$  for any symbol  $\alpha \in \Sigma$  with  $\alpha \neq w[i]$ . Let  $\pi' = w[: i - 1]xw[i + 1 :]$ . It is clear that  $w \preceq \pi'$ . Since  $|\Sigma| \geq 5$ , there are symbols  $a, b$  and  $c$  in  $\Sigma$  such that  $a, b, c$  and  $w[i]$  are mutually distinct. By Lemma 11, since  $w_{i,a} = \pi'\{x := a\} \preceq \pi_*$ ,  $w_{i,b} = \pi'\{x := b\} \preceq \pi_*$  and  $w_{i,c} = \pi'\{x := c\} \preceq \pi_*$ , we have  $\pi' \preceq \pi_*$ . From  $w \preceq \pi'$  and  $\pi' \preceq \pi_*$ ,  $w \in L(\pi_*)$ . This contradicts with  $w \notin L(\{\pi_*\})$ . Therefore, there is a symbol  $\alpha \in \Sigma$  with  $w_{i,\alpha} \in L(\Gamma_*) - L(\{\pi_*\})$ .  $\square$

By using Lemma 10 and Lemma 12, we can show Lemma 13.

**Lemma 13.** Let  $\tau$  be a regular pattern output by the procedure *LEARN\_TAU1*. There is no regular pattern  $\tau'$  such that  $|\tau'| = |\tau|$ ,  $\tau \prec \tau'$  and  $L(\{\tau'\}) \subseteq L(\Gamma_*)$ .

It is clear that  $|w| = |\tau|$ . We assume that there is a regular pattern  $\tau'$  such that  $|\tau'| = |\tau|$ ,  $\tau \prec \tau'$  and  $L(\{\tau'\}) \subseteq L(\Gamma_*)$ . By Lemma 1 and Lemma 2,  $\Gamma_*$  is in the case (iii) or (iv), and  $\pi \equiv \pi_*$ . At First, we assume that  $\Gamma_*$  is in the case (iii). If  $\tau' \preceq \pi$ , then  $w \preceq \pi$ . This contradicts with  $w \notin L(\{\pi\})$ . It implies that  $\tau' \not\preceq \pi$ . Since  $\tau' \not\preceq \pi$  and  $L(\{\tau'\}) \subseteq L(\Gamma_*)$ , we have  $\tau' \preceq \tau_*$ . By Lemma 5,  $|\tau| = |\tau_*|$ . Since  $|\tau'| = |\tau|$  and  $\tau \prec \tau'$ , there is a positive integer  $i \in \{1, \dots, |\tau|\}$  such that  $\tau[i] \in \Sigma$  and  $\tau'[i] \in X$ . From  $\tau' \preceq \tau_*$  and  $|\tau| = |\tau'| = |\tau_*|$ ,  $\tau_*[i] \in X$ . By Lemma 12, there is a symbol  $\alpha \in \Sigma$  such that  $\alpha \neq w[i]$  and  $w_{i,\alpha} \in L(\Gamma_*) - L(\{\pi\})$ . This contradicts with  $\tau[i] \in \Sigma$ . Therefore,  $\Gamma_*$  is not in the case (iii).

In case that  $\Gamma_*$  is in the case (iv), we can show in a similar way. Thus, there is no regular pattern  $\tau'$  such that  $|\tau'| = |\tau|$ ,  $\tau \prec \tau'$  and  $L(\{\tau'\}) \subseteq L(\Gamma_*)$ .  $\square$

By Lemma 13, the procedure *LEARN\_TAU1* outputs a regular pattern  $\tau$  satisfying Condition C-2 for  $\pi$ . Since  $|w| \leq |\pi_*||\tau_*|$ , it uses  $O(|\pi_*||\tau_*|)$  membership queries, and runs in  $O(|\pi_*|^2|\tau_*|^2)$  time.

The procedure *LEARN\_TAU2* of Fig. 3 takes regular patterns  $\pi$  and  $\tau$  such that  $\pi$  satisfies Condition C,  $\tau$  satisfies Condition C-2 for  $\pi$ , and  $L(\Gamma_*) \not\subseteq L(\{\pi, \tau\})$  as input. The procedure outputs a regular pattern  $\tau'$  with  $\tau' \equiv \tau_*$ .

**Theorem 3.** The algorithm *LEARN\_REFS* of Fig. 3 identifies any EFS  $\Gamma \in r\mathcal{EFS}$  in polynomial time using  $O(|\Gamma_*|^2)$  membership queries and  $O(|\Gamma_*|^2)$  superset queries, where  $|\Sigma| \geq 5$ .

By Lemma 8, the procedure *LEARN\_PI* outputs a regular pattern  $\pi$  satisfying Condition A. For  $\pi$ , we consider the following cases:

1. In case that  $L(\Gamma_*) \subseteq L(\{\pi\})$ . If the procedure *LEARN\_PLTAU1* outputs  $\pi$ , then there are no regular patterns  $\pi'$  and  $\tau'$  satisfying Condition B-2 for  $\pi$ . By Lemma 1 and Lemma 3,  $\Gamma_*$  is in the case (i) and  $\pi \equiv \pi_*$ . Thus,  $L(\Gamma_*) = L(\{\pi\})$ . By Lemma 3, if the procedure *LEARN\_PLTAU1* outputs regular patterns, then  $\Gamma_*$  is in the case (ii). By Lemma 4 and Lemma 9, the procedure *LEARN\_PLTAU2* outputs regular patterns  $\pi''$  and  $\tau''$  with  $\pi'' \equiv \pi_*$  and  $\tau'' \equiv \tau_*$ , or  $\pi'' \equiv \tau_*$  and  $\tau'' \equiv \pi_*$ . Thus,  $L(\Gamma_*) = L(\{\pi'', \tau''\})$ .

**Procedure** *LEARN\_TAU2*( $\pi, \tau$ )

*Input:* Regular patterns  $\pi$  and  $\tau$   
 such that  $\pi$  satisfies Condition C,  
 $\tau$  satisfies Condition C-2 for  $\pi$ ,  
 and  $L(\Gamma_*) \not\subseteq L(\{\pi, \tau\})$ ;

**begin** $\tau' := \varepsilon; i := 1; j := 1;$ **while**  $j \leq |\tau| - |\pi| + 1$  **do begin** $\pi' := \tau[j : j + |\pi| - 1];$ **if**  $\pi \equiv \pi'$  **then begin** $\tau' := \tau' \tau[i : j - 1] x_j;$  $i := j + |\pi|;$  $j := j + |\pi|;$ **end else begin** $j := j + 1;$ **end;****end;** $\tau' := \tau' \tau[j : ];$ **output**  $\tau'$ ;**end.****Algorithm** *LEARN\_REFS*

*Given:* An oracle  $\text{Sup}_{\Gamma_*}$  for the target  $\Gamma_*$ ;

**begin** $\ell := \text{LENGTH1};$  $\pi := \text{LEARN\_PI}(\ell);$ **if**  $\text{Sup}_{\Gamma_*}(\{\pi\}) = \text{"yes"}$  **then begin****if**  $\text{LEARN\_PI\_TAU1}(\pi) = \text{"no"}$  **then** $H := \{\pi\}$ **else begin**Let  $\pi'$  and  $\tau'$  be regular patternsoutput by  $\text{LEARN\_PI\_TAU1}(\pi);$  $\{\pi'', \tau''\} := \text{LEARN\_PI\_TAU2}(\pi', \tau');$  $H := \{\pi'', \tau''\};$ **end;****end else begin** $w := \text{LENGTH2}(\pi);$  $\tau := \text{LEARN\_TAU1}(\pi, w);$ **if**  $\text{Sup}_{\Gamma_*}(\{\pi, \tau\}) = \text{"yes"}$  **then** $H := \{\pi, \tau\}$ **else begin** $\tau' := \text{LEARN\_TAU2}(\pi, \tau);$  $H := (\pi, \tau');$ **end;****end;****output**  $H;$ **end.****Fig. 3.** Procedure *LEARN\_TAU2* and Algorithm *LEARN\_REFS*

2. In case that  $L(\Gamma_*) \not\subseteq L(\{\pi\})$ . By Lemma 2,  $\Gamma_*$  is in the case (iii) or (iv). By Lemma 1,  $\pi \equiv \pi_*$ . By Lemma 13, the procedure *LEARN\_TAU1* outputs a regular pattern  $\tau$  satisfying Condition C-2 for  $\pi$ . By Lemma 6 and Lemma 7, if  $L(\Gamma_*) \subseteq L(\{\pi, \tau\})$ , then  $\Gamma_*$  is in the case (iii) and  $\tau \equiv \tau_*$ . Thus,  $L(\Gamma_*) = L(\{\pi, \tau\})$ . By Lemma 6 and Lemma 7, if  $L(\Gamma_*) \not\subseteq L(\{\pi, \tau\})$ , then  $\Gamma_*$  is in the case (iv) and  $\tau \equiv \tau_{*\pi_*}$ . Then the procedure *LEARN\_TAU2* outputs a regular pattern  $\tau'$  with  $\tau' \equiv \tau_*$ . Thus,  $L(\Gamma_*) = L((\pi, \tau'))$ .

Therefore, the algorithm outputs an EFS  $H \in r\mathcal{EFS}$  with  $L(\Gamma_*) = L(H)$ .

In case that  $\Gamma_*$  is in the case (i) or (ii), the algorithm runs in  $O(|\pi_*|^3)$  time and uses  $O(|\pi_*|^2)$  superset queries. In case that  $\Gamma_*$  is in the case (iii) or (iv), it runs in  $O(|\pi_*|^2 |\tau_*|^2)$  time and uses  $O(|\pi_*| |\tau_*|)$  superset queries and  $O(|\pi_*| |\tau_*|)$  membership queries. Thus, it runs in  $O(|\pi_*|^2 (|\pi_*| + |\tau_*|^2))$  time, and uses  $O(|\pi_*| |\tau_*|)$  membership queries and  $O(|\pi_*| (|\pi_*| + |\tau_*|))$  superset queries. Since  $|\pi_*| \leq |\Gamma_*|$  and  $|\tau_*| \leq |\Gamma_*|$ , it runs in  $O(|\Gamma_*|^4)$  time and uses  $O(|\Gamma_*|^2)$  membership queries and  $O(|\Gamma_*|^2)$  superset queries.  $\square$

**Table 1.** Our results and future works

	Exact Learning		Inductive Inference from Positive Data
$r\mathcal{EFS}$	sufficiency [This work]	insufficiency [This work]	<i>Open</i>
	superset query & membership query	membership query equivalence query subset query	
$\mathcal{PL}$	restricted superset query [2]		<i>Yes</i> [1]
$\mathcal{RPL}$	membership query & one positive example [5]		polynomial time [10]

## 5 Hardness Results on Learnability

In this section, we show the insufficiency of learning  $r\mathcal{EFS}$  in the query learning model. By Lemma 14, we have Theorem 4. We omit the proof of Theorem 4.

**Lemma 14.** [2] Suppose the hypothesis space contains a class of distinct sets  $L_1, \dots, L_N$ , and there exists a set  $L_\cap$  which is not a hypothesis, such that for any pair of distinct indices  $i$  and  $j$ ,  $L_\cap = L_i \cap L_j$ . Then any algorithm that exactly identifies each of the hypotheses  $L_i$  using equivalence, membership, and subset queries must at least  $N - 1$  queries in the worst case.

**Theorem 4.** Any learning algorithm that exactly identifies all strings of length  $n$  using equivalence, membership and subset queries must make at least  $5^n - 1$  queries in the worst case.

## 6 Conclusions

In this paper, we have investigated exact identification of an EFS in  $r\mathcal{EFS}$  using queries. We have shown that any EFS in  $r\mathcal{EFS}$  is exactly identifiable in  $O(|\Gamma_*|^4)$  time using  $O(|\Gamma_*|^2)$  membership queries and  $O(|\Gamma_*|^2)$  superset queries, where  $|\Sigma| \geq 5$ . Moreover, we showed that there exists no polynomial time learning algorithm which identifies any EFS in  $r\mathcal{EFS}$  using membership, equivalence and subset queries. As future works, we will consider the learnability of  $r\mathcal{EFS}$  in the framework of inductive inference from positive data. We summarize our results and future works in Table 1. We denote by  $\mathcal{PL}$  (resp.,  $\mathcal{RPL}$ ) the set of all patterns (resp., regular patterns).

## References

- [1] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Science*, 21:46–62, 1980.
- [2] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
- [3] S. Arikawa. Elementary formal systems and formal languages - simple formal systems. *Memoirs of Faculty of Science, Kyushu University, Series A, Mathematics*, 24:47–75, 1970.

- [4] S. Arikawa, T. Shinohara, and A. Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, 95:97–113, 1992.
- [5] S. Matsumoto and A. Shinohara. Learning pattern languages using queries. *Proc. EuroCOLT-97, Springer-Verlag, LNAI 1208*, pages 185–197, 1997.
- [6] S. Matsumoto, A. Shinohara, H. Arimura, and T. Shinohara. Learning subsequence languages. In *Information Modelling and Knowledge Bases VIII*, pages 335–344. IOS Press, 1997.
- [7] S. Matsumoto, T. Shoudai, T. Miyahara, and T. Uchida. Learning of finite unions of tree patterns with internal structured variables from queries. *Proc. AI-2002, Springer LNAI 2557*, pages 523–534, 2002.
- [8] H. Sakamoto, K. Hirata, and H. Arimura. Learning elementary formal systems with queries. *Theoretical Computer Science*, 298:21–50, 2003.
- [9] M. Sato, Y. Mukouchi, and D. Zheng. Characteristic sets for unions of regular pattern languages and compactness. In *Proc. ALT-98, Springer-Verlag, LNAI 1501*, pages 220–233. Springer, 1998.
- [10] T. Shinohara. Inductive inference of formal systems from positive data. *Bulletin of Informatics and Cybernetics*, 22:9–18, 1986.
- [11] J. Uemura and M. Sato. Learning of erasing primitive formal systems from positive examples. In *Proc. ALT-2003, Springer-Verlag, LNAI 2842*, pages 69–83. Springer-Verlag, 2003.

# Gold-Style and Query Learning Under Various Constraints on the Target Class

Sanjay Jain,<sup>1,\*</sup> Steffen Lange,<sup>2</sup> and Sandra Zilles<sup>3</sup>

<sup>1</sup> School of Computing, National University of Singapore, Singapore 117543  
`sanjay@comp.nus.edu.sg`

<sup>2</sup> Fachhochschule Darmstadt, FB Informatik, Haardtring 100,  
64295 Darmstadt, Germany  
`s.lange@fbi.fh-darmstadt.de`

<sup>3</sup> DFKI GmbH, Erwin-Schrödinger-Straße, 67663 Kaiserslautern, Germany  
`sandra.zilles@dfki.de`

**Abstract.** In language learning, strong relationships between Gold-style models and query models have recently been observed: in some quite general setting Gold-style learners can be replaced by query learners and vice versa, without loss of learning capabilities. These ‘equalities’ hold in the context of learning indexable classes of recursive languages.

Former studies on Gold-style learning of such indexable classes have shown that, in many settings, the enumerability of the target class and the recursiveness of its languages are crucial for learnability. Moreover, studying query learning, non-indexable classes have been mainly neglected up to now. So it is conceivable that the recently observed relations between Gold-style and query learning are not due to common structures in the learning processes in both models, but rather to the enumerability of the target classes or the recursiveness of their languages.

In this paper, the analysis is lifted onto the context of learning arbitrary classes of r.e. languages. Still, strong relationships between the approaches of Gold-style and query learning are proven, but there are significant changes to the former results. Though in many cases learners of one type can still be replaced by learners of the other type, in general this does not remain valid vice versa. All results hold even for learning classes of recursive languages, which indicates that the recursiveness of the languages is not crucial for the former ‘equality’ results. Thus we analyse how constraints on the algorithmic structure of the target class affect the relations between two approaches to language learning.

## 1 Introduction

In order to model different aspects of human learning and machine learning, different abstract approaches have to be considered. Each model analysed within the scope of learning theory addresses only special facets of learning. For example, in Gold’s [9] model of  $\Psi$ -learning is interpreted as

---

\* Sanjay Jain was supported in part by NUS grant number R252-000-127-112.

a limiting process of generating and improving hypotheses about a target concept. These hypotheses are built upon instances of the target concept offered to the learner. In the limit, the output of the learner is supposed to stabilize on a correct guess, but during the learning process one never knows whether or not the current hypothesis is already correct. The potential of changing its mind is a crucial quality of the learner.

In contrast to that, Angluin's [3, 4] model of  $\text{EXACT}_q$  is concerned with learning as a finite process in which a learner and a teacher interact. The learner asks questions of a specified type about the target concept and the teacher answers these reliably. After finitely many steps the learner is required to return a single hypothesis, which then correctly describes the target concept. Here the crucial characteristics of the learner are its access to special information on the target concept and its confinements in terms of mind changes. Since a query learner identifies the target concept with just a single hypothesis, we allude to this scheme as  $\text{EXACT}_q$ .<sup>1</sup>

Recently, the combination of these two approaches [11, 12] as well as the common features of learners in either model [14, 15] have gained interest in the learning theory community. [14, 15] contributes a systematic analysis of common features of both approaches, thereby focussing on the identification of formal languages, ranging over indexable classes of recursive languages, as target concepts, see [2, 13, 19]. Characterising different types of Gold-style language learning in terms of query learning has pointed out correspondences between the two models. In particular, [14, 15] demonstrate how learners identifying languages in the limit can be replaced by one-shot learners without loss of learning power—and vice versa. That means, under certain circumstances the capabilities of limit learners are equal to those of one-shot learners using queries. An important parameter in this context is the range of possible hypothesis spaces/query spaces used during the learning process. Despite the fundamental differences in the definitions of the two learning paradigms, there are strong relations—at least in the case of learning indexable families of recursive languages.

The latter restriction had initially been made, since many natural language classes are indexable. Former studies [19] on Gold-style learning of indexable classes of languages have shown that, in many settings, the enumerability of the target class may be the crucial reason for positive learnability results. Moreover, when studying query learning, non-indexable classes have been mainly neglected up to now. So it is conceivable that the strong relationships between Gold-style and query learning observed in [14, 15] are not caused by common structures in the learning processes in both models, but rather by the enumerability of the target classes or maybe at least by the recursiveness of the target languages themselves. In order to determine the actual cause for the relationships observed before, we now lift the analysis thereof onto more complex classes of languages.

Therefore the current paper concerns the relationships of Gold-style learning and query learning for the case that arbitrary classes of r.e. languages form the

---

<sup>1</sup> Most studies on query learning mainly deal with the efficiency of query learners, whereas, in what follows, we are only interested in qualitative learnability results.

target. This is additionally based on the following observation: when trying to learn a class of recursive languages, a certain type of learner may sometimes be successful only in case the learner uses a hypothesis space comprising more than the languages to be learned—such as for instance a hypothesis space given by an r.e. indexing of r.e. languages. Then a natural question might be whether it is possible to learn not only the initial target class, but additionally the languages represented by further queries a learner asks or further hypotheses a learner states during learning the initial target languages. This again leads to the problem of learning r.e. languages. Literature, see e.g. [7], knows more examples of lifting results on learning recursive languages, as in [2], to learning r.e. languages.

From now on assume that arbitrary classes of r.e. languages form the target classes. Below we prove that in almost all cases, where equivalences between two learning models  $A$  and  $B$  had been witnessed for learning indexable classes of recursive languages, learners of type  $A$  can be replaced by learners of type  $B$  without loss of learning power—but no longer vice versa. So, although most of the equivalences between Gold-style models and query models no longer hold, at least some of the inclusions hold, thereby forming a hierarchy of inference types. This shows that huge parts of the relationships shown for learning indexable classes of recursive languages are maintained; the cause must be common structures of learning processes in Gold-style and query learning! An important parameter in the final hierarchy is again the underlying hypothesis space/query space.

Interestingly, all separations of inference types in the final hierarchy can be witnessed even by (non-indexable) classes of recursive languages. This raises the question whether the main reason for the equivalence results in [14, 15] is the fact that the classes considered are enumerable and not that the languages themselves are recursive. So we analysed whether the results in [14, 15] can be lifted to the case of learning enumerable classes of r.e. languages. The relationships observed are somewhat dismal: several of the equivalence results do not hold for learning enumerable classes of r.e. languages, but at least one of them does. That means that in most but not in all cases, the main reason for the equivalence results in [14, 15] lies not only in the enumerability of the target classes.

## 2 Preliminaries

Familiarity with standard recursion theoretic notions is assumed, see [17, 10]. From now on, a fixed finite alphabet  $\Sigma$  with  $\{a, b\} \subseteq \Sigma$  is given. A word is any element from  $\Sigma^*$  and a set any subset of  $\Sigma^*$ . The complement of a language  $L$ , denoted  $\bar{L}$ , is the set  $\Sigma^* \setminus L$ . Any total function  $t : \mathbb{N} \rightarrow \Sigma^*$  with  $\{t(i) \mid i \in \mathbb{N}\} = L$  is called a *realizer* for  $L$ . A text  $t$  is often identified with an infinite sequence  $(t(i))_{i \in \mathbb{N}}$ . Then, given  $n \in \mathbb{N}$ ,  $t_n$  is the initial segment  $(t(0), \dots, t(n))$  and  $\text{content}(t_n)$  denotes the set  $\{t(0), \dots, t(n)\}$ .

In the sequel,  $\varphi$  is a Gödel numbering of all partial recursive functions and  $K = \{i \in \mathbb{N} \mid \varphi_i(i) \text{ is defined}\}$ . The language family  $(W_i)_{i \in \mathbb{N}}$  is given by  $W_i = \{w_j \mid \varphi_i(j) \text{ is defined}\}$  for all  $i \in \mathbb{N}$ , where  $(w_j)_{j \in \mathbb{N}}$  is a repetition-free effective enumeration of  $\Sigma^*$ . Then  $W_{i,s}$ ,  $s \in \mathbb{N}$ , is the set of all words  $w_j$ , such that  $j < s$



and  $\varphi_i(j)$  terminates within  $s$  steps. Given  $A \subseteq \mathbb{N}$ , an  $A$ -recursive function is a function recursive using an oracle for the set  $A$ .

A family  $(A_i)_{i \in \mathbb{N}}$  of languages is *uniformly recursive* if there is a recursive (partial recursive) function  $f$  such that  $A_i = \{w \in \Sigma^* \mid f(i, w) = 1\}$  for all  $i \in \mathbb{N}$ . For uniformly recursive families membership is uniformly decidable. A family  $(A_i)_{i \in \mathbb{N}}$  is *uniformly recursively enumerable* if there is a recursive function  $g$  such that  $A_i = \{w \in \Sigma^* \mid g(i, w, n) = 1 \text{ for all but finitely many } n\}$  for all  $i \in \mathbb{N}$ . A class  $\mathcal{C}$  of recursive languages over  $\Sigma^*$  is called an *uniformly recursively enumerable class* (or *uniformly r.e. class* for short), if there is a uniformly recursive family  $(L_i)_{i \in \mathbb{N}}$  of all and only the languages in  $\mathcal{C}$ .

### 2.1 Gold-Style Language Learning

Let  $\mathcal{C}$  be a class of r.e. languages,  $\mathcal{H} = (A_i)_{i \in \mathbb{N}}$  a language family (a *hypothesis space*). An *algorithmic device*  $M$  is an algorithmic device that reads longer and longer initial segments  $\sigma$  of a text and outputs numbers  $M(\sigma)$ . Returning  $i$ ,  $M$  is construed to hypothesize the language  $A_i$ .

The following definition of learning in the limit is based on [9]. Given a text  $t$  for  $L \in \mathcal{C}$ ,  $M$  identifies  $L$  from text  $t$  if the sequence of hypotheses output by  $M$ , when fed  $t$ , stabilizes on a number  $i$  (i.e., past some point  $M$  always outputs the hypothesis  $i$ ) with  $A_i = L$ .  $M$  identifies  $\mathcal{C}$  from text with respect to  $\mathcal{H}$ , if it identifies every  $L' \in \mathcal{C}$  from every text for  $L'$ . In what follows, we focus our studies on uniformly r.e. families as hypothesis spaces.  $\mathcal{C}'$  denotes the collection of all classes  $\mathcal{C}'$  for which there is a uniformly r.e. hypothesis space  $\mathcal{H}$  and an IIM  $M'$  identifying  $\mathcal{C}'$  in the limit from text with respect to  $\mathcal{H}$ . A quite natural and often studied modification of  $\mathcal{C}'$  is defined by the model of *conservative IIM*, see [2, 13] for this concept in the context of learning recursive languages.  $M$  is a *conservative IIM* for  $\mathcal{C}$  with respect to  $\mathcal{H} = (A_i)_{i \in \mathbb{N}}$ , if  $M$  performs only justified mind changes, i.e., if  $M$ , on some text  $t$  for some  $L \in \mathcal{C}$ , outputs hypotheses  $i$  and later  $j$ , then  $M$  must have seen some element  $w \notin A_i$  before returning  $j$ . The collection of all classes identifiable from text by a conservative IIM with respect to some uniformly r.e. hypothesis space is denoted by  $\mathcal{C}'_{\text{conservative}}$ . Note that  $\mathcal{C}'_{\text{conservative}} \subset \mathcal{C}'$ , as witnessed by the indexable class used in [19] to separate  $\mathcal{C}'$ -learnable indexable classes from  $\mathcal{C}'_{\text{conservative}}$ -learnable indexable classes. Another often studied version of Gold-style language learning is *behaviourally correct learning* [6]: If  $\mathcal{C}$  is a class of r.e. languages,  $\mathcal{H} = (A_i)_{i \in \mathbb{N}}$  any hypothesis space,  $M$  an IIM, then  $M$  is a *behaviourally correct learner* for  $\mathcal{C}$  from text with respect to  $\mathcal{H}$ , if for each  $L \in \mathcal{C}$  and each text  $t$  for  $L$ , for all but finitely many  $n$ ,  $A_{M(t_n)} = L$  is fulfilled. Here  $M$  may alternate different correct hypotheses arbitrarily often instead of converging to a single hypothesis. Defining the notion  $\mathcal{C}'_{\text{behaviourally correct}}$  as usual yields  $\mathcal{C}'_{\text{behaviourally correct}} \supset \mathcal{C}'_{\text{conservative}}$  [6].

Since we analyse learning from text, we assume in the sequel that all target languages are *indexable*. One main aspect of human learning is modelled in the approach of learning in the limit: the ability to change one's mind. Thus learning is a process in which the learner may change its hypothesis arbitrarily often until

reaching its final correct guess. In particular, it is in general impossible to find out when the final hypothesis has been reached, i.e., when a success in learning has eventuated. The main concern of our analysis will be comparisons of such inference types to query learning models resulting in a hierarchy reflecting the capabilities of the corresponding learners.

Finally, note that each class in  $\mathcal{C}_{\text{r.e.}}$ ,  $\mathcal{C}_{\text{co-r.e.}}$ ,  $\mathcal{C}_{\text{c.e.}}$  can be learned using the hypothesis space  $(W_i)_{i \in \mathbb{N}}$ . We will use this property in our proofs below. These notions of learning are closely related to the notion of  $\mathcal{C}_{\text{r.e.}}$ -stabilizing [8]. If  $\mathcal{H} = (A_i)_{i \in \mathbb{N}}$  is a hypothesis space,  $M$  an IIM, and  $L$  a language, then any finite text segment  $\sigma$  of  $L$  is called a  $\mathcal{C}_{\text{r.e.}}$ -stabilizing sequence (a  $\mathcal{C}_{\text{co-r.e.}}$ -stabilizing sequence) for  $M$ ,  $L$ , and  $\mathcal{H}$ , if  $M(\sigma) = M(\sigma\sigma')$  ( $A_{M(\sigma)} = A_{M(\sigma\sigma')}$ ) for all finite text segments  $\sigma'$  of  $L$ . If  $L$  is  $\mathcal{C}_{\text{r.e.}}$ -learned by  $M$  ( $\mathcal{C}_{\text{co-r.e.}}$ -learned by  $M$ ) respecting  $\mathcal{H}$ , then there exists a  $\mathcal{C}_{\text{r.e.}}$ -stabilizing sequence (a  $\mathcal{C}_{\text{co-r.e.}}$ -stabilizing sequence) for  $M$ ,  $L$ , and  $\mathcal{H}$ .

### 2.2 Language Learning via Queries

In the query learning model, a learner has access to a teacher that truthfully answers queries of a specified kind. An algorithmic device that, depending on the reply on the previous queries, either computes a new query or returns a hypothesis and halts [3]. Its queries and hypotheses are coded as natural numbers; both will be interpreted with respect to an underlying  $\mathcal{C}_{\text{r.e.}}$ . We adapt Angluin’s original definition here for learning r.e. languages as follows: when learning a class  $\mathcal{C}$  of r.e. languages, any family  $(A_i)_{i \in \mathbb{N}}$  of languages may form a hypothesis space.

More formally, let  $\mathcal{C}$  be a class of r.e. languages, let  $L \in \mathcal{C}$ , let  $\mathcal{H} = (A_i)_{i \in \mathbb{N}}$  be a hypothesis space, let  $M$  be a query learner.  $M$   $\mathcal{C}$ -learns  $L$  with respect to  $\mathcal{H}$  if it eventually halts and its only hypothesis, say  $i$ , represents  $L$ , i.e.,  $A_i = L$ . So  $M$  returns its unique and correct guess  $i$  after finitely many queries. Moreover,  $M$   $\mathcal{C}$ -learns  $\mathcal{H}$  if it learns every  $L' \in \mathcal{C}$  with respect to  $\mathcal{H}$  using queries of the specified type. If  $L$  is a target language, a query learner  $M$  may ask:

- **Superset query.** The input is an index of a language  $L' \in \mathcal{C}$ . The answer is ‘yes’ or ‘no’, depending on whether or not  $L'$  is a superset of  $L$ .
- **Disjointness query.** The input is an index of a language  $L' \in \mathcal{C}$ . The answer is ‘yes’ or ‘no’, depending on whether or not  $L'$  and  $L$  are disjoint.

The term ‘restricted’ is used to distinguish these inference types from learning with superset (disjointness) queries, where, with each negative reply to a query  $j$  the learner is provided a counterexample, i.e., a word in  $L \setminus A_j$  (in  $L \cap A_j$ ).

$\mathcal{C}_{\text{r.e.}}$  and  $\mathcal{C}_{\text{co-r.e.}}$  denote the collections of all classes  $\mathcal{C}'$  for which there is a uniformly r.e. hypothesis space  $\mathcal{H}$  and a query learner  $M'$  learning  $\mathcal{C}'$  with respect to  $\mathcal{H}$  using restricted superset and restricted disjointness queries, respectively. In the sequel we will omit the term ‘restricted’ for convenience and will again without loss of generality assume that  $\mathcal{C}_{\text{r.e.}}$ -learners and  $\mathcal{C}_{\text{co-r.e.}}$ -learners always use the hypothesis space  $\mathcal{H} = (W_i)_{i \in \mathbb{N}}$ . In the literature, see Angluin [3, 4], more types of queries, such as (restricted) subset queries, membership

queries, and equivalence queries have been analysed, but in what follows we concentrate on the two types explained above. Obviously, superset and disjointness queries are in general not decidable, i.e., the teacher may be non-computable.

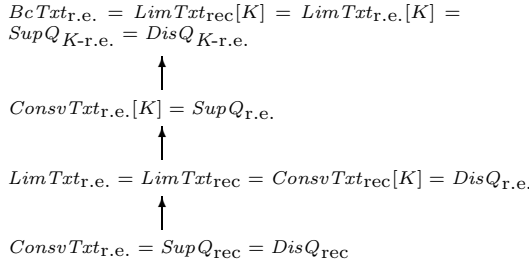
Note that, in contrast to the models of Gold-style language learning introduced above, learning via queries focusses the aspect of one-shot learning, i.e., it is concerned with scenarios in which learning eventuates without mind changes.

### 3 Learning Indexable Classes of Recursive Languages

Numerous studies on language learning restrict their focus on indexable classes, since, first, these include many natural classes of languages, and second, many conceptions can be simplified in this context. In particular, uniformly recursive families may be considered as hypothesis spaces in the approaches of both Gold-style and query learning (indicated by a subscript  $_{rec}$  instead of  $_{r.e.}$ ). In this section, all results referred to hold for indexable classes only. Recent studies [14, 15] have shown astonishing relations between the two approaches witnessed by equivalences of pairs of inference types, such as  $\text{BcTxt}_{rec} = \text{LimTxt}_{rec} = \text{ConsvTxt}_{r.e.}$  ( $= \text{ConsvTxt}_{rec}$ , see a result by Jain in [16]) and  $\text{SupQ}_{r.e.} = \text{DisQ}_{r.e.}$ . In these equalities, all inference types are considered restricted to indexable classes.

Concerning characterisations of  $\text{BcTxt}_{r.e.}$  and  $\text{LimTxt}_{rec}$  by similar means, oracle-IIMs as well as more general hypothesis spaces have been useful. Firstly, an oracle-IIM is an IIM which is recursive relative to an arbitrary oracle, i.e. its computation depends on according to which oracle it currently accesses, see e.g. [18]. For instance, using a  $K$ -oracle, such an IIM  $M$  becomes a  $K$ -recursive IIM  $M^K$ . Thus, e.g.,  $\text{ConsvTxt}_{r.e.}[K]$  denotes the collection of classes  $\text{ConsvTxt}_{r.e.}$ -learnable with the help of a  $K$ -oracle. Restricting such inference types to indexable classes, one obtains for instance  $\text{ConsvTxt}_{r.e.}[K] = \text{BcTxt}_{r.e.}$ .

Secondly, in order to characterise  $\text{LimTxt}_{rec}$ , uniformly  $K$ -r.e. hypothesis spaces have been introduced for query learning, indicated by a subscript  $_{K-r.e.}$  as in  $\text{SupQ}_{K-r.e.}$ . This has lead to the result  $\text{BcTxt}_{K-r.e.} = \text{LimTxt}_{K-r.e.} = \text{ConsvTxt}_{r.e.}$ .



**Figure 1.** This graph illustrates the relations between different inference types restricted to indexable classes as studied in [14, 15]. Arrows indicate proper inclusions of inference types.

### 4 Learning Classes of r.e. Languages

In the sequel, a hierarchy as in Figure 1 is established for arbitrary classes of r.e. languages. A section on query learning with uniformly r.e. hypothesis spaces is followed by a section dealing with uniformly  $K$ -r.e. hypothesis spaces. Note that indexable hypothesis spaces in general are obsolete here.

### 4.1 Results for Uniformly r.e. Hypothesis Spaces

Our first comparison already yields a change to the former hierarchy: when learning arbitrary classes of r.e. languages,  $\mathcal{U}_{\text{r.e.}}$ -learners can in general no longer be replaced by  $\mathcal{U}_{\text{r.e.}}$ -learners.

**Theorem 1.**  $\mathcal{U}_{\text{r.e.}} \# \mathcal{U}_{\text{r.e.}} \neq \emptyset$ .

$\mathcal{U}_{\text{r.e.}} \setminus \mathcal{U}_{\text{r.e.}} \neq \emptyset$  follows from Figure 1. For  $\mathcal{U}_{\text{r.e.}} \setminus \mathcal{U}_{\text{r.e.}} \neq \emptyset$ , consider the following class  $\mathcal{C}$ .

Let  $L_i = \{a^i b^x \mid x \in \mathbb{N}\}$ . Let  $L_i^S = \{a^i b^x \mid x \in S\}$  for any set  $S \subseteq \mathbb{N}$ .

Let  $\mathcal{C}_1 = \{L_i^S \mid i \in \mathbb{N}, \text{card}(S) < \infty, \exists e [\text{card}(S \cap \{x \mid x \leq 2e\}) > e + 1]\}$ .

Below, we will define a recursive function  $f$  such that, for all  $i$ , the following two properties hold:

(a)  $W_{f(i)} \subseteq L_i$  and  $W_{f(i)}$  is recursive (although an index for the characteristic function of  $W_{f(i)}$  in general cannot be obtained from  $i$ );

(b) For all  $e$ ,  $\text{card}(W_{f(i)} \cap \{a^i b^x \mid x \leq 2e\}) \leq e + 1$ .

Let  $\mathcal{C}_2 = \{W_{f(i)} \mid i \in \mathbb{N}\}$ . Let  $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ . Thus,  $\mathcal{C}$  is uniformly r.e. and consists only of recursive languages (however,  $\mathcal{C}$  is not an indexed family).

$\mathcal{C} \in \mathcal{U}_{\text{r.e.}}$  is witnessed by an IIM  $M$  which, on target  $L$ , first acquires an  $i$  with  $L \subseteq L_i$ .  $M$  outputs  $f(i)$ , until an  $e$  with  $\text{card}(\{x \mid a^i b^x \in L, x \leq 2e\}) > e + 1$  is found. In the latter case  $M$  runs the learning procedure for finite sets.

Let  $(M_i)_{i \in \mathbb{N}}$  be an enumeration of all  $\mathcal{U}_{\text{r.e.}}$ -learners. We now define  $f$ , such that (a) and (b) above are fulfilled and for each  $i$ ,  $M_i$  either does not  $\mathcal{U}_{\text{r.e.}}$ -identify  $W_{f(i)}$ , or it does not  $\mathcal{U}_{\text{r.e.}}$ -identify  $\mathcal{C}_1$ .

For any  $i, s \in \mathbb{N}$ , let  $W_{f(i)}^s$  denote the subset of  $W_{f(i)}$  enumerated before stage  $s$ . Let  $W_{f(i)}^0 = \{a^i\}$ , i.e., the word  $a^i$  is enumerated in  $W_{f(i)}$  at stage 0. Go to stage 0. In general, stage  $s$  reads as follows.

- Step 1: Run  $M_i$  for  $s$  steps, where each disjointness query  $j$  (representing  $W_j$ ) of  $M_i$  is answered ‘yes’, if  $W_{j,s} \cap W_{f(i)}^s = \emptyset$ ; ‘no’ otherwise.
- Step 2: If  $M_i$  does not output a hypothesis within  $s$  steps, go to stage  $s + 1$ . Else dovetail steps 2.1 and 2.2 until one of them succeeds. If 2.1 succeeds before 2.2, then go to stage  $s + 1$ , else if 2.2 succeeds, then go to step 3.
  - (2.1) Find a query  $j$  from step 1 which was answered ‘yes’, but  $W_j \cap W_{f(i)}^s \neq \emptyset$ .
  - (2.2) Find a query  $j$  from step 1 which was answered ‘yes’, and  $a^i b^y \in W_j$  for some  $y > 2s$ .
- Step 3: Let  $j, y$  be as found in step 2.2. Enumerate  $a^i b^y$  in  $W_{f(i)}$  and go to stage  $s + 1$  (otherwise stage  $s$  never ends).

Fix  $i$ . By construction,  $W_{f(i)}$  fulfils the conditions (a) (as either  $W_{f(i)}$  is finite, or  $a^i b^s \in W_{f(i)}$ , iff it is enumerated in  $W_{f(i)}$  before stage  $s$ ) and (b) (as at most  $s + 1$  elements are enumerated before stage  $s$ , and every element enumerated at or after stage  $s$  is of form  $a^i b^y$  for some  $y > 2s$ ). We consider two cases.

$\mathcal{U}_{\text{r.e.}} \setminus \mathcal{U}_{\text{r.e.}} \neq \emptyset$ . In this case clearly  $W_{f(i)}$  is finite. Now, since step 2.1 did not succeed, all questions of  $M_i$  in step 1 above for the input being  $W_{f(i)} = W_{f(i)}^s$ , are answered correctly at stage  $s$ , and  $M_i$  outputs

a hypothesis on  $W_{f(i)}$ . Furthermore, all questions  $j$  of  $M_i$  on  $W_{f(i)}$  which are answered ‘yes’, have the property that  $W_j \cap L_i$  is finite (since step 2.2 did not succeed). Thus, there exists a finite set  $S$  with  $L_i^S \in \mathcal{C}_1$  such that  $M_i$  behaves the same way on  $L_i^S$  as it does on  $W_{f(i)}$ . To see this, let  $S = \{x \mid a^i b^x \in W_{f(i)}\} \cup \{z \mid e \leq z \leq 2e\}$ , where  $e = 1 + \max(\{y \mid a^i b^y \in W_j \text{ for some query } j \text{ asked by } M_i \text{ on input } W_{f(i)}, \text{ and answered ‘yes’}\})$  (\* note that for each question  $j$  asked by  $M_i$  on input  $W_{f(i)}$  and answered ‘yes’,  $W_j \cap L_i$  is finite \*). Now,  $M_i$  can  $\mathcal{P}$ -r.e.-identify at most one of  $W_{f(i)}$  and  $L_i^S$ , both of which are in  $\mathcal{C}$ .

Consider the following subcases:

$\mathcal{P}$ -r.e.  $M_i$   $\mathcal{P}$ -r.e.-identifies  $W_{f(i)} \in \mathcal{C}$ . In this case clearly,  $M_i$  does not  $\mathcal{P}$ -r.e.-identify  $W_{f(i)} \in \mathcal{C}$ .

In this case let stage  $s$  be large enough so that, if  $j$  is a question asked by  $M_i$  on  $W_{f(i)}$  (when all the questions are answered correctly), and  $W_j \cap W_{f(i)} \neq \emptyset$ , then  $W_{j,s} \cap W_{f(i)}^s \neq \emptyset$ . Note that then beyond stage  $s$  all questions of  $M_i$  are answered correctly in step 1. Now step 2.1 and 2.2 cannot succeed. Thus the only way infinitely many stages can exist is by  $M_i$  not returning a hypothesis. A contradiction.

From the above cases it follows that  $M_i$  does not  $\mathcal{P}$ -r.e.-identify  $\mathcal{C}$ .

Thus we obtain  $\mathcal{P}\text{-r.e.} \# \mathcal{P}\text{-r.e.}$ . □

This incoherency holds since presently  $\mathcal{P}\text{-r.e.}$  no longer equals  $\mathcal{P}\text{-r.e.}$ . However, an inclusion as in Thm. 2 still indicates a relation between Gold-style and query learning, albeit weaker than when restricted to indexable classes.

**Theorem 2.**  $\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}$

For the inclusion the congruent proof for indexable classes in [14] can be adopted. The inequality follows from Thm. 1 and  $\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}$  □

The relationship between  $\mathcal{P}\text{-r.e.}$  and  $\mathcal{P}\text{-r.e.}$  remains unchanged from the former hierarchy, as Thm. 3 shows.

**Theorem 3.**  $\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}$

The proof of  $\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}$  is omitted. The underlying idea is similar to that in the proof of  $\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}$  in [14].  $\mathcal{P}\text{-r.e.} \setminus \mathcal{P}\text{-r.e.} \neq \emptyset$  is even witnessed by a uniformly recursive family of languages, see [14]. □

Interestingly, the characterisation  $\mathcal{P}\text{-r.e.} = \mathcal{P}\text{-r.e.}[K]$  persists when learning classes of r.e. languages. Here the proof for indexable classes [15] applies.

**Theorem 4.**  $\mathcal{P}\text{-r.e.} = \mathcal{P}\text{-r.e.}[K]$

Though  $\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}[K]$  persists (Thm. 5), the relation between  $\mathcal{P}\text{-r.e.}$  and  $\mathcal{P}\text{-r.e.}$  changes significantly for arbitrary classes of r.e. languages, see Thm. 7. The reason is that  $\mathcal{P}\text{-r.e.}[K]$  no longer equals  $\mathcal{P}\text{-r.e.}$  (Thm. 6).

**Theorem 5.**  $\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}[K]$

$\mathcal{P}\text{-r.e.} \subseteq \mathcal{P}\text{-r.e.}[K]$  follows from Thm. 4, since  $\mathcal{P}\text{-r.e.}[K]$  comprises  $\mathcal{P}\text{-r.e.}[K]$ . As  $\mathcal{P}\text{-r.e.} \setminus \mathcal{P}\text{-r.e.}[K] \neq \emptyset$  [15], Thm. 4 yields  $\mathcal{P}\text{-r.e.} \setminus \mathcal{P}\text{-r.e.} \neq \emptyset$ . Thm. 6 then implies  $\mathcal{P}\text{-r.e.}[K] \setminus \mathcal{P}\text{-r.e.} \neq \emptyset$ . □

**Theorem 6.**  $\text{LimTxt}_{\text{r.e.}} \subseteq \text{LimTxt}_{\text{r.e.}}[K]$

To show  $\text{LimTxt}_{\text{r.e.}} \subseteq \text{LimTxt}_{\text{r.e.}}[K]$  suppose  $\mathcal{C}$  is a class of r.e. languages in  $\text{LimTxt}_{\text{r.e.}}$ . Let  $M$  be an IIM identifying  $\mathcal{C}$  behaviourally correctly in  $(W_i)_{i \in \mathbb{N}}$ .

The following oracle-IIM  $M'$   $\text{LimTxt}_{\text{r.e.}}$ -identifies  $\mathcal{C}$  with respect to  $(W_i)_{i \in \mathbb{N}}$  using an oracle for  $K$ . Given a text segment  $t_n$  of length  $n + 1$ ,  $M'$  first computes  $M(t_n)$ . If  $n = 0$ , then  $M'(t_n) = M(t_n)$ . If  $n > 0$ , then  $M'$  uses the  $K$ -oracle to determine whether or not there is a word  $w_x$  for some  $x \leq n$ , such that  $w_x \in W_{M(t_n)} \Delta W_{M(t_{n-1})}$ . If no such word exists, then  $M'(t_n) = M'(t_{n-1})$ . Else  $M'(t_n) = M(t_n)$ . It is not hard to prove that  $M'$  learns all languages in  $\mathcal{C}$  in the limit respecting  $(W_i)_{i \in \mathbb{N}}$ . Thus  $\text{LimTxt}_{\text{r.e.}} \subseteq \text{LimTxt}_{\text{r.e.}}[K]$ .

$\text{LimTxt}_{\text{r.e.}}[K] \setminus \text{LimTxt}_{\text{r.e.}} \neq \emptyset$  is witnessed by the class  $\mathcal{C}_R = \{L_f \mid f \text{ is a recursive function}\}$ , where for each partial recursive function  $f$  we define  $L_f = \{a^x b^{f(x)} \mid x \in \mathbb{N}\}$ . (\*  $\mathcal{C}_R$  consists only of recursive languages. \*) If  $\mathcal{C}_R$  was  $\text{LimTxt}_{\text{r.e.}}$ -learnable, then the class of all recursive functions would be  $\text{LimTxt}_{\text{r.e.}}$ -learnable as defined in [5]. The latter contradicts a result in [5]. On the other hand,  $\mathcal{C}_R$  is  $\text{LimTxt}_{\text{r.e.}}$ -learnable: if  $L \in \mathcal{C}_R$  is the target language, a learner can find the least  $i$  with  $L_{\varphi_i} \supseteq L$ . Then  $L_{\varphi_i}$  must equal  $L$ . By Thm. 4, then  $\mathcal{C}_R \in \text{LimTxt}_{\text{r.e.}}[K] \subseteq \text{LimTxt}_{\text{r.e.}}[K]$ .<sup>2</sup> This establishes  $\text{LimTxt}_{\text{r.e.}} \subseteq \text{LimTxt}_{\text{r.e.}}[K]$ .  $\square$

**Theorem 7.**  $\text{LimTxt}_{\text{r.e.}} \not\subseteq \text{LimTxt}_{\text{r.e.}}$

For  $\text{LimTxt}_{\text{r.e.}} \setminus \text{LimTxt}_{\text{r.e.}} \neq \emptyset$  see [15]. The class  $\mathcal{C}_R$  used to prove Thm. 6 witnesses  $\text{LimTxt}_{\text{r.e.}} \setminus \text{LimTxt}_{\text{r.e.}} \neq \emptyset$ .  $\square$

### 4.2 Results for Uniformly $K$ -r.e. Hypothesis Spaces

Finally, we consider  $K$ -r.e. hypothesis spaces for query learning as in [14, 15]. A family  $(A_i)_{i \in \mathbb{N}}$  is  $\text{LimTxt}_{K\text{-r.e.}}$ , if there is a recursive function  $g$  with  $A_i = \{w \in \Sigma^* \mid g(i, w, n) = 1 \text{ for all but finitely many } n\}$  for all  $i \in \mathbb{N}$ . As it turns out, all the equality results from former studies, as illustrated in Figure 1, now turn into proper inclusions. So, though there are strong relations between the corresponding inference types, these are not as strong as in the context of learning indexable classes. Thms. 8 and 9 state this formally.

**Theorem 8.**  $\text{LimTxt}_{\text{r.e.}}[K] \subseteq \text{LimTxt}_{K\text{-r.e.}}$

First, we prove  $\text{LimTxt}_{\text{r.e.}}[K] \subseteq \text{LimTxt}_{K\text{-r.e.}}$ . For that purpose, suppose that  $\mathcal{C}$  is a class of r.e. languages in  $\text{LimTxt}_{\text{r.e.}}[K]$ . Let  $M$  be an oracle-IIM identifying  $\mathcal{C}$  in the limit with respect to  $(W_i)_{i \in \mathbb{N}}$ , using a  $K$ -oracle.

Suppose  $L \in \mathcal{C}$  is the target language. Let  $(V_i)_{i \in \mathbb{N}}$  be a uniformly  $K$ -r.e. family, in which grammars for all queries as posed in the instructions below can be computed (\* such a family exists \*). A  $\text{LimTxt}_{K\text{-r.e.}}$ -learner  $M'$  for  $L$  with respect to  $(V_i)_{i \in \mathbb{N}}$  is defined to act on the following instructions, starting in stage 0. Stage  $n$  reads as follows:

<sup>2</sup>  $\mathcal{C}_R \in \text{LimTxt}_{\text{r.e.}}[K]$  also follows from a result in [1], which proves that the access to an oracle for  $K$  permits to learn the class of all recursive functions in the limit, as defined in [9]. This finally yields  $\text{LimTxt}_{\text{r.e.}}[K]$ -learnability of  $\mathcal{C}_R$ .

- Ask disjointness queries for  $\{w_0\}, \dots, \{w_n\}$ . Let  $L_{[n]}$  be the set of words  $w_x, x \leq n$ , for which the corresponding query is answered with ‘no’.  
 (\* Note that  $L_{[n]} = L \cap \{w_x \mid x \leq n\}$ . \*)
- Let  $(\sigma_x^n)_{x \in \mathbb{N}}$  be an effective enumeration of all finite text segments for  $L_{[n]}$ . For all  $x, y \leq n$  compute  $M(\sigma_x^y)$  (\* note that for these computations a  $K$ -oracle must be simulated \*) as follows: whenever  $M$  wants to access a  $K$ -oracle in order to determine whether  $k \in K$  for some  $k \in \mathbb{N}$ , then pose a disjointness query for the language

$$V'_k = \begin{cases} \Sigma^*, & \text{if } k \in K, \\ \emptyset, & \text{otherwise.} \end{cases}$$

If the answer is ‘yes’, then transmit the answer ‘no’ to  $M$  and vice versa.

- For each  $x, y \leq n$ , pose a disjointness query for the  $K$ -r.e. language  $\overline{W_{M(\sigma_x^y)}}$ . Let  $\text{Cand}_n = \{\sigma_x^y \mid x, y \leq n \text{ and } \overline{W_{M(\sigma_x^y)}} \cap L = \emptyset\}$  be the set of those segments, for which the query has been answered with ‘yes’.  
 (\* Note that  $\text{Cand}_n = \{\sigma_x^y \mid x, y \leq n \text{ and } L \subseteq W_{M(\sigma_x^y)}\}$ . \*)
- For all  $\sigma \in \text{Cand}_n$ , pose a disjointness query for the  $K$ -r.e. language

$$V'_\sigma = \begin{cases} \Sigma^*, & \text{if, given access to a } K\text{-oracle as requested,} \\ & M(\sigma\sigma') \neq M(\sigma) \text{ for some text segment } \sigma' \text{ of } W_{M(\sigma)}, \\ \emptyset, & \text{otherwise.} \end{cases}$$

(\*  $V'_\sigma \cap L = \emptyset$  iff  $\sigma$  is a  $\Sigma^*$ -stabilizing sequence for  $M$  and  $W_{M(\sigma)}$ .) \*

If all these queries are answered ‘no’, then go to stage  $n+1$ . Else, if  $\sigma \in \text{Cand}_n$  is minimal with  $V'_\sigma \cap L = \emptyset$ , then hypothesize a  $j$  with  $V_j = W_{M(\sigma)}$  and stop.

$M'$  identifies  $L$  with disjointness queries in  $(V_i)_{i \in \mathbb{N}}$ ; the proof is omitted. So  $\mathcal{C} \in \mathfrak{V}_{K\text{-r.e.}}$  and  $\mathfrak{V}_{K\text{-r.e.}}[K] \subseteq \mathfrak{V}_{K\text{-r.e.}}$ .

$\mathfrak{V}_{K\text{-r.e.}}[K] \neq \mathfrak{V}_{K\text{-r.e.}}$  can be verified as follows:

We say that an oracle-IIM  $M$  is  $\mathfrak{V}_{K\text{-r.e.}}$ -identified, if for all oracles  $A$  and all languages  $L$ , [if  $M^A$  has a stabilizing sequence on  $L$ , then every text for  $L$  starts with a stabilizing sequence for  $M^A$  on  $L$ ]. Note that from any oracle-IIM  $M$ , one can effectively find an oracle-IIM  $M'$  such that  $M'$  is nice, and for all  $A, \mathfrak{V}_{K\text{-r.e.}}[A]$ -identifies at least as much as  $M$  (a construction in [8] can be seen to easily relativize).

Thus, let  $M_0, M_1, \dots$  be a recursive sequence of nice oracle-IIMs, such that any class in  $\mathfrak{V}_{K\text{-r.e.}}[K]$  is  $\mathfrak{V}_{K\text{-r.e.}}[K]$ -identified by some  $M_i$ .

Let  $X_i = \{a^i b^x \mid x \in \mathbb{N}\}$ . Let  $t^i$  be the canonical text  $a^i b^0, a^i b^1, a^i b^2, \dots$  for  $X_i$ . Let  $X_i^n = \text{content}(t^n_i) = \{a^i b^x \mid x \leq n\}$ .

Define  $L_i$  as follows. If there is no stabilizing sequence for  $M_i^K$  on  $X_i$ , then let  $L_i = X_i$ . Else, let  $t^n_i$  be a stabilizing sequence for  $M_i^K$  on  $X_i$  (where  $n$  is the least non-zero number such that  $t^n_i$  is a stabilizing sequence for  $M_i^K$  on  $X_i$ ). Then if  $W_{M_i^K}(t^n_i) \supset X_i^n$ , then let  $L_i = X_i^n$ ; else let  $L_i = X_i^{n+1}$ .

Let  $\mathcal{C} = \{L_i \mid i \in \mathbb{N}\}$ . (\* Note that  $\mathcal{C}$  consists only of recursive languages. \*)

Note that  $M_i^K$  does not  $\mathfrak{V}_{K\text{-r.e.}}[K]$ -identify  $L_i$ . Thus  $\mathcal{C} \notin \mathfrak{V}_{K\text{-r.e.}}[K]$ .

We now show how to get a  $K$ -r.e. grammar for  $L_i$  from  $i$ . This is clearly enough to verify  $\mathcal{C} \in \mathfrak{V}_{K\text{-r.e.}}$  (as  $i$  can be obtained by asking disjointness queries for  $L_0, L_1, \dots$ , until the unique  $i$  to cause a ‘no’-reply is found).

Now  $a^i b^n \in L_i$  iff: (i)  $n = 0$  or  
 (ii)  $\forall y \leq n$  [ $t_y^i$  is not a stabilizing sequence for  $M_i^K$  on  $X_i$ ] or  
 (iii)  $\forall y < n$  [ $t_y^i$  is not a stabilizing sequence for  $M_i^K$  on  $X_i$ ] and  $W_{M_i^K(t_n^i)} \not\subseteq X_i^n$ .  
 This is a  $K$ -r.e. predicate, hence one can obtain a  $K$ -r.e. grammar for  $L_i$ .

**Theorem 9.**  $\mathcal{C}_{K\text{-r.e.}} \subseteq \mathcal{C}_{K\text{-r.e.}}$

First we show  $\mathcal{C}_{K\text{-r.e.}} \subseteq \mathcal{C}_{K\text{-r.e.}}$ . Let  $\mathcal{C}$  be a class of r.e. languages  $\mathcal{C}_{K\text{-r.e.}}$ -learnable by some  $M$  in a uniformly  $K$ -r.e. hypothesis space  $(V_i)_{i \in \mathbb{N}}$ . Let  $(V'_i)_{i \in \mathbb{N}}$  be a uniformly  $K$ -r.e. family, in which grammars for all superset queries needed below can be computed (\* such a family exists \*). For a target language  $L$ , an IIM  $M'$  is defined to execute stage 0.

Stage  $n$ : Simulate  $M$ . If  $M$  poses a disjointness query for  $V_j$ , determine the set  $\text{Cand}_n$  of all  $w \in \{w_0, \dots, w_n\}$ , for which a superset query concerning

$$V'_w = \begin{cases} \Sigma^*, & \text{if } w \in V_j, \\ \emptyset, & \text{if } w \notin V_j, \end{cases}$$

is answered with ‘yes’. (\*  $\text{Cand}_n = \{w_0, \dots, w_n\} \cap V_j$ . \*)

Then pose a superset query for all languages  $\Sigma^* \setminus \{w\}$  with  $w \in \text{Cand}_n$ . If all the answers are ‘yes’ (\*  $\text{Cand}_n \cap L = \emptyset$  \*), then transmit the answer ‘yes’ to  $M$ , else transmit the answer ‘no’ to  $M$ . (\* ‘no’-answers are always correct. \*)

If  $M$  has not returned a hypothesis within  $n$  steps, then go to stage  $n + 1$ , else, if  $M$  has guessed the language  $V_i$ , pose a superset query representing  $V_i$  in  $(V'_s)_{s \in \mathbb{N}}$ . If the answer is ‘no’, then go to stage  $n + 1$ . If the answer is ‘yes’, then let  $J$  be the set of indices of queries of  $M$ , which have been answered with ‘yes’. For all  $j \in J$  pose a superset query for

$$V'_j = \begin{cases} \Sigma^*, & \text{if } V_j \cap V_i \neq \emptyset, \\ \emptyset, & \text{if } V_j \cap V_i = \emptyset. \end{cases}$$

If all these queries are answered ‘no’, then return a grammar for  $V_i$  in  $(V'_k)_{k \in \mathbb{N}}$  (\* because all queries are then answered correctly for the language  $V_i \supseteq L$  and for  $L$ —so the hypothesis  $V_i$  of  $M$  must be correct for  $L$  \*). Else go to stage  $n + 1$ .

It is not hard to show that  $M'$  learns  $L$  respecting  $(V'_i)_{i \in \mathbb{N}}$ . So  $\mathcal{C} \in \mathcal{C}_{K\text{-r.e.}}$ .

Second, a class in  $\mathcal{C}_{K\text{-r.e.}} \setminus \mathcal{C}_{K\text{-r.e.}}$  can be defined as follows.

Let  $A$  be a  $\Pi_3$ -complete set. Let  $L_i = \{a^i b^{j+1} a^{x+1} \mid j, x \in \mathbb{N}\}$  and  $L_i^s = \{a^i b^{j+1} a^{x+1} \mid j \in \mathbb{N}, x \leq s\}$  for all  $i, s \in \mathbb{N}$ . Finally, let  $\mathcal{C} = \{L_i \mid i \in A\} \cup \{L_i^s \mid i \notin A, s \in \mathbb{N}\}$ . (\* Note that  $\mathcal{C}$  consists only of recursive languages. \*)

We first show  $\mathcal{C} \notin \mathcal{C}_{K\text{-r.e.}}$ . Suppose by way of contradiction that  $M$  witnesses  $\mathcal{C} \in \mathcal{C}_{K\text{-r.e.}}$  in some uniformly  $K$ -r.e. family  $(V_i)_{i \in \mathbb{N}}$ . We establish a contradiction by concluding  $A \in \Sigma_3$ , though  $A$  is  $\Pi_3$ -complete. For that purpose, fix recursive sets  $Q, R$  with  $i \in A$  iff  $\forall x \exists y \forall z [Q(i, x, y, z)]$ ;  $w \in V_i$  iff  $\exists y \forall z [R(i, w, y, z)]$ . Define a  $\Sigma_3$ -procedure  $P$  on input  $i \in \mathbb{N}$  to begin in stage 0.

Stage  $n$ : a) Test whether or not  $\exists y \forall z [Q(i, n, y, z)]$  is true. If not, then stop with the output ‘ $i \notin A$ ’. Else go to b).

b) Simulate  $M$  for  $n + 1$  steps. Thereby, whenever  $M$  poses a disjointness query  $k$ , transmit the answer ‘yes’ to  $M$  in case  $\exists j, x \exists y \forall z [R(k, a^i b^{j+1} a^{x+1}, y, z)]$



is  $i \in A$  true (\* i.e., if  $L_i \cap V_k = \emptyset$  \*); the answer ‘no’, otherwise. In case  $M$  does not return any hypothesis within  $n + 1$  steps of computation, go to stage  $n + 1$ . Else stop with the output ‘ $i \in A$ ’. (\* If  $i \notin A$ , then there would be some  $s$ , such that, in the scenario above, all answers transmitted to  $M$  would be correct for the languages  $L_i^s, L_{i+1}^s, L_{i+2}^s, \dots$ , which would all belong to  $\mathcal{C}$ . Since  $M$  returns a hypothesis,  $M$  would fail for infinitely many languages in  $\mathcal{C}$ —a contradiction. \*)

$P$  decides  $A$  in  $\Sigma_3$ . This contradiction implies that  $\mathcal{C} \notin \text{K-r.e.}$ .

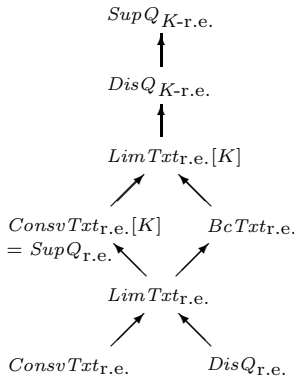
We now show that  $\mathcal{C} \in \text{K-r.e.}$ .

Let  $q_x^i(t) = \min(\{t\} \cup \{r \leq t \mid \forall z \leq t [Q(i, x, r, z)]\})$ . Note that  $q_x^i(t)$  is recursive, and  $\lim_{t \rightarrow \infty} q_x^i(t)$  exists iff  $\exists y \forall z [Q(i, x, y, z)]$  is true. Let

$$f_i(u, t) = \begin{cases} 1, & \text{if } \forall x \leq u [q_x^i(t) = q_x^i(t + 1)], \\ 0, & \text{otherwise.} \end{cases}$$

It is easy to verify that  $X_i = \{w_u \mid \lim_{t \rightarrow \infty} f_i(u, t) = 1\}$ , is finite if  $i \notin A$ , and equal to  $\Sigma^*$ , if  $i \in A$ . Moreover,  $f_i$  is a  $K$ -r.e. function for  $X_i$ . Thus, a  $K$ -r.e. grammar for  $X_i$  can be obtained effectively from  $i$ .

Now define  $M$  as follows. For a target language  $L$ , query  $\Sigma^* \setminus \{a^i b^{j+1} a^{x+1}\}$ , for various values of  $i, j, x$ , until  $i, j, x$  are found such that  $\Sigma^* \setminus \{a^i b^{j+1} a^{x+1}\} \not\supseteq L$ . By definition of  $\mathcal{C}$ , this implies that  $L$  is of the form  $L_i$  or  $L_i^s$  for some  $s \in \mathbb{N}$ . Now, pose a superset query for  $X_i$ . Note that if  $i \in A$ , then  $X_i = \Sigma^* \supseteq L$ , and if  $i \notin A$ , then  $X_i \not\supseteq L$  (as  $X_i$  would then be finite, while  $L$  is infinite). Thus  $M$  can determine whether or not  $i \in A$ . If  $i \in A$ , then  $M$  outputs a grammar for  $L_i$ . If not, then  $M$  searches for the minimal  $s \in \mathbb{N}$  such that  $L_i^s \supseteq L$ . Now  $L = L_i^s$ , and  $M$  can output a grammar for  $L_i^s$ . Thus  $\mathcal{C} \in \text{K-r.e.}$   $\square$



**Fig. 2.** This graph illustrates the relations between different inference types studied above. Arrows indicate proper inclusions of inference types. Two inference types which are not connected by a path of arrows are incomparable.

## 5 Discussion

Above we have seen that many of the equivalences of Gold-style and query inference types in the context of learning indexable classes no longer hold, if arbitrary target classes of r.e. languages are considered. Nevertheless, these two approaches of learning reveal strong relations, expressed in an inclusion hierarchy

of inference types. Altogether, this shows that in many cases all learners of the one kind of inference types can be transformed into learners of the other kind without loss of learning power, though in general not vice versa. Interestingly, our proofs for the inclusions are constructive, i.e. the transformations of learners can be done uniformly and indicate the essential reasons for the strong relations.

Another outcome is that all our separation results are witnessed by classes of recursive languages:  $\mathcal{U}_{\text{r.e.}} \setminus \mathcal{U}_{\text{K-r.e.}} \neq \emptyset$ ,  $\mathcal{U}_{\text{K-r.e.}} \setminus \mathcal{U}_{\text{r.e.}} \neq \emptyset$ , and  $\mathcal{U}_{\text{r.e.}}[K] \setminus \mathcal{U}_{\text{K-r.e.}} \neq \emptyset$  are obtained in [14, 15] using indexable classes of recursive languages; the other separations displayed in Figure 2 have been verified with non-indexable classes of recursive languages. For the latter, note that all classes used in our proofs above consist only of recursive languages. Of course these proofs would not have worked with indexable classes of recursive languages, since the corresponding separations do not hold for indexable classes, see Figure 1. So the equalities are not due to the recursiveness of the target languages alone. The fact that the target classes are indexable is crucial. This raises the question whether the new inequalities obtained above hold for uniformly r.e. classes. As it turns out, at least one of them does not, while some of them do.

When restricting the focus to learning indexable classes, [14] has shown that the capabilities of  $\mathcal{U}_{\text{K-r.e.}}$ -learners and  $\mathcal{U}_{\text{r.e.}}[K]$ -learners are equal, which does not hold for general classes of recursive languages, as witnessed in the proof of Thm. 8. Interestingly, the enumerability of the target class is the crucial reason for the equality result in [14], as the following theorem illustrates.

**Theorem 10.**  $\mathcal{U}_{\text{r.e.}} \subseteq \mathcal{U}_{\text{K-r.e.}}$  and  $\mathcal{U}_{\text{K-r.e.}} \subseteq \mathcal{U}_{\text{r.e.}}[K]$

Proof.  $\mathcal{U}_{\text{r.e.}} \subseteq \mathcal{U}_{\text{K-r.e.}}$  by Thm. 8. So suppose  $\mathcal{C} \in \mathcal{U}_{\text{K-r.e.}}$  is uniformly r.e. Let  $f$  be a recursive function such that  $\mathcal{C} = \{W_{f(i)} \mid i \in \mathbb{N}\}$ . Let  $M$  be a  $\mathcal{U}_{\text{K-r.e.}}$ -learner for  $\mathcal{C}$  in a  $K$ -r.e. hypothesis space  $(V_i)_{i \in \mathbb{N}}$ . Let  $g$  be a recursive function such that  $w \in V_i$  iff  $\lim_{t \rightarrow \infty} g(i, w, t) = 1$ .

The idea is to construct a  $\mathcal{U}_{\text{r.e.}}[K]$ -learner  $M'$  for  $\mathcal{C}$  by simulating  $M$ . Given a text segment  $t_n$ ,  $M'$  searches for a language in  $\mathcal{C}$ , which is consistent with  $t_n$  and for which the behaviour of the known learner  $M$  seems reasonable, at least when taking  $t_n$  into consideration. The length of the given text segment serves as a bound for  $M'$  when trying to analyse whether the behaviour of  $M$  is reasonable. Define  $M'(t_n)$  (using an oracle for  $K$ ) as follows:

- If there exists a  $j \leq n$  such that the following three conditions are satisfied:
  1.  $\text{content}(t_n) \subseteq W_{f(j)}$ .
  2.  $M$  outputs a hypothesis if the questions  $k$  of  $M$  are answered as follows:
    - ‘no’, if there exists a  $w \in \text{content}(t_n)$  with  $g(k, w, n') = 1$  for all  $n' \geq n$ .
    - ‘yes’, otherwise.
  3. For any query  $k$  made by  $M$  in the simulation in 2 above:
    - if there exists a  $w \in W_{f(j),n}$  with  $g(k, w, n') = 1$  for all  $n' \geq n$ , then there also exists a  $w \in \text{content}(t_n)$  with  $g(k, w, n') = 1$  for all  $n' \geq n$ . (\* i.e., the seeming ‘yes’-answers for  $\text{content}(t_n)$  as a target language also seem to be ‘yes’-answers for  $W_{f(j)}$  as a target language \*).
 then output  $f(j)$  for the least such  $j$ . Else output 0.

Note that these simulations can be done using an oracle for  $K$ . We now claim that  $M'$  uniformly  $K$ -r.e.-identifies  $\mathcal{C}$  using a  $K$ -oracle. To see this, suppose  $L \in \mathcal{C}$  and  $t$  is a text for  $L$ . Let  $j$  be minimal with  $W_{f(j)} = L$ . Fix  $n$  large enough such that:

A. If all the questions of  $M$  are answered correctly in the simulation made by  $M'(t_n)$ , then  $M$  outputs a hypothesis.

B.  $j \leq n$ .

C. Let  $Q$  denote the set of questions asked as in A above. Then, for all  $k \in Q$ , if  $V_k \cap L \neq \emptyset$ , then for some  $w \in \text{content}(t_n)$ , for all  $n' \geq n$ ,  $g(k, w, n') = 1$  (\* i.e., answers can be given correctly based on  $t_n$  \*).

D. For all  $j' < j$ , if  $W_{f(j')} \not\supseteq L$ , then  $W_{f(j')} \not\supseteq \text{content}(t_n)$ .

E. For all  $j' < j$ , if  $W_{f(j')} \supset L$ , then for some  $k \in Q$  with  $V_k \cap L = \emptyset$  and  $x \in W_{f(j'),n}$ , for all  $n' \geq n$ ,  $g(k, x, n') = 1$ .

Such an  $n$  exists (for (A–D), clearly; for (E), if for  $j' < j$  no such  $n$  existed, then  $t_n$  could be extended to a text segment for  $W_{f(j)}$  and  $W_{f(j')}$ , with answers as in (A) being correct for both  $W_{f(j)}$  and  $W_{f(j')}$ . So  $M$  would fail to uniformly  $K$ -learn at least one of them). Now,  $M'(t_{n'}) = f(j)$  follows for all  $n' \geq n$ .  $\square$

For the other separations, except for  $\text{uniformly } K\text{-r.e.} \setminus \text{uniformly } K\text{-r.e.} \neq \emptyset$ , we will now prove that even enumerability of the target class is not sufficient for achieving the equality results from [14, 15]. Whether or not a similar result holds for the separation of  $\text{uniformly } K\text{-r.e.}$  and  $\text{uniformly } K\text{-r.e.}$  remains an open question.

**Theorem 11.**  $\text{uniformly } K\text{-r.e.} \setminus \text{uniformly } K\text{-r.e.} \neq \emptyset$ .  
 $\text{uniformly } K\text{-r.e.} \setminus \text{uniformly } K\text{-r.e.} \neq \emptyset$ .  
 $\text{uniformly } K\text{-r.e.} \setminus \text{uniformly } K\text{-r.e.} \neq \emptyset$ .

This is already witnessed by the proof of Thm. 2, where a uniformly r.e. class of recursive languages is used for the separation.

The idea is to use a class comprising the class  $\mathcal{C}_R$  used in the proof of Thm. 6. For that purpose, choose a uniformly r.e. family  $(L_i)_{i \in \mathbb{N}}$  of recursive languages satisfying the following demands:

- for all  $i \in \mathbb{N}$ ,  $L_i$  is either finite or  $L_i \in \mathcal{C}_R$ ,
- for all  $L \in \mathcal{C}_R$  there is some  $i \in \mathbb{N}$  such that  $L = L_i$ ,
- $L_0 = \emptyset$ ,
- for all  $i, j \in \mathbb{N}$  with  $L_i \neq \emptyset$ , if  $L_i \subset L_j$ , then  $i < j$ .

Such a family can be constructed with standard methods. Now, if  $\mathcal{C} = \{L_i \mid i \in \mathbb{N}\}$ , then  $\mathcal{C}$  obviously comprises  $\mathcal{C}_R$  and thus  $\mathcal{C} \notin \text{uniformly } K\text{-r.e.}$ . In contrast to that,  $\mathcal{C} \in \text{uniformly } K\text{-r.e.}$ : for identifying some  $L \in \mathcal{C}$  a query learner can find the least  $i$  such that  $L_i \supseteq L$ . The properties of  $\mathcal{C}$  then imply  $L_i = L$ .

Thus,  $\mathcal{C} \in \text{uniformly } K\text{-r.e.}$  and, by Thm. 5, also  $\mathcal{C} \in \text{uniformly } K\text{-r.e.}[K]$ .  $\square$

Thus we have seen that in several cases the equivalence results for indexable classes from previous work are diminished to strict inclusions, regardless of whether or not the target class (i) consists of recursive languages only, or (ii) is enumerable (the latter with one exception, see Thm. 10). This shows that

indexable target classes yield a specific situation for Gold-style and query learning. Strong relationships between the two models are already witnessed in the general case of learning arbitrary classes of r.e. languages, but only a restriction to even indexable target classes further intensifies these relationships.

**Acknowledgement.** Many thanks are due to the anonymous referees for their helpful comments, especially, for a simplified proof of Theorem 11–(2).

## References

- [1] L. Adleman, M. Blum. Inductive inference and unsolvability. *J. Symbolic Logic*, 56:891–900, 1991.
- [2] D. Angluin. Inductive inference of formal languages from positive data. *Inform. Control*, 45:117–135, 1980.
- [3] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
- [4] D. Angluin. Queries revisited. *Theoret. Comput. Sci.*, 313:175–194, 2004.
- [5] J. Barzdins. Two theorems on the limiting synthesis of functions. *Theory of Algorithms and Programs*, Latvian State University, Riga 210:82–88, 1974.
- [6] J. Case, C. Lynes. Machine inductive inference and language identification. In: *Proc. ICALP 1982*, LNCS 140, pp. 107–115, Springer, 1982.
- [7] D. De Jongh, M. Kanazawa. Angluin’s theorem for indexed families of r.e. sets and applications. In: *Proc. COLT 1996*, pp. 193–204, ACM Press, 1996.
- [8] M. Fulk. Prudence and other conditions for formal language learning. *Inform. Comput.*, 85:1–11, 1990.
- [9] E.M. Gold. Language identification in the limit. *Inform. Control*, 10:447–474, 1967.
- [10] J.E. Hopcroft, J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 1979.
- [11] S. Jain, E. Kinber. Learning languages from positive data and negative counterexamples. In: *Proc. ALT 2004*, LNAI 3244, pp. 54–68, Springer, 2004.
- [12] S. Jain, E. Kinber. Learning languages from positive data and a finite number of queries. In: *Proc. FST&TCS 2004*, LNAI 3328, pp. 360–371, Springer, 2004.
- [13] S. Lange, T. Zeugmann. Language learning in dependence on the space of hypotheses. In: *Proc. COLT 1993*, pp. 127–136, ACM Press, 1993.
- [14] S. Lange, S. Zilles. Replacing limit learners with equally powerful one-shot query learners. In: *Proc. COLT 2004*, LNAI 3120, pp. 129–143, Springer, 2004.
- [15] S. Lange, S. Zilles. Comparison of query learning and Gold-style learning in dependence of the hypothesis space. In: *Proc. ALT 2004*, LNAI 3244, pp. 99–113, Springer, 2004.
- [16] S. Lange, S. Zilles. Relations between Gold-style learning and query learning. Submitted to *Inform. Comput.*
- [17] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*, MIT Press, 1987.
- [18] F. Stephan. *Degrees of Computing and Learning*. Habilitationsschrift, Ruprecht-Karls-Univ., Heidelberg, 1999.
- [19] T. Zeugmann, S. Lange. A guided tour across the boundaries of learning recursive languages. In: *Algorithmic Learning for Knowledge-Based Systems*, LNAI 961, pp. 190–258, Springer, 1995.

# Non U-Shaped Vacillatory and Team Learning

Lorenzo Carlucci<sup>1,\*</sup>, John Case<sup>2,\*\*</sup>, Sanjay Jain<sup>3,\*\*\*</sup>, and Frank Stephan<sup>4,†</sup>

<sup>1</sup> Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716-2586, USA and Dipartimento di Matematica, Università di Siena, Pian dei Mantellini 44, Siena, Italy

`carlucci5@unisi.it`

<sup>2</sup> Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716-2586, USA

`case@cis.udel.edu`

<sup>3</sup> School of Computing, 3 Science Drive 2, National University of Singapore, Singapore 117543

`sanjay@comp.nus.edu.sg`

<sup>4</sup> School of Computing and Department of Mathematics, National University of Singapore, 3 Science Drive 2, Singapore 117543

`fstephan@comp.nus.edu.sg`

**Abstract.** *U-shaped learning behaviour* in cognitive development involves learning, unlearning and relearning. It occurs, for example, in learning irregular verbs. The prior cognitive science literature is occupied with how humans do it, for example, general rules versus tables of exceptions. This paper is mostly concerned with whether U-shaped learning behaviour may be *necessary* in the abstract mathematical setting of inductive inference, that is, in the computational learning theory following the framework of Gold. All notions considered are learning from text, that is, from positive data. Previous work showed that U-shaped learning behaviour is necessary for behaviourally correct learning but not for syntactically convergent, learning in the limit (= explanatory learning). The present paper establishes the necessity for the whole hierarchy of classes of vacillatory learning where a behaviourally correct learner has to satisfy the additional constraint that it vacillates in the limit between at most  $k$  grammars, where  $k \geq 1$ . Non U-shaped vacillatory learning is shown to be restrictive: Every non U-shaped vacillatorily learnable class is already learnable in the limit. Furthermore, if vacillatory learning with the parameter  $k = 2$  is possible then non U-shaped behaviourally correct learning is also possible. But for  $k = 3$ , surprisingly, there is a class witnessing that this implication fails.

## 1 Introduction and Motivation

*U-shaped learning behaviour* is a learning behaviour in which the learner first learns the correct behaviour, then abandons the correct behaviour and finally returns to

---

\* Supported in part by NSF grant number NSF CCR-0208616.

\*\* Supported in part by NSF grant number NSF CCR-0208616.

\*\*\* Supported in part by NUS grant number R252-000-127-112.

† Supported in part by NUS grant number R252-000-212-112.

the correct behaviour once again. This pattern of learning behaviour has been observed by cognitive and developmental psychologists in a variety of child development phenomena, such as language learning [6, 15, 22], understanding of temperature [22], understanding of weight conservation [5, 22], object permanence [5, 22] and face recognition [7].

The case of language acquisition is paradigmatic. In the case of the past tense of english verbs, it has been observed that children learn correct syntactic forms (call/called, go/went), then undergo a period of overregularization in which they attach regular verb endings such as ‘ed’ to the present tense forms even in the case of irregular verbs (break/breaked, speak/spoke) and finally reach a final phase in which they correctly handle both regular and irregular verbs. The irregular verb examples of U-shaped learning behaviour has figured so prominently in the so-called “Past Tense Debate” in cognitive science that competing models of human learning are often judged on their capacity for modeling the U-shaped learning phenomenon [15, 19, 23].

The prior literature is typically concerned with modeling how humans achieve U-shaped behaviour. Recently, Baliga, Case, Merkle, Stephan and Wiehagen [1] looked at abstract mathematical models which give some indication how humans exhibit this seemingly inefficient behaviour. Is it a mere harmless evolutionary accident or is it necessary for full human learning power? Specifically, are there some learning tasks for which U-shaped behaviour is logically necessary? In the present paper we continue this line of work.

In order to explain our results, we have to be a bit more formal. Although we refer to Section 2 below for an explanation of the mathematical terms used, we summarize for the reader’s convenience the basics of inductive inference, that is, Gold’s formal model of language learning from positive data [13].

The learning task is given by a subclass  $\mathcal{C}$  of the class of all recursively enumerable (r.e.) subsets of the natural numbers which are indexed by natural numbers in an acceptable way [16, Section II.5]. The learner is then required to learn all the languages in the class  $\mathcal{C}$ . Here, a learner  $\mathbf{M}$  learns a language  $L$  if it produces, in parallel to reading a text for  $L$  (that is, an infinite sequence of all elements of  $L$  in arbitrary order), a sequence  $e_0, e_1, \dots$  of hypotheses such that almost all of these hypotheses are the same index/grammar of the set  $L$ . This criterion is called **TextEx** and stands for “explanatory learning from text” (see [13]).

The above criterion has been relaxed to **TextBc** [9, 17] where it is only required that almost all  $e_n$  are grammars for  $L$ , but each  $e_n$  can be different from all previous ones. This criterion **TextBc** is more general than **TextEx** since languages have infinitely many grammars and the equality problem of the grammars is undecidable.

**TextFex<sub>b</sub>** [8] is the intermediate criterion where the learner succeeds iff there is an  $n$  such that  $\{e_n, e_{n+1}, \dots\}$  is actually a finite set of up to  $b$  correct grammars; the learner is then said to be  $b$ -close between these grammars. The criteria **TextFex<sub>1</sub>**, **TextFex<sub>2</sub>**,  $\dots$ , **TextFex<sub>\*</sub>** form a proper hierarchy between **TextEx** and **TextBc**.

Within this paper we continue the investigation of these standard criteria with the requirement that the learner is *U-shaped*, (which would require that  $e_{n+1}$  generates the language to be learnt whenever  $e_n$  does).

Baliga, Case, Merkle, Stephan and Wiehagen [1] initiated the Gold style learning theoretic study of U-shaped learning behaviour and showed that it is circumventable for **TxtEx**-learning, see Theorem 5. In contrast to this, Fulk, Jain and Osherson's proof of [12, Theorem 4] shows that U-shaped learning behaviour is necessary for the full learning power of **TxtBc**-learning. We show in Theorem 7 below that U-shaped learning behaviour is also necessary for full learning power for the whole hierarchy of the learning criteria **TxtFex<sub>b</sub>** between **TxtEx** and **TxtBc**. While Case [8] proved that the **TxtFex<sub>b</sub>** criteria form a hierarchy of more and more powerful learning criteria, Theorem 7 of the present paper shows that **TxtFex<sub>b</sub>**-learners are not more powerful than **TxtEx**-learners. In other words, there are classes of languages that can be **TxtFex<sub>n</sub>**-identified, for  $n > 1$ , but these learners can be U-shaped on some texts.

What if we consider the more liberal criterion **TxtBc**? Our Theorem 18 strengthens the collapse result of Theorem 7 considerably by showing that there are classes in **TxtFex<sub>3</sub>** that cannot be **TxtBc**-learned by a non U-shaped learner. This means that U-shaped learning behaviour cannot be dispensed with for learning such classes, even if we only require behavioural convergence and permit convergence to possibly infinitely many syntactically different correct hypotheses. By contrast, one of our main results, Theorem 17, shows that every class of languages that can be **TxtFex<sub>2</sub>**-identified can be **TxtBc**-identified by a non U-shaped learner. Hence, for only this early stage of the hierarchy, the cases in which **TxtFex<sub>2</sub>**-identification necessitates U-shaped learning behaviour can be circumvented by shifting to **TxtBc**-identification.

A further interesting aspect is that this paper gives a close relation between vacillatory learning and team learning. Theorem 2 gives the basic connection: A class is **TxtFex<sub>b</sub>**-learnable iff there is a team of  $b$  learners where all team-members converge on every text of a language to be learned and at least one of the team-members has to be correct. Furthermore, in Sections 3 and 5 some general inclusions for non U-shaped team learning are established.

We note that the relevance of Gold style learning to cognitive science has been supported in the cognitive science literature, for example, in [14, 18]. The publications [8, 24] critically discuss the relevance of the well studied criterion **TxtBc** to human learning; in order to avoid a mere interpolation of the data, one might want that a learner does not generate larger and larger hypotheses but tries to concentrate on finding a few correct ones. When **TxtEx**-learning is impossible, this can only be done by vacillating between some few hypotheses. Case [8] formalized this approach by introducing the criteria **TxtFex<sub>b</sub>** for small, feasibly sized  $b > 1$ . Case also argues that these criteria better fit the human case than the **TxtEx** criterion. Certainly, then, the new results of the present paper, regarding, for example, the **TxtFex<sub>3</sub>** criterion are of interest for cognitive science, and they inform regarding the human case.

## 2 Preliminaries

$\mathbb{N}$  denotes the set of natural numbers,  $\{0,1,2,\dots\}$ .  $\text{card}(D)$  denotes the cardinality of a set  $D$ .  $\text{card}(D) \leq *$  means that  $\text{card}(D)$  is finite. The symbol  $*$  is used to denote the “finite with no preassigned bound”. The symbols  $\subseteq, \subset, \supseteq, \supset$  respectively denote the subset, proper subset, superset and proper superset relation between sets. The quantifiers  $\forall^\infty$  and  $\exists^\infty$  mean “for all but finitely many” and “there exists infinitely many”, respectively.

A pair  $\langle \cdot, \cdot \rangle$  stands for an arbitrary, computable one-to-one encoding of all pairs of natural numbers onto  $\mathbb{N}$  [20]. Similarly we can define  $\langle \cdot, \dots, \cdot \rangle$  for encoding  $n$ -tuples of natural numbers, for  $n > 1$ , onto  $\mathbb{N}$ .

$\varphi$  denotes a fixed  $\dots$  programming system for the partial-recursive functions [20].  $\varphi_e$  denotes the partial-recursive function computed by the program with code number  $e$  in the  $\varphi$ -system. We will unambiguously refer to programs using their code number in the  $\varphi$ -system.  $W_e$  denotes domain of  $\varphi_e$ .  $W_{e,s}$  denotes  $W_e$  enumerated within  $s$  computation stages [4]. For our purposes, we need  $W_{e,s}$  to satisfy the following additional constraints, which can be easily ensured using standard techniques: (a)  $W_{e,s} \subseteq \{0, \dots, s-1\}$ , (b)  $\{(x, s) : x \in W_{e,s}\}$  is primitive recursive for all  $e$  and (c) for every primitive-recursive enumeration  $A_s$  of some set  $A$  with  $A_0 = \emptyset \wedge (\forall s) [A_s \subseteq A_{s+1} \subseteq \{0, \dots, s\}]$  there is an index  $e$  with  $(\forall s) [W_{e,s} = A_s]$ ; furthermore,  $e$  can be computed from an index of the enumeration for  $A_s$ . Any unexplained recursion-theoretic notions are from [16, 20].

We now introduce the basic definitions of Gold-style computational learning theory.

A  $\dots$   $\sigma$  is a mapping from an initial segment of  $\mathbb{N}$  into  $\mathbb{N} \cup \{\#\}$ . An  $\dots$  is a mapping from  $\mathbb{N}$  into  $\mathbb{N} \cup \{\#\}$ . The content of a finite or infinite sequence  $\sigma$  is the set of natural numbers occurring in  $\sigma$  and is denoted by  $\text{content}(\sigma)$ . The length of a sequence  $\sigma$  is the number of elements in the domain of  $\sigma$  and is denoted by  $|\sigma|$ . For a subset  $L$  of  $\mathbb{N}$ ,  $\text{seg}(L)$  denotes the set of sequences  $\sigma$  with  $\text{content}(\sigma) \subseteq L$ . An infinite sequence  $T$  is a  $\dots$   $L$  iff  $L = \text{content}(T)$ .

Intuitively, a  $\dots$  for a language  $L$  is an infinite stream or sequential presentation of  $\dots$  the elements of the language  $L$  in any order with the  $\#$ 's representing pauses in the presentation of the data. For example, the only text for the empty language is an infinite sequence of  $\#$ 's. Furthermore,  $T[n]$  denotes the first  $n$  elements of a text  $T : \mathbb{N} \rightarrow \mathbb{N} \cup \{\#\}$ .

A learner will map sequences from  $(\mathbb{N} \cup \{\#\})^*$  to hypotheses. These are represented by natural numbers and interpreted as codes for programs in the  $\varphi$ -system.  $\mathbf{M}$ , with possible superscripts and subscripts, is intended to range over language learning machines.

**Definition 1.** [1, 2, 8, 9, 10, 13, 17] A  $\dots$   $\mathbf{M}$  is a computable mapping from  $\text{seg}(\mathbb{N})$  into  $\mathbb{N}$ .  $\mathbf{M}$  **TextBc**-learns a class  $\mathcal{L}$  of r.e. languages iff for every  $L \in \mathcal{L}$  and every text  $T$  for  $L$ , almost all hypotheses  $\mathbf{M}(T[n])$  are indices for the language  $L$  to be learned.



A **TxtBc**-learner  $\mathbf{M}$  for  $\mathcal{L}$  is a **TxtFex** $_*$ -learner iff for every  $L \in \mathcal{L}$  and every text  $T$  for  $L$  the set  $\{M(T[n]) : n \in \mathbb{N}\}$  is finite.

A **TxtFex** $_*$ -learner  $\mathbf{M}$  for  $\mathcal{L}$  is a **TxtFex** $_b$ -learner for a  $b \in \{1, 2, \dots\}$  iff there are for every  $L \in \mathcal{L}$  and every text  $T$  for  $L$  at most  $b$  indices which  $\mathbf{M}$  outputs infinitely often, that is,  $|\{e : (\exists^\infty n)[e = \mathbf{M}(T[n])]\}| \leq b$ .

A **TxtBc**-learner  $\mathbf{M}$  for  $\mathcal{L}$  is a **TxtEx**-learner iff for every  $L \in \mathcal{L}$  and every text  $T$  for  $L$  almost all hypotheses  $\mathbf{M}(T[n])$  are the same grammar for  $L$ .

A **TxtBc**-learner  $\mathbf{M}$  for  $\mathcal{L}$  is non U-shaped iff for every  $L \in \mathcal{L}$  and every text  $T$  for  $L$  there are no three numbers  $k, m, n$  such that  $k < m < n$  and  $W_{\mathbf{M}(T[k])} = L$ ,  $W_{\mathbf{M}(T[m])} \neq L$  and  $W_{\mathbf{M}(T[n])} = L$ . Furthermore, **NUShTxtBc**-learners, **NUShTxtFex** $_b$ -learners and **NUShTxtEx**-learners (for a class  $\mathcal{L}$ ) are those learners which are non U-shaped and at the same time a **TxtBc**-learner, **TxtFex** $_b$ -learner and **TxtEx**-learner, respectively (for  $\mathcal{L}$ ).

The criteria **TxtBc**, **TxtFex** $_b$ , **TxtEx**, **NUShTxtBc**, **NUShTxtFex** $_b$ , **NUShTxtEx** are the sets consisting of all those classes which are learnable by a learner satisfying the respective above defined requirements.

The historically most important learning criterion is **TxtEx** where the learner has to converge syntactically to a single index of the language to be learned [13]. **TxtEx** stands for “explanatory identification from text”. Intuitively, the notion **TxtBc** captures what could be called learning in the most general sense, where “Bc” stands for “behaviourally correct” identification. In this, the learner outputs correct grammars almost always. A class  $\mathcal{L}$  of r.e. languages is **TxtFex** $_b$  identified by a machine  $\mathbf{M}$  iff when  $\mathbf{M}$  is given as input  $\langle \cdot, \cdot \rangle$ , listing  $T$  of  $\cdot$ ,  $L \in \mathcal{L}$ , it outputs a sequence of grammars such that, past some point in this sequence, no more than  $b$  different grammars occur and each of them is a grammar for  $L$ . **TxtFex** stands for ‘finite explanatory identification from text’. **TxtFex** $_1$  is equivalent to **TxtEx**. Osherson and Weinstein [17] first studied the case with  $b = *$ , later Case [8] studied the whole hierarchy with  $b \in \mathbb{N}^+$ .

We say  $\sigma$  is a **TxtEx**-locking-sequence [11] for a learner  $\mathbf{M}$  on a set  $L$  iff  $\sigma \in \text{seg}(L)$  and  $\mathbf{M}(\sigma\tau) = \mathbf{M}(\sigma)$  for all  $\tau \in \text{seg}(L)$ . Furthermore a **TxtEx**-stabilizing-sequence  $\sigma$  is called a **TxtEx**-locking-sequence [3] for  $\mathbf{M}$  on  $L$  iff  $W_{\mathbf{M}(\sigma)} = L$ . Note that stabilizing and locking sequence definitions can be generalized to other learning criteria such as **TxtFex** and **TxtBc**; we often drop “**TxtEx**” (respectively, “**TxtBc**”, “**TxtFex**”) from “**TxtEx**-stabilizing-sequence” and “**TxtEx**-locking-sequence”, when it is clear from context.

Smith [21] studied learning by teams of machines, and, we can show vacillatory learning can be characterized by teams as follows.

**Theorem 2.** Let  $\mathcal{L}$  be a class of r.e. languages. Then  $\mathcal{L} \in \text{TxFex}_b$  iff there exist  $b$  machines  $\mathbf{M}_1, \dots, \mathbf{M}_b$  such that for every  $L \in \mathcal{L}$  and every text  $T$  for  $L$  there are indices  $e_1, \dots, e_b$  such that  $e_i \in L$  and  $\mathbf{M}_i(T) = e_i$  for all  $i \in \{1, \dots, b\}$ .

Case [8] showed **TxtFex** $_1 \subset \text{TxFex}_2 \subset \dots \subset \text{TxFex}_*$ , as stated in the following Theorem.

**Theorem 3.** [8]  $\neg_i$ .  $b \in \{1, 2, \dots\}$   $\mathcal{H}_b = \{W_e : e \in W_e \wedge |W_e \cap \{0, \dots, e\}| \leq b + 1\}$ ,  $\mathbf{TxtFex}_{b+1} - \mathbf{TxtFex}_b \neg_i$   $\mathcal{H}_* = \{W_e : W_e \neq \emptyset \wedge e \leq \min(W_e)\}$ ,  $\mathbf{TxtFex}_* - \bigcup_{b \in \{1, 2, \dots\}} \mathbf{TxtFex}_b$

**Proposition 4.**  $\mathbf{NUShTxtBc} \not\subseteq \mathbf{TxtFex}_*$

The next theorem is from [1] and states that being non U-shaped is not restrictive for **TxtEx**-learning. Its proof can also be obtained by letting  $b = 1$  in Theorem 12 below. Its fundamental equality will be extended to all classes  $\mathbf{NUShTxtFex}_b$  in Theorem 7.

**Theorem 5.** [1]  $\mathbf{NUShTxtEx} = \mathbf{TxtEx}$

Hence, for **TxtEx**, U-shaped behaviour is not necessary for full learning power. By contrast, an easy adaptation of the proof of Theorem 4 in [12] shows that, for **TxtBc**, U-shaped behaviour is necessary for full learning power.

**Theorem 6.** [1, 12]  $\mathbf{NUShTxtBc} \subseteq \mathbf{TxtBc}$

### 3 Non U-Shaped Vacillatory Learning

We show that the  $\mathbf{TxtFex}_b$ -hierarchy collapses if U-shaped behaviour is forbidden.

**Theorem 7.**  $\mathbf{NUShTxtFex}_* \subseteq \mathbf{TxtFex}_1$

**Proof.** Let  $\mathcal{L} \in \mathbf{NUShTxtFex}_*$  and let  $\mathbf{M}$  be a learner witnessing this fact. We define a new learner  $\mathbf{N}$  witnessing that  $\mathcal{L} \in \mathbf{TxtFex}_1$  as follows.

On a text  $T$ ,  $\mathbf{N}$  keeps a list of all of  $\mathbf{M}$ 's conjectures in order of appearance and without repetitions and outputs the most recent entry in the list.

If  $T$  is a text for a language  $L \in \mathcal{L}$ , then  $\mathbf{M}$  outputs on  $T$  only finitely many different hypotheses and at least one of them, say  $e$ , infinitely often, and  $W_e = L$ . Furthermore,  $\mathbf{N}$  converges to that hypothesis  $e'$  which goes into the list last. If  $e = e'$  then  $\mathbf{N}$  is correct on  $T$ . If  $e \neq e'$  then  $\mathbf{M}$  has output  $e'$  the first time after having already output  $e$  at least once. Since  $\mathbf{M}$  is not U-shaped and  $e$  is correct, so is  $e'$ . Thus  $\mathbf{N}$  is correct on  $T$  again. ■

Theorems 5 and 7 give  $\mathbf{NUShTxtFex}_1 = \mathbf{TxtFex}_1$  and  $\mathbf{NUShTxtFex}_* \subseteq \mathbf{NUShTxtFex}_1$ . Furthermore, the definition of  $\mathbf{NUShTxtFex}_b$  immediately gives  $\mathbf{NUShTxtFex}_1 \subseteq \mathbf{NUShTxtFex}_b \subseteq \mathbf{NUShTxtFex}_*$ . Thus all these criteria coincide.

**Corollary 8.**  $(\forall b \in \{1, 2, \dots, *\}) [\mathbf{NUShTxtFex}_b = \mathbf{TxtFex}_1]$

The result  $\mathbf{NUShTxtFex}_1 = \mathbf{TxtFex}_1$  stands in contrast to the fact that  $\mathbf{TxtFex}_1 \subset \mathbf{TxtFex}_2 \subset \dots \subset \mathbf{TxtFex}_*$ . Thus we have that the following inclusions are proper.

**Corollary 9.**  $(\forall b \in \{2, 3, \dots, *\}) [\mathbf{NUShTxtFex}_b \subset \mathbf{TxtFex}_b]$

Corollaries 8 and 9 show that U-shaped learning behaviour is forbidden for the full learning power of  $\mathbf{TxtFex}_b$ -identification for  $b > 1$  in a strong sense: if U-shaped learning behaviour is forbidden, the hierarchy collapses to  $\mathbf{TxtFex}_1$ . Hence,  $\mathbf{TxtFex}_*$ -learnability of any class in  $(\mathbf{TxtFex}_* - \mathbf{TxtFex}_1)$  implies U-shaped learning behaviour.

**Corollary 10.**  $b \in \{1, 2, \dots, *\}$   $\mathcal{H}_b \in \mathbf{TxtFex}_b$  iff  $\mathcal{H}_b \in \mathbf{TxtFex}_{b+1}$ .

A non U-shaped learner does not make a mind change from a correct to an incorrect hypothesis since it cannot learn the set otherwise. This property is enforced on all machines for the case of team learning.

**Definition 11.** A team learning a class  $\mathcal{L}$  is  $[a, b]$ - $\mathbf{NUShTxtEx}$ , iff no machine in the team on any text for any language  $L \in \mathcal{L}$  ever makes a mind change from an index for  $L$  to an index for a different language. In particular, the class  $\mathcal{L}$  is in  $[a, b]$ - $\mathbf{NUShTxtEx}$  iff there are  $b$  machines such that on any text for any language in  $\mathcal{L}$  at least  $a$  machines converge to an index for that language and no machine makes a mind change from a correct to an incorrect hypothesis. For any learning criterion  $I$ ,  $[a, b]I$  is the corresponding team variant of this criterion.

The next result shows that Theorem 2 can be extended such that every class in  $\mathbf{TxtFex}_b$  is learnable by a non U-shaped team. So the restriction  $\mathbf{NUShTxtFex}_b = \mathbf{TxtFex}_1$  is caused by the fact that the hypothesis of the learner have to be brought into an ordering and cannot be done in parallel as in the case of the team below. Actually Theorem 12 enables us to achieve more properties of the team than that it is just non U-shaped.

**Theorem 12.**  $b \in \mathbb{N}^+$   $\mathcal{L} \in \mathbf{TxtFex}_b$  iff there are machines  $\mathbf{M}_1, \dots, \mathbf{M}_b$  such that for every  $L \in \mathcal{L}$  there is a text  $T \in L$  and  $n \in \mathbb{N}$  such that

- (1)  $T[n] \in L$  and  $\mathbf{M}_a(T[m]) = \mathbf{M}_a(T[n])$  for all  $m \geq n$ .
- (2)  $a \in \{1, \dots, b\}$  and  $\mathbf{M}_a(T[n]) \in L$ .
- (3)  $a \in \{1, \dots, b\}$  and  $\mathbf{M}_a(T[m]) \in L$  for all  $m \geq n$ .

iff  $\mathcal{L} \in [1, b]$ - $\mathbf{NUShTxtEx}$ .

**Proof.** By Theorem 2 there is a team  $\mathbf{N}_1, \dots, \mathbf{N}_b$  of  $\mathbf{TxtEx}$ -learners for  $\mathcal{L}$  such that for every  $L \in \mathcal{L}$  and every text  $T$  for  $L$ , every machine converges on  $T$  to some hypothesis and at least one of these hypotheses is an index for  $L$ .

The basic idea of the proof is to search for a  $\sigma \in \text{seg}(L)$ , which is a  $\mathbf{TxtEx}$ -stabilizing-sequence for each member of the team  $\mathbf{N}_1, \dots, \mathbf{N}_b$  on  $L$ . Additionally, we will also find a maximal set  $D \subseteq \{1, \dots, b\}$  such that  $\sigma$  is a stabilizing sequence for each  $\mathbf{N}_i$ ,  $i \in \{1, \dots, b\}$  on  $W_{\mathbf{N}_j(\sigma)}$ ,  $j \in D$ . Before such  $\sigma, D$ , is obtained, we will make sure that the output of  $\mathbf{M}_a$  below is not a grammar for

*L.* Once such  $\sigma, D$  is obtained, we will have that the learners  $\mathbf{M}_a$  do not change their hypothesis and one of them correctly outputs a grammar for  $L$ . We now proceed formally.

Let  $E$  to be an infinite recursive set such that  $E \cup \tilde{E} \notin \mathcal{L}$  for all finite sets  $\tilde{E}$ . Such an  $E$  can be defined as follows. Let  $\mathbf{M}$  be a  $\mathbf{TxtFex}_b$ -learner for  $\mathcal{L}$ . If  $\mathbb{N} \notin \mathcal{L}$ , then let  $E = \mathbb{N}$ . If  $\mathbb{N} \in \mathcal{L}$ , then there exists a  $\mathbf{TxtFex}_b$ -locking sequence  $\sigma'$  for  $\mathbf{M}$  on  $\mathbb{N}$ . Now we can take  $E$  to be any infinite and coinfinite recursive set such that  $\text{content}(\sigma') \subseteq E$ .

Let  $\sigma \sqsubseteq \tau$  denote that  $\text{content}(\sigma) \subseteq \text{content}(\tau)$  and  $|\sigma| \leq |\tau|$ . Furthermore, let  $T_e$  be the canonical text for  $W_e$ , that is,  $T_e$  is the text generated by some standard enumeration of  $W_e$ .

As long as the content of the input is  $\emptyset$  or no  $\sigma$  is found in the algorithm below, all machines  $\mathbf{M}_1, \dots, \mathbf{M}_b$  output the least index of  $\emptyset$ . The  $\sigma$  searched for on input  $\tau = T[t]$  has to satisfy the following conditions:

- (a)  $\sigma \sqsubseteq T[t]$  and  $\text{content}(\sigma) \neq \emptyset$ ;
- (b)  $\mathbf{N}_a(\sigma\eta) = \mathbf{N}_a(\sigma)$  for all  $a \in \{1, \dots, b\}$  and  $\eta \sqsubseteq T[t]$ .

Once having  $\sigma$ , this is only replaced by a  $\sigma'$  on a future input  $T[t']$  iff  $\sigma'$  but not  $\sigma$  satisfies (a) and (b) with respect to  $T[t']$  (if there are several choices to replace  $\sigma$ , the first one with respect to some fixed recursive enumeration of  $\text{seg}(\mathbb{N})$  is taken). Having  $\sigma$ , define  $D$  as follows.

- (c)  $D = \{a \in \{1, \dots, b\} : (\forall \eta \sqsubseteq T_{\mathbf{N}_a(\sigma)}[t]) (\forall c \in \{1, \dots, b\}) [\mathbf{N}_c(\sigma\eta) = \mathbf{N}_c(\sigma)]\}$ .

Having  $\sigma$  and  $D$ ,  $\mathbf{M}_a(\tau) = F(\sigma, D, a)$  where  $W_{F(\sigma, D, a)}$  is the set of all  $x$  for which there is an  $s$  such that the conditions (d) and either (e) or (f) below hold.

- (d)  $a \in D$  and  $\sigma \sqsubseteq \text{content}(T_{\mathbf{N}_a(\sigma)}[s])$ ;
- (e)  $x \in \text{content}(T_{\mathbf{N}_a(\sigma)}[s])$  and  $\mathbf{N}_c(\sigma\eta) = \mathbf{N}_c(\sigma)$  for all  $c \in \{1, \dots, b\}$ ,  $d \in D$  and  $\eta \sqsubseteq T_{\mathbf{N}_d(\sigma)}[s]$ ;
- (f)  $x \in E$  and  $\mathbf{N}_c(\sigma\eta) \neq \mathbf{N}_c(\sigma)$  for some  $c \in \{1, \dots, b\}$ ,  $d \in D$  and  $\eta \sqsubseteq T_{\mathbf{N}_d(\sigma)}[s]$ .

It is easy to see that the sets  $W_{F(\sigma, D, a)}$  are uniformly recursively enumerable and thus the specified function  $F$  can be taken to be recursive. Thus also the learners  $\mathbf{M}_1, \dots, \mathbf{M}_b$  are recursive. Verification of properties (1), (2) and (3) is omitted. ■

The next result gives a further inclusion between vacillatory learning and team-learning.

**Theorem 13.**  $\mathbf{TxtFex}_b \subseteq [2, b + 1]\mathbf{NUShTxtEx}_{,1, \dots, b}$ ,  $b \in \{1, 2, 3, \dots\}$   
 $\mathbf{TxtFex}_b \subset [1, b]\mathbf{NUShTxtEx}_{,1, \dots, b}$ ,  $b \in \{2, 3, \dots\}$

As  $\mathbf{TxtFex}_2$ -learning is more general than  $\mathbf{TxtEx}$ -learning, one gets the following corollary.

**Corollary 14.**  $[2, 3]\mathbf{NUShTxtEx} \not\subseteq \mathbf{NUShTxtEx}$

### 4 Vacillatory Versus Behaviourally Correct Learning

As every **NUShTxtFex<sub>b</sub>**-learner can be turned into a **NUShTxtEx**-learner identifying the same class, the restriction to U-shaped learning without loss of learning power is only possible in the least class **TxtFex<sub>1</sub>** of the **TxtFex<sub>b</sub>** hierarchy. But, the next, quite surprising result shows that in the case of **TxtFex<sub>2</sub>** one can avoid U-shaped learning behaviour if one gives up the constraint that the learner has to vacillate between many indices. That is, **TxtFex<sub>2</sub>**  $\subseteq$  **NUShTxtBc**. In Theorem 16 it is shown that there is a uniform learner **U** which is given a set  $E$  of up to 2 indices and **NUShTxtBc**-identifies every  $\{W_e : e \in E\}$  such that every hypothesis is a subset of an  $W_e$  with  $e \in E$ . Then this result is combined with Theorem 12 to show the inclusion **TxtFex<sub>2</sub>**  $\subseteq$  **NUShTxtBc**. But before turning to Theorem 16, the following auxiliary proposition gives a method to enforce that the sets  $W_{i'}, W_{j'}$  mentioned there are represented by corresponding approximations  $W_i, W_j$  which differ from one another at all stages of their enumerations.

**Proposition 15.** Given  $F = \{i', j'\}$  and  $G(F) = \{i, j\}$

$$\begin{aligned} & W_{i,s} \cup W_{j,s} = \emptyset, \quad W_{i,s} \neq W_{j,s}. \\ & W_i \subseteq W_{i'} \quad \text{and} \quad W_{i'} \subseteq W_i \quad \text{and} \quad W_i = W_{i'}. \\ & W_j \subseteq W_{j'} \quad \text{and} \quad W_{j'} \subseteq W_j \quad \text{and} \quad W_j = W_{j'}. \\ & \{W_{i'}, W_{j'}\} \subseteq \{W_i, W_j\} \end{aligned}$$

$$j' = i' \quad \text{and} \quad F = \{i'\}$$

**Theorem 16.**  $\forall F, L, H, |F| \leq 2 \quad \mathbf{U} \text{ NUShTxtBc, } \{L, H\}$

- (1)  $W_{\mathbf{U}(F,\sigma)} \subseteq L, \quad W_{\mathbf{U}(F,\sigma)} \subseteq H.$
- (2)  $L = H, \quad W_{\mathbf{U}(F,\sigma)} \in \{\emptyset, L\}$

$$L = H, \quad W_{\mathbf{U}(F,\sigma)} \in \{\emptyset, L\}$$

Given  $F$ , let  $G(F)$  be as in Proposition 15. Figure 1 gives the algorithm witnessing this inclusion. We omit the proof that it works.

**Theorem 17.** **TxtFex<sub>2</sub>**  $\subseteq$  **NUShTxtBc**

**Proof.** Given a **TxtFex<sub>2</sub>**-learnable class  $\mathcal{L}$ , there is by Theorem 2 a pair of two learners  $\mathbf{N}_1, \mathbf{N}_2$  which converge on every language from  $\mathcal{L}$  and  $[1, 2]$ **TxtEx**-identify  $\mathcal{L}$ . Obtain from these two learners the team  $\mathbf{M}_1, \mathbf{M}_2$  as done in the proof of Theorem 12. Let  $F$  be as defined in the proof of Theorem 12. Let  $\sigma_\tau, D_\tau$  be the values of  $\sigma, D$  computed on input  $\tau$  by the algorithm for  $\mathbf{M}_1, \mathbf{M}_2$  in the proof of Theorem 12. Note that one can, for each input  $\sigma_\tau, D_\tau$ , check in the limit whether  $W_{F(\sigma_\tau, D_\tau, a)}$  enumerates some elements using part (f) of the algorithm in the proof of Theorem 12. Let **U** be as in Theorem 16.

**Uniform non U-shaped Behaviourally Correct Learner U**Parameter:  $F$ . Input:  $\sigma$ . Output:  $k$ , specified implicitly.Algorithm to enumerate  $W_k = \bigcup_r W_{k,r}$ .**(Start)** Let  $u = |\sigma|$ ,  $C = \text{content}(\sigma)$  and  $s = 0$ .Let  $W_{k,t} = \emptyset$  for all  $t < |\sigma|$ .If  $C = \emptyset$  or  $W_{e,u} = \emptyset$  for all  $e \in G(F)$ , Then go to (Empty).Let  $\tau = \sigma[|\sigma| - 1]$ .Select  $i, j, x$  such that

- (a)  $\{i, j\} = G(F)$ ;
- (b)  $x = \min((W_{i,u} - W_{j,u}) \cup (W_{j,u} - W_{i,u}))$ ;
- (c)  $x \in W_{i,u} \Leftrightarrow x \in C$ .

Go to (Branch).

**(Branch)** If  $C \cup W_{i,s} \subseteq W_{\mathbf{U}(F,\tau),u}$  and  $(W_{i,u} - W_{i,|\sigma|}) \cup (W_{j,u} - W_{j,|\sigma|}) \neq \emptyset$  Then go to (Copy) Else go to (Enum).**(Enum)** Let  $t$  be the maximal element of  $\{s, \dots, u\}$  such that one of the following conditions holds:**(Min)**  $t = s$ ;**(Equal)**  $C \subseteq W_{i,t} \cup W_{j,t} \subset W_{i,u} \cap W_{j,u}$ ;**(Inf)**  $C \subseteq W_{i,t} \subset W_{i,u} \wedge (\forall y \leq x) [y \notin (W_{i,u} - W_{i,|\sigma|}) \cup (W_{j,u} - W_{j,|\sigma|})]$ ;**(Diff)**  $C \subseteq W_{i,t} \subset W_{i,u}$  and  $W_{j,u} = W_{i,s}$ ;**(Sub)**  $C \subseteq W_{i,t} \subseteq W_{\mathbf{U}(F,\tau),u}$ ;**(Exact)**  $C = W_{i,t}$  and  $t = |\sigma|$ .Let  $W_{k,u} = W_{i,t}$ , update  $s = t$ ,  $u = u + 1$  and go to (Branch).**(Copy)** Let  $W_{k,u} = W_{\mathbf{U}(F,\tau),u}$ , update  $u = u + 1$  and go to (Copy).**(Empty)** Let  $W_{k,u} = \emptyset$ , update  $u = u + 1$  and go to (Empty).**Fig. 1.** Algorithm to enumerate  $W_{\mathbf{U}(F,\sigma)}$  from Theorem 16

Now one builds the following new learner  $\mathbf{U}'(\tau)$  which outputs the least index of  $\emptyset$  if  $\text{content}(\tau) = \emptyset$  or, in the definition of  $\mathbf{M}_1, \mathbf{M}_2$  on input  $\tau$ , the search for  $\sigma$  satisfying (a), (b) (as in proof of Theorem 12) fails. Otherwise  $\mathbf{U}'(\tau)$  is defined as follows.

$$W_{\mathbf{U}'(\tau)} = W_{\mathbf{U}(\{e_1, e_2\}, \tau)} \text{ where, for } a = 1, 2,$$

$$W_{e_a} = \begin{cases} W_{F(\sigma_\tau, D_\tau, a)} & \text{if (f) is not used for} \\ & W_{F(\sigma_\tau, D_\tau, 1)} \text{ or } W_{F(\sigma_\tau, D_\tau, 2)}; \\ W_{F(\sigma_\tau, D_\tau, 1)} \cup W_{F(\sigma_\tau, D_\tau, 2)} & \text{if (f) is used for} \\ & W_{F(\sigma_\tau, D_\tau, 1)} \text{ or } W_{F(\sigma_\tau, D_\tau, 2)}. \end{cases}$$

Let  $L \in \mathcal{L}$  and  $T$  be a text for  $L$ . Let  $n$  be the first number where  $L \in \{W_{\mathbf{M}_1(T[n])}, W_{\mathbf{M}_2(T[n])}\}$ . Then, for any  $m \geq n$  and any  $a \in \{1, 2\}$ ,  $W_{\mathbf{M}_a(T[m])} = W_{\mathbf{M}_a(T[n])}$  and  $W_{\mathbf{M}_a(T[m])}$  is not of the form  $E \cup \tilde{E}$ , where  $E$  is as introduced in the proof of Theorem 12 and  $\tilde{E}$  is finite. Then  $\mathbf{U}$  is fed with the same parameter set  $\{e_1, e_2\}$  for all  $m \geq n$  and one of the  $e_1, e_2$  enumerates  $L$ . Thus  $\mathbf{U}'$  **TxtBc**-learns  $L$  on  $T$ .

It remains to show that  $\mathbf{U}'$  is non U-shaped on  $T$ . This is clearly true if  $L$  is the empty set. So assume  $L \neq \emptyset$ . Consider any  $m$  with  $W_{\mathbf{U}'(T[m])} = L$ . If case (f) of the algorithm for  $F(\sigma_{T[m]}, D_{T[m]}, 1)$  or  $F(\sigma_{T[m]}, D_{T[m]}, 2)$  applies, then  $W_{e_1}$  and  $W_{e_2}$  are the same infinite set  $E \cup \tilde{E}$  for some finite set  $\tilde{E}$ . It follows by the additional property (2) of  $\mathbf{U}$  in Theorem 16 that  $\mathbf{U}'(T[m])$  either outputs an index for the empty set or for  $E \cup \tilde{E}$ ; both sets are different from  $L$ , thus case (f) does not apply. Hence,  $T[m]$  is a stabilizing sequence for both  $\mathbf{M}_1, \mathbf{M}_2$  on those sets  $W_{\mathbf{M}_1(T[m])}, W_{\mathbf{M}_2(T[m])}$  which are not empty. Since one of these is a superset of  $L$  by the additional property (1) of  $\mathbf{U}$  in Theorem 16, it follows that  $\mathbf{M}_1, \mathbf{M}_2$  do not change mind on  $T$  beyond  $T[m]$ . For  $a = 1, 2$  the parameter  $e_a$  is defined as above for  $\tau = T[m]$  and it holds that  $W_{e_a} = W_{\mathbf{M}_a(T[m])}$ . Thus  $\mathbf{U}'(T[o])$  coincides with  $\mathbf{U}(\{e_1, e_2\}, T[o])$  for all  $o \geq m$  and  $\mathbf{U}$  with the parameter set  $\{e_1, e_2\}$  is non U-shaped on the text  $T$  for  $L$ . The same holds for  $\mathbf{U}'$ . Thus  $\mathbf{U}'$  **NUShTxtBc**-learns  $\mathcal{L}$ .  $\blacksquare$

From Theorem 7 it is already known that, for all  $b > 1$ , U-shaped learning behaviour is necessary for **TxtFex<sub>b</sub>** identification of any class in **TxtFex<sub>b</sub>** – **TxtFex<sub>1</sub>**. Theorem 18 strengthens this result by showing that, for some classes of languages in **TxtFex<sub>b</sub>** for  $b > 2$ , the necessity of U-shaped behaviour cannot be circumvented by allowing infinitely many correct grammars in the limit, that is, by shifting to the more liberal criterion of **TxtBc**-identification. This is one of the rare cases in inductive inference where the containment in a class defined without numerical parameters holds for level 2 but not for level 3 and above of a hierarchy. The proof is a diagonalization proof reminiscent of the proof of Theorem 4 in [12].

**Theorem 18.** **TxtFex<sub>3</sub>  $\not\subseteq$  NUShTxtBc**

**Proof.** Let  $L_{i,j} = \{\langle i, j, k \rangle : k \in \mathbb{N}\}$ ,  $I_{i,j} = W_i \cap L_{i,j}$  and  $J_{i,j} = W_j \cap L_{i,j}$  for  $i, j \in \mathbb{N}$ . The class  $\mathcal{L} = \{L_{i,j} : i, j \in \mathbb{N}\} \cup \{I_{i,j}, J_{i,j} : i, j \in \mathbb{N} \wedge I_{i,j} \subset J_{i,j} \wedge |I_{i,j}| < \infty\}$  witnesses the separation.

To see that  $\mathcal{L}$  is in **TxtFex<sub>3</sub>**, consider the following machines  $\mathbf{N}_I, \mathbf{N}_J, \mathbf{N}_L$  which initially output indices of the empty set. Each of them waits for the first tuple of the form  $\langle i, j, k \rangle$  for some  $k$  to come up in the input. From then on,  $\mathbf{N}_I$  outputs an index for  $I_{i,j}$  forever,  $\mathbf{N}_J$  an index for  $J_{i,j}$  forever and  $\mathbf{N}_L$  an index for  $L_{i,j}$  forever. So, for every  $i, j \in \mathbb{N}$ ,  $\mathbf{N}_I$  learns the set  $I_{i,j}$ ,  $\mathbf{N}_J$  the set  $J_{i,j}$  and  $\mathbf{N}_L$  the set  $L_{i,j}$ . The class  $\mathcal{L}$  is learnable by a team of three machines which converge on every text for every language in  $\mathcal{L}$  to some index. It follows from Theorem 2 that  $\mathcal{L}$  is in **TxtFex<sub>3</sub>**.

So it remains to show that  $\mathcal{L}$  is not in **NUShTxtBc**, that is, to show that any given **TxtBc**-learner for  $\mathcal{L}$  is U-shaped on some text for some language in  $\mathcal{L}$ . Given the learner  $\mathbf{M}$ , one defines the following function  $F$  by an approximation from below.

$$F_s(i, j) = \begin{cases} F_{s-1}(i, j) & \text{if } s > 0 \text{ and} \\ & W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F_{s-1}(i, j) \rangle)}, s \subseteq L_{i, j}; \\ k & \text{otherwise where } k \text{ is the first number} \\ & \text{found with } k > F_t(i + j) + s, \text{ for all } t < s, \text{ and} \\ & \{\langle i, j, 0 \rangle, \langle i, j, 1 \rangle, \dots, \langle i, j, k \rangle\} \subset W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, k \rangle)}. \end{cases}$$

Since  $\langle i, j, 0 \rangle, \langle i, j, 1 \rangle, \dots$  is a text for  $L_{i, j}$  and  $\mathbf{M}$  **TxtBc**-learns  $L_{i, j}$ , almost all hypotheses  $\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, k \rangle)$  are indices for  $L_{i, j}$ . Thus the  $k$  is always found in the second part of the definition of  $F_s$  and  $F_s$  is well-defined. Furthermore, if  $F_{s-1}(i, j)$  is sufficiently large, the condition

$$W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F_{s-1}(i, j) \rangle)}, s \subseteq L_{i, j}$$

holds for all  $s$  and thus  $F_s(i, j) = F_{s-1}(i, j)$ . So the limit  $F(i, j)$  of all  $F_s(i, j)$  exists and is approximated from below. By considering the first  $s$  where  $F(i, j) = F_s(i, j)$  and the fact that it is then no longer updated, one has

$$\{\langle i, j, 0 \rangle, \langle i, j, 1 \rangle, \dots, \langle i, j, F(i, j) \rangle\} \subset W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F(i, j) \rangle)} \subseteq L_{i, j}.$$

Now there are r.e. sets  $W_a, W_b$  such that

$$W_a = \{\langle i, j, l \rangle : i, j \in \mathbb{N} \wedge l \in \{0, 1, \dots, F(i, j)\}\},$$

$$W_b = \{\langle i, j, l \rangle : i, j \in \mathbb{N} \wedge (\exists t > l) [\langle i, j, l \rangle \in W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F_t(i, j) \rangle)}, l]\}.$$

Now fix the parameters  $i, j$  such that  $i = a$  and  $j = b$ ; the cases where  $i \neq a$  or  $j \neq b$  are not important in the considerations below.

Assume that  $\langle i, j, l \rangle \in W_b$  using a parameter  $t$  with  $F_t(i, j) \neq F(i, j)$ . Let  $s$  be the first stage with  $F_s(i, j) = F(i, j)$ ; note that  $s > t$ . Then by the definition of  $F_s$ ,  $F(i, j) = F_s(i, j) > s > t$  and  $\{\langle i, j, 0 \rangle, \langle i, j, 1 \rangle, \dots, \langle i, j, F_s(i, j) \rangle\} \subset W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F_s(i, j) \rangle)}$ . So  $\langle i, j, l \rangle$  is in  $W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F(i, j) \rangle)}$  as well. Thus  $\{\langle i, j, 0 \rangle, \langle i, j, 1 \rangle, \dots, \langle i, j, F(i, j) \rangle\} = W_i \cap L_{i, j} = I_{i, j} \subset J_{i, j} = W_j \cap L_{i, j} = W_{\mathbf{M}(\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F(i, j) \rangle)}$  and  $I_{i, j}$  is finite. Hence  $I_{i, j}, J_{i, j} \in \mathcal{L}$ .

Now consider a text  $T$  for  $J_{i, j}$  formed as follows. Let  $\sigma$  be the sequence  $\langle i, j, 0 \rangle \langle i, j, 1 \rangle \dots \langle i, j, F(i, j) \rangle$ . Note that  $\mathbf{M}(\sigma)$  outputs an index for  $J_{i, j}$ . Let  $\tau = \sigma \#^r$ , for some  $r$ , be such that  $\mathbf{M}(\tau)$  is an index for  $I_{i, j}$ . Note that there exists such  $\tau$  since  $\mathbf{M}$  **TxtBc**-identifies  $I_{i, j}$ . Let  $T$  be a text for  $J_{i, j}$  starting with  $\tau$ . Now  $\mathbf{M}$  on  $T$  has to output an index  $J_{i, j}$  beyond  $\tau$ . Hence,  $\mathbf{M}$  is U-shaped on text  $T$ , and thus  $\mathbf{M}$  is not a **NUSHTxtBc**-learner for  $\mathcal{L}$ . Since  $\mathbf{M}$  was chosen arbitrarily,  $\mathcal{L}$  is not **NUSHTxtBc**-learnable. ■

Since **TxtFex**<sub>3</sub>  $\subset$  **TxtFex**<sub>4</sub>  $\subset \dots \subset$  **TxtFex**<sub>\*</sub>, one immediately gets the following corollary.

**Corollary 19.**  $(\forall b \in \{3, 4, \dots, *\}) [\mathbf{TxtFex}_b \not\subseteq \mathbf{NUSHTxtBc}]$

A further corollary is that the counterpart of Theorem 16 does not hold for sets of three indices. Indeed, if such an algorithm would exist, then one could



**NUShTxtBc**-learn  $\mathcal{L}$  from Theorem 18 by conjecturing  $\emptyset$  until the first triple  $\langle i, j, k \rangle$  comes up and then simulating the uniform learner with a set of three indices for the sets  $I_{i,j}, J_{i,j}, L_{i,j}$  from then on without changing this parameter set anymore. But Theorem 18 clearly showed that such a learner does not exist.

**Corollary 20.**  $\{W_e : e \in F\} \notin \text{NUShTxtBc}$  for any finite set  $F$  with  $|F| \geq 3$ .

### 5 Teams Revisited

Classes in **TxtFex**<sub>2</sub> are in **TxtBc** and in  $[1, 2]\text{NUShTxtEx}$ . The next proposition shows that one cannot weaken the condition of being in **TxtFex**<sub>2</sub> to the combination of the two consequences in Theorem 17. Furthermore the condition that the team members converge on every text for a language in  $\mathcal{L}$  is essential in Theorem 2.

**Proposition 21.**  $\mathcal{L} \notin \text{NUShTxtBc}$  for any  $[1, 2]\text{NUShTxtEx}$ -learnable  $\mathcal{L}$  with  $\text{TxtFex}_3$ .

**Remark 22.**  $\text{TxtFex}_* \not\subseteq [1, b]\text{TxtEx}$  for all  $b \in \mathbb{N}^+$ , as witnessed by  $\mathcal{H}_*$ . Note that by Proposition 21 it can be that a class in  $\text{TxtFex}_{b+1} - \text{TxtFex}_b$  is already  $[1, b]\text{TxtEx}$ -learnable.

A further interesting question is whether one can at least obtain non U-shaped team learning for arbitrary team learnable classes. This is true for  $[1, 1]\text{TxtEx}$  by Theorem 12 but it fails for  $[1, 2]\text{TxtEx}$ -learning.

**Theorem 23.**  $b \in \{2, 3, \dots\} \implies [1, b]\text{NUShTxtEx} \subset [1, b]\text{TxtEx}$   
 $a, b \geq 1 \implies 1 \leq a \leq b \implies [a, b]\text{TxtEx} \subseteq [a, a + b]\text{NUShTxtEx}$

### 6 Conclusion

The following results were obtained.

- $\text{TxtFex}_b \subset [1, b]\text{NUShTxtEx}$  for all  $b \in \{2, 3, \dots\}$ .
- $\text{TxtFex}_1 = \text{TxtEx} = \text{NUShTxtEx} = [1, 1]\text{NUShTxtEx}$ , see also [1].
- $[1, b]\text{NUShTxtEx} \subset [1, b]\text{TxtEx}$ , for all  $b \in \{2, 3, \dots\}$ .
- $\text{NUShTxtFex}_b = \text{NUShTxtEx}$  for all  $b \in \{1, 2, \dots, *\}$ .
- $\text{TxtFex}_2 \subseteq \text{NUShTxtBc}$ .
- $\text{TxtFex}_3 \not\subseteq \text{NUShTxtBc}$ .

These results and the facts known from previous work [1, 8] are summarized in Figure 2. Single-headed arrows in the diagram denote proper inclusions. Double-headed arrows denote equality. All transitive closures of the inclusions displayed are valid and no other inclusions hold between language learning criteria in the diagram.

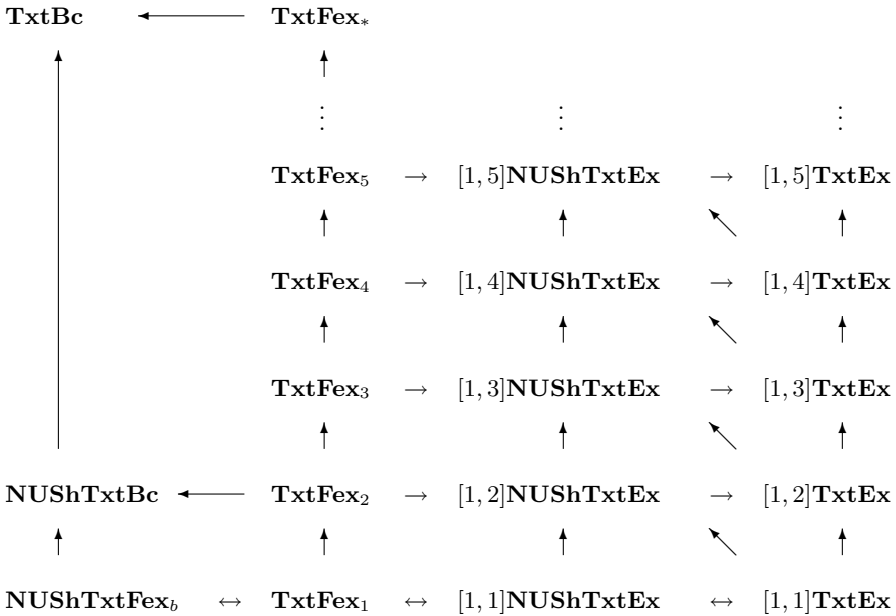


Fig. 2. Summary of the results for  $b \in \{1, 2, 3, 4, 5, *\}$

We note that our proof that  $\text{TxtFex}_3 \not\subseteq \text{NUShTxtBc}$  intriguingly features learning finite tables versus general rules, but does  $\text{TxtFex}_3$ , as might be expected from some models of the human case of U-shaped learning, feature, among other things, learning an incorrect general rule followed by learning a general rule augmented by a correcting finite table. This difference may be significant or, more likely, nothing more than an artifact of our particular proof. Not explored herein, but very interesting to investigate in the future, are complexity-issues of U-shaped learning.

## References

- [1] Ganesh Baliga, John Case, Wolfgang Merkle, Frank Stephan and Rolf Wiehagen. *When unlearning helps*. <http://www.cis.udel.edu/~case/papers/decisive.ps>, Manuscript, 2005. Preliminary version of the paper appeared at ICALP, Springer LNCS 1853:844–855, 2000.
- [2] Janis Bārzdīņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.
- [3] Lenore Blum and Manuel Blum. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
- [4] Manuel Blum. A machine independent theory of the complexity of the recursive functions. *Journal of the Association for Computing Machinery* 14:322–336, 1967.

- [5] T. G. R. Bower. Concepts of development. In *Proceedings of the 21st International Congress of Psychology*. Presses Universitaires de France, pages 79–97, 1978.
- [6] Melissa Bowerman. Starting to talk worse: Clues to language acquisition from children’s late speech errors. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*. Academic Press, New York, 1982.
- [7] Susan Carey. Face perception: Anomalies of development. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*, Developmental Psychology Series. Academic Press, pages 169–190, 1982.
- [8] John Case. The power of vacillation in language learning. *SIAM Journal on Computing*, 28(6):1941–1969, 1999.
- [9] John Case and Chris Lynes. Machine inductive inference and language identification. In M. Nielsen and E. M. Schmidt, editors, *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 140, pages 107–115. Springer-Verlag, 1982.
- [10] John Case and Carl H. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [11] Mark Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85:1–11, 1990.
- [12] Mark Fulk, Sanjay Jain and Daniel Osherson. Open problems in “Systems That Learn”. *Journal of Computer and System Sciences*, 49:589–604, 1994.
- [13] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [14] David Kirsh. PDP learnability and innate knowledge of language. In S. Davis, editor, *Connectionism: Theory and Practice*, pages 297–322. Oxford University Press, 1992.
- [15] Gary Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T. John Rosen and Fei Xu. *Overregularization in Language Acquisition*. Monographs of the Society for Research in Child Development, vol. 57, no. 4. University of Chicago Press, 1992. Includes commentary by Harold Clahsen.
- [16] Piergiorgio Odifreddi. *Classical Recursion Theory*. North Holland, Amsterdam, 1989.
- [17] Daniel Osherson and Scott Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.
- [18] Steven Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.
- [19] Kim Plunkett and Virginia Marchman. U-shaped learning and frequency effects in a multi-layered perceptron: implications for child language acquisition. *Cognition*, 38(1):43–102, 1991.
- [20] Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967. Reprinted, MIT Press, 1987.
- [21] Carl H. Smith. The power of pluralism for automatic program synthesis. *Journal of the Association of Computing Machinery*, 29:1144–1165, 1982.
- [22] Sidney Strauss and Ruth Stavy, editors. *U-Shaped Behavioral Growth*. Developmental Psychology Series. Academic Press, 1982.
- [23] Niels A. Taatgen and John R. Anderson. Why do children learn to say broke? A model of learning the past tense without feedback. *Cognition*, 86(2):123–155, 2002.
- [24] Kenneth Wexler. On extensional learnability. *Cognition*, 11:89–95, 1982.

# Learning Multiple Languages in Groups

Sanjay Jain<sup>1,\*</sup> and Efim Kinber<sup>2</sup>

<sup>1</sup> School of Computing, National University of Singapore, Singapore 117543  
`sanjay@comp.nus.edu.sg`

<sup>2</sup> Department of Computer Science, Sacred Heart University, Fairfield, CT  
06432-1000, U.S.A.  
`kinbere@sacredheart.edu`

**Abstract.** We consider a variant of Gold’s learning paradigm where a learner receives as input  $n$  different languages (in form of one text where all input languages are interleaved). Our goal is to explore the situation when a more “coarse” classification of input languages is possible, whereas more refined classification is not. More specifically, we answer the following question: under which conditions, a learner, being fed  $n$  different languages, can produce  $m$  grammars covering all input languages, but cannot produce  $k$  grammars covering input languages for any  $k > m$ . We also consider a variant of this task, where each of the output grammars may not cover more than  $r$  input languages. Our main results indicate that the major factor affecting classification capabilities is the difference  $n - m$  between the number  $n$  of input languages and the number  $m$  of output grammars. We also explore relationship between classification capabilities for smaller and larger groups of input languages. For the variant of our model with the upper bound on the number of languages allowed to be represented by one output grammar, for classes consisting of disjoint languages, we found complete picture of relationship between classification capabilities for different parameters  $n$  (the number of input languages),  $m$  (number of output grammars), and  $r$  (bound on the number of languages represented by each output grammar). This picture includes a combinatorial characterization of classification capabilities for the parameters  $n, m, r$  of certain types.

## 1 Introduction

In this paper, we continue a line of research where the learner is required to learn unions of different concepts [JNT05]. This situation occurs, for example, when children living in a multilingual environment learn several languages simultaneously. In this case, in ideal, children are able to learn each individual language. A more complex case is the problem of multilayered classification, where descriptions of families of objects on a higher level are rather “coarse”, while descriptions on a lower level are much more specific/refined. An example of this type of classification is the theory of species: as a result of learning process (inductive synthesis of concepts from examples), life can be classified as just animals and plants, or, more specifically, as families - fishes, birds, mammals, etc.,

---

\* Supported in part by NUS grant number R252-000-127-112.

or, even more specifically as different species, etc. An important issue here is that a more coarse classification is typically easier to achieve than a more refined one. For example, an alien civilization, learning life on Earth (from examples), most likely, will have much more difficulty describing different birds, than making distinction between birds and fishes. A child, learning classical music, has much easier time determining if a piece uses  $3/4$  or  $4/4$  time signature, than telling apart waltz, mazurka, or polonaise (each of them uses time signature  $3/4$ ).

Our goal in this paper is to determine if, and under which circumstances, a more coarse classification, as a result of learning process, is possible, whereas a more refined classification is not. More specifically, we explore the following general situation: under which circumstances, a learner, facing a union of  $n$  languages on the input, is able to learn descriptions of  $m$  (larger) groups of languages from the union, but is not able to learn descriptions of  $k$  (smaller) groups of input languages for  $k > m$ . For example, we would like to find out when a learner, facing a union of 6 languages on the input, can learn descriptions of 3 groups of languages, but cannot learn descriptions for each individual language. We are also interested in situations when learning larger groups of languages can be easier than smaller ones.

To model the process of learning, we employ the well-known Gold's learning paradigm [Gol67]: the learner receives all members of the union of languages in some random order and produces a sequence of descriptions (grammars) that stabilizes to a correct description. This model is known in literature as **TextEx** (where Ex stands for "explanatory learning"). Exploration of this model provided a robust advice to cognition theory (see, for example, [WC80]). We consider also a popular variant of this model, **TextBc**, (introduced in [CL82, OW82]) where a learner produces a sequence of conjectures, almost all of which are correct descriptions of the target language (but not necessarily the same - BC stands here for "behaviourally correct learning").

Among several papers in this line of research, the closest to our inquiry is the paper [JNT05], where the authors primarily explore the issues of learnability of larger unions of languages versus smaller unions of languages from the same families. In particular, they define the concept of  $\forall_{i \in I} \exists_{j \in I}$  learnability, when a learner is required to learn each of the members of the union, and compare this notion with the situation when the learner may provide one description for the whole union. Relevant results from [JNT05] can be viewed as the first step in our line of research.

Our main results can be summarized as follows. On one hand, if  $n - m > n' - m'$  then there exists a class of languages such that it is possible to learn unions of  $n$  languages from this class in  $m$  groups, but it is not possible to learn unions of  $n'$  languages from this class in  $m'$  groups (Theorem 7). That is, the difference between the number of input languages in the union and the number of conjectures ultimately produced by the learner is the major factor affecting learning capabilities. On the other hand, if a family consists of only disjoint languages, then, if it is possible to learn unions of  $n$  languages in  $m$  groups, then it is possible to learn unions of  $n - 1$  languages in  $m - 1$  groups. For example,

learnability of unions of any 6 disjoint languages in 3 groups implies learnability of any union of 5 languages in 2 groups (Theorem 9). We also extend our results to the case when the number of languages in learned groups is bounded (in the general case, the learner, when required to produce at least 3 groups for the union of 6 input languages, can include 4 languages into one group and only one language into each of the remaining two groups); the corresponding results are presented in Corollaries 14 and 15, and Theorem 16. The last result of this paper, Theorem 18, presents a combinatorial characterization (when the language classes consist of disjoint languages) for the remaining cases not solved by Corollaries 14 and 15, and Theorem 16 (i.e., the circumstances under which learnability of unions of  $n$  languages in  $m$  groups describing at most  $r$  languages in each of the groups implies learnability for other corresponding parameters  $n', m', r'$ , where  $n' \leq n, r \leq r'$  and  $n' - m' \leq n - m$ ).

## 2 Notation and Preliminaries

Any unexplained recursion theoretic notation is from [Rog67].  $N$  denotes the set of natural numbers,  $\{0, 1, 2, 3, \dots\}$ .  $\emptyset$  denotes the empty set.  $\subseteq, \subset, \supseteq, \supset$  respectively denote subset, proper subset, superset and proper superset.  $D_x$  denotes the finite set with (canonical) index  $x$  [Rog67]. We sometimes identify finite sets with their canonical indices. The quantifier ‘ $\forall^\infty$ ’ essentially from [Blu67], means ‘for all but finitely many’.

$\uparrow$  denotes undefined.  $\max(\cdot)$  denotes the maximum of a set, where  $\max(\emptyset) = 0$ .  $\langle \cdot, \cdot \rangle$  stands for an arbitrary, computable, one-to-one encoding of all pairs of natural numbers onto  $N$  [Rog67]. Similarly we can define  $\langle \cdot, \dots, \cdot \rangle$  for encoding tuples of natural numbers onto  $N$ .

$\varphi$  denotes a fixed universal programming system for the partial computable functions:  $N \rightarrow N$  [Rog58, Rog67, MY78].  $\varphi_i$  denotes the partial computable function computed by program  $i$  in the  $\varphi$ -system.  $W_i$  denotes  $\text{domain}(\varphi_i)$ .  $W_i$  is, then, the r.e. set/language ( $\subseteq N$ ) accepted (or equivalently, generated) by the  $\varphi$ -program  $i$ .  $\mathcal{E}$  will denote the set of all r.e. languages.  $\mathcal{L}$ , with or without decorations, ranges over  $\mathcal{E}$ .  $\text{DisjClass} = \{\mathcal{L} \mid (\forall L, L' \in \mathcal{L})[L = L' \text{ or } L \cap L' = \emptyset]\}$ , i.e., **DisjClass** is the collection of language classes which consist of disjoint languages.  $\mathcal{L}$ , with or without decorations, ranges over subsets of  $\mathcal{E}$ .

We now consider some basic notions in language learning. We first introduce the concept of data that is presented to a learner. A text  $T$  is a mapping from  $N$  into  $(N \cup \{\#\})$  (see [Gol67]). The content of a text  $T$ , denoted  $\text{content}(T)$ , is the set of natural numbers in the range of  $T$ .  $T$  is a text for  $L$  iff  $\text{content}(T) = L$ .  $T[n]$  denotes the initial segment of  $T$  of length  $n$ . We let  $T$ , with or without superscripts, range over texts. Intuitively,  $\#$ 's in the texts denote pauses in the presentation of data. For example, the only text for the empty language is just an infinite sequence of  $\#$ 's.

A finite sequence  $\sigma$  is an initial segment of a text.  $\text{content}(\sigma)$ , is the set of natural numbers in the range of  $\sigma$ .  $|\sigma|$  denotes the length of  $\sigma$ , and if  $n \leq |\sigma|$ , then  $\sigma[n]$  denotes the initial segment of  $\sigma$  of length  $n$ .  $\sigma \smallfrown \tau$  denotes the concatenation of initial segments.

An algorithmic device which computes a mapping from finite initial segments of texts into  $N$ . We let  $\mathbf{M}$ , with or without decorations, range over learning machines. We say that  $\mathbf{M}(T)\downarrow = i \Leftrightarrow (\forall^\infty n)[\mathbf{M}(T[n]) = i]$ .

We now introduce criteria for a learning machine to be considered on languages. Our first criteria is based on learner, given a text for the language, converging to a grammar for the language.

**Definition 1.** [Gol67] (a)  $\mathbf{M} \text{ TxtEx}$ -identifies  $L$  (written:  $L \in \text{TxtEx}(\mathbf{M})$ )  $\Leftrightarrow (\forall \text{ texts } T \text{ for } L)(\exists i \mid W_i = L)[\mathbf{M}(T)\downarrow = i]$ .  
 (b)  $\text{TxtEx} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \text{TxtEx}(\mathbf{M})]\}$ .

The influence of Gold’s paradigm [Gol67] to human language learning is discussed by various authors, for example [Pin79, WC80, OSW86].

The following definition is based on learner semantically rather than syntactically converging to the grammar(s) for the language. Here note that equivalence of grammars is non-computable. The corresponding notion for learning functions was introduced by [Bär74, CS83].

**Definition 2.** [CL82, OW82]. (a)  $\mathbf{M} \text{ TxtBc}$ -identifies  $L$  (written:  $L \in \text{TxtBc}(\mathbf{M})$ )  $\Leftrightarrow (\forall \text{ texts } T \text{ for } L)(\forall^\infty n)[W_{\mathbf{M}(T[n])} = L]$ .  
 (b)  $\text{TxtBc} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \text{TxtBc}(\mathbf{M})]\}$ .

It can be shown that  $\text{TxtEx} \subset \text{TxtBc}$  (for example, see [CL82, OW82]).

### 3 Learning Languages in Groups

Now we give formal definition of our main learning model - under which a learner, being fed the union of  $n$  different languages, outputs in the limit at least  $m$  grammars, each representing a number of input languages, so that the union of all  $m$  grammars covers the union of all input languages.

**Definition 3.** (a) We say that  $\mathbf{M} [m, n]\text{MultEx}$ -identifies  $\mathcal{L}$ , iff for all distinct languages  $L_1, \dots, L_n$  in  $\mathcal{L}$ , for all texts  $T$  for  $L_1 \cup \dots \cup L_n$ , there exist  $i_1, \dots, i_k$ , (where  $k \geq m$ ) such that  $\mathbf{M}(T)$  converges on  $T$  to the canonical index for the set  $\{i_1, \dots, i_k\}$  and there exists a partition  $G_1, \dots, G_k$  of  $\{1, 2, \dots, n\}$  such that  
 (i) each  $G_i$  is non-empty,  
 (ii) for  $1 \leq j \leq k$ ,  $i_j$  is a grammar for  $\bigcup_{r \in G_j} L_r$ .  
 (b)  $[m, n]\text{MultEx} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M} [m, n]\text{MultEx}\text{-identifies } \mathcal{L}]\}$ .

Note that requiring  $k = m$  in the above does not change the class of languages which can be  $[m, n]\text{MultEx}$ -identified, as one can just combine  $k - m + 1$  of the language groups into one. However, this may make a difference in some modifications we consider later. Our next definition is a modification of our model for behaviorally correct type of learning.

**Definition 4.** (a) We say that  $\mathbf{M} [m, n]\text{MultBc}$ -identifies  $\mathcal{L}$ , iff for all distinct languages  $L_1, \dots, L_n$  in  $\mathcal{L}$ , for all texts  $T$  for  $L_1 \cup \dots \cup L_n$ , for all but finitely

many  $t$ , there exist  $i_1, \dots, i_k$ , (where  $k \geq m$ ) such that  $\mathbf{M}(T[t]) =$  the index for  $\{i_1, \dots, i_k\}$  and there exists a partition  $G_1, \dots, G_k$  of  $\{1, 2, \dots, n\}$  such that

- (i) each  $G_i$  is non-empty,
  - (ii) for  $1 \leq j \leq k$ ,  $i_j$  is a grammar for  $\bigcup_{r \in G_j} L_r$ .
- (b)  $[m, n]\mathbf{MultBc} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M} [m, n]\mathbf{MultBc}\text{-identifies } \mathcal{L}]\}$ .

Our first result demonstrates that, for some families of languages, one can **Bc**-learn each individual language from the input union of  $n$  languages, while no **Ex**-learner can converge to a correct single grammar representing the whole union.

**Theorem 5.**  $1 \leq m < 1 \leq n$   $[n, n]\mathbf{MultBc} - [1, m]\mathbf{MultEx} \neq \emptyset$   
 DisjClass

Let  $\mathcal{L}$  be a class of languages in  $\mathbf{TxtBc} - \mathbf{TxtEx}$  (such a class exists by results from [CS83, CL82]). Let  $cyl_L^i = \{\langle i, x \rangle \mid x \in L\}$ . It is easy to verify that for all  $i$ , one can find an  $L_i \in \mathcal{L}$  such that  $\mathbf{M}_i$  does not **TxtEx** identify  $\bigcup_{i^*m \leq j < (i+1)^*m} cyl_{L_i}^j$  (otherwise, one can easily show that  $\mathcal{L} \in \mathbf{TxtEx}$ ). Let  $\mathcal{L} = \{cyl_{L_i}^j \mid i \in N, i^*m \leq j < (i+1)^*m\}$ . It now follows that  $\mathcal{L} \notin [1, m]\mathbf{MultEx}$ . On the other hand,  $\mathcal{L} \in [n, n]\mathbf{MultBc}$  easily follows as for any  $S_1, \dots, S_n$  in the class  $\mathcal{L}$ , from a text for  $S_1 \cup \dots \cup S_n$ , one can easily obtain texts for  $S_1, \dots, S_n$  and then use **TxtBc** learning procedure for each of them. ■

Now we show that, if the number of the languages in the union given to an **Ex**-learner is smaller than the number of languages given to a **Bc**-learner, then the reverse of above result also holds: for some family of languages, an **Ex**-learner can correctly infer grammars for each individual language from a smaller input union, while **Bc**-learnability of even one grammar covering the whole larger union of languages is impossible.

**Theorem 6.**  $1 \leq n$   $[n, n]\mathbf{MultEx} - [1, n + 1]\mathbf{MultBc} \neq \emptyset$   
 DisjClass

[JNT05] (Theorem 21) constructed a class  $\mathcal{L} \in \mathbf{DU}^n\mathbf{TxtEx} - \mathbf{U}^{n+1}\mathbf{TxtEx}$  (see [JNT05] for definitions of  $\mathbf{DU}^n\mathbf{TxtEx}$  and  $\mathbf{U}^{n+1}\mathbf{TxtEx}$  — informally,  $\mathbf{U}^{n+1}\mathbf{TxtEx}$  is similar to  $[1, n + 1]\mathbf{MultEx}$  and  $\mathbf{DU}^n\mathbf{TxtEx}$  is similar to  $[n, n]\mathbf{MultEx}$ ). It is easy to verify that  $\mathbf{DU}^n\mathbf{TxtEx} \subseteq [n, n]\mathbf{MultEx}$ . Furthermore, the diagonalization proof in [JNT05] also shows that  $\mathcal{L} \notin [1, n + 1]\mathbf{MultEx}$ . This diagonalization can be easily generalized to show that  $\mathcal{L} \notin [1, n + 1]\mathbf{MultBc}$  and  $\mathcal{L} \notin \mathbf{U}^{n+1}\mathbf{TxtBc}$ . We omit the details. ■

Yet another type of situation when **Ex**-learnability of unions in groups may be possible but **Bc**-learnability might be not is presented in the following result: for some class of languages, if the difference  $n - m$  between the size  $n$  of the union of languages and the number of learned groups  $m$  is greater than  $n' - m'$  (where  $m' \geq 2$ ), then an **Ex**-learner can infer grammars for  $m$  groups representing an input union of  $n$  languages, while **Bc**-learning of any union  $n'$  input languages in  $m'$  groups is not possible.



**Theorem 7.**  $1 \leq m \leq n$ ,  $2 \leq m' \leq n'$ ,  $n - m > n' - m'$ .  
 $[m, n]\mathbf{MultEx} - [m', n']\mathbf{MultBc} \neq \emptyset$

**DisjClass**

For  $i \in N$ , define

(i) for  $x \leq n - m$ , let

$$X_{i,s_0,s_1,\dots,s_{i-1}}^x = \{\langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, 2x \rangle, \langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, 2x + 1 \rangle\}.$$

(ii) for  $x < n - m$ ,  $i \in N$ , let

$$Y_{i,s_0,s_1,\dots,s_{i-1}}^x = \{\langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, 2x + 1 \rangle, \langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, 2x + 2 \rangle\};$$

$$Y_{i,s_0,s_1,\dots,s_{i-1}}^{n-m} = \{\langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, 2(n - m) + 1 \rangle, \langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, 0 \rangle\}.$$

(iii) for  $x < n' - (n - m + 1)$ , let  $Z_{i,s_0,s_1,\dots,s_{i-1}}^x = \{\langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, 2n + x \rangle\}$ .

Note that  $\bigcup_{x \leq n-m} X_{i,s_0,s_1,\dots,s_{i-1}}^x = \bigcup_{x \leq n-m} Y_{i,s_0,s_1,\dots,s_{i-1}}^x$  (this will be utilized for diagonalization against  $\mathbf{M}_i$ ).

For any binary sequence of  $s_i$ 's, let  $\mathcal{L}_{s_0,s_1,\dots} = \{X_{i,s_0,s_1,\dots,s_{i-1}}^x \mid i \in N, x \leq n - m, s_i = 0\} \cup \{Y_{i,s_0,s_1,\dots,s_{i-1}}^x \mid i \in N, x \leq n - m, s_i = 1\} \cup \{Z_{i,s_0,s_1,\dots,s_{i-1}}^x \mid i \in N, x < n' - (n - m + 1)\}$ .

$\mathcal{L}_{s_0,s_1,\dots} \in [m, n]\mathbf{MultEx}$ , for any fixed binary values of  $s_i$ 's.

Fix  $s_0, s_1, \dots$ . Consider any text  $T$  for  $L_1 \cup \dots \cup L_n$  being given as input. Let  $i = \max(\{i' \mid (\exists x, y) \langle i', x, y \rangle \in \text{content}(T)\})$ . Let  $s'_0, \dots, s'_{i-1}$  be such that, for some  $y$ ,  $\langle i, \langle s'_0, s'_1, \dots, s'_{i-1} \rangle, y \rangle \in \text{content}(T)$ . Note that it must be that  $s'_w = s_w$ . Thus,  $i$  and  $s_0, \dots, s_{i-1}$  can be determined in the limit.

Now the learner (in the limit) outputs the index for set  $S$  defined as follows:

(A) For  $j < i$ ,  $x \leq n - m$ , if  $s_j = 0$  and  $X_{j,s_0,\dots,s_{j-1}}^x \subseteq \text{content}(T)$ , then  $S$  contains a grammar for  $X_{j,s_0,\dots,s_{j-1}}^x$ .

(B) For  $j < i$ ,  $x \leq n - m$ , if  $s_j = 1$  and  $Y_{j,s_0,\dots,s_{j-1}}^x \subseteq \text{content}(T)$ , then  $S$  contains a grammar for  $Y_{j,s_0,\dots,s_{j-1}}^x$ .

(C) For  $j \leq i$ ,  $x < n' - (n - m + 1)$ , if  $Z_{j,s_0,\dots,s_{j-1}}^x \subseteq \text{content}(T)$ , then  $S$  contains a grammar for  $Z_{j,s_0,\dots,s_{j-1}}^x$ .

(D)  $S$  contains a grammar for  $\bigcup_{x \leq n-m} X_{i,s_0,s_1,\dots,s_{i-1}}^x \cap \text{content}(T)$  (assuming this set is non-empty).

It is easy to verify that above method outputs a set of at least  $m$  grammars which partition the input languages (the only case of more than one language in the input being combined to form a single grammar is via case (D) above). Claim follows.  $\square$

There exist values of  $s_0, s_1, \dots$  such that  $\mathcal{L}_{s_0,s_1,\dots} \notin [m', n']\mathbf{TxtBc}$ .

For each  $i \in N$ , define  $s_i$  inductively as follows.

Let  $T$  be a text for  $\bigcup_{x \leq n-m} X_{i,s_0,\dots,s_{i-1}}^x \cup \bigcup_{x < n' - (n - m + 1)} Z_{i,s_0,\dots,s_{i-1}}^x$ .

Note that if  $\mathbf{M}_i$ , on text  $T$ , infinitely often outputs an index for a set which contains a grammar enumerating all of  $\bigcup_{x \leq n-m} X_{i,s_0,\dots,s_{i-1}}^x$ , then  $\mathbf{M}_i$  does not  $[m', n']\mathbf{MultBc}$ -identify the input (since it combines  $n - m + 1$  languages in the same group). In this case one can choose  $s_i$  arbitrarily.

On the other hand, suppose for all but finitely many  $t$ ,  $\mathbf{M}_i(T[t])$  is an index for a set which contains at least two grammars which enumerate part

of  $\bigcup_{x \leq n-m} X_{i,s_0,\dots,s_{i-1}}^x$ . Then, there must exist a  $w \leq 2(n-m)$ , such that  $\langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, w \rangle$  and  $\langle i, \langle s_0, s_1, \dots, s_{i-1} \rangle, w+1 \rangle$  end up being enumerated by different grammars in the index set output by  $\mathbf{M}_i(T[t])$ , for infinitely many  $t$ . Now consider the following cases.

Case 1:  $w$  is even. Let  $s_i = 0$ . In this case one can easily verify that  $\mathbf{M}_i$  does not  $[m', n']\mathbf{MultBc}$ -identify  $\{X_{i,s_0,\dots,s_{i-1}}^x \mid x \leq m-n\} \cup \{Z_{i,s_0,\dots,s_{i-1}}^x \mid x < n' - (n-m+1)\}$ .

Case 2:  $w$  is odd. Let  $s_i = 1$ . In this case one can easily verify that  $\mathbf{M}_i$  does not  $[m', n']\mathbf{MultBc}$ -identify  $\{Y_{i,s_0,\dots,s_{i-1}}^x \mid x \leq m-n\} \cup \{Z_{i,s_0,\dots,s_{i-1}}^x \mid x < n' - (n-m+1)\}$ .  $\square$

Theorem follows from above claims. ■

Following technical proposition is helpful for our results.

**Proposition 8.**  $\forall 1 \leq n, m \leq n, \mathcal{L} \in \mathbf{DisjClass}, \mathcal{L} \in [1, n]\mathbf{MultEx}, 1 \leq m \leq n, L_1, \dots, L_m \in \mathcal{L}, T = L_1 \cup \dots \cup L_m, g_1, \dots, g_{n-m} \in \mathcal{L}, W_{g_i} \in \mathcal{L} - \{L_1, \dots, L_m\}, g = L_1 \cup \dots \cup L_m, \mathcal{L} \in [1, m]\mathbf{MultEx}$

Let  $e_1, \dots, e_{2n-m}$  be such that  $W_{e_1}, \dots, W_{e_{2n-m}}$  are distinct languages in  $\mathcal{L}$ . Let  $\mathbf{M}$  be  $[1, n]\mathbf{MultEx}$  learner for  $\mathcal{L}$ .

Now given any text  $T$  as in the hypothesis, one can effectively search for distinct  $g_1, \dots, g_{n-m}, g'_1, \dots, g'_{n-m} \in \{e_1, \dots, e_{2n-m}\}$  such that

$$[\bigcup_{1 \leq r \leq n-m} (W_{g_r} \cup W'_{g'_r})] \cap \text{content}(T) = \emptyset.$$

Let  $i$  be the grammar to which  $\mathbf{M}$  converges on a text for  $\text{content}(T) \cup \bigcup_{1 \leq r \leq n-m} W_{g_r}$  and  $i'$  be the grammar to which  $\mathbf{M}$  converges on a text for  $\text{content}(T) \cup \bigcup_{1 \leq r \leq n-m} W'_{g'_r}$ .

It is now easy to verify that,  $g_1, \dots, g_{n-m}$  satisfy part (a) of the Proposition, and  $L_1 \cup \dots \cup L_m = W_i \cap W_{i'}$  (which allows us to find  $g$  as required for part (b)). ■

Our next result shows that, for classes in **DisjClass**, reducing the number of languages in the input union and the number of learned groups by the same parameter does not affect **MultEx** and **MultBc**-learnability.

**Theorem 9.**  $\mathcal{L} \in \mathbf{DisjClass}, 1 \leq m \leq n, 1 \leq s \leq n, m \geq 2, \mathcal{L} \in [m, n]\mathbf{MultEx}, \mathcal{L} \in [m-1, n-1]\mathbf{MultEx}, m \geq 2, \mathcal{L} \in [m, n]\mathbf{MultBc}, \mathcal{L} \in [m-1, n-1]\mathbf{MultBc}, \mathcal{L} \in [m, n]\mathbf{MultEx}, \mathcal{L} \in [1, s]\mathbf{MultEx}, \mathcal{L} \in [m, n]\mathbf{MultBc}, \mathcal{L} \in [1, s]\mathbf{MultBc}$

We show part (a). Part (b) can be shown similarly. Part (c) follows from Proposition 8(b), and part (d) can be proved similarly.

(a) Without loss of generality assume  $\mathcal{L}$  is infinite. Suppose  $\mathbf{M} [m, n]\mathbf{MultEx}$ -identifies  $\mathcal{L}$ . Define  $\mathbf{M}'$  as follows.

Suppose a text  $T$  for  $L_1 \cup \dots \cup L_{n-1}$  is given as input. Let  $g_1$  be a grammar such that  $W_{g_1} \cap \text{content}(T) = \emptyset$  and  $W_{g_1} \in \mathcal{L}$ . Let  $g$  be a grammar for  $\text{content}(T)$  (Note that by Proposition 8, one can find such a  $g, g_1$  in the limit).

Let  $T'$  be a text for  $\text{content}(T) \cup W_{g_1}$ . Suppose  $\mathbf{M}(T')$  converges to the index for  $\{i_1, \dots, i_m\}$ . For  $1 \leq r \leq m$ , let  $i'_r$  be a grammar for  $W_{i_r} \cap W_g$ . Then,  $\mathbf{M}'(T)$  converges to the index for  $\{i'_r \mid 1 \leq r \leq m \text{ and } W_{i'_r} \neq \emptyset\}$ . It is easy to verify that  $\mathbf{M}' [m - 1, n - 1]\mathbf{MultEx}$ -identifies  $\mathcal{L}$  (as at most one of  $W_{i'_r}$  is empty). ■

**Corollary 10.**  $\mathcal{L} \in \mathbf{DisjClass} \implies \mathcal{L} \in [m, n]\mathbf{MultEx} \implies \mathcal{L} \in [m', n']\mathbf{MultEx}$   
 $\mathcal{L} \in [m, n]\mathbf{MultBc} \implies \mathcal{L} \in [m', n']\mathbf{MultBc}$

Now we will demonstrate complexity advantages of  $[1, m]\mathbf{MultEx}$ -learnability over  $[m, m]\mathbf{MultEx}$ -learnability (for classes of languages which are learnable under both criteria). First we need a technical proposition.

**Proposition 11.** Let  $S = \{e \mid \text{card}(W_e) \leq e \text{ and } \lim_{t \rightarrow \infty} h(e, t) = 1\}$ .  
 Let  $n \in \mathbb{N}$ . Let  $h = \langle h(e, t) \mid e \in S, t \in \mathbb{N} \rangle$ .  
 Let  $e \in \mathbb{N}$ . Let  $h(e, t) = 1$  for  $t \geq e$  and  $h(e, t) = 0$  for  $t < e$ .

Above proposition can be easily proved using Kleene's recursion theorem [Rog67]. We omit the details.

If  $\mathbf{M}(T[r]) \neq \mathbf{M}(T[r + 1])$ , then we say that  $\mathbf{M}$  made a mind change at  $T[r + 1]$ .

**Theorem 12.** Let  $2 \leq m \in \mathbb{N}$ . Let  $\mathcal{L} \in [m, m]\mathbf{MultEx}$ .  
 Let  $n \in \mathbb{N}$ . Let  $\mathbf{M} [m, m]\mathbf{MultEx}$ -identify  $\mathcal{L}$ .  
 Let  $L_1, \dots, L_m$  be languages such that  $L_1 \cup \dots \cup L_m \in \mathcal{L}$ .  
 Let  $\mathbf{M} [1, m]\mathbf{MultEx}$ -identify  $\mathcal{L}$ .

Let  $\mathcal{L}_e^1 = \{\langle e, 2x \rangle, \langle e, 2x + 1 \rangle \mid x < m\}$ .  
 Let  $\mathcal{L}_e^2 = \{\langle e, 2x + 1 \rangle, \langle e, 2x + 2 \rangle \mid x < m - 1\} \cup \{\langle e, 2m - 1 \rangle, \langle e, 0 \rangle\}$ .  
 Note that  $\bigcup \mathcal{L}_e^1 = \bigcup \mathcal{L}_e^2$ , for all  $e$ .  
 Let  $S = \{e \mid \text{card}(W_e) \leq e, \text{ and } \text{card}(W_e) \text{ is odd}\}$ .  
 Let  $\mathcal{L} = \bigcup_{e \in S} \mathcal{L}_e^1 \cup \bigcup_{e \notin S} \mathcal{L}_e^2$ .

It is easy to verify that  $\mathcal{L} \in [m, m]\mathbf{MultEx}$  (one determines in the limit the  $2m$  elements that constitute the  $m$  input languages, and whether  $e \in S$  or not, for each  $e$  such that, for some  $y$ ,  $\langle e, y \rangle$  belongs to the input text. This information is enough to determine the individual languages which constitute the input).

Furthermore,  $\mathcal{L} \in [1, m]\mathbf{MultEx}$  via a learner which makes no mind changes (one just needs to wait until at least  $2m$  elements appear in the input. At which point the learner can output the grammar which enumerates these  $2m$  elements).

However, a learner which  $[m, m]$ **MultEx**-identifies  $\mathcal{L}$  using at most  $n$  mind changes, also gives us a method to decide  $S$  limit effectively using at most  $n$  mind changes. An impossible task by Proposition 11. ■

### 4 Some Extensions

In this section we consider learnability of unions in groups under additional constraint: the number of languages in learned groups may be limited.

**Definition 13.** (a) We say that  $\mathbf{M} [m, s, n]$ **MultEx**-identifies  $\mathcal{L}$ , iff for all distinct languages  $L_1, \dots, L_n$  in  $\mathcal{L}$ , for all texts  $T$  for  $L_1 \cup \dots \cup L_n$ , there exist  $i_1, \dots, i_k$ , ( $k \geq m$ ) such that  $\mathbf{M}(T)$  converges on  $T$  to the index for the set  $\{i_1, \dots, i_k\}$  and there exists a partition  $G_1, \dots, G_k$  of  $\{1, 2, \dots, n\}$  such that

- (i) each  $G_i$  is non-empty and of size at most  $s$ ,
- (ii) for  $1 \leq j \leq k$ ,  $i_j$  is a grammar for  $\bigcup_{r \in G_j} L_r$ .

(b)  $[m, s, n]$ **MultEx** =  $\{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M} [m, s, n]$ **MultEx**-identifies  $\mathcal{L}]\}$ .

One can similarly define  $[m, s, n]$ **MultBc**. Note that  $[m, n]$ **MultEx** is same as  $[m, s, n]$ **MultEx**, for any  $s \geq n - m + 1$ . Thus, often when  $s \geq n - m + 1$ , we just use  $[m, \infty, n]$ **MultEx** to show that there is no restriction on individual groups except the one forced by values of  $m, n$ .

We first consider some results which follow from the results/proofs of Theorems in the previous section. As a corollary to Theorem 6, we get

**Corollary 14.**  $\dots, \dots, \dots$   $1 \leq n < n'$   $\dots$   $1 \leq m' \leq n'$   $\dots$   $[n, 1, n]$ **MultEx** –  $[m', \infty, n']$ **MultBc**  $\neq \emptyset$

**Corollary 15.**  $\dots, \dots, \dots$   $1 \leq r' < r$   $1 \leq m \leq n - r + 1$   $1 \leq m' \leq n' - r' + 1$   $\dots$   $[m, r, n]$ **MultEx** –  $[m', r', n']$ **MultBc**  $\neq \emptyset$

$\dots, \dots, \dots$  By Proof of Theorem 7, we have that  $[n - r + 1, r, n]$ **MultEx** –  $[m', r', n']$ **MultBc**  $\neq \emptyset$ . Corollary follows. ■

Proof of Theorem 9 essentially shows the following theorem also.

**Theorem 16.**  $\dots, \dots, \dots$   $n - m \leq n' - m'$   $r \leq r'$   $\dots$   $n' \leq n$   $\dots, \dots, \dots$   $\mathcal{L} \in$  **DisjClass**  $\dots$

- $\dots$   $[m, r, n]$ **MultEx**  $\subseteq$   $[m', r', n']$ **MultEx**
- $\dots$   $n' \leq n$   $[m, r, n]$ **MultEx**  $\subseteq$   $[1, n', n']$ **MultEx**

We now give a result which solves the remaining cases for the relationship between different  $[m, r, n]$ **MultEx**-learnability for classes in **DisjClass**. Suppose  $n' \leq n$ ,  $r \leq r'$  and  $n' - m' < n - m$ : since the other cases have been handled in corollaries and theorem above. For classes in **DisjClass**, the next result, Theorem 18, shows when  $[m, r, n]$ **MultEx** can be simulated by  $[m', r', n']$ **MultEx**. This depends on a complex relationship between  $m, r, n$  and  $m', r', n'$ , which we express as the following property.

Intuitively, the property says that if we distribute  $n$  balls in at least  $k \geq m$  non-empty boxes, such that each box has at most  $r$  balls, and then take out  $n - n'$  balls, then at least  $m'$  boxes will remain non-empty.

**Definition 17.** Suppose  $1 \leq m' \leq n'$ ,  $1 \leq m \leq n$ ,  $1 \leq n' \leq n$ ,  $1 \leq r \leq r'$  and  $n' - m' \leq n - m$ .

We say that  $Prop(m, r, n, m', r', n')$  holds iff for any  $k \geq m$ ,  $a_1, \dots, a_k, b_1, \dots, b_k$ , if (A) to (D) hold, then (E) also holds.

- (A)  $\sum_{1 \leq i \leq k} b_i = n - n'$
- (B)  $\sum_{1 \leq i \leq k} a_i = n$ ,
- (C) for  $1 \leq i \leq k$ ,  $b_i \leq a_i \leq r$ ,
- (D) for  $1 \leq i \leq k$ ,  $1 \leq a_i \leq r$ ,
- (E)  $\text{card}(\{i \mid 1 \leq i \leq k, a_i > b_i\}) \geq m'$ .

**Theorem 18.**  $n' \leq n, r \leq r', n' - m' < n - m$   
 $Prop(m, r, n, m', r', n') \iff \mathcal{L} \in \text{DisjClass}$   
 $\mathcal{L} \in [m, r, n]\text{MultEx} \iff \mathcal{L} \in [m', r', n']\text{MultEx}$   
 $Prop(m, r, n, m', r', n') \iff [m, r, n]\text{MultEx} - [m', r', n']\text{MultEx} \neq \emptyset$   
 $\text{DisjClass}$

(a) Suppose  $Prop(m, r, n, m', r', n')$  holds. Let  $\mathbf{M}$  be  $[m, r, n]\text{MultEx}$  learner for  $\mathcal{L} \in \text{DisjClass}$ .

If  $\mathcal{L}$  is finite, then the simulation is trivial. So assume  $\mathcal{L}$  is infinite.  $\mathbf{M}'$  behaves as follows.

Given any text  $T$ ,  $\mathbf{M}'$  finds (i)  $g_1, \dots, g_{n-n'}$  such that  $W_{g_1}, W_{g_2}, \dots, W_{g_{n-n'}} \in \mathcal{L}$  and input text  $T$  does not contain any element of  $\bigcup_{1 \leq i \leq n-n'} W_{g_i}$ , and (ii) a grammar  $g$  for  $\text{content}(T)$  (note that by Proposition 8 this can be done in the limit).  $\mathbf{M}'$  runs  $\mathbf{M}$  on a text  $T'$  for  $\text{content}(T) \cup \bigcup_{1 \leq i \leq n-n'} W_{g_i}$ .

Suppose  $\mathbf{M}$  converges on  $T'$  to the index for set  $\{i'_1, \dots, i'_k\}$ . Then,  $\mathbf{M}'$  forms grammars  $\{i'_w, \dots, i'_k\}$ , such that for  $1 \leq w \leq k$ ,  $W_{i'_w} = W_{i_w} \cap W_g$ . Then,  $\mathbf{M}'$  outputs the index for the set  $\{i'_w \mid 1 \leq w \leq k, W_{i'_w} \neq \emptyset\}$ .

It is straightforward to verify, using the definition of  $Prop$ , that  $\mathbf{M}'$   $[m', r', n']\text{MultEx}$ -identifies  $\mathcal{L}$ .

(b) Suppose  $Prop(m, r, n, m', r', n')$  does not hold. Let  $a_1, \dots, a_k, b_1, \dots, b_k$ , be such that (A) to (D) are satisfied but (E) does not hold in Definition 17.

The diagonalizing class  $\mathcal{L}$  will consist of  $L_{j,i,w}$ , for  $w < a_i, j \in N, 1 \leq i \leq k$ .

$L_{j,i,w}$  will satisfy the following properties. For the following, let  $code_j$  be the index for a set of grammars for the languages in  $\{L_{j',i,w} \mid j' < j, 1 \leq i \leq k, w < a_i\}$ .

- (P1)  $L_{j,i,w} \subseteq \{ \langle j, code_j, i, x \rangle \mid x < 2r \}$ .
- (P2)  $\text{card}(L_{j,i,w}) \geq 2$ .
- (P3)  $L_{j,i,w}$  are disjoint for different values of  $w$ .

We first claim that  $\mathcal{L}$  is in  $[m, r, n]\text{MultEx}$ , irrespective of what the exact chosen  $L_{j,i,w}$  are as long as above properties are satisfied.

On any input text, a learner can first determine the largest  $j$  and corresponding  $code_j$  such that,  $\langle j, code_j, i, x \rangle$  belongs to the input text, for some  $i, x$ . Now the learner can determine (in the limit) grammars for:

(i) any language from  $\mathcal{L}$  which is of form  $L_{j',i',w'}$ , for some  $j' < j$ , and  $L_{j',i',w'} \subseteq \text{content}(T)$  (this can be done using  $code_j$ ).

(ii)  $\text{content}(T) \cap \{\langle j, \text{code}_j, i, x \rangle \mid x < 2r\}$ , for each  $i \in N$ , such that  $\text{content}(T) \cap \{\langle j, \text{code}_j, i, x \rangle \mid x < 2r\} \neq \emptyset$ .

The learner can then, in the limit, converge to the index for set of grammars obtained in (i) and (ii). It immediately follows that the learner  $[m, r, n]\mathbf{MultEx}$ -identifies  $\mathcal{L}$ .

We now show that for appropriate choice of  $L_{j,i,w}$ ,  $\mathcal{L} \notin [m', r', n']\mathbf{MultEx}$ .

Suppose by way of contradiction that, for some  $j$ , one cannot choose appropriate  $L_{j,i,w}$ ,  $1 \leq i \leq k$ ,  $w < a_i$  such that  $\mathbf{M}_j$  fails to  $[m', r', n']\mathbf{MultEx}$ -identify  $\{L_{j,i,w} \mid 1 \leq i \leq k, w < a_i\}$ . Let  $j$  be least such number.

Define  $L_{j,i,w}$  for  $1 \leq i \leq k$ ,  $w < b_i$ , as  $\{\langle j, \text{code}_j, i, 2w \rangle, \langle j, \text{code}_j, i, 2w + 1 \rangle\}$ . Give as input a text  $T$  to  $\mathbf{M}_j$ , where  $\text{content}(T) = \{\langle j, \text{code}_j, i, x \rangle \mid a_i > b_i \text{ and } 2b_i \leq x < 2r\}$ . (Note that  $\text{code}_j$  is determined by languages chosen for  $L_{j',i,w}$  for  $j' < j$ .)

Suppose  $\mathbf{M}_j$  on  $T$  converges to the index set  $\{s_1, \dots, s_{k'}\}$ . Now  $W_{s_w}$ ,  $1 \leq w \leq k'$  must be non-empty, and  $k' \geq m'$  (otherwise clearly, one can choose appropriate  $L_{j,i,w}$  such that  $\mathbf{M}_j$  fails to  $[m', r', n']\mathbf{MultEx}$ -identify  $\{L_{j,i,w} \mid 1 \leq i \leq k, w < a_i\}$ ). Furthermore note that there cannot be an  $i$ ,  $1 \leq i \leq k$ ,  $a_i > b_i$ , such that for two distinct  $w$  and  $w'$ ,  $W_{s_w}, W_{s_{w'}}$  intersect with  $\{\langle j, \text{code}_j, i, x \rangle \mid 2b_i \leq x < 2r\}$  (since otherwise, one may take  $L_{j,i,b_i}$  to contain one element from both  $W_{s_w}, W_{s_{w'}}$ , and other  $L_{j,i,w}$  appropriately, to contradict  $\mathbf{M}_j$   $[m', r', n']\mathbf{MultEx}$ -identifying  $\mathcal{L}$ ). Now for  $i$ ,  $1 \leq i \leq k$ , such that  $a_i > b_i$ , for  $b_i \leq w < a_i$ , define  $L_{j,i,w}$  such that each  $L_{j,i,w}$  contains at least 2 elements and  $\bigcup_{b_i \leq w < a_i} L_{j,i,w} = \{\langle j, \text{code}_j, i, x \rangle \mid 2b_i \leq x < 2r\}$ .

Now each  $i$  in  $\{i \mid a_i > b_i\}$  can be mapped to a  $w$ ,  $1 \leq w \leq k'$  such that  $W_{s_w}$  contains  $\{\langle j, \text{code}_j, i, x \rangle \mid 2b_i \leq x < 2r\}$ . Thus,  $\{i \mid a_i > b_i\} \geq k' \geq m'$ . A contradiction.

This completes the proof of the theorem. ■

**Corollary 19.**  $[m, r, n]\mathbf{MultEx} - [m', \infty, n']\mathbf{MultEx} \neq \emptyset$  if  $n - m > n' - m'$  and  $n' \geq n - m + \lceil \frac{n-m}{r-1} \rceil$ .

Let  $a_1, a_2, \dots, a_m, b_1, \dots, b_m$ , be defined as follows.

$$a_i = r, b_i = 0, \text{ for } 1 \leq i \leq \lfloor \frac{n-m}{r-1} \rfloor.$$

$$a_i = b_i = 1, \text{ for } \lceil \frac{n-m}{r-1} \rceil < i \leq m.$$

If  $\frac{n-m}{r-1}$  is not an integer, then

$$\text{let } a_{\lceil \frac{n-m}{r-1} \rceil} = 1 + (n - m) - [\lfloor \frac{n-m}{r-1} \rfloor * (r - 1)], b_{\lceil \frac{n-m}{r-1} \rceil} = 0.$$

Now, as  $m - m' < n - n'$ , and  $n' \geq n - m + \lceil \frac{n-m}{r-1} \rceil$ , we immediately have that  $\text{Prop}(m, r, n, m', r', n')$  cannot hold (as there are  $< m'$   $w$ 's in  $\{1, \dots, m\}$  such that  $a_w - b_w > 0$ ). Corollary follows. ■

## 5 Conclusions

In this paper we explored relationships between a more coarsened and a more refined classification from the standpoint of computability. Some of our main

results (for example, Theorems 9, 16, and 18(a)) worked on the assumption that underlying targets of classification were pairwise distinct. While it is true for many cognitive classification tasks, there are classification problems where such an assumption cannot be made. For example, when one wants to classify all classical music pieces as being in major or minor, this can probably be done, relatively easily, for everything written within Western musical tradition before the 20th century. However, this has changed by impressionism, introduction of the atonal scale, etc. in the 20th century. Many pieces written by contemporary composers often alternate between major and minor several times, which makes the task of classifying such pieces much harder. To model a situation of this kind, one has to lift the requirement of classification targets being pairwise distinct, or, at least, replace it by a much weaker requirement allowing intersections of classification targets being finite. It would be interesting to explore the issues discussed in the paper in such a setting. We leave it for a future research.

**Acknowledgements.** We thank the anonymous referees of ALT for several helpful comments.

## References

- [Bär74] J. Bārzdīņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.
- [BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
- [Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.
- [CL82] J. Case and C. Lynes. Machine inductive inference and language identification. In M. Nielsen and E. M. Schmidt, editors, *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 107–115. Springer-Verlag, 1982.
- [CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [JNT05] Sanjay Jain, Yen Kaow Ng, and Tiong Seng Tay. Learning languages in a union. 2005. Preliminary version appeared in ALT 2001.
- [MY78] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms*. North Holland, New York, 1978.
- [OSW86] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*. MIT Press, 1986.
- [OW82] D. Osherson and S. Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.
- [Pin79] S. Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.
- [Rog58] H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

- [Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted by MIT Press in 1987.
- [WC80] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition*. MIT Press, 1980.



# Inferring Unions of the Pattern Languages by the Most Fitting Covers

Yen Kaow Ng<sup>1</sup> and Takeshi Shinohara<sup>2</sup>

<sup>1</sup> Kyushu Institute of Technology,  
Graduate School of Computer Science and Systems, Iizuka, 820, Japan  
kalngyk@daisy.ai.kyutech.ac.jp

<sup>2</sup> Kyushu Institute of Technology,  
Department of Artificial Intelligence, Iizuka, 820, Japan  
shino@ai.kyutech.ac.jp

**Abstract.** We are interested in learning unions of the pattern languages in the limit from positive data using strategies that guarantee some form of minimality during the learning process. It is known that for any class of languages with so-called finite elasticity, any learning strategy that finds a language that is minimal with respect to inclusion (MINL) ensures identification in the limit. We consider a learning strategy via another form of minimality — the minimality in the number of elements that are shorter than a specified length. A search for languages with this minimality is possible in many cases, and the search can be adapted to identify any class where every language in the class has a characteristic set within the class. We compare solutions using this strategy to those from MINL to illustrate how we may obtain solutions that fulfill both notions of minimality. Finally, we show how the results are relevant using some subclasses of the pattern languages.

## 1 Introduction

In this paper we study some strategies for learning from positive data [4] the unions of the pattern languages [1] and its subclasses. We are interested in getting hypotheses that not only converge to the presented languages in the limit, but also guarantee other desirable properties, such as some form of minimality.

One such minimality is that with respect to set inclusion. Given a class of languages where the membership is uniformly decidable (called an indexed family of recursive languages [1]), and a property known as *finite elasticity* [17] is fulfilled, every language in the class can be identified in the limit by finding, for any sample received at each stage, a language which (1) includes all the samples, and (2) is minimal with respect to set inclusion among the other languages in the class that contain all the samples. An algorithm that computes (1) and (2) is known as MINL in the literature [1, 14].

The unions of up to a bounded number of the non-erasing pattern languages is particularly interesting in this case because the class has finite elasticity. That is, the ability to compute MINL is sufficient for identifying languages in the class. Arimura *et al.* devised an efficient algorithm (MMG) that computes MINL

for the bounded unions of languages where a condition known as *MINL* can be fulfilled [3]. Sato [13] showed that for the *MINL* problem (that is, patterns where each variable appears at most once), the property can be fulfilled when the number of pattern languages allowed in a union is less than a bound that depends on the alphabet size [13]. In general, the *MINL* problem is difficult to compute, partly due to the difficulty in deciding inclusion between infinite languages [6, 7].

On the other hand, it may be desirable to achieve another form of minimality that is not guaranteed by *MINL*. We illustrate this with an example using the regular pattern languages. Let  $S = \{“cabcc”, “cbcac”\}$ , and consider the language generated by the pattern “\*a\*” (that is, the set of strings obtained by replacing the variables “\*”s with any non-empty string over  $a, b$ , and  $c$ ). Since no other language that includes  $S$  is a proper subset of it, “\*a\*” is minimal with respect to inclusion and hence is a potential output of *MINL*. However, another possible solution of *MINL*, “\*bc\*” generates less number of strings of any given length than “\*a\*”, and may be preferred in some applications [10].

Based on this we propose another approach to minimality. We propose to find, among the hypotheses that are consistent with the given sample, for one that contains the least number of elements of up to a specified length. More precisely, an instance of the problem would ask for a set of languages by specifying the following: (1) a class  $\mathcal{L}$  of languages from which the output is drawn, (2)  $k$ , the number of languages allowed in the output, (3) a set  $S$  of samples that the output languages must together contain, and (4) a length  $\ell$ , where the output must contain the least number of elements of up to length  $\ell$ , among all the sets of languages that fulfill (1)–(3). We denote an instance of the problem specified in such a way  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$ . (The efficient computability of such a problem for some subclasses of the pattern languages has been studied in [11].)

Now let us look at how we may use an algorithm that computes  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$  for learning. First of all, we do not want the learning strategy to favor any particular indexing of the languages, that is, we want to base our conjectures solely upon the output from an algorithm that computes  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$ . When there are more than one solutions for  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$ , we want to treat them as equals. We then consider if it is possible to bound  $\ell$  to some value. We can, perhaps, set  $\ell$  to the length of the longest element in  $S$ . Now suppose there are two languages  $L, L' \in \mathcal{L}$  where  $L$  is finite,  $L \subseteq L'$ , and the two languages contains exactly the same elements of within the length ( $m$  say) of the longest element in  $L$ . In this case, given a text for  $L$  where all its elements are represented, both  $L$  and  $L'$  will be in  $\mathbf{FCP}(\mathcal{L}, L, k, m)$ , and hence the output of  $\mathbf{FCP}(\mathcal{L}, L, k, m)$  alone is insufficient for us to decide on the conjecture  $L$ . However, since this problem does not arise for a class that consists only of infinite languages (where there is no bound to the sample length), in this paper we consider the following two cases separately: (Case 1) allow only infinite languages in the class, and compute  $\mathbf{FCP}(\mathcal{L}, S, k, m)$  where  $m$  is the length of the longest element in  $S$ ; and (Case 2) allow both finite and infinite languages in the class, but allow  $\ell$  to extend indefinitely beyond the lengths of the samples.

We found similar results in both cases: such a learner learns a class of languages if and only if for every language  $L$  in the class, there exists a finite set  $S \subseteq L$  that in some way, witnesses the minimality of  $L$  in the FCP sense. While this may seem exceedingly restrictive, it turns out that any class of languages where each language in it has a *finite witness set* [13] fulfills this requirement. This includes any class with finite elasticity [12, 8], e.g. the class of *regular pattern languages* [17]. Also, even though the full class of the erasing pattern languages does not have finite elasticity, many of its interesting subclasses do, e.g. the *non-erasing regular pattern languages* [14].

We then turn to the question of obtaining unions of the pattern languages that are at the same time minimal with respect to MINL and FCP. We observe that for any  $S$ , if there is a way to decide a length  $n$  for which every union of up to  $k$  languages in  $\{L \in \mathcal{L} \mid S \cap L \neq \emptyset\}$  has a characteristic set consisting of elements of no longer than  $n$ , then every solution in  $\mathbf{FCP}(\mathcal{L}, S, k, n)$  also fulfills the minimality of MINL. This allows us to use previous results by Sato [13] to show that for the non-erasing regular pattern languages, if the number  $k$  of unions allowed is less than  $(|\Sigma| - 1)/2$  where  $|\Sigma|$  is the alphabet size, then for any  $\mathcal{L}$  and  $S$ , any output from  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$  also fulfills the minimality required by MINL, if  $\ell$  is at least as long as the longest element in  $S$ .

Through the same method we also show how solutions that fulfill both MINL and FCP can be obtained for unions of the languages generated respectively by the *regular pattern languages* [15] and *non-erasing regular pattern languages* [5]. Finally, we show that for the class  $\mathcal{L}$  of languages generated by patterns where only one variable occurrence is allowed, any solution for  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$ , where  $\ell$  is at least  $2 \lfloor \frac{k-1}{\text{card}(\Sigma)-1} \rfloor + 1$  plus the length of the longest string in  $S$ , also fulfills the minimality required by MINL.

## 2 Preliminaries

The symbol  $N$  denotes the set of natural numbers. The symbol  $N^+$  denotes the set of positive natural numbers. Cardinality of a set  $S$  is written  $\text{card}(S)$ . A *word* over a non-empty alphabet  $A$  is a finite string of symbols taken from  $A$ . The *empty word* is a null string and denoted by  $\epsilon$ .  $A^*$ ,  $A^+$ , and  $A^{\leq n}$  denote the sets of all the words, non-empty words, and words of length  $n$  or less over  $A$ , respectively.

Let  $\Sigma$  be a finite alphabet with  $|\Sigma| \geq 2$ , and  $V = \{x_1, x_2, \dots\}$  be a countable alphabet disjoint from  $\Sigma$ . Elements in  $\Sigma$  are called *constants* and elements in  $V$  are called *variables*. A non-empty word over  $V \cup \Sigma$  is called a *pattern*, and the length of a pattern  $p$  is denoted  $|p|$ . Each constant or variable has unit length. For a set of patterns  $P$ , the length of the longest word in it is denoted  $\lceil P \rceil$  while the length of the shortest word in it is denoted  $\lfloor P \rfloor$ .

A *substitution* is a homomorphism from patterns to patterns that maps every constant to itself. The image of a pattern  $p$  under a substitution  $\theta$  is written  $p\theta$ . A substitution is *erasing* if it maps some or all variables to the empty word, and it is *non-erasing* otherwise. Unless stated otherwise, substitutions are non-erasing in this paper. We write  $p \preceq q$  iff  $p = q\theta$  for some substitution  $\theta$ . We

write  $p \prec q$  just in case  $p \preceq q$  but not  $q \preceq p$ . Given two sets of patterns  $P$  and  $Q$ , we write  $P \sqsubseteq Q$  iff for each  $p \in P$ ,  $p \preceq q$  for some  $q \in Q$ .  $P \sqsubset Q$  just in case  $P \sqsubseteq Q$  but not  $Q \sqsubseteq P$ .

A  $\langle x_1, \dots, x_n \rangle$  is a pattern where each variable appears at most once [14]. A pattern is a  $\langle x_1, \dots, x_n, s \rangle$  just in case it is of the form “ $x_1 s x_2$ ” and  $s \in \Sigma^+$ . A pattern is a  $\langle x_1 a_1 x_2 a_2 \dots x_n a_n x_{n+1} \rangle$  just in case it is of the form “ $x_1 a_1 x_2 a_2 \dots x_n a_n x_{n+1}$ ” where the subscripted “ $a$ ”s are elements of  $\Sigma$  [5]. We denote the sets of all regular patterns, substring patterns and subsequence patterns by **RP**, **Psub**, and **Psubseq**, respectively.

A  $\langle x_1, \dots, x_n \rangle$  is a set of words over  $\Sigma$ . The  $\langle x_1, \dots, x_n \rangle$  of a pattern  $p$ , written  $L(p)$ , is the set  $\{w \in \Sigma^* \mid w \preceq p\}$ . For a set  $P$  of patterns,  $L(P)$  denotes the language  $\bigcup_{p \in P} L(p)$  and  $\mathcal{L}(P)$  denotes the class of languages  $\{L(p) \mid p \in P\}$ .

### 2.1 Inductive Inference

We present the concepts in inductive inference [4] in this section. A  $\langle x_1, \dots, x_n \rangle$  is an ordered presentation of elements in  $\Sigma^*$  where repetitions are allowed. For each sequence  $\sigma, \langle x_1, \dots, x_n \rangle(\sigma)$  denotes the set of elements that appeared in  $\sigma$ . SEQ denotes the set of all finite sequences. A  $\langle x_1, \dots, x_n \rangle$  of a language  $L$  is an infinite sequence  $w_1, w_2, \dots$  such that  $\{w_i \mid i \in N\} = L$ . The initial segment of length  $n$  of a text  $T$  is denoted  $T[n]$ . An  $\langle x_1, \dots, x_n \rangle$  (IIM) is an algorithmic device which computes a mapping from SEQ into  $N$ . We say that an IIM  $M$   $\langle x_1, \dots, x_n \rangle$  on a text  $T$  to  $i$  just in case there exists  $n \in N^+$  such that for all  $n' \geq n$ ,  $M$  outputs  $i$  on input  $T[n']$ .

A collection of non-empty languages  $\mathcal{L} = \{L_i \mid i \in N\}$  is  $\langle x_1, \dots, x_n \rangle$  just in case there exists a computable function  $f$  such that for each  $i \in N$  and for each  $w \in \Sigma^*$ ,  $f(i, w) = 1$  if  $w \in L_i$  and  $f(i, w) = 0$  otherwise. The family  $L_1, L_2, \dots$  is said to be **TextEx**,  $\langle x_1, \dots, x_n \rangle$  IIM  $M$  just in case for each  $L_i$ , and for each text  $T$  for  $L_i$ ,  $M$  converges on  $T$  to  $j$ , and  $L_j = L_i$ .  $\mathcal{L}$  is  $\langle x_1, \dots, x_n \rangle$  just in case some IIM  $M$  **TextEx**-identifies it. A  $\langle x_1, \dots, x_n \rangle$  for a language  $L$  within a class  $\mathcal{L}$  is a finite set  $S \subseteq L$  such that there exists no language  $L' \in \mathcal{L}$  where  $S \subseteq L' \subset L$ . An indexed family of recursive languages  $\{L_i \mid i \in N\}$  is learnable just in case there exists a procedure, effective in each  $i$ , that enumerates a finite tell-tale for  $L_i$  within the class [2].

A collection of languages  $\mathcal{L}$  has  $\langle x_1, \dots, x_n \rangle$  just in case for each  $w \in \Sigma^*$ ,  $\{L \in \mathcal{L} \mid w \in L\}$  is finite. A collection of languages  $\mathcal{L}$  has  $\langle x_1, \dots, x_n \rangle$  just in case there exists an infinite sequence of pairwise distinct words,  $w_0, w_1, w_2, \dots$  from  $\Sigma^*$ , and an infinite sequence of pairwise distinct languages,  $A_1, A_2, \dots$ , such that for each  $k \in N^+$ ,  $\{w_i \mid i < k\} \subseteq A_k$ , but  $w_k \notin A_k$ .  $\mathcal{L}$  is said to have  $\langle x_1, \dots, x_n \rangle$  just in case  $\mathcal{L}$  does not have infinite elasticity [17, 9]. A  $\langle x_1, \dots, x_n \rangle$   $L$ ,  $\langle x_1, \dots, x_n \rangle$   $\mathcal{L}$  is a finite set  $S_L$  such that for each  $L' \in \mathcal{L}$ ,  $S_L \subseteq L' \Rightarrow L \subseteq L'$ . Sato [12] and Kobayashi [8] independently showed that a class of languages  $\mathcal{L}$  has finite elasticity if and only if every language  $L \subseteq \Sigma^*$  has a characteristic set within  $\mathcal{L}$ .<sup>1</sup>

<sup>1</sup> Note that for a class with finite elasticity, every language (that is, including languages outside of  $\mathcal{L}$ ) has a characteristic set within  $\mathcal{L}$ .

### 3 Fittest Cover Problem

We now formalize our notion of minimality.

**Definition 1.** Let  $\mathcal{L} = \{L_i \mid i \in N^+\}$  be an indexed family of languages,  $S \subseteq \bigcup_{L \in \mathcal{L}} L$  a sample,  $k, \ell \in N^+$  and  $\mathcal{L}' = \{L \in \mathcal{L} \mid L \cap S \neq \emptyset\}$ . The **Fittest Cover Problem**  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$  is to find a set  $P \subseteq \mathcal{L}'$  such that  $S \subseteq \bigcup_{L \in P} L$  and  $|P| \leq k$  and  $\max_{L \in P} |L| \leq \ell$ .

As an example consider the sample  $S = \{\text{“cabcc”}, \text{“cbcac”}\}$ . It is easy to verify that  $L(\text{“}x_1bcx_2\text{”})$  is a solution for  $\mathbf{FCP}(\mathcal{L}(\mathbf{RP}), S, 1, [S])$ , but  $L(\text{“}x_1ax_2\text{”})$  is not. Note that both are solutions for MINL.

For a given class of patterns  $\mathcal{P}$ , we say that a set of patterns  $P \subseteq \mathcal{P}$  is a solution for  $\mathbf{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell)$  just in case  $L(P)$  is a solution for  $\mathbf{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell)$ . That is, we identify a pattern set  $P$  with the language it generates whenever the context is clear.

**Definition 2.** Let  $\mathcal{P}$  be a class of patterns,  $k, \ell \in N^+$  and  $S \subseteq \Sigma^*$ .  $\mathbf{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell) = \{P \subseteq \mathcal{P} \mid P \text{ is a solution for } \mathbf{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell)\}$ .

#### 3.1 FCP as a Strategy for Inductive Inference

An immediate concern regarding the use of such a strategy is that its minimality is partial. That is, while the minimality in MINL is with respect to  $\Sigma^*$ , an FCP solution is only minimal for up to a specified length. In this section we show how the requirement of such a minimality restricts learning. To simplify the discussion in this section we use only  $\mathbf{FCP}(\mathcal{L}, S, 1, \ell)$ , that is, the case without union. This does not make the discussion any less relevant to the study of unions since the finite unions of an indexed family is also an indexed family.

Given an indexed family of recursive languages  $\mathcal{L} = \{L_i \mid i \in N\}$ , and  $\lambda \in N$ , we let  $\mathbf{InferByFCP}(\ell = \lambda)$  be a computation that, upon given each initial segment  $T[n]$  of a text  $T$ , conjectures the language with the smallest index among the solutions for  $\mathbf{FCP}(\mathcal{L}, (T[n]), 1, \lambda)$ . In this paper we study the two cases: (1)  $\mathbf{InferByFCP}(\ell = [S])$ , which uses the solutions for  $\mathbf{FCP}(\mathcal{L}, (T[n]), 1, [S])$ , and (2)  $\mathbf{InferByFCP}(\ell = n)$ , which uses the solutions for  $\mathbf{FCP}(\mathcal{L}, (T[n]), 1, n)$ . We refer to the learner of (1) by  $\mathbf{FCPLearner1}$ , and that of (2) by  $\mathbf{FCPLearner2}$ .

Note that the least index is used in  $\mathbf{InferByFCP}(\ell = \lambda)$  to simplify convergence and its use does not add a dependency on the numberings for the cases we consider (where  $\ell$  extends indefinitely), as the results will show. In the case that  $\mathbf{FCP}(\mathcal{L}, (T[n]), 1, \lambda)$  is not computable due to that  $\mathcal{L}$  is infinite, we may alternatively compute for  $\mathbf{FCP}(\{L_1, L_2, \dots, L_n\}, (T[n]), 1, \lambda)$ . Since the correct language is included in the hypothesis space in the limit, this does not weaken the strength of the learner in terms of **TextEx**-identification.

**Definition 3.** Let  $\mathcal{L} = \{L_i \mid i \in N\}$  be an indexed family of languages,  $L \in \mathcal{L}$  and  $S \subseteq L$ .

witness of minimality for  $L$  within  $\mathcal{L}$   $\ell \in N$   
 $L' \in \mathcal{L} \ S \subseteq L' \Rightarrow \ell' \geq \ell \ (L \leq^{\leq \ell}) \leq (L' \leq^{\leq \ell'})$   
 witness of neighborhood minimality for  $L$  within  $\mathcal{L}$   
 $S' \ S \subseteq S' \subseteq L \ L' \in \mathcal{L} \ S' \subseteq L' \Rightarrow (L \leq^{\lceil S' \rceil}) \leq (L' \leq^{\lceil S' \rceil})$

**Theorem 1. *FCPLearner1 TextEx***

$\mathcal{L} = \{L_i \mid i \in N\}$   $L_i \in \mathcal{L}$

( $\Leftarrow$ ) For  $L_i \in \mathcal{L}$ , let  $T$  be any text for  $L_i$ , and let  $S_i$  be a witness of neighborhood minimality for  $L$  within  $\mathcal{L}$ . Let  $\ell \in N$  and finite  $A_i \subseteq \Sigma^*$  be such that for each language  $L_j \neq L_i$  where  $j < i$ , (1) if  $L_i \subset L_j$ , then  $(L_i \leq^{\leq \ell}) < (L_j \leq^{\leq \ell})$ , (2) otherwise  $A_i$  contains an element in  $L_i - L_j$ . Let  $x_i$  be an element longer than  $\ell$  in  $L_i$ . Let  $n \in N$  (where we furthermore require that  $n \geq i$  if **FCPLearner1** is made to compute **FCP** from the class  $\{L_1, \dots, L_n\}$  instead of  $\mathcal{L}$ ) be such that  $A_i \cup S_i \cup \{x_i\} \subseteq (T[n])$ . (Note that we do not require such  $n$  to be recursively determined.) Then, for all  $T[n']$  where  $n' \geq n$ , in the case that  $j < i$ , either (1)  $L_j = L_i$ , (2)  $(T[n']) \not\subseteq L_j$ , or (3)  $(L_i \leq^{\leq n}) < (L_j \leq^{\leq n})$ ; and in the case that  $j > i$ ,  $(L_i \leq^{\leq n}) \leq (L_j \leq^{\leq n})$ . From this it is not difficult to see that **FCPLearner1** will converge to some  $i' \leq i$  where  $L_{i'} = L_i$  after reading  $T[n]$ .

( $\Rightarrow$ ) Given a language  $L_i$  where no such witness of neighborhood minimality exists, we show how to construct a text  $T$  for  $L_i$  for which **FCPLearner1** fails to converge on to an index for a language for  $L_i$ . At each stage  $l$  of the construction we look for a finite set  $S^l \subseteq L_i$  and sequence  $\sigma^l$  where every  $S^{l+1}$  includes  $L_i \leq^{\lceil S^l \rceil}$  and every sequence  $\sigma^{l+1}$  begins with  $\sigma^l$ . At stage 0, let  $\sigma^0$  be the empty word and let  $S^0 = \emptyset$ . At every stage  $l + 1$ , we find a language  $L_j \neq L_i$  and  $S^{l+1}$  where  $L_i \leq^{\lceil S^l \rceil} \subseteq S^{l+1} \subseteq L_j \cap L_i$  and  $(L_i \leq^{\lceil S^{l+1} \rceil}) > (L_j \leq^{\lceil S^{l+1} \rceil})$ .

This search will always succeed, because otherwise  $L_i \leq^{\lceil S^l \rceil}$  is a witness of neighborhood minimality for  $L_i$  within  $\mathcal{L}$ . Let  $\sigma^{l+1}$  be a sequence (of length  $\geq j$  if **FCPLearner1** is made to compute **FCP** from the class  $\{L_1, \dots, L_n\}$  instead of  $\mathcal{L}$ ) extending  $\sigma^l$  where  $(\sigma^{l+1}) = S^{l+1}$ . Let  $T$  be a text where each initial portion begins with a sequence  $\sigma^l$  for some  $l$ . Since at every stage  $l + 1$ ,  $\sigma^{l+1}$  contains all the elements in  $L_i$  of up to length  $\lceil S^l \rceil$ ,  $T$  is a text for  $L_i$ . However, **FCPLearner1** fails to converge on  $T$  to an index for  $L_i$ . ■

**Corollary 1.**  $\mathcal{L}$  **TextEx**, **FCPLearner1**

Let  $a \in \Sigma$  and let  $a^n$  denote the word in  $\{a\}^*$  of length  $n$ . For each  $i, j \in N^+, j \geq 2$ , let  $L_{ij} = \{a^n \mid n \leq i \vee n \equiv i \pmod{j}\}$ . Let  $\mathcal{L} = \{L_{ij} \mid i, j \in N^+, j \geq 2\}$ . For each  $i, j \in N$ , let  $S_{ij}$  be a finite set including  $\{a^n \mid n \leq i\}$  and  $\{a^{i+j}\}$ . It is easy to verify that  $S_{ij}$  is a finite tell-tale for  $L_{ij}$  and hence  $\mathcal{L}$  is learnable. We now show that there is no witness of neighborhood minimality

for any language in the class. For any  $i, j$ , let  $S$  be any finite subset of  $L_{ij}$ . Let  $i', j' \in N$  be such that: (1)  $i' \geq \max\{i, \lceil S \rceil\}$ , (2)  $i' + j' \equiv i \pmod{j}$ , and (3)  $i + (i' + j' - i)/j > i' + 1$ . Let  $S' = S \cup \{a^{i'+j'}\}$ . By (1),  $S' \subseteq L_{i'j'}$ ; and by (2),  $S' \subseteq L_{ij}$ . However, there are at least  $i + (i' + j' - i)/j (> i' + 1)$  elements in  $L_{ij}^{\leq i'+j'}$ , but only  $i' + 1$  elements in  $L_{i'j'}^{\leq i'+j'}$ . ■

**Theorem 2. *FCPLearner2 Text*,**  $\mathcal{L} = \{L_i \mid i \in N\}$  *FCPLearner2 Text*,  $\mathcal{L}$

Theorem 2 can be shown using methods similar to that in proving Theorem 1.

**Corollary 2.**  $\mathcal{L}$  *FCPLearner2 Text*,  $\mathcal{L}$

**Corollary 3.**  $k \in N$  *FCPLearner2 Text*,  $\mathcal{L}$

For an example of a class learnable by *FCPLearner2* where no language in it has a characteristic set within the class, let  $w_0, w_1, \dots$  be an enumeration of  $\Sigma^*$  and consider the class  $\mathcal{L} = \{\Sigma^* - \{w_i\} \mid i \in N\}$ . Also, the class of unbounded unions of the erasing languages of substring patterns has infinite elasticity, but each language in it has a characteristic set within the class [15].

### 4 Comparing FCP to MINL and MMG

The problems MINL and MMG [3] have been introduced in the past as learning strategies. MINL, as mentioned, aims to find languages that are minimal with respect to set inclusion. MMG, on the other hand, looks for patterns that are minimal with respect to the  $\sqsubseteq$  relation. In this section we see how the FCP problem relates to these problem, being motivated by the possibility of obtaining solutions that fulfill more than one of these notions of minimality. We first formally state the MINL and MMG problems.

**Definition 4.**  $\mathcal{P}$   $S \subseteq \Sigma^*$ ,  $k \in N^+$

**MINL**( $\mathcal{L}(\mathcal{P}), S, k$ ),  $k, P \subseteq \mathcal{P}$ ,  $S \subseteq L(P)$ ,  $k, P' \subseteq \mathcal{P}$ ,  $S \subseteq L(P')$ ,  $L(P') \subset L(P)$

**MMG**( $\mathcal{L}(\mathcal{P}), S, k$ ),  $k, P \subseteq \mathcal{P}$ ,  $S \subseteq L(P)$ ,  $k, P' \subseteq \mathcal{P}$ ,  $S \subseteq L(P')$ ,  $P' \sqsubset P$

$MINL(\mathcal{L}(\mathcal{P}), S, k) = \{P \subseteq \mathcal{P} \mid P, S \subseteq L(P), \text{ and } \nexists P' \subseteq \mathcal{P}, S \subseteq L(P'), L(P') \subset L(P)\}$

$MMG(\mathcal{L}(\mathcal{P}), S, k) = \{P \subseteq \mathcal{P} \mid P, S \subseteq L(P), \text{ and } \nexists P' \subseteq \mathcal{P}, S \subseteq L(P'), P' \sqsubset P\}$

Note that  $\mathcal{FCP}$  and  $\mathcal{MINL}$  are defined to be classes of patterns rather than classes of languages solely to simplify the comparison with  $\mathcal{MMG}$ . The results in this section regarding  $\mathcal{FCP}$  and  $\mathcal{MINL}$  can be easily adapted to remove the dependency on a pattern class  $\mathcal{P}$ .

For any class  $\mathcal{P}$  of patterns, since (from [1])  $(\forall P, Q \subseteq \mathcal{P}) [P \sqsubseteq Q \Rightarrow L(P) \subseteq L(Q)]$ , a solution  $P$  for  $\mathbf{MINL}(\mathcal{L}(\mathcal{P}), S, k)$  is either a solution for  $\mathbf{MMG}(\mathcal{L}(\mathcal{P}), S, k)$ , or there exists some solution  $Q$  for  $\mathbf{MMG}(\mathcal{L}(\mathcal{P}), S, k)$  where  $L(P) = L(Q)$ . On the other hand,  $\mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k) \subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$  if for all  $P, Q \in \mathcal{P}$ ,  $L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q$ , as noted in [3].

The following states necessary and sufficient conditions for the solutions of  $\mathbf{FCP}(\mathcal{L}, S, k, \ell)$  to be solutions for  $\mathbf{MMG}(\mathcal{L}, S, k)$  and  $\mathbf{MINL}(\mathcal{L}, S, k)$ .

- Proposition 1.**  $\forall k, \ell \in \mathbb{N}^+, \mathcal{P} \subseteq \Sigma^*, S \subseteq \Sigma^*$
- 1  $\mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell) \subseteq \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k)$   $\Leftrightarrow \forall P \in \mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell) \exists Q \subseteq \mathcal{P} \text{ s.t. } S \subseteq L(Q) \text{ and } P \sqsubseteq Q \Rightarrow L(Q)^{\leq \ell} - L(P)^{\leq \ell} = \emptyset$
  - 2  $\mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell) \subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$   $\Leftrightarrow \forall P \in \mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell) \exists Q \subseteq \mathcal{P} \text{ s.t. } S \subseteq L(Q) \text{ and } L(P) \subset L(Q) \Rightarrow L(Q)^{\leq \ell} - L(P)^{\leq \ell} \neq \emptyset$

The following result relates the classes  $\mathcal{FCP}$  and  $\mathcal{MINL}$  using the characteristic sets.

- Theorem 3.**  $\forall \mathcal{P} \subseteq \Sigma^*, S \subseteq \Sigma^*, k \in \mathbb{N}^+, \mathcal{P}' = \{p \in \mathcal{P} \mid L(p) \cap S \neq \emptyset\}, \mathcal{L}^k = \{L(P) \mid P \subseteq \mathcal{P}' \wedge |P| \leq k\}, \ell \in \mathbb{N}^+, \mathcal{L}^k \subseteq \Sigma^*, L(P) \in \mathcal{L}^k \Rightarrow S_P \subseteq L(P)^{\leq \ell}, \mathcal{L}^k \subseteq \mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell) \subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$

Let  $P \in \mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \ell)$ . Suppose there exists  $L(Q) \in \mathcal{L}^k$  where  $L(P) \subset L(Q)$  and  $L(Q)^{\leq \ell} - L(P)^{\leq \ell} = \emptyset$ . Let  $S_Q \subseteq L(Q)^{\leq \ell}$  be a characteristic set of  $L(Q)$  within  $\mathcal{L}^k$ . Since  $L(Q)^{\leq \ell} - L(P)^{\leq \ell} = \emptyset$ , we would have  $S_Q \subseteq L(P)$  and hence  $L(Q) \subseteq L(P)$ , a contradiction. Hence no such  $L(Q)$  exists, and therefore by part 2 of Proposition 1,  $P \in \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$ . ■

Hence it suffices that we specify a sufficiently large  $\ell$ , depending only on the maximum length of the elements in the characteristic sets, in the  $\mathbf{FCP}$  problem to obtain  $\mathbf{MINL}$  solutions. A caveat here is that even though a characteristic set is known to exist for every non-erasing pattern language within the class (since the class has finite elasticity), such maximum length of the elements in a characteristic set cannot be recursively determined for every non-erasing pattern language. The reason is that, if such lengths can be decided, then inclusion for any two patterns can be decided — by recursively generating their constituent elements of up to that length. However, Jiang [4] has shown that the problem of inclusion between any two given non-erasing pattern languages is undecidable [7].

Nevertheless, the same problem does not arise for the regular patterns nor its bounded unions. The following result by Sato [5] gives a bound on the



length of such characteristic sets for the bounded unions of the regular pattern languages, under specific conditions.

**Definition 5 ([13]).** For a set of patterns  $P$  and a set  $S_n(p)$  of length  $\leq n$ , the characteristic set  $S_n(P) = \bigcup_{p \in P} S_n(p)$ .

Sato [13] showed that it is possible to use  $S_n$  for some fixed  $n$  as the characteristic sets for some unions of the regular pattern languages [13, 16]. For example, every union of up to  $k$  regular pattern languages  $L(P)$  has  $S_1(P)$  as a characteristic set provided that  $|\Sigma| \geq 2k + 1$  (Theorem 11 in [13]). This gives us the following Corollary.

**Corollary 4.** For  $S \subseteq \Sigma^*$ ,  $k \in \mathbb{N}^+$ ,  $2k + 1 \leq |\Sigma|$ ,  $\mathcal{FCP}(\mathcal{L}(\mathbf{RP}), S, k, \lceil S \rceil) \subseteq \mathcal{MINL}(\mathcal{L}(\mathbf{RP}), S, k)$ .

We first note that given any  $S$ , the set  $\mathcal{P} = \{p \in \mathbf{RP} \mid L(p) \cap S \neq \emptyset\}$  consists only of patterns of length up to  $\lceil S \rceil$ .  $S_1(P)$  where  $P \subseteq \mathcal{P}$  hence contains only elements of up to length  $\lceil S \rceil$ . Theorem 3 and Theorem 11 in [13] then give us the result. ■

#### 4.1 Substring Patterns and Subsequence Patterns

For some subclasses of the regular patterns, stronger results relating FCP to MINL and MMG can be obtained.

**Lemma 1.** For  $|\Sigma| \geq 2$ ,  $\mathcal{P} \in \{\mathbf{Psub}, \mathbf{Psubseq}\}$ ,  $p \in \mathcal{P}$ ,  $w \in L(p)$ ,  $q \in \mathcal{P}$ ,  $p \not\preceq q \implies w \notin L(q)$ .

We first show the case for **Psub**. Let  $p = x_1ux_2$  where  $x_1, x_2 \in V$  and  $u \in \Sigma^+$ . We claim that any word  $w = aub$  for some (possibly equal)  $a, b \in \Sigma$  is not in  $L(q)$  if  $p \not\preceq q$ . Towards a contradiction let  $q = y_1vy_2$  where  $y_1, y_2 \in V$ ,  $v \in \Sigma^+$ , and suppose  $w \in L(q)$ . Since  $w \in L(q)$ ,  $w = aw_1vw_2b$  for some  $w_1, w_2 \in \Sigma^*$ . However, the substitution  $\theta$  where  $y_1\theta = x_1w_1$  and  $y_2\theta = w_2x_2$  then witnesses that  $p \preceq q$ , a contradiction. We now show the case for **Psubseq**. Let  $p = x_1a_1x_2a_2 \dots a_nx_{n+1}$  where  $x_1, \dots, x_{n+1} \in V$  and  $a_1, \dots, a_n \in \Sigma$ . Let  $w = p\theta$  where  $\theta$  is a substitution transforming each  $x_i$  to  $a_i$  for  $1 \leq i \leq n$ , and transforming  $x_{n+1}$  to  $a_n$ . Now for any  $q \in \mathbf{Psubseq}$ , if  $w \preceq q$ , it is not difficult to see that  $q$  must be of the form  $y_1a_{i_1}y_2a_{i_2} \dots a_{i_k}y_{k+1}$  where  $1 \leq i_1 < i_2 < \dots < i_k \leq n$  and  $y_1, \dots, y_{k+1} \in V$ . However, then  $p \preceq q$ . ■

Note that Lemma 1 implies  $L(P) \subseteq L(Q) \implies P \sqsubseteq Q$ . In the following Lemma, for a set of patterns  $P$ ,  $c(P)$  denotes the maximum of  $\{\lceil L(p) \rceil \mid p \in P\}$ .

**Lemma 2.** For  $|\Sigma| \geq 2$ ,  $\mathcal{P} \in \{\mathbf{Psub}, \mathbf{Psubseq}\}$ ,  $P, Q \subseteq \mathcal{P}$ ,  $P \sqsubseteq Q \implies L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} \neq \emptyset$ ,  $L(P) \subset L(Q) \implies L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} \neq \emptyset$ ,  $P \subset Q \iff L(P) \subset L(Q)$ .

We first proof Part (a). Suppose there exists  $P, Q \subseteq \mathcal{P}$  where  $P \sqsubset Q$  and  $L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} = \emptyset$ . Since  $L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} = \emptyset$ , by Lemma 1,  $q \in Q$  implies  $q \preceq p$  for some  $p \in P$ , and hence  $Q \sqsubseteq P$ , a contradiction. For Part (c), that  $P \sqsubset Q \Rightarrow L(P) \subset L(Q)$  is from Part (a) and the fact that  $P \sqsubset Q \Rightarrow L(P) \subseteq L(Q)$ . We now show  $L(P) \subset L(Q) \Rightarrow P \sqsubset Q$ . By Lemma 1,  $L(P) \subset L(Q)$  implies that  $P \sqsubseteq Q$ . If furthermore  $Q \sqsubseteq P$  then  $L(P) = L(Q)$ , contradicting  $L(P) \subset L(Q)$ . Hence we have  $P \sqsubset Q$ . Part (b) follows from (a) and (c). ■

Note also that  $L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q$  is sufficient to show Lemma 2(c).

**Theorem 4.**  $(\Sigma) \geq 2, \mathcal{P} \in \{\mathbf{Psub}, \mathbf{Psubseq}\}, S \subseteq \Sigma^*, k \in N^+, \mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \lceil S \rceil) \subseteq \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k) = \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$

We first note that given any  $S$ , every pattern in  $\mathcal{P}' = \{p \in \mathcal{P} \mid L(p) \cap S \neq \emptyset\}$  must have their shortest element not longer than  $\lceil S \rceil$ . Hence for any  $P, Q \subseteq \mathcal{P}'$ ,  $P \sqsubset Q$  or  $L(P) \subset L(Q)$  (by Lemma 2(a,b))  $\Rightarrow L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} \neq \emptyset \Rightarrow L(Q)^{\leq \lceil S \rceil} - L(P)^{\leq \lceil S \rceil} \neq \emptyset$ . Proposition 1 then gives us  $\mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \lceil S \rceil) \subseteq \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k)$  and  $\mathcal{FCP}(\mathcal{L}(\mathcal{P}), S, k, \lceil S \rceil) \subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$ .

That  $\mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k) = \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k)$  is from Lemma 2(c). ■

In general,  $\mathcal{MINL}(\mathcal{L}, S, k) - \mathcal{FCP}(\mathcal{L}, S, k, \lceil S \rceil)$  is non-empty. For example we have the following cases.

Let  $\Sigma = \{a, b\}$ . (**Psub**) Let  $P = \{“x_1aaax_2”, “x_1bbbx_2”\}, Q = \{“x_1abax_2”, “x_1bbax_2”\}$ , and  $S = L(P) \cap L(Q) \cap \Sigma^{\leq 9}$ . It can be verified that both  $P, Q \in \mathcal{MINL}(\mathcal{L}(\mathbf{Psub}), S, 2)$ , but  $P \notin \mathcal{FCP}(\mathcal{L}(\mathbf{Psub}), S, 2, \lceil S \rceil)$ . (**Psubseq**) Let  $P = \{“x_1ax_2ax_3ax_4”, “x_1bx_2bx_3bx_4”\}, Q = \{“x_1ax_2bx_3ax_4”, “x_1bx_2bx_3ax_4”\}$ , and  $S = L(P) \cap L(Q) \cap \Sigma^{\leq 8}$ . It can be verified that both  $P, Q \in \mathcal{MINL}(\mathcal{L}(\mathbf{Psubseq}), S, 2)$ , but  $P \notin \mathcal{FCP}(\mathcal{L}(\mathbf{Psubseq}), S, 2, \lceil S \rceil)$ .

Nevertheless, on some very specific samples, for **Psub** and **Psubseq**, the FCP problem is no different from the MINL or MMG problem, as the following result shows.

**Proposition 2.**  $(\Sigma) \geq 2, \mathcal{P} \in \{\mathbf{Psub}, \mathbf{Psubseq}\}, P \subseteq \mathcal{P}, S_P \subseteq \Sigma^*, k \geq 1, (P) \subseteq S'_P, S'_P \subseteq L(P), \mathcal{FCP}(\mathcal{L}(\mathcal{P}), S'_P, k) = \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S'_P, k) = \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S'_P, k)$

### 4.2 Patterns with at Most 1 Variable Occurrence

In the subclasses discussed so far, any FCP or MMG solution is at the same time a solution for MINL. We now show a class of pattern languages where an FCP solution is also a solution for MINL, but an MMG solution is not necessarily so.

Let  $\mathbf{P}_{\leq 1}$  be the class of patterns where only one variable occurrence is allowed in each pattern. It is easy to see that the condition  $(\forall P, Q \in \mathbf{P}_{\leq 1})[L(P) \subseteq L(Q)$

$\Rightarrow P \sqsubseteq Q$ ] does not hold when  $k \geq \dots$  ( $\Sigma$ ). As an example, let  $\Sigma = \{a, b\}$ , then  $L("x_1aa") \subset L("ax_1a") \cup L("bx_1a")$ , but neither " $x_1aa$ "  $\preceq$  " $ax_1a$ " nor " $x_1aa$ "  $\preceq$  " $bx_1a$ " holds. Hence a solution of MMG is not necessarily a solution for MINL. In the following, for any rational number  $r$ ,  $\lfloor r \rfloor$  denotes the largest natural number smaller than  $r$ .

**Theorem 5.**  $\mathcal{L}(\mathbf{P}_{\leq 1})^k = \{L(P) \mid P \subseteq \mathbf{P}_{\leq 1} \wedge \dots (P) \leq k\}$   
 $P \subseteq \mathbf{P}_{\leq 1} \ S_t(P)$   $t = 2\lfloor \frac{k-1}{\text{card}(\Sigma)-1} \rfloor + 2$   $L(P)$   
 $\mathcal{L}(\mathbf{P}_{\leq 1})^k$

Let variables  $V = \{x\}$  and constants  $\Sigma = \{a_1, \dots, a_n\}$ . We define two sets of substitutions,

- (1)  $\Theta_1 = \{\theta \mid x\theta = a_i x a_j, 1 \leq i, j \leq n\}$ , and
- (2)  $\Theta_2 = \{\theta \mid x\theta \in \Sigma^{\leq 2}\}$ .

For any set of patterns  $P$  and for any set of substitutions  $\Theta$ , we let  $\Theta \cdot P$  denote the set of patterns  $\{p\theta \mid p \in P \wedge \theta \in \Theta\}$ . Furthermore, for  $l \in N$ , we let  $(\Theta)^l \cdot P$  denote the set of patterns obtainable from  $l$  applications of substitutions from  $\Theta$  on patterns in  $P$ , that is, for example,  $(\Theta)^2 = \Theta \cdot (\Theta \cdot P)$ . Note that  $S_{2l}(P) = \bigcup_{0 \leq i \leq l-1} \Theta_2 \cdot ((\Theta_1)^i \cdot P)$ .

For any  $P \subseteq \mathbf{P}_{\leq 1}$ , and finite  $Q \subseteq \mathbf{P}_{\leq 1}$ ,  $L(P) \subseteq L(Q)$  if and only if there exists  $l \in N$  such that (I)  $(\Theta_1)^l \cdot P \sqsubseteq Q$ , and (II)  $\bigcup_{0 \leq i \leq l-1} \Theta_2 \cdot ((\Theta_1)^i \cdot P) \subseteq L(Q)$ .

( $\Leftarrow$ ) is not difficult to see. We show ( $\Rightarrow$ ) part. We claim that (I) must hold for some  $l \in N$ . Let  $Q = \{u_i x v_i \mid 1 \leq i \leq \dots (Q) \wedge u_i, v_i \in \Sigma^*\}$  be, and let  $m$  be the longest length of such  $u_i$  and  $v_i$ . Now consider  $P' = (\Theta_1)^{m+1} \cdot P$ . Clearly, every pattern in  $P'$  is of the form  $u x v$  where  $u, v \in \Sigma^*$ ,  $|u| > m$  and  $|v| > m$ . Suppose  $P' \not\sqsubseteq Q$ , then there exists  $u x v \in P'$ ,  $u, v \in \Sigma^*$  such that for every  $u_i x v_i \in Q$ ,  $u$  does not begin with  $u_i$  or  $v$  does not end in  $v_i$ , and hence  $u a_1 v \notin L(Q) \Rightarrow L(P) \not\subseteq L(Q)$ , a contradiction. Now note that if (II) fails for any  $l \in N$  then  $L(P) \not\subseteq L(Q)$  also. Claim follows from these observations.  $\square$

The following computation uses the principle in the Claim to decide, given  $p \in \mathbf{P}_{\leq 1}$  and finite  $Q \subseteq \mathbf{P}_{\leq 1}$ , if  $L(p) \subseteq L(Q)$ . At each stage  $l \in N$ , the computation is as follows:

stage 0:  
 If  $p \preceq q$  for some  $q \in Q$ , output true.  
 Otherwise let  $P_1 = \{p\}$  and  $Q_1 = Q$ .  
 Enter stage 1.

stage  $l$ :  
 If  $\Theta_2 \cdot P_l \not\subseteq L(Q_l)$ , output false.  
 If  $\Theta_1 \cdot P_l \sqsubseteq Q_l$ , output true.  
 Otherwise let  
 $P_{l+1} \subseteq \Theta_1 \cdot P_l$  be the set of patterns not refinable from any  $q$  in  $Q_l$ ,  
 $Q_{l+1} \subseteq Q_l$  be the set of patterns  $q \in Q$  where either  
 1.  $q$  contains a variable, but refines to nothing in  $\Theta_1 \cdot P_l$ , or

2.  $q$  contains no variable, and  $|q| \geq |p| + 2l$ .

Enter stage  $l + 1$ .

Note that at any stage  $l$ , patterns in  $Q_l - Q_{l+1}$  are excluded from further evaluation because they do not refine to any pattern in  $\Theta_1 \cdot P_{l'}$  and  $\Theta_2 \cdot P_{l'}$  for any  $l' > l$ . From the computation, we see that at any stage  $l$ , if  $Q_l = \emptyset$ , then either

1.  $P_l = \emptyset$ , in which case  $(\Theta_1)^{l-1} \cdot \{p\} \sqsubseteq Q$  and (by the earlier Claim)  $L(p) \subseteq L(Q)$ , or
2.  $\Theta_2 \cdot P_l \not\subseteq L(Q)$ , hence  $S_{2l}(p) \not\subseteq L(Q)$ .

Hence in the case that  $Q_l = \emptyset$ ,  $S_{2l}(p) \subseteq L(Q) \Leftrightarrow L(p) \subseteq L(Q)$ . We now show that for any  $l > \lfloor (k-1)/(n-1) \rfloor$ ,  $Q_l \neq \emptyset$  implies that  $S_{2l}(p) \not\subseteq L(Q)$ . For each stage  $l$ , let  $Q'_l = Q_l - Q_{l+1}$ , that is,  $Q'_l$  are the patterns removed from  $Q_l$  at stage  $l$ . Now without loss of generality we let each  $q \in Q$  which contains a variable refine to some  $p' \in \Theta_1 \cdot P_m$  for some  $m \in N$ , since otherwise  $L(q) \cap L(p) = \emptyset$ , in which case  $q$  can be discarded from consideration. Hence each  $q \in Q$ , with or without a variable occurrence, is in  $Q'_m$  for some  $m \in N$ .

Now note that for any  $l \in N$ , each  $q \in Q'_m$  refines to at most  $n^{2l-m}$  words of length  $\lceil S_{2l}(p) \rceil$  in  $L(p)$  if  $m \leq l$  and at most  $n^l$  such words in  $L(p)$  if  $m > l$ . However, there are  $n^{2l}$  words in  $L(p)$  of length  $\lceil S_{2l}(p) \rceil$ . Hence, for any  $p \in \mathbf{P}_{\leq 1}$  and  $Q \subseteq \mathbf{P}_{\leq 1}$ , for  $S_{2l}(p) \subseteq L(Q)$  to hold, it is necessary for the inequality

$$\left( \dots (Q) - \sum_{1 \leq m \leq l} \dots (Q'_m) \right) n^l + \sum_{1 \leq m \leq l} \dots (Q'_m) * n^{2l-m} \geq n^{2l} \quad (1)$$

to hold. However, given that

$$\sum_{1 \leq m \leq l} \dots (Q'_m) \leq \dots (Q) \leq k,$$

there are no solution for non-zero  $\dots (Q'_m)$  where  $m > \lfloor (k-1)/(n-1) \rfloor$  that fulfills inequality (1) (via a simpler analysis involving only  $\dots (\Theta_1 \cdot P_m)$  and  $\dots (Q'_m)$ ). That every  $q \in Q$  must be in  $Q'_m$  for some  $m$  implies that  $Q_m = \emptyset$  for  $m > \lfloor (k-1)/(n-1) \rfloor$ .

Let  $l' = \lfloor (k-1)/(n-1) \rfloor + 1$ . In the case that  $Q_{l'} = \emptyset$ , as argued,  $S_{2l'}(p) \subseteq L(Q) \Leftrightarrow L(p) \subseteq L(Q)$ ; and in the case that  $Q_{l'} \neq \emptyset$ ,  $S_{2l'}(p) \not\subseteq L(Q)$ . It follows that for any  $P \subseteq \mathbf{P}_{\leq 1}$ ,  $S_t(P)$  where  $t = 2(\lfloor (k-1)/(n-1) \rfloor + 1)$  is a characteristic set for  $L(P)$  within  $\mathcal{L}(\mathbf{P}_{\leq 1})^k$ . ■

**Corollary 5.**  $\dots S \subseteq \Sigma^* \dots k \in N^+ \mathcal{FCP}(\mathcal{L}(\mathbf{P}_{\leq 1}), S, k, \lceil S \rceil + 2 \lfloor \frac{k-1}{\text{card}(\Sigma)-1} \rfloor + 1) \subseteq \mathcal{MLN}\mathcal{L}(\mathcal{L}(\mathbf{P}_{\leq 1}), S, k)$

While for space reasons we have dealt only with non-erasing pattern languages, most of the results in this paper can be extended to the erasing case.

## 5 Conclusions

We gave learnability results for hypotheses with a form of minimality, and showed how two notions of minimality can be related using the idea of characteristic sets. We gave a result regarding such characteristic sets, namely, that for the bounded unions of  $\mathcal{L}(\mathbf{P}_{\leq 1})$ . We are interested in results which extend this.

## Acknowledgement

We would like to thank the reviewers for corrections and helpful comments. Yen Kaow Ng is supported by the Japanese Government Scholarship of the Ministry of Education, Science, Sports, Culture and Technology of Japan.

## References

- [1] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
- [2] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.
- [3] H. Arimura, T. Shinohara, and S. Otsuki. Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. In *Proc. of the 11th Ann. Symp. on Theoretical Aspects of Comp. Sci. (STACS'94)*, volume 775 of *Lecture Notes in Computer Science*, pages 649–660. Springer-Verlag, 1994.
- [4] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [5] M. Hirao, H. Hoshino, A. Shinohara, M. Takeda, and S. Arikawa. A practical algorithm to find the best subsequence patterns. In *Proceedings of the Third International Conference on Discovery Science*, volume 1967 of *Lecture Notes in Artificial Intelligence*, pages 141–154, 2000.
- [6] T. Jiang, E. Kinber, A. Salomaa, K. Salomaa, and S. Yu. Pattern languages with and without erasing. *International Journal Of Computer Mathematics*, 50:147–163, 1994.
- [7] T. Jiang, A. Salomaa, K. Salomaa, and S. Yu. Decision problems for patterns. *Journal of Computer and System Sciences*, 50:53–63, 1995.
- [8] S. Kobayashi and T. Yokomori. Identifiability of subspaces and homomorphic images of zero-reversible languages. In *Algorithmic Learning Theory: Eighth International Workshop (ALT '97)*, volume 1316 of *Lecture Notes in Artificial Intelligence*, pages 48–61, 1997.
- [9] T. Motoki, T. Shinohara, and K. Wright. The correct definition of finite elasticity: Corrigendum to identification of unions. In L. Valiant and M. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, page 375. Morgan Kaufmann, 1991.
- [10] Y. K. Ng, H. Ono, and T. Shinohara. Measuring over-generalization in the minimal multiple generalizations of biosequences. In *Proceedings of The Eighth Conference on Discovery Science (DS'05)*. to appear.

- [11] H. Ono and Y. K. Ng. Best fitting fixed-length substring patterns for a set of strings. In *Proceedings of The Eleventh International Computing and Combinatorics Conference (COCOON'05)*. to appear.
- [12] M. Sato. Inductive inference of formal languages. *Bulletin of Informatics and Cybernetics*, 27(1):85–106, 1995.
- [13] M. Sato, Y. Mukouchi, and D. Zheng. Characteristic sets for unions of regular pattern languages and compactness. In *Algorithmic Learning Theory: Ninth International Conference (ALT' 98)*, volume 1501 of *Lecture Notes in Computer Science*, pages 220–233. Springer-Verlag, 1998.
- [14] T. Shinohara. Polynomial time inference of extended regular pattern languages. In *RIMS Symposia on Software Science and Engineering, Kyoto, Japan*, volume 147 of *Lecture Notes in Computer Science*, pages 115–127. Springer-Verlag, 1982.
- [15] T. Shinohara and H. Arimura. Inductive inference of unbounded unions of pattern languages from positive data. *Theoretical Computer Science A*, 241:191–209, 2000.
- [16] J. Uemura and M. Sato. Compactness and learning of classes of unions of erasing regular pattern languages. In *Algorithmic Learning Theory: Thirteenth International Conference (ALT' 02)*, vol. 2533, pages 293–307. Springer-Verlag, 2002.
- [17] K. Wright. Identification of unions of languages drawn from an identifiable class. In R. Rivest, D. Haussler, and M. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 328–333. Morgan Kaufmann, 1989.

# Identification in the Limit of Substitutable Context-Free Languages

Alexander Clark<sup>1</sup> and Rémi Eyraud<sup>2</sup>

<sup>1</sup> Department of Computer Science,  
Royal Holloway University of London,  
Egham, Surrey, TW20 0EX, UK  
alexcs@cs.rhul.ac.uk

<http://www.cs.rhul.ac.uk/home/alex/>

<sup>2</sup> EURISE,

23, rue du Docteur Paul Michelon,  
42023 Saint-Étienne Cedex 2, France  
remi.eyraud@univ-st-etienne.fr

<http://eurise.univ-st-etienne.fr/~eyraud/>

**Abstract.** This paper formalises the idea of substitutability introduced by Zellig Harris in the 1950s and makes it the basis for a learning algorithm from positive data only for a subclass of context-free grammars. We show that there is a polynomial characteristic set, and thus prove polynomial identification in the limit of this class. We discuss the relationship of this class of languages to other common classes discussed in grammatical inference. We also discuss modifications to the algorithm that produces a reduction system rather than a context-free grammar, that will be much more compact. We discuss the relationship to Angluin's notion of reversibility for regular languages.

## 1 Introduction

Current techniques for grammatical inference have for a long time been focussed to a great extent on learnable subclasses of regular languages. For many application domains though, there are structural dependencies in the data that are more naturally modelled by context-free grammars of various types. One of the oldest ideas for a grammatical inference algorithm, and one geared towards context-free inference, is Harris's use of substitutability [4, 7]. Though this has formed the intuitive motivation for a number of grammatical inference algorithms before, it has never been adequately formalized. In this paper we present an explicit mathematical formalization of this idea of substitutability and use it to define a subclass of context-free languages that we call the  $\text{substitutable}$  languages, that can be learned according to the polynomial identification in the limit paradigm [5]. These languages are not comparable to the very simple languages, but seem better suited to be the basis for algorithms that can learn natural languages.

In this paper we use a polynomial variant of Gold's identification in the limit paradigm, working from positive data only. We hope in the future to be able to

extend this to a more practical PAC-learning result, but in the meantime work in this paradigm allows some foundational issues to be addressed. The contribution of the work presented in this paper lies in two main directions: first we capture the essential language theoretic property that a certain class of algorithms must rely on, and show that this criterion is sufficient to guarantee identification in the limit.

The key to the Harris approach for learning a language  $L$ , is to look at pairs of strings  $u$  and  $v$  and to see whether they occur in the same contexts; that is to say, to look for pairs of strings of the form  $lur$  and  $lvr$  that are both in  $L$ . This can be taken as evidence that there is a nonterminal symbol that generates both strings. In the informal descriptions of this, there is an ambiguity between two ideas. The first is that they should appear in the same contexts; and the second is that they should appear in the same contexts. We can write the first criterion as follows: (we define our notation more formally in the next section, but we hope the reader will bear with us for the moment)

$$\forall l, r \text{ } lur \in L \text{ if and only if } lvr \in L \tag{1}$$

The second, weaker, criterion is

$$\exists l, r \text{ } lur \in L \text{ and } lvr \in L \tag{2}$$

The problem is then that to draw conclusions about the structure of the language, one needs the former; but all one can hope for by observation of given data is the latter. In general, the class of context-free grammars will be unlearnable: certainly according to the Gold style approach we take in this paper since it is a superfinite class. Therefore to obtain learnability results we must define subclasses of the languages that sufficiently restrict the class so that learning can take place. The restriction we consider here is that whenever two strings have one context in common, then they have all contexts in common: Equation 2 implies Equation 1. We call these the *context-closed* languages.

Our main result is that this simple, but powerful constraint on languages – and note that it is expressed in purely language theoretic terms – sufficiently restricts the class of context-free languages to the extent that it can be learned using a simple polynomial algorithm. In this case, we can learn according to the IIL criterion, and the algorithm will be polynomial in the amount of data it needs (the characteristic set) and in computation.

## 2 Definitions

We start by defining some standard notation.

An alphabet  $\Sigma$  is a finite nonempty set of symbols called letters. A string  $w$  over  $\Sigma$  is a finite sequence  $w = a_1a_2 \dots a_n$  of letters. Let  $|w|$  denote the length of  $w$ . In the following, letters will be indicated by  $a, b, c, \dots$ , strings by  $u, v, \dots, z$ , and the empty string by  $\lambda$ . Let  $\Sigma^*$  be the set of all strings, the free monoid generated by  $\Sigma$ . By a language we mean any subset  $L \subseteq \Sigma^*$ . The set of all



substrings of a language  $L$  is denoted  $Sub(L) = \{u \in \Sigma^+ : \exists l, r, lur \in L\}$  (notice that the empty word does not belong to  $Sub(L)$ ). We shall assume an order  $\prec$  or  $\preceq$  on  $\Sigma$  which we shall extend to  $\Sigma^*$  in the normal way by saying that  $u \prec v$  if  $|u| < |v|$  or  $|u| = |v|$  and  $u$  is lexicographically before  $v$ .

Many classes of languages have been investigated in the literature. In general, the definition of a class  $\mathbb{L}$  relies on a class  $\mathbb{R}$  of abstract machines, here called *representations*, together with a function  $\mathcal{L}$  from representations to languages, that characterize all and only the languages of  $\mathbb{L}$ : (1)  $\forall R \in \mathbb{R}, \mathcal{L}(R) \in \mathbb{L}$  and (2)  $\forall L \in \mathbb{L}, \exists R \in \mathbb{R}$  such that  $\mathcal{L}(R) = L$ . Two representations  $R_1$  and  $R_2$  are *equivalent* if  $\mathcal{L}(R_1) = \mathcal{L}(R_2)$ .

**Definition 1 (grammar).** A grammar  $G = \langle V, \Sigma, P, S \rangle$  consists of a finite alphabet  $\Sigma$ , a finite set of non-terminals  $V$ , a finite set of productions  $P$ , and a start symbol  $S \in V$ . We assume  $P \subseteq V \times (\Sigma \cup V)^+$  and  $S$  does not appear on the right hand side of any production. We write  $T \rightarrow w$  for  $(T, w) \in P$ .

We will write  $uTv \Rightarrow uwv$  when  $T \rightarrow w \in P$ .  $\Rightarrow^*$  is the reflexive and transitive closure of  $\Rightarrow$ .

We denote by  $\mathcal{L}(G) = \{w \in \Sigma^* : S \xRightarrow{*}_G w\}$  the language defined by the grammar. Since we do not allow rules with an empty right hand side this language cannot contain  $\lambda$ .

**Definition 2 (syntactic congruence).**

The syntactic congruence  $\equiv_L$  on  $\Sigma^*$  is defined by  $u \equiv_L v$  if and only if  $w.r.t. \forall l, r \in \Sigma^* lur \in L \iff lvr \in L$ .

We can think of this syntactic congruence as the strong notion of substitutability. Note two things: first this is clearly an equivalence relation, and secondly, it is a congruence of the monoid  $\Sigma^*$  i.e.

$$u \equiv_L v \text{ implies } \forall l, r \text{ } lur \equiv_L lvr$$

The syntactic monoid of the language  $L$  is just the quotient of  $\Sigma^*$  by this relation. It is a standard result that this will be finite if and only if  $L$  is regular.

Another way of looking at this relation is to define the set of contexts of a string:

**Definition 3.** The set of contexts  $C_L(u)$  of a string  $u \in \Sigma^*$  is defined by  $C_L(u) = \{(l, r) | lur \in L\}$ .

Using this definition we can say that  $u \equiv_L v$  if and only if  $C_L(u) = C_L(v)$ .

The weaker idea of substitutability that we will use is defined in the following way.

**Definition 4 (weak substitutability).**

The weak substitutability  $\dot{\equiv}_L$  on  $\Sigma^*$  is defined by  $u \dot{\equiv}_L v$  if and only if  $\forall l, r \in \Sigma^* lur \in L \iff lvr \in L$ .

Note that this is in general not a congruence or even transitive. Normally we will have a finite sample  $S$  of the language  $L$ : clearly  $u \dot{=}_S v$  implies  $u \dot{=}_L v$ .

We now can define the class of languages that we are concerned with:

**Definition 5.** A language  $L$  is *substitutable* if and only if  $u \dot{=}_L v$  implies  $u \equiv_L v$ .

In terms of contexts we can say that a language is substitutable, if whenever the set of contexts of two strings have non-empty intersection, they are identical. The substitutable context-free languages are just those languages that are both substitutable and context-free.

### 2.1 Learning

We now define our learning criterion. This is identification in the limit from positive text [6], with polynomial bounds on data and computation, but not on errors of prediction [5].

A learning algorithm  $A$  for a class of representations  $\mathbb{R}$ , is an algorithm that computes a function from a finite sequence of strings  $s_1, \dots, s_n$  to  $\mathbb{R}$ . We define a presentation of a language  $L$  to be an infinite sequence of elements of  $L$  such that every element of  $L$  occurs at least once. Given a presentation, we can consider the sequence of hypotheses that the algorithm produces, writing  $R_n = A(s_1, \dots, s_n)$  for the  $n$ th such hypothesis.

The algorithm  $A$  is said to identify the class  $\mathbb{R}$  in the limit if for every  $R \in \mathbb{R}$ , for every presentation of  $\mathcal{L}(R)$ , there is an  $N$  such that for all  $n > N$ ,  $R_n = R$  and  $\mathcal{L}(R) = \mathcal{L}(R_n)$ .

We further require that the algorithm needs only polynomially bounded amounts of data and computation. We use the slightly weaker notion defined by de la Higuera [5].

**Definition 6.** An algorithm  $A$  identifies a class  $\mathbb{R}$  in the limit with polynomial bounds if for every  $R \in \mathbb{R}$ , for every presentation  $S$  of  $\mathcal{L}(R)$ , there is an  $m$  such that for all  $n > m$ ,  $R_n = R$  and  $\mathcal{L}(R) = \mathcal{L}(R_n)$ . Moreover, there is a constant  $C$  such that for all  $n$ ,  $|A(S_n)| \leq Cn$ .

## 3 Algorithm

We now define an algorithm *SGL* (substitution graph learner), that will learn a context-free grammar from a sample of positive strings of a language.

The primary data structure of our algorithm can be conceived of as a graph, where each node of the graph corresponds to a substring of a string in the sample,

and where there is an arc between any two nodes corresponding to substrings  $u, v$  if and only if  $u \doteq_S v$  where  $S$  is the set of positive examples.

In the next section we will define a characteristic set of examples for a context-free grammar, and show that whenever the context-free grammar is substitutable, and the sample contains the characteristic set, then  $SGL$  will produce a grammar weakly equivalent (ie. that generates the same language) to the target.

**Definition 7 (substitution graph).**

$$SG(S) = (V, E)$$

$$V = \{u \in \Sigma^+ : \exists l, r \in \Sigma^*, lur \in S\}$$

$$E = \{(u, v) \in \Sigma^+ \times \Sigma^+ : u \doteq_S v\} = \{(u, v) \in \Sigma^+ \times \Sigma^+ : \exists l, r \in \Sigma^*, lur \in S \wedge lwr \in S\}$$

This graph will consist of a number of components, in the usual graph theoretic sense. If the language is substitutable, then every member of the same component will be syntactically congruent, and can thus be freely swapped with each other without altering language membership. Of course in general, there may be more than one component corresponding to the same congruence class, since we are deriving the graph from a small finite sample.

First, note that since syntactic congruence is transitive, and we are interested in substitutable languages, we can compute the transitive closure of the graph, by adding any edges  $(u, w)$  when we have edges  $(u, v), (v, w)$ . We will write  $\cong_S$  for the transitive closure of  $\doteq_S$ . If  $S$  is a subset of a substitutable language  $L$  then  $u \cong_S v$  implies  $u \equiv_L v$ .

We can write  $SG / \cong_S$  for the set of components of the substitution graph and  $[u]_{\cong_S}$  for each element. We will normally omit the subscript where there is no risk of confusion.

**3.1 Constructing the Grammar**

Given the SG we now construct a grammar  $\hat{G} = \langle \Sigma, \hat{V}, \hat{P}, \hat{S} \rangle$ .

We define the set of nonterminals to be the set of components of the substitution graph,  $\hat{V} = SG / \cong_S$ . First note that there will be precisely one component of the substitution graph that will contain all the strings in the sample  $S$ . This is because they will all appear in the empty context  $(\lambda, \lambda)$ . This component will be  $\hat{S}$ .

We now define the set of productions for the grammar. These consist of two types. First for every letter in the alphabet, and we can assume without loss of generality that they occur as substrings in the language, we have a production

$$[a] \rightarrow a$$

Note that if we have two letters such that  $a \doteq b$ , then  $[a] = [b]$  and the same nonterminal will have two productions rewriting it.

The second set of productions is defined for every substring of length greater than 1. For every node in the substitution graph  $u$ , if  $|u| > 1$ , for every pair of non-empty strings  $v, w$  such that  $u = vw$  add a production  $[u] \rightarrow [v][w]$ . Again note that if the component has more than one node in it, then all of the productions will have the same left hand side.

We can define the set of productions formally as:

$$\hat{P} = \{[u] \rightarrow [v][w] \mid u = vw, u \in V \text{ of } SG, |v| > 0, |w| > 0\} \cup \{[a] \rightarrow a \mid a \in \Sigma\}$$

To be explicit about the algorithm, we show it in Algorithm 1, rather than relying on the characteristic set.

---

**Algorithm 1:** SGL algorithm

---

**Data:** A sequence of strings  $s_1, s_2 \dots$

**Result:** A sequence of CFGs  $G_1, G_2 \dots$

$G$  = Grammar generating the empty language ;

**while** ... **do**

    read next string  $s_n$ ;

**if**  $s_n \notin \mathcal{L}(G)$  **then**

        set  $SG$  to be the substitution graph generated from  $\{s_1, \dots, s_n\}$ ;

        set  $G$  to be the grammar generated from  $SG$ ;

**end**

    output  $G$ ;

**end**

---

### 3.2 Examples

... Suppose the sample consists of the two strings  $S = \{a, aa\}$ .  $Sub(S) = \{a, aa\}$ . It is clear that  $a \doteq_S aa$ . Therefore there is only one component in the substitution graph, associated with the nonterminal  $\hat{S}$ . The grammar will thus have productions  $[aa] \rightarrow [a][a]$  which is  $\hat{S} \rightarrow \hat{S}\hat{S}$  and  $[a] \rightarrow a$  which is  $\hat{S} \rightarrow a$ . Thus the learned grammar will generate the language  $a^+$ .

... Consider the language  $L = \{a^n cb^n \mid n \geq 0\}$ . Suppose we have a large sample of strings from this language. The substitution graph will have components  $C_i \subset \{a^n cb^{n+i} \mid n \geq 0\}$  for integer values of  $i$ ,  $A_i = \{a^i\}$  and  $B_i = \{b^i\}$ , for positive values of  $i$ , with  $\hat{S} = C_0$ . The grammar generated from this sample will then have rules of the form (for  $i \geq 0$ )

$$\begin{aligned} C_i &\rightarrow C_j B_{i-j} \\ C_{-i} &\rightarrow A_{i-j} C_j \\ A_{i+j} &\rightarrow A_i A_j, A_1 \rightarrow a \\ B_{i+j} &\rightarrow B_i B_j, B_1 \rightarrow b \end{aligned}$$

Thus, the set of nonterminals can be substantially larger than that of the original grammar.

### 3.3 Polynomial Time

We now show, rather crudely, that SGL runs in a time bounded by a polynomial in the total length of the sample. Suppose the sample is  $S = \{w_1, \dots, w_n\}$ . We can define  $N = \sum |w_i|$ , and  $L = \max |w_i|$ . Clearly  $L \leq N$ , and  $n \leq N$ . The total number of substrings, and thus nodes in the graph, is less than  $N^2$ . The cost of computing, for a given pair of strings  $u, v$ , all of the substrings  $u', v'$  such that  $u' \doteq_S v'$  can be done in time less than  $L^2$ , and thus assuming a constant time map from substrings to nodes in the graph, we can compute all the edges in the graph in time less than  $L^2 n^2$ . Computing the transitive closure of  $\doteq$  or equivalently identifying the components of the substitution graph, can be done in time linear in the sum of the number of nodes and edges which are both polynomially bounded. When constructing the grammar, the number of rules defined by each component/nonterminal is clearly bounded by the number of different ways of splitting the strings in the component, and thus the total number of rules must be bounded by  $LN^2$ , and each rule can be constructed in constant time.

There are much more efficient algorithms that could be used: hashing from contexts to components and using a union-find algorithm to identify the components, for example.

## 4 Proof

**Theorem 1.** *Let  $L_*$  be a target language. Then there exists a sample  $S$  such that  $SGL(S)$  identifies  $L_*$ .*

To prove this theorem, we first need to define a characteristic set, that is to say a subset of a target language  $L_*$  which will ensure the desired algorithm will output a grammar  $G$  such that  $\mathcal{L}(G) = L_*$ .

**Construction of the characteristic sample.** let  $G_* = \langle V, \Sigma, P, S \rangle$  be a target grammar. We are going to define a set  $CS$  of words of  $L_*$ , such that the algorithm  $SGL$  will identify  $L_*$  from any superset of  $CS$ .

We define  $w(\alpha) \in \Sigma^*$  to be the smallest word, according to  $\prec$ , generated by  $\alpha \in (\Sigma \cup V)^+$ . For each nonterminal  $N \in V$  define  $c(N)$  to be the smallest pair of terminal strings  $(l, r)$  (extending  $\prec$  from  $\Sigma^*$  to  $\Sigma^* \times \Sigma^*$ , in some way), such that  $S \xrightarrow{*} lNr$ .

We can now define the characteristic set  $CS = \{lwr | (N \rightarrow \alpha) \in P, (l, r) = c(N), w = w(\alpha)\}$ . The cardinality of this set is at most  $|P|$  which is clearly polynomially bounded.

**Convergence.** We now must show that for any substitutable context-free grammar  $G$ , if  $CS(G) \subseteq S \subseteq \mathcal{L}(G)$  then if  $\hat{G}$  is the output  $SGL$  produces on the sample

$S, \mathcal{L}(\hat{G}) = \mathcal{L}(G)$ . We will start by showing that the grammar derived from the original substitution graph will define the right grammar.

To do this we will show first the following lemma:

**Lemma 1.**  $\mathcal{L}(G) \subseteq \mathcal{L}(\hat{G})$

Proof. By the definition of the characteristic set, for each production  $N \rightarrow \alpha$  in the original grammar, there is a substring  $w(N)$  and a substring  $w(\alpha)$  in the same component of the substitution graph. If  $\alpha = a_1 \dots a_n$  where  $a_i \in \Sigma \cup V$ , then  $w(\alpha) = w(a_1) \dots w(a_n)$ <sup>1</sup>. Therefore by the definition of  $\hat{P}$ , we will have a total of  $n - 1$  productions, one of the form  $[w(\alpha)] \rightarrow [w(a_1)][w(a_2) \dots w(a_n)]$  and  $n - 2$  of the form  $[w(a_i) \dots w(a_n)] \rightarrow [w(a_i)][w(a_{i+1}) \dots w(a_n)]$ . These productions suffice to show that  $[w(N)] \xrightarrow{*}_{\hat{G}} [w(a_1)] \dots [w(a_n)]$ <sup>2</sup>. Given the definition of the start symbols, and the definition of the non-terminal symbols of the form  $[u]$  where  $u \in \Sigma$  (i.e. the pre-terminals), this suffices to show that if  $S \xrightarrow{*}_{\hat{G}} u$  then  $[w(S)] \xrightarrow{*}_{\hat{G}} u$ . Intuitively this is because by the construction of the characteristic set, the set of productions in the hypothesis is going to be a superset of the set of productions in the target, and thus by an induction on the length of the derivation, the defined language will be a superset of the target language. QED.

Next we show that the grammar does not define a language that is too large. Recall that  $w(\alpha)$  for some sequence of non terminals and terminals is the smallest (**w.r.t.**  $\prec$ ) string  $v$  such that  $\alpha \xrightarrow{*} v$ . The basic lemma here is that derivation with respect to  $\hat{G}$  maintains syntactic congruence. Note first that by the construction of the grammar:  $[u] \xrightarrow{*}_{\hat{G}} u$ .

**Lemma 2.**  $\neg, \dots, v \in \Sigma^*, \dots, u \in Sub(S) \quad [u] \xrightarrow{*}_{\hat{G}} v \implies \dots, u \equiv_L v$

Proof. By induction on the maximum length of both derivations  $k$ . Base step:  $k = 1$ . This means the derivation must be a single production of the form  $[u] \rightarrow v$ . This will only be the case if  $|v| = 1$  and  $v$  is in the same component as  $u$ ; therefore  $u \equiv_L v$ .

Inductive step: suppose this is true for all derivations of length less than  $k$ . Suppose we have a derivation of length  $k > 1$ . Suppose we have  $[u] \Rightarrow [v][w] \xrightarrow{*}_{\hat{G}} x$ . There must be strings  $l, r$  such that  $x = lr$  and  $[v] \xrightarrow{*}_{\hat{G}} l$  and  $[w] \xrightarrow{*}_{\hat{G}} r$  with derivations of length less than  $k$ . Therefore by the inductive hypothesis,  $v \equiv_L l$  and  $w \equiv_L r$ . Since we have a production  $[u] \rightarrow [v][w]$  in  $\hat{P}$ , there must be strings  $v', w'$  such that  $v'w'$  is a string in the same component as  $u$ , and  $v' \equiv_L v$  and  $w' \equiv_L w$  and  $u \equiv_L v'w'$ . Since  $\equiv_L$  is a monoid congruence, we have  $u \equiv_L v'w' \equiv_L vw' \equiv_L vw \equiv_L lw \equiv_L lr = x$ . QED

This lemma suffices to establish that  $\mathcal{L}(\hat{G}) \subseteq \mathcal{L}(G)$ , since if  $v$  is in the sample  $S$ , then if  $\hat{S} = [v] \xrightarrow{*}_{\hat{G}} u$ , then  $u \equiv_L v$  implies  $u \in L$ . Therefore  $\mathcal{L}(\hat{G}) = \mathcal{L}(G)$ , and Theorem 1 follows immediately.

<sup>1</sup> This is clear by definition of the partial order  $\prec$  above.

<sup>2</sup> This is just the right binarization of the production. We have not made the assumption that the grammar is in Chomsky Normal Form which would remove the need for this step in the proof.

## 5 Reduction System

As described here the algorithm is not practical, since the number of nonterminals will often become very large. There are a number of algorithms for reducing the number of nonterminals. Clearly one can recursively remove all nonterminals that only have one production by replacing the nonterminal on the left hand side of the production with the right hand side, wherever it occurs. Secondly, one can remove nonterminals, one by one, and test whether the grammar continues to accept all of the sample, and thus arrive at a minimal CFG.

In this section we describe a variant algorithm that is efficient and practical for large data sets, but that produces a reduction system, rather than a grammar.

The key point here is to reduce the substitution graph, by removing strings that are potentially redundant. In particular if we have one component that contains the strings  $u$  and  $v$ , where  $u < v$  and another that contains the strings  $lur$  and  $lvr$ , we can reduce the graph by removing the string  $lvr$ . This is equivalent to reducing the reduction system associated with the graph.

### 5.1 Definitions

We will briefly describe semi-Thue systems or reduction systems [3].

**Definition 8 (Reduction system).**

$$T \subset \Sigma^* \times \Sigma^* \quad T \text{ is a set of pairs } (u, v) \text{ where } u \vdash_T v \text{ and } v < u$$

By extension, we will denote  $lur \vdash lvr$  when  $u \vdash v \in T$ .  $\vdash^*$  is the reflexive and transitive closure of  $\vdash$ .

**Definition 9 (Confluent and weakly confluent reduction system).**

- $w \vdash w1 \quad w \vdash w2 \quad T \quad w, w1, w2 \quad w1 \vdash e \quad w2 \vdash e$
- $w \vdash w1 \quad w \vdash w2 \quad S \quad w, w1, w2 \in S \quad e \in S \quad w1 \vdash^* e \quad w2 \vdash^* e$

Finally a reduction system is **confluent** if there is no infinite sequence of reductions. This defines a congruence relation where  $u$  and  $v$  are congruent if and only if they can be reduced to the same element. Being confluent and Noetherian means that there is a simple algorithm to determine this congruence: each string belong to only one congruence class. If we have the strict requirement that the reductions must be length reducing ( $|v| < |u|$ ), then the maximum number of reductions is the length of the string you start with. Since we have a looser definition ( $v < u$ ), this number can be exponential.

Given a reduction system one can define a language as the union of finitely many congruence classes. Thus given a set of irreducible strings  $A$ , and a reduction system  $T$ , we can define a language  $L(T, A) = \{v : \exists a \in A \wedge v \vdash_T^* a\}$ . These

<sup>3</sup> This differs slightly from the standard definition which requires  $|v| < |u|$ .

are the congruential languages. In some cases, this is a more natural way of defining the structure of a language than systems from the traditional Chomsky hierarchy.

For example consider the reduction system  $T = \{(aca, c), (bcb, c)\}$ , and the axiom  $c$  (i. e. we are looking at the congruence class of  $c$ ). The language defined by  $L(T, \{c\})$  is exactly the palindrome language over  $a, b$  with center marker  $c$ .

### 5.2 Reduction of a Substitution Graph

Given a substitution graph  $SG = \langle V, E \rangle$ , we say that  $SG$  reduces to  $SG' = \langle V', E' \rangle$  if and only if there exists  $(u, v) \in E : v \prec u$ , and  $(l, r), |l| + |r| > 0$ , such that  $lur \in V, V' = (V \setminus \{lur\}) \cup \{lvr\}, E' = \{(x, y) \in V' \times V' : (x, y) \in E \vee ((lur, y) \in E \wedge x = lvr)\}$ .

We say that a substitution graph  $SG$  is *reduced* if there exists no other substitution graph  $SG'$  such that  $SG$  reduces to  $SG'$ .

Given this reduced graph, we define a reduction system directly from the graph.

In this case we will define the set of reductions to be exactly the set of all pairs  $v \vdash u$ , where  $u \prec v$  and  $u, v$  are nodes in the same component of the substitution graph. We can also limit  $u$  to be the unique least node (w.r.t.  $\prec$ ) in each component.

Assuming that we have a set of examples generated from a substitutable CFG that contains the characteristic set, it is easy to prove the following lemmas.

**Lemma 3.**  $N \in V \implies N \xRightarrow{*} u, u \in \Sigma^* \implies u \vdash^* w(N)$

Proof. Suppose  $N = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \dots \Rightarrow \alpha_n = u$  is a derivation of  $u$ . Map this to a sequence  $(w(N), w(\alpha_1), \dots, w(\alpha_n), u)$  of strings from  $\Sigma^*$ . Consider a single step  $\alpha_i = lMr$  and  $\alpha_{i+1} = l\beta r$  and there is a production  $M \rightarrow \beta$  in  $P$ .  $w(\alpha_i) = w(l)w(M)w(r)$  and  $w(\alpha_{i+1}) = w(l)w(\beta)w(r)$ . Therefore  $w(\alpha_i) \vdash_T^* w(\alpha_{i+1})$ . QED.

**Lemma 4.**  $v \vdash u \implies v \in L \implies u \in L$

Proof.  $v \vdash u$  implies  $\exists(x, y) \in P$  and  $l, r \in \Sigma^*$  such that  $v = lxr$  and  $u = lyr$ .  $x \doteq_S y$  implies  $x \doteq_L y$  implies  $x \equiv_L y$  implies  $lxr \in L$  iff  $lyr \in L$ . QED.

The reduction system will be weakly confluent on  $L$ , and it is Noetherian, since the number of strings smaller (w.r.t.  $\prec$ ) than a given string is clearly finite. Unfortunately in general we will not be able to compute an irreducible string for any given word  $u$  in a polynomial (in the size of  $u$ ) number of reductions. Thus though the reduction system itself may be much smaller, in some cases the “parsing” algorithm, determining whether a word is in the language, may be exponential. Subject to this caveat, we can define an efficient, small reduction system that represents the same language, namely the set of all strings that reduces to the least string  $w(S)$  (w.r.t  $\prec$ ) in the language.



## 6 Substitutable Languages

We now give some examples of substitutable CFLs, as well as some simple CFLs that are not substitutable, and discuss the relationship of this class of languages to other standard classes. This is without a doubt a restricted class of languages but contains some interesting examples. They are not closed under any standard operation except reversal.

Since we are learning under a Gold style paradigm, we cannot hope to learn all finite languages [6]. Indeed, the more complex the languages we hope to learn, the smaller the set of finite languages we will be able to learn.

### 6.1 Examples

- $\Sigma^*$  is substitutable
- Any language consisting of only one string is substitutable.
- The finite language  $\{a, aa\}$  is substitutable. The algorithm presented here would return the hypothesis  $\{a^n | n > 0\}$
- $\{a^n | n > 0\}$  is substitutable.
- $\{a^n b^n | n > 0\}$  is substitutable. This is because  $a \doteq aab$ , but they are clearly not syntactically congruent.
- $\{a^n cb^n | n > 0\}$  is substitutable. Here the addition of a center marker removes the problem.
- $\{w c w^R | w \in (a, b)^*\}$  (the palindrome with center marker) is substitutable.
- Strictly deterministic regular languages [13] are substitutable. Since the automaton is forward and backwards deterministic, and any given string can only be generated by a unique sequence of states, we can see easily that if  $u \doteq v$  then the sequence of states that generates  $u$  must start and stop in exactly the same state that  $v$  starts and stops in.

Recall that very simple grammars [14] consist of CFGs in Greibach normal form such that no terminal symbol is used in more than one production. Some very simple grammars are not substitutable: an example is the grammar with productions  $S \rightarrow bN, S \rightarrow aNP, N \rightarrow xM, N \rightarrow n, P \rightarrow rMP, P \rightarrow p, M \rightarrow m$ . This generates the language  $bn, bxm, anp, axmp, anrmp, \dots$ . We can see that  $x \doteq nr$  but it is not the case that  $x \equiv nr$ , since  $bxm$  is in the language but  $bnrm$  is not. Nonetheless we note that the three grammars in [14] Example 2 are all substitutable languages.

We also note the relationship to NTS grammars[11]; which can be seen to be relevant in the next section. NTS grammars have the property that if  $N \xrightarrow{*} v$  and  $M \xrightarrow{*} uvw$  then  $M \xrightarrow{*} uNw$ . We conjecture that all substitutable languages are NTS languages.

### 6.2 Relation to Other Language Classes

Substitutable context-free languages are properly included within the class of congruential languages [3]. They are incomparable with the classes of finite languages, regular languages, and very simple languages. It properly includes the class of strictly deterministic regular languages.

## 7 Discussion

This work is related to two other strands of work. First work that proves polynomial IIL of other subclasses of context-free grammars. In [14], Yokomori shows that the class of very simple languages can be polynomially identified in the limit. Unfortunately the complexity is  $N^{|\Sigma|+1}$  and the alphabet size is equal to the number of productions in a very simple grammar, so this algorithm is not practical for large scale problems. Secondly, we can relate it to the work of Adriaans [1], who uses a similar heuristic to identify languages. Finally, we can mention the similar work of [12] who shows an identification in the limit result of a class of grammars called “left-aligned R grammars”. This work defines a rather complicated family of grammars, and shows how constituents can be identified. We also note [8] who show a learnable subclass of CFGs.

We can compare substitutability with reversibility [2, 9]. Recall that a language is reversible if whenever  $uw$  and  $vw$  are in the language then  $ux$  is in the language if and only if  $vx$  is in the language. Thus reversibility is the exact analogue of substitutability for regular languages. Note that reversibility is a weaker criterion than substitutability. Substitutability implies reversibility, but not vice versa, as can be seen from the language  $\{ab, bb\}$  which is reversible but not substitutable.

We can also compare the substitutability to  $\mu$ -distinguishability for inference of regular languages [10]. Ron uses a measure of similarity of residual languages, rather than of contexts as we use here. Considered in this way, our measure is very crude, and brittle – contexts are equal if they have non empty intersection. Nonetheless the techniques of Ron et al., suggest a way that this technique could be extended to a PAC-learning result, using a bound on a statistical property of the distribution. There are some technical problems to be overcome, since the number of syntactic congruence classes will be infinite for non regular languages, and thus the distinguishability will not in general be bounded from below. A more serious problem is that the worst case sample complexity, if the data is drawn randomly, is clearly exponential, since the chance of getting two strings that differ only in a single point is in general exponential in the derivational entropy of the grammar.

Algorithms for learning regular languages focus on identifying the states of a deterministic automaton. When trying to move to learning context-free languages, the obvious way is to try to identify configurations (i.e. pairs of states and strings of stack symbols) of a deterministic push down automaton. A problem here is that the structure of this set depends on the representation, the automaton. One way of viewing the work presented in this paper, is to say that a better way is to try to identify the elements of the syntactic monoid. This monoid represents in the barest form the combinatorial structure of the language. From a learnability point of view this is interesting because it is purely syntactic – it is not semantic as it does not depend on the representation of the language but only on the language itself. Since we are interested in algorithms that learn from unstructured data – strings from the language that are not annotated with structural information – this seems a more natural approach.

Importantly, our algorithm does not rely on identifying constituents: that is to say on identifying which substrings have been generated by the non terminals of the target,  $\dots$ . This has up to now been considered the central problem in context-free grammatical inference, though it is in some sense an ill-posed problem since there may be many different grammars with different constituent structure that are nonetheless weakly equivalent, that is to say, define the same language.

One of the weaknesses in the work is the fact that we do not yet have a grammatical characterisation of substitutability, nor an algorithm for determining whether a grammar defines a substitutable language. It is clear from standard results in the field that this property will be undecidable in general, but it might be possible to decide it for NTS grammars [11].

Looking at our approach more generally, it is based on identifying syntactically congruent substrings. Substitutable languages have a property that allows a trivial procedure for determining when two substrings are congruent, but it is easy to think of much more robust methods of determining this that rely on more complex properties of the context distributions. Thus in principle we can use any property of the sample from the context distribution: average length, substring counts, marginal distributions at certain offsets and so on.

To conclude, we have shown how a simple formalization of Harris's substitutability criterion can be used to polynomially learn an interesting subclass of context-free languages.

## Acknowledgements

This work has benefitted from the support of the EU funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning. We would like to thank Colin de la Higuera, Jean-Christophe Janodet and Brad Starkie for helpful comments and discussions.

## References

- [1] P. Adriaans, M. Trautwein, and M. Vervoort. Towards high speed grammar induction on large text corpora. In *SOFSEM 2000*, pages 173–186. Springer Verlag, 2000.
- [2] D. Angluin. Inference of reversible languages. *Communications of the ACM*, 29:741–765, 1982.
- [3] R. Book and F. Otto. *String rewriting systems*. Springer Verlag, 1993.
- [4] N. Chomsky. Systems of syntactic analysis. *Journal of Symbolic Logic*, 18(3), 1953.
- [5] C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997.
- [6] E. M. Gold. Language indentification in the limit. *Information and Control*, 10(5):447 – 474, 1967.
- [7] Z. Harris. Distributional structure. *Word*, 10(2-3):146–62, 1954.

- [8] J. A. Laxminarayana and G. Nagaraja. Inference of a subclass of context free grammars using positive samples. In *Proceedings of the Workshop on Learning Context-Free Grammars at ECML/PKDD 2003*, 2003.
- [9] E. Mäkinen. On inferring zero-reversible languages. Technical Report A-1998-7, University of Tampere, 1998.
- [10] D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences (JCSS)*, 56(2):133–152, 1998.
- [11] G. Senizergues. The equivalence and inclusion problems for NTS languages. *J. Comput. Syst. Sci.*, 31(3):303–331, 1985.
- [12] B. Starkie. *Identifying Languages in the Limit using Alignment Based Learning*. PhD thesis, University of Newcastle, Australia, 2004.
- [13] T. Yokomori. On polynomial-time learnability in the limit of strictly deterministic automata. *Machine Learning*, 19(2):153–179, 1995.
- [14] T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 298(1):179–206, 2003.

# Algorithms for Learning Regular Expressions

(Extended Abstract)

Henning Fernau<sup>1,2,\*</sup>

<sup>1</sup> University of Hertfordshire, College Lane, Hatfield, Herts AL10 9AB, UK

<sup>2</sup> Wilhelm-Schickard-Institut für Informatik, Universität Tübingen,  
Sand 13, D-72076 Tübingen, Germany

henningfernau@yahoo.de

**Abstract.** We describe algorithms that *directly* infer regular expressions from positive data and characterize the regular language classes that can be learned this way.

## 1 Introduction

Over the last about forty years, many algorithms have been proposed and implemented that are capable to infer a regular language, given only a finite number of samples of the language. Probably most of them, especially the ones in actual use, are mere heuristics in the sense that there does not exist any concise characterization of the class of languages that can be learned in this way.

In more mathematical terms, the “generalization capability” sketched in the preceding paragraph is best captured by Gold’s learning paradigm of *learning in the limit* as proposed by Gold [10]. In this well-established model, a language class  $\mathcal{L}$  (defined via a class of language describing devices  $\mathcal{D}$  as, e.g., grammars or automata) is said to be *learnable in the limit* if there is a so-called *learner*  $I$  to which as input an arbitrary language  $L \in \mathcal{L}$  may be enumerated (possibly with repetitions) in an arbitrary order, i.e.,  $I$  receives an infinite input stream of words  $E(1), E(2), \dots$ , where  $E : \mathbb{N} \rightarrow L$  is an enumeration of  $L$ , i.e., a surjection, and  $I$  reacts with an output device stream  $D_i \in \mathcal{D}$  such that there is an  $N(E)$  so that, for all  $n \geq N(E)$ , we have  $D_n = D_{N(E)}$  and, moreover, the language defined by  $D_{N(E)}$  equals  $L$ .

In practical applications of this learning scenario, learning regular languages often means to infer regular expressions (REs), because REs are arguably the more suitable model to specify regular languages, especially for human beings. Therefore they (or variants thereof) are probably used in well-known tools as

---

\* Most of the work on this project has been done while the author was with The University of Newcastle, School of Electrical Engineering and Computer Science, University Drive, NSW 2308 Callaghan, Australia; the assistance by a New Staff grant from the University of Newcastle that made possible to employ Linda Buisman, aka Postniece, to implement the algorithms described in this paper, is gratefully acknowledged.

`grep` in UNIX. Unfortunately, to our knowledge all learning algorithms with known characterizations of the learned language class are based on grammars and/or automata.<sup>1</sup> The disadvantage is that when you finally turn your results into REs (as the “target structure” towards human beings) by standard algorithms as contained in any introductory book to automata theory (e.g., [11]), these are often quite clumsy and lengthy, containing lots of “nested loops” and seeming repetitions of subsequences. Quite easily a human who wants to somehow check the outcome of the automatic learning procedure is simply abhorred by such output. We made some computer experiments in the context of inferring DTDs for XML documents with the aid of known DFA learners that verified these considerations [8]. Hence, although the (syntactic) task of DTD inference basically boils down to learning regular expressions, most systems that are actually used for this purpose rely on heuristics, see [9] for the description of one such system, as well as further references.

In order to improve on the readability of the resulting REs and probably also giving a better intuitive “explanation” of the observed data (sample strings), we are going to propose a learning procedure which is only generating REs of star height one, i.e., starred expressions get never nested. Although the corresponding class of languages is that restricted, many cases that occur in practice are covered.

## 2 Blockwise Grouping and Alignment

The basic technique we are using can be best described as follows. This means the following: Given some sample strings, say

$$ababb, aabb, ababa, abc,$$

we would first transform them into

$$[a][b][a][bb], [aa][bb], [a][b][a][b][a], [a][b][c],$$

where we use square brackets to denote the “block letters.” Actually, and more technically, we will call any  $[x^n]$  a block letter whenever  $x$  is a character in the alphabet over which the input words are formed.  $x$  is called the basic letter of  $[x^n]$ . When we only use block letters  $[x]$  representing  $x^n$  for any  $n \geq 1$ , we also say that we use block letters. In our transformation, we also require that the basic letters of neighboring block letters are different. At first glance, this appears to mean that we are dealing with infinite alphabets. But since we will only use block letters as derived from input samples, there are at any moment only finitely many of them, so that they can be conveniently handled.

In a second step, we try to align the blocks from left to right:

---

<sup>1</sup> There are even only few learning algorithms that directly deal with regular expressions, see [4, 5, 12] and the literature quoted therein, but they are not addressing the text learning model we are using here.

```

[a] [b] [a] [bb]
[aa] [bb]
[a] [b] [a] [b] [a]
[a] [b] [c]

```

As an aside, let us mention that committing and restricting ourselves to leftmost alignments seems to be quite suited to what humans are doing in such a case, as well. Namely, when given the task of describing the common nature of the given four strings, most people I dare to claim would describe this nature along the following lines: “each string starts with one or two *a*, then there are one or two *b*, possibly followed by...” But there other obvious alternatives, as well, e.g.: “each string starts with *ab*, followed by one or two *a*, then one or two *b*, and finally (optionally) one *a* or one *c*.” Observe that in this second description, the second and fourth sample was aligned starting with the third block of the other two samples, i.e.:

```

[a] [b] [a] [bb]
      [aa] [bb]
[a] [b] [a] [b] [a]
      [a] [b] [c]

```

Due to the above “psychologic” argument and because the general problem of finding “best alignments” is NP hard (corresponding to the multiple sequence alignment problem), we will restrict our attention to leftmost alignments in what follows.

So, sticking to our first “generalization” we would end up with the language describable with the following RE:

$$(a|aa)(b|bb)(\lambda|a(b|bb)(\lambda|a)|c).$$

Since this is still denoting a finite language, we might wish to further generalize. Then, it is quite natural to consider “one or two repetitions” as a (very) special case of “one or more repetitions”. Having this in mind, we might then arrive at:

$$a^+b^+(\lambda|ab^+(\lambda|a)|c),$$

which, moreover, gives a shorter “explanation” of the given “observations” and is hence preferable by the “minimum description length” principle, likewise known as “Occam’s razor,” see [9, 13, 14] for a discussion with respect to Grammatical Inference.

Note that we were writing down Kleene plus rather than Kleene star operations to be sure not to destroy the “blockwise readings” we originally provided.

There is still one issue we did not discuss up to now: how should we, in general terms, arrive at a suitable “explanation” when facing the block letters  $[x^{k_1}], \dots, [x^{k_j}]$  within one block (e.g., assuming that  $\{x^{k_1}, \dots, x^{k_j}\}$  was the given input sample)? Without loss of generality, assume that  $0 < k_1 < k_2 < \dots < k_j$ . Of course, we could generalize every time directly to  $x^+$ , but this might override to quickly the “multiplicity information” contained in the sample. We will instead choose integer numbers  $\ell \geq 0$  and  $r \geq 1$  such that, for each  $1 \leq i \leq j$ ,  $r$  divides

$k_i - \ell$ , and such that  $r$  is maximal with this property. So,  $r = 1$  and  $\ell = 0$  will always satisfy the first part of the property (and this would correspond to the generalization  $x^+$  mentioned above), but larger  $r$  will reveal more information within the loop. More concretely, if  $k_i = 2i + 1$ , i.e., given  $x^3, x^5, x^7, \dots$ , the “explanation”  $x(xx)^+$  might look better than simply guessing at  $x^+$ . In other words, we would have taken  $\ell = 1$  and  $r = 2$  in that case.

On the downside, let us mention that this way only so-called strictly bounded languages can be derived, i.e., subsets of  $x_1^*x_2^*\dots x_k^*$ , where the  $x_i$  are characters of the original alphabet. Especially, this means that the universal language  $\Sigma^*$  cannot be derived for any input alphabet  $\Sigma$  with more than one letter. We will therefore also discuss strategies how to overcome this sort of dilemma, see Sec. 4.

### 3 Working out Details of RE Learning Strategies

We start analyzing a simplified version of our learning strategy.

#### 3.1 Creating the Start Tree

This algorithm can be explained as follows for a basic alphabet  $\Sigma$  and an input sample  $I_+ \subset \Sigma^+$ , yielding the recursive procedure `create-tree(I)`, which initially takes as input  $I$  the sample  $I_+$ .

`create-tree(I)`

1. Left-align the input words.  
 Due to the recursion within this procedure, aligning according to the first symbols is sufficient, so that this step can be very efficiently implemented.
2. This gives, for each symbol  $a \in \Sigma$ , the set

$$I(a) = \{w \in I \mid w \text{ starts with } a\}.$$

This yields a partition of  $I$  into the different sets  $I(a)$ .

3. For each  $I(a)$ , form the set

$$I^a = \{v \in b\Sigma^* \mid b \neq a \wedge \exists n > 0 (a^n v \in I(a))\}.$$

4. Recursively call, for all  $a \in \Sigma$ , `create-tree(Ia)`.

Let us introduce some further notations: Given a set  $I \subset \Sigma^+$ , consider

$$I(a) = \{a^{n_1}v_1, a^{n_2}v_2, \dots, a^{n_m}v_m\}$$

with  $v_i \in (\Sigma \setminus \{a\})\Sigma^* \cup \{\lambda\}$  and  $n_j \leq n_{j+1}$ ; then, the  $m = |I(a)|$ -tuple  $S(I(a)) = (n_1, n_2, \dots, n_m)$  is also called the  $a$ -*signature* of  $I$ .

Further denotations are:  $\alpha(S(I(a))) = n_1$ ,  $\omega(S(I(a))) = n_m$ .

The set  $J^b$  with  $J = I^a$  is also denoted  $I^{ab}$ . In accordance,  $I^\lambda = I$ .

It is obvious that we can associate to  $I_+$  a rooted labeled directed tree, the root being labeled  $I_+ = I_+^\lambda$ , and the children of a node labeled  $I_+^x$  being



labeled with the  $I_+^{xa}$  for  $a \in \Sigma$  if  $I_+^x(a)$  is non-empty. The arc from  $I_+^x$  to  $I_+^{xa}$  is then labeled  $a$ . This tree is in direct correspondence with the recursion tree of `create-tree`( $I_+$ ) and will be also called the *prefix block tree* for  $I_+$ . Sometimes, it is more convenient to label non-root vertices of this tree  $I^x(a)$  instead of  $I^{xa}$ . Then, the root will be labeled  $r$ .

Getting back to our start example, we illustrate these notions.

... We take

$$I_+ = \{ababb, aabb, ababa, abc\} \subset \Sigma^+ = \{a, b, c\}^+$$

Hence,  $I_+(a) = I_+$ , and moreover,

$$I_+^a = \{babb, bb, baba, bc\}.$$

Since  $I_+^a(b) = I_+^a$ , we get

$$I_+^{ab} = \{abb, aba, c\}.$$

Note that  $bb \in I_+^a(b)$  did not leave any trace in  $I_+^{ab}$ . Now,  $I_+^{ab}(a) = \{abb, aba\}$  and  $I_+^{ab}(c) = \{c\}$ . Since  $I_+^{abc} = \emptyset$ , we continue with aligning  $I_+^{aba} = \{bb, ba\}$ , finally giving  $I_+^{abab} = \{a\}$ .

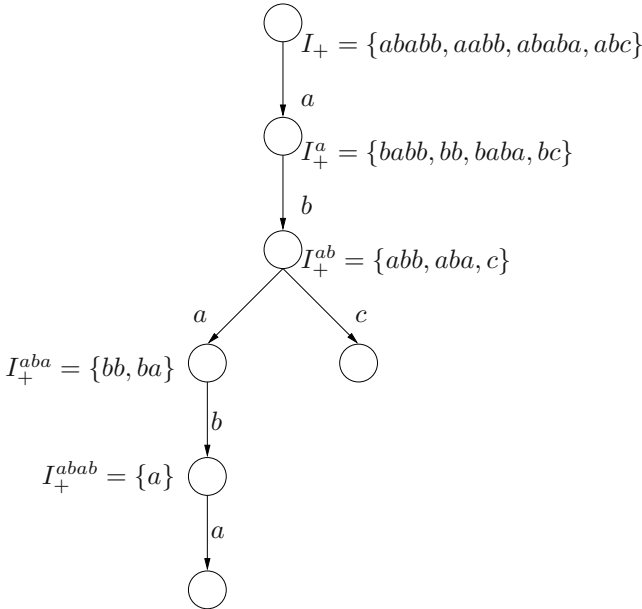
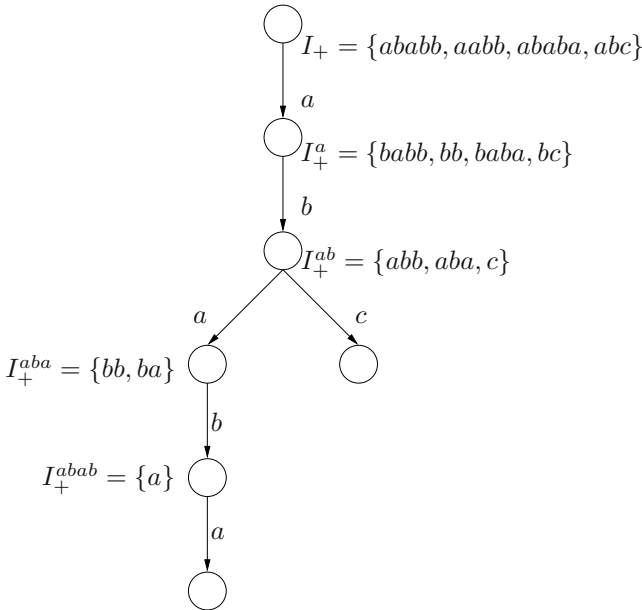


Fig. 1. A simple prefix block tree

In our example, the nodes of the start tree are labeled  $I_+$ ,  $I_+^a$ ,  $I_+^{ab}$ ,  $I_+^{aba}$ ,  $I_+^{abc}$ ,  $I_+^{abab}$ , and  $I_+^{ababa}$ . Leaves in this tree are always labeled with the empty set. This gives the picture of Fig. 1. The alternative node labeling convention is displayed in Fig. 2.

### 3.2 Constructing NFAs by Alignment

Recall now the notion of a  $I^x(a)$  start tree. The arcs of such an NFA may be labeled with whole words (or even finite set of words), not just single symbols. By way of contrast, a  $I^x(a)$  is a generalized NFA which maintains the “determinism” property; therefore, we require that the labels of the arcs emanating from an arbitrary node form a prefix code.



**Fig. 2.** The resulting NFA

It is now easy to turn the start tree into a (generalized) NFA: If a node is carrying the label  $I^x(a)$ , then label the arc ending in  $I^x(a)$  with the finite set  $\{a^j \mid j \in S(I^x(a))\}$ . The resulting NFA in our example is displayed in Fig. 2. The root obviously becomes the start state, and the terminal states are (as usual) indicated by double-circles. Observe that this leads to some generalization (e.g., now the word  $ab \notin I_+$  will be accepted), according to the following  $I^x(a)$  algorithms [15]:

Aligned pieces of words are considered to be interchangeable.

However, this kind of generalization is very cautious, since it will always create only finite languages, given a finite sample. In order to arrive at infinite languages, loops must be introduced.

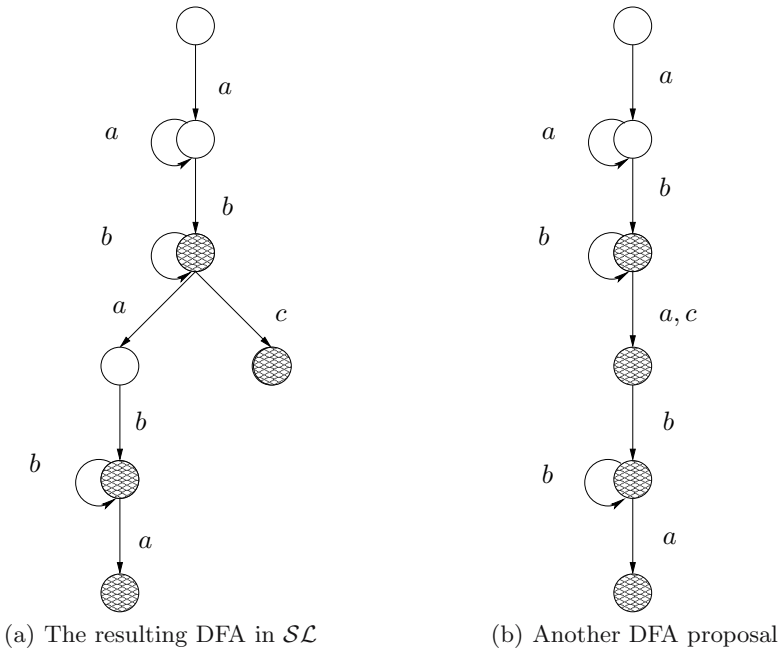


Fig. 3. Two ways to generalize

### 3.3 Introducing Loops by Determinization

We will introduce loops by turning the (generalized) NFA we obtained so far into a (generalized) DFA. To this end, if an arc  $A$  in the NFA is labeled with a finite set of words, say  $\{a^{n_1}, \dots, a^{n_m}\}$  containing more than one word, (i.e.,  $m > 1$ ) then only keep the shortest among these words as a label of  $A$  and enable the generation of the longer words by introducing a loop labeled  $a$  at the node  $A$  points to. The resulting generalized automaton is indeed deterministic, since in the situation sketched above, in the original generalized NFA there was no arc labeled  $a$  starting at the node  $A$  points to due to the alignment we performed in the very beginning. The resulting DFA for our example is depicted in Fig. 3(a). We can also summarize both generalization steps into one algorithm as follows:

**generalize-simple**

If a node is carrying the label  $J(a)$ , then

- (re)label the arc ending in  $J(a)$  with  $a^{\alpha(S(J(a)))}$ ;
- moreover, if  $\omega(S(J(a))) > \alpha(S(J(a)))$ , then introduce a loop on the node labeled  $J(a)$ ; this loop arc is labeled  $a$ .

Let us have a look at another example where we actually obtain generalized automata.

Let  $I_+ = \{aaa, aab\}$ . The generalized NFA resulting from the start tree has three states (labeled  $r, I_+(a)$  and  $I_+^a(b)$ , the first being the start state

and the latter two being final states). The arc from  $r$  to  $I_+(a)$  is labeled  $\{aa, aaa\}$  and the arc from  $I_+(a)$  to  $I_+^a(b)$  is labeled  $b$ . This yields a generalized DFA with the same states and the following three arcs:

- One arc labeled  $aa = a^{\alpha(S(I_+(a)))}$  from  $r$  to  $I_+(a)$ .
- One loop arc at  $I_+(a)$  labeled  $a$ .
- One arc labeled  $b$  from  $I_+(a)$  to  $I_+^a(b)$ .

Obviously, it is the label of the arc from the root  $r$  to  $I_+(a)$  which makes the automata “truly generalized.”

### 3.4 The Language Class $\mathcal{SL}$

Which language class can we learn this way? In fact, this is an interesting question which can be posed for many “heuristics” which have been proposed in the literature (or merely presented as programs, nowadays on the web) for learning regular expressions (or more generally, certain classes of [descriptions of] languages). We now give two definitions, restricting the usual notion of a regular expressions and the usual notion of a DFA, which turn out to be two characterizations of the languages which can be learned by the algorithm `generalize-simple`.

**Definition 1 (simple looping expressions).** Let  $\alpha$  be a union-free regular expression

$$b_1^{y_1} b_2^{y_2} \dots b_i^{y_i}$$

$$c_1^{z_1} c_2^{z_2} \dots c_j^{z_j}$$

with  $i, j \geq 1$ ,  $\Sigma = \{a, b, c, \dots\}$ ,  $y_\ell = z_\ell$ ,  $i = 0$ ,  $\beta = \lambda$ ,  $\alpha$  is left-aligned,  $y_k = z_k$ ,  $1 \leq k \leq K$ ,  $K \leq \min(i, j)$ ,  $b_k = c_k$ ,  $1 \leq k \leq K$ ,  $b_{K+1} \neq c_{K+1}$ ,  $b_{i+1} = c_{j+1} = \$$ .

simple looping

normal form  $\alpha = \lambda$

$$\alpha = a_1^{k_1} a_1^{x_1} a_2^{k_2} a_2^{x_2} \dots a_K^{k_K} a_K^{x_K}$$

- $a_i$   $\Sigma$
- $a_i$   $a_{i+1}$ ,  $i < K$
- $k_i$

- $x_i = 0 \text{ or } 1, \dots, x_i = 0 \text{ or } 1, \dots, a_i^0$
- $L$  simple looping
- $L$  simple looping  $\mathcal{SL}$

**Definition 2 (simple looping automata).** A DFA  $A$  is simple looping if

- $A$  is a skeleton graph
- $A$  is simple looping
- $A$  is simple looping  $\mathcal{SL}$

Observe that the first property of the previous definition ensures that the only cycles in the automaton graph are loops.

**Lemma 1.** If  $L$  is simple looping, then  $A(L)$  is simple looping.

If  $a \neq b$  are two letters, then  $A(\{a, b\})$  has two states, while this automaton is not simple looping, since it contains multiple arcs, namely two arcs (labeled  $a$  and  $b$ , respectively) from the start state to the final state. The corresponding simple looping DFA has three states (and is hence not minimal): one of its final states is reached via an  $a$ -transition from the start state, while the other final state is reached by a  $b$ -transition from the start state.

**Theorem 1 (Characterization Theorem).** A language  $L$  is simple looping if and only if

- $L \subseteq \Sigma^*$
- $L$  is simple looping
- $L$  is simple looping

If  $L$  is simple looping, then there is a simple looping regular expression  $E$  describing  $L$ , consisting of pairwise left-aligned union-free subexpressions  $E_i, 1 \leq i \leq m$ . Each  $E_i$  can be straightforwardly turned into a DFA  $A_i$  which is simple looping. A part  $a^k$  of  $E_i$  is thereby turned into a path of length  $k$  within the automaton graph of  $A_i$  where each arc is labeled  $a$ ; if  $a^*$  follows  $a^k$  in  $E_i$ , then the endpoint of the mentioned part will carry a loop labeled  $a$ . Since the expressions  $E_i$  are left-aligned, we can “overlay” the different  $A_i$  graphs to produce a DFA  $A$  which is simple looping.

Conversely, if  $L$  is generated by a simple looping DFA  $A$ , we can decompose  $A$  (due to the underlying “tree structure”) into a collection of paths  $A_i$  (which may contain loops), such that each  $A_i$  contains only one final state, namely the last one on the path. (This means that “intermediate” final states are not considered to be final in “larger paths.”) Each such  $A_i$  can be turned into a regular expression  $E_i$  whose collection yields the required simple looping regular expression  $E$ .

Since (as remarked upon introducing simple looping regular expressions) union-free “subexpressions” only turn up once, we can conclude:

**Corollary 1.**

The following lemma relates  $\mathcal{SL}$  with our learning algorithm. Recall that a generalized DFA can be turned into a “usual” DFA by possibly introducing “intermediate states” used for spelling the words labeling arcs. If  $A$  is a generalized DFA, let  $\text{DFA}(A)$  denote the DFA obtained in this way. The following lemma is immediate from the definition, yet crucial.

**Lemma 2.**

$A$ , generalize-simple

$(A)$

**3.5  $\mathcal{SL}$  Can be Learned from Positive Data**

Let us now discuss the identifiability of  $\mathcal{SL}$ . To this end, let us first describe how to obtain a characteristic sample for a simple looping DFA  $A$ .

For every final state  $s$ , there is a unique path from the initial state  $s_0$  of  $A$  to  $s$  in the skeleton graph (tree) associated to  $A$ : upon reading the labels of the arcs, this yields a unique word  $w_s \in L(A)$ . In fact,  $w_s$  is the shortest word in  $L(A)$  that is accepted via state  $s$ . Observe that the collection  $L_F$  of all those  $w_s$  pass through all states of  $A$ , since  $A$  has no useless states. Therefore, we may extend our notation, so that for each state  $s$  (not only for final ones),  $w_s$  refers to some word from  $L_F$  such that  $w_s$  passes through  $s$ . Then, let  $u_s$  be the initial part of  $w_s$  that describes how to get from the initial state  $s_0$  into  $s$ . Let  $v_s$  refer to the remaining part of  $w_s$ , i.e.,  $w_s = u_s v_s$ . Now, if  $A$  has a loop labeled  $a$  at state  $s$ , let  $w_s^\circ = u_s a v_s$ . Let  $L^\circ$  collect all such loop indicating words  $w_s^\circ$ . Then,  $\chi(L) = L_F \cup L^\circ$  is a characteristic sample of  $L$ .

Namely, upon getting the sample  $\chi(L)$ ,  $L_F$  will provide the information to construct the skeleton graph (which corresponds to the prefix block tree discussed above, except that block words are spelled out). Then, the loops are reconstructed using the words from  $L^\circ$ .

In our example from Fig. 3(a), the characteristic sample would be

$$\chi = \{ab, abc, abab, ababa, aab, abb, ababb\}.$$

Notice that this is larger than the input sample  $I_+$  in Ex. 1 due to the following reasons: (i)  $ab$  and  $abab$  are prefixes of other words in  $L_F$  and could hence be omitted; (ii) we introduce one loop-generating word per loop in our systematic construction of  $L^\circ$ , ignoring the possibility of describing more than one loop in a (longer) word.

Finally notice that the algorithm we presented so far for learning  $\mathcal{SL}$  languages is a “bulk algorithm” in the sense that it expects the whole sample at one gulp. From the point of view of practice, also incremental algorithms are

interesting. It is not too hard to convert the given algorithm into an incremental one. Firstly, we would scan a new word  $w$  with the automaton derived so far. If  $w$  is accepted, we continue with the next input word. If not, this can have two reasons: (i) either, we have reached a state that has not been marked final; then, marking this state final will cope with the situation; or (ii) we would have to use a yet non-existing transition in a certain state. In that case, we might (a) either introduce a loop in the state we are just dealing with, or (b) we will leave the state via a new arc to a new state, and the remainder of the word will create more new states. Notice that also the case (a) might in continuation lead to cases (i) or (ii)(b) upon further reading of  $w$  by means of the (modified) automaton.

We summarize our findings:

**Theorem 2.**  $\mathcal{SL}$  is closed under the operation of adding loops to transitions.

## 4 Extensions of Our Approach

### 4.1 Dealing with Multiplicities

In Sec. 2, we described another way of dealing with loops: namely, we described how to actually count multiple occurrences of a letter upon looping. It is not hard to see how this can be actually incorporated into the framework developed in the preceding section.

More precisely, we would now allow looping regular expressions that contain loops of the form  $(a^n)^*$ , or  $a^{n*}$  for short. This would have to be incorporated into the left-alignment definition (allowing  $y_\ell/z_\ell$  to be 1 or  $n^*$ , not only 1 or  $*$ , in the notation of that definition). This way, we can describe a superclass  $\mathcal{SL}'$  of  $\mathcal{SL}$ . This would naturally also entail that the corresponding looping DFA may contain “larger loops.” The inference algorithm itself is again very similar, except that when it observes, e.g.,  $\{b, bbb\}$  as arc label of a resulting NFA, it would create a loop with arc label  $bb$  in the resulting (general) DFA. Keeping in mind the intention of the characteristic sample definition (to exercise each transition at least once), also that definition can be extended.

This reasoning allows us to conclude:

**Theorem 3.**  $\mathcal{SL}'$  is closed under the operation of adding loops to transitions.

### 4.2 Introducing Wildcards

There is another type of generalization that one easily comes up with when thinking of regular expressions: the use of wildcards, maybe in the slightly general form of introducing character groups. It is exactly the generalizing capacity formalized in  $\mathcal{SL}$  together with the ability of forming character groups that has been realized in the SWYN system of Blackwell [4].

Introducing wildcards (for simplicity) on top of the  $\mathcal{SL}$ -mechanism means that we are finally arriving at an identifiable language class different from  $\mathcal{SL}$  (so not generalizing  $\mathcal{SL}$  as we did in the preceding section). How can we find

places where we might wish to insert wildcards (denoted by the letter '?' in what follows, assuming that this is not part of the usual alphabet we are dealing with)? The easiest way to find such places is to look at it in terms of preprocessing, given an input sample  $I_+$ :

**preprocess-wildcards( $I_+$ )**

1. Form the set of blocks  $[I_+]$  consisting of all sequences of block letters of all words from  $I_+$  while ignoring multiplicity information.
2. Pairwisely left-align the blocks in  $[I_+]$ ; whenever we find a pair  $(x, y)$  with
  - $x = [a_1] \dots [a_m][a][a_{m+2}] \dots [a_n]$  and
  - $y = [b_1] \dots [b_m][b][b_{m+2}] \dots [b_\ell]$
 such that, for all  $i = 1, \dots, m, m + 2, \dots, \min(n, \ell)$ , either  $a_i = ?$  or  $b_i = ?$  or  $a_i = b_i$ ; then we can replace both  $a$  and  $b$  by '?'. Moreover, we would replace  $a_i$  by '?' if  $b_i = ?$  and (conversely)  $b_i$  by '?' if  $a_i = ?$ . Notice that in accordance with our usage of block letters, the actual interpretation of '?' depends on its context: so, if we result in  $[?][?]$ , then this refers to two different basic letters upon unfolding into the usual alphabet.
3. If the second step permits no further changes, we go back to the original input  $I_+$  and replace letters by '?' whenever such a change was indicated by the second step (working on block letters), yielding a new instance  $I_+^?$ .

Then, we run the procedure for  $\mathcal{SL}$ -learning (or say for  $\mathcal{SL}'$ -learning) on  $I_+^?$ .

Let us explain the wildcard usage by means of a little example. Consider  $I_+ = \{ab, ac, bc\}$ . Hence,  $[I_+] = \{[a][b], [a][c], [b][c]\}$ . Matching  $[a][b], [a][c]$  gives  $[a][?]$ . Alternatively, we can match  $[a][?], [b][c]$  yielding  $[?][?]$ . Since this last expression matches (subsumes) all original ones, we arrive at  $I_+^? = \{??\}$ .

Starting with the less pathological sample  $I_+ = \{ababb, aabb, ababa, abc\}$  from Ex. 1, we get

$$[I_+] = \{[a][b][a][b], [a][b], [a][b][a][b][a], [a][b][c]\}.$$

Matching  $[a][b][a][b]$  against  $[a][b][c]$  yields  $[a][b][?][b]$  and  $[a][b][?]$ . Similarly, we might match  $[a][b][a][b][a]$  against  $[a][b][c]$ . Hence,  $I_+^? = \{ab?bb, aabb, ab?ba, ab?\}$ . The (finally) resulting DFA is depicted in Fig. 3(b).

How can we characterize the class of languages that is identifiable in this way? This can be easiest explained using simple looping expressions that may contain the special letter '?'. We can define a compatibility relation of two left-aligned, union-free expressions if we consider the "mergibility" of the corresponding block letter words as described above. Then, we would only allow looping expressions (that may contain '?') that are decomposed as finite union of expressions which are incompatible in the sense sketched above. For reasons of space, we omit the corresponding technical details. These details become even more cumbersome if we aim to generalize  $\mathcal{SL}'$  or if try to introduce, e.g.,  $\{a, b, c, \dots, z, 0, 1, \dots, 9\}$  (as "lowercase letters", "alphanumeric symbols", etc., as known from certain application areas), thus enabling to have different sorts of wildcards. However, let us state that all these ways of introducing wildcards lead to identifiable language classes that can be syntactically characterized by certain forms of regular expressions.



## 5 A Possible Application: Learning DTDs

The expectations surrounding XML (eXtensible Markup Language) as universal syntax for data representation and exchange are huge, as underlined by the amount of effort being committed to XML by the World Wide Web Consortium (W3C) (see [www.w3.org/TR/REC-XML](http://www.w3.org/TR/REC-XML)), by the huge number of academics involved in the research of the backgrounds of XML, as well as by numerous private companies. Moreover, many applications arise which make use of XML, although they are not directly related to the world wide web. For example, nowadays XML plays an important role in the integration of manufacturing and management in highly automated fabrication processes such as in car companies [7]. For further information, refer to [www.oasis-open.org/cover/xmlIntro.html](http://www.oasis-open.org/cover/xmlIntro.html).

The syntactic part of the XML language describes the relative position of pairs of corresponding tags. This description is done by means of a Document Type Definition (DTD). Ignoring attributes of tags, a DTD is a special form of a context-free grammar. This sort of grammar formalism has been formalized and studied by Berstel and Boasson [3], defining

As already worked out by Ahoen, grammatical inference (GI) techniques can be very useful for automatic document processing, see [1, 2]. Building upon her work, we described [8] three possible applications of grammatical inference in the context of DTD inference:

- to assist designing grammars for (semi-) structured documents;
- to create views and sub-documents; and
- to optimize the performance of database queries based on XML by the help of adequate DTDs.

To underline the first of these applications, let us quote Tim Bray, one of the “fathers” of XML, who wrote (see [www.xml.com/axml/notes/Any1.html](http://www.xml.com/axml/notes/Any1.html)):

Suppose you’re given an existing well-formed XML document and you want to build a DTD for it. One way to do this is as follows:

1. Make a list of all the element types that actually appear in the document, and build a simple DTD which declares each and every one of them as ANY. Now you’ve got a DTD (not a very useful one) and a valid document.
2. Pick one of the elements, and work out how it’s actually used in the document. Design a rule, and replace the ANY declaration with a ...content declaration. This, of course, is *the tricky part*, particularly in a large document.
3. Repeat step 2, working through the elements ..., until you have a useful DTD.

Instead of explaining these notions formally (as done in [8]), let us rather discuss a small but realistic example.

Fig. 4 shows the first lines of a short novel in somewhat simplified XML format. Disregarding the actual contents (i.e., the novel itself), we obtain the following possible samples of what a paragraph could be: it could consist in one sentence (see the headline etc.), of three sentences (first paragraph of the

```

<book>
  <part>
    <chapter>
      <paragraph>
        <sentence>Die Verwandlung</sentence>
      </paragraph>
      <paragraph>
        <sentence>von Franz Kafka</sentence>
      </paragraph>
    </chapter>
    <chapter>
      <paragraph>
        <sentence>I</sentence>
      </paragraph>
      <paragraph>
        <sentence>Als Gregor Samsa eines Morgens aus unruhigen Träumen erwachte,
          fand er sich in seinem Bett zu einem ungeheueren Ungeziefer verwandelt.
        </sentence>
        <sentence>Er lag auf seinem panzerartig harten Rücken und sah, wenn er den Kopf ein wenig hob, seinen gewölbtten,
          braunen, von bogenförmigen Versteifungen geteilten Bauch, auf dessen Höhe sich die Bettdecke,
          zum gänzlichen Niedergleiten bereit, kaum noch erhalten konnte.
        </sentence>
        <sentence>Seine vielen, im Vergleich zu seinem sonstigen Umfang kläglich dünnen Beine
          flimmerten ihm hilflos vor den Augen.
        </sentence>
      </paragraph>
      <paragraph>
        <sentence>"Was ist mit mir geschehen?",</sentence> dachte er. </sentence>
        <sentence>Es war kein Traum. </sentence>
      </paragraph>
    </chapter>
  </part>
</book>

```

**Fig. 4.** The first lines of Kafka’s short novel “Verwandlung” in XML format

novel itself) or two (due to cutting short the novel after the introductory part). Hence, our learner would generalize these examples to express that a paragraph could consist of one or more sentences. However, all learning algorithms that we proposed would insist in a chapter consisting of at least two paragraphs according to the examples seen up to now. Both ways of generalization are according to common sense: chapters with one paragraph are rarely observed.

## 6 Conclusions

We are aware of the fact that our proposed learners can only generate very simple kinds of regular expressions. However, first of all this is intrinsic to the learning model we used, since text learning (identification in the limit from positive samples) does not allow to learn all regular languages (be them encoded by automata or expressions); even the class of languages that definable by all REs (based on the operations union, catenation and star) without nested stars is not identifiable from positive data: ponder the sample  $w_n = (ab)^n$  versus  $(ab)^*$ .

Secondly, there do exist applications for at least similarly simplistic versions of regular expressions. For example, the path expressions as discussed in [6] in the context of XML path queries are of a similar simplicity.

Moreover, many web browsers do only admit a limited form of regular expressions which pretty much resemble the subset we propose. We already mentioned the SWYN system [4] that employs practically the same sorts of expressions.<sup>2</sup> However, it would be nice to extend the results of this paper in a direction that

<sup>2</sup> From the system description, it is not quite clear how the alignment is actually performed, so there might be some technical differences.

allows longer strings (containing different sorts of letters) to be starred. Finding a reasonable subclass of regular languages that can be inferred in this way remains a topic of future study, obviously limited by the observed sample in the first paragraph.

We have also considered the (non-)closure properties of  $\mathcal{SL}$  (all standard operations but intersection show non-closure behavior) and have derived an alternative, more compact characterization of  $\mathcal{SL}$  in terms of REs. We skipped details here for reasons of space.

## References

- [1] H. Ahonen. Disambiguation of SGML content models. In C. Nicholas and D. Wood, eds., *Principles of Document Processing PODP'96*, volume 1293 of *LNCS*, pp. 27–37. Springer, 1997.
- [2] H. Ahonen, H. Mannila, and E. Nikunen. Forming grammars for structured documents: an application of grammatical inference. In R. C. Carrasco and J. Oncina, eds., *International Colloquium on Grammatical Inference ICGI'94*, volume 862 of *LNCS/LNAI*, pp. 153–167. Springer, 1994.
- [3] J. Berstel and L. Boasson. Formal properties of XML grammars and languages. *Acta Informatica*, 38(9):649–671, August 2002.
- [4] A.F. Blackwell. SWYN: A visual representation for regular expressions. In H. Lieberman, ed., *Your wish is my command: Giving users the power to instruct their software*, pp. 245–270. Morgan Kaufmann, 2001.
- [5] A. Brazma. Efficient learning of regular expressions from approximate examples. In R. Greiner, T. Petsche, and S. J. Hanson, eds., *Computational Learning Theory and Natural Learning Systems, Vol. IV: Making Learning Systems Practical*, chapter 19, pp. 337–352. Cambridge (MA) USA: MIT Press, 1997.
- [6] Y. D. Chung, J. W. Kim, and M. H. Kim. Efficient preprocessing of XML queries using structured signatures. *Information Processing Letters*, 87:257–264, 2003.
- [7] CZ-Redaktion. Maschinenmenschen plaudern per XML mit der Unternehmens-IT. *Computer Zeitung*, (50):30, December 2000.
- [8] H. Fernau. Learning XML grammars. In P. Perner, ed., *Machine Learning and Data Mining in Pattern Recognition MLDM'01*, volume 2123 of *LNCS/LNAI*, pp. 73–87. Springer, 2001.
- [9] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: learning document type descriptors from XML document collections. *Data Mining and Knowledge Discovery*, 7:23–56, 2003.
- [10] E. M. Gold. Language identification in the limit. *Information and Control (now Information and Computation)*, 10:447–474, 1967.
- [11] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Reading (MA) USA: Addison-Wesley, 1979.
- [12] Ph. D. Laird. *Learning from Good and Bad Data*. Norwell (MA) USA: Kluwer Academic Publishers, 1988.
- [13] C. G. Nevill-Manning and I. H. Witten. Online and offline heuristics for inferring hierarchies of repetitions in sequences. *Proc. IEEE*, 88:1745–1755, 2000.
- [14] T. C. Smith, I. H. Witten, J. G. Cleary, and S. Legg. Objective evaluation of inferred context-free grammars. In *Proc. Australian and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia*, November 1994.
- [15] M. van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD, School of Computing, University of Leeds, UK, September 2001.

# A Class of Prolog Programs with Non-linear Outputs Inferable from Positive Data

M. R. K. Krishna Rao

Information and Computer Science Department,  
King Fahd University of Petroleum and Minerals,  
Dhahran 31261, Saudi Arabia  
krishna@ccse.kfupm.edu.sa

**Abstract.** In this paper, we study inferability of Prolog programs from positive examples alone. We define a class of Prolog programs called recursion bounded programs that can capture non-linear relationships between inputs and outputs and yet inferable from positive examples. This class is rich enough to include many programs like append, delete, insert, reverse, permute, count, listsum, listproduct, insertion-sort, quick-sort on lists, various tree traversal programs and addition, multiplication, factorial, power on natural numbers. The relation between our results and the known results is also discussed. In particular, the class of recursion bounded programs contains all the known terminating linearly-moded Prolog programs of Krishna Rao [7] and additional programs like power on natural numbers which do not belong to the class of linearly-moded programs and the class of safe programs of Martin and Sharma [12].

## 1 Introduction

Many problems in Machine Learning are concerned with investigating and formalizing human learning processes in order to utilize the results in designing computer programs. In particular, children's ability to learn their mother tongue on the basis of incomplete and ambiguous information motivated various abstract learning models which try to reflect the special quality of language acquisition. A well studied approach in this field is the theory of formal language learning introduced by Gold [6] and Blum and Blum [5]. The general situation investigated in language learning in the limit can be described as follows. An inductive inference machine is an algorithmic device that is fed more and more information about a language to be inferred. This information can consist of positive and negative examples or only positive ones. When fed a presentation for a concept  $C$ , the inductive inference machine has to produce hypotheses about  $C$ . The hypotheses the learner produces have to be members of an admissible set of hypotheses (a hypothesis space) and the sequence of hypotheses has to converge to a hypothesis correctly describing the concept  $C$  to be learned. If the learner converges for every presentation of  $C$  to a correct description of  $C$ , then it is said to identify the concept  $C$  in the limit. A learner identifies a collection of concepts in the limit if and only if it identifies each member of this collection in the limit. In this

paper we consider the case, where the learner is fed all positive examples of concept  $C$  to be inferred but no negative examples, i.e., positive presentation. If the learner converges for every positive presentation for  $C$  to a correct description of  $C$ , then it is said to identify the concept  $C$  in the limit from positive data. A learner identifies a collection of concepts in the limit from positive data if and only if it identifies each member of this collection in the limit from positive data. The study of inferability from positive data alone is important because negative information is hard to obtain in practice – positive examples are much easier to obtain from collections like bookmarks and to generate by conducting experiments than the negative examples in general.

In his seminal paper [6] on inductive inference, Gold proved that even simple classes of concepts like the class of regular languages cannot be inferred from positive examples alone. This strong negative result disappointed the scientists in the field until Angluin [1] has given a characterization of the classes of concepts that can be inferred from positive data alone and exhibited a few nontrivial classes of concepts inferable from positive data. This influential paper inspired further research on the inductive inference from positive data. Since then many positive results are published about inductive inference of logic programs and pattern languages from positive data (see a.o., [15, 3, 4, 16, 7, 12, 8]).

Logic programs with elegant and simple declarative semantics can be used as representations of the concepts to be learned. Continuing Angluin's line of research, Shinohara [15] presented a few classes of concepts inferable from positive data. The class of linear Prolog programs of Shapiro [14] is notable among them. Unfortunately, the class of linear programs is very restrictive from the programming point of view as they do not allow local variables, which play the fundamental role of sideways information passing in the paradigm of logic programming. Using moding annotations on predicates, Arimura and Shinohara [4] introduced a class of linearly covering programs which have local variables and yet inferable from positive data. Though they allow local variables, the class of linearly covering programs is still restrictive as they view the logical variable as a point-to-point communication channel. Krishna Rao [7] proposed the class of linearly-moded programs, which includes all the above mentioned programs and established inferability of linearly-moded programs from positive data. The class of linearly-moded programs is rich enough to contain many programs from Sterling and Shapiro's book [17] including `append`, `merge`, `split`, `insert`, `insertion-sort`, `preorder` and `inorder` traversal of binary trees, polynomial recognition, derivatives, sum of a list of natural numbers.

The main shortcoming of the class of linearly-moded programs is that the programs in this class can only capture linear relationships between input and output. In fact, the size of output is bounded by the size of input. This means that programs for multiplication and exponentiation are beyond the class of linearly-moded programs. In this paper, we propose a class of Prolog programs inferable from positive examples and yet include programs with nonlinear relationships between input and output. The programs in this class are called

recursion bounded programs and this class includes all the known linear-moded programs that are terminating.

The rest of the paper is organized as follows. The next section gives preliminary definitions and results needed about inductive inference. In section 3, we define the class of recursion bounded programs and present a few examples. In Section 4, we establish a few properties of these programs. We then prove inferability of recursion bounded programs from positive data in Section 5. The class of recursion bounded programs is compared with the classes of linear-moded and safe programs in Section 6.

## 2 Preliminaries

We assume that the reader is familiar with the basic terminology of logic programming and inductive inference and use the standard terminology from [6, 10]. In the following, we recall some definitions and results needed in the sequel.

### 2.1 Inductive Inference from Positive Data

**Definition 1.** Let  $U$  and  $E$  be two recursively enumerable sets, whose elements are called  $u$  and  $e$  respectively.

- A  $\Gamma \subseteq U$  is a subset  $\Gamma \subseteq U$ .
- An  $\langle A, a \rangle$  is a tuple  $\langle A, a \rangle$  where  $A \in U$  and  $a = \text{true}$  or  $\text{false}$ . Example  $\langle A, a \rangle$  is  $\langle A, \text{true} \rangle$  if  $a = \text{true}$  and  $\langle A, \text{false} \rangle$  otherwise.
- A concept  $\Gamma$  is  $\Gamma \subseteq U$  with a sequence of examples  $\langle A_1, a_1 \rangle, \dots, \langle A_m, a_m \rangle$  when  $A_i \in \Gamma$  if and only if  $a_i = \text{true}$ , for each  $i \in [1, m]$ .
- A  $R \subseteq E$  is a finite subset  $R \subseteq E$ .
- A  $\Phi$  is a mapping  $\Phi$  from formal systems to concepts.
- We say that a formal system  $R$  is a concept  $\Gamma$  if  $\Phi(R) = \Gamma$ .

**Definition 2.** A  $\langle U, E, \Phi \rangle$  is a triple  $\langle U, E, \Phi \rangle$  of a universe  $U$  of objects, a universe  $E$  of expressions and a semantic mapping  $\Phi$ .

**Definition 3.** A class of concepts  $C = \{R_1, R_2, \dots\}$  is an  $\text{algorithm}$  if there exists an algorithm that decides whether  $w \in R_i$  for any object  $w$  and natural number  $i$ .

Here onwards, we fix a concept defining framework  $\langle U, E, \Phi \rangle$  arbitrarily and only consider indexed families of recursive concepts.

**Definition 4.** A  $w_1, w_2, \dots$  of a nonempty concept  $R \subseteq U$  is an infinite sequence  $w_1, w_2, \dots$  of objects such that  $\{w_i \mid i \geq 1\} = R$ .

An  $g$  is an effective procedure that requests an object as an example from time to time and produces a concept (or a formal system defining a concept) as a conjecture from time to time. Given a positive presentation  $\sigma = w_1, w_2, \dots$ , an inference machine IM generates a sequence of conjectures  $g_1, g_2, \dots$ . We say that IM  $g$  on input  $\sigma$  if the sequence of conjectures  $g_1, g_2, \dots$  is finite and ends in  $g$  or there exists a positive integer  $k_0$  such that  $g_k = g$  for all  $k \geq k_0$ .

**Definition 5.** A class  $C$  of concepts is *computable* if there exists an inference machine IM such that for any  $R \in C$  and any positive presentation  $\sigma$  of  $R$ , IM converges to a formal system  $g$  such that  $\Phi(g) = R$ .

We need the following result of Shinohara [15] in proving our results.

**Definition 6.** A semantic mapping  $\Phi$  is *bounded* if  $\Gamma \subseteq \Gamma'$  implies  $\Phi(\Gamma) \subseteq \Phi(\Gamma')$ . A formal system  $\Gamma$  is *bounded* if  $S \subseteq U$  if  $S \subseteq \Phi(\Gamma)$  and  $S \not\subseteq \Phi(\Gamma')$  for any proper subset  $\Gamma' \subset \Gamma$ .

**Definition 7.** A concept defining framework  $\mathcal{C} = \langle U, E, \Phi \rangle$  has *bounded finite thickness* if  $\Phi$  is bounded and  $\mathcal{C}^m = \{\Phi(\Gamma) \mid \Gamma \subseteq E, |\Gamma| \leq m\}$  for all  $m \geq 0$ .

**Theorem 1. (Shinohara [15])**

If a concept defining framework  $\mathcal{C} = \langle U, E, \Phi \rangle$  has bounded finite thickness, then

$$\mathcal{C}^m = \{\Phi(\Gamma) \mid \Gamma \subseteq E, |\Gamma| \leq m\}$$

for all  $m \geq 1$ .

**2.2 Basic Concepts from Logic Programming**

The alphabet of a first order language  $L$  is a triple  $\langle \Pi, \Sigma, \mathcal{X} \rangle$  of mutually disjoint sets such that  $\Pi$  and  $\Sigma$  are finite. The elements of  $\Pi, \Sigma$  and  $\mathcal{X}$  are called *predicate symbols*, *function symbols* and *variables*. We denote arity of a predicate/function symbol  $f$  by *arity*( $f$ ). In the following,  $\mathcal{T}(\Sigma, \mathcal{X})$  denotes the set of terms constructed from the function symbols in  $\Sigma$  and the variables in  $\mathcal{X}$ .  $\mathcal{A}(\Pi, \Sigma, \mathcal{X})$  denotes the set of atoms constructed from these terms and the predicate symbols in  $\Pi$ . Terms (atoms) which do not contain any variable are called *ground terms* (atoms). For a predicate  $p$ , the set of ground atoms  $\mathcal{A}(\{p\}, \Sigma, \phi)$  is denoted by  $B(p)$ . The sets  $\mathcal{T}(\Sigma, \phi)$  and  $\mathcal{A}(\Pi, \Sigma, \phi)$  are also called *Herbrand universe*  $U_L$  and *Herbrand base*  $B_L$  respectively of  $L$ . In the following, we sometimes denote the sequence of terms  $t_1, \dots, t_n$  by  $\mathbf{t}$ . The size of a term  $t$ , denoted by  $|t|$ , is defined as the number of symbols (except the punctuation symbols) occurring in it. The size of an atom  $p(t_1, \dots, t_n)$ , denoted by  $|p(t_1, \dots, t_n)|$ , is defined as the sum  $1 + |t_1| + \dots + |t_n|$ .

**Definition 8.** A logic program  $P$  is a finite set of definite clauses of the form  $H \leftarrow B_1, \dots, B_n$ , where  $H, B_1, \dots, B_n$  are atoms. The atom  $H$  is the head and  $B_1, \dots, B_n$  is the body and  $n$  is the body-length of this clause. The size of a clause  $H \leftarrow B_1, \dots, B_n$ , denoted by  $Size(H \leftarrow B_1, \dots, B_n)$ , is defined as the sum  $|H| + |B_1| + \dots + |B_n|$  of the sizes of atoms in it.

The declarative semantics of a logic program  $P$  is usually given by its least Herbrand model, i.e., the smallest set that contains all the ground atoms which are entailed by the clauses in  $P$ . The least Herbrand model of  $P$  is denoted by  $M(P)$ . The least Herbrand model semantics of logic programs serves as a

monotonic semantic mapping. We use a special predicate symbol  $c$  to denote the target predicate to be learned. Our concept defining frameworks are of the form  $\langle B(c), L, M_c \rangle$ , where  $L$  is the class of Prolog clauses under consideration and  $M_c$  is a semantic mapping such that  $M_c(P)$  is the set of all atoms of the target predicate  $c$  in the least Herbrand model of  $P$ .

The procedural semantics of logic programs is given by the SLD-resolution (see Lloyd [10] for more details). It is well known that the declarative semantics and the procedural semantics of definite logic programs coincide. In the following, we are concerned with Prolog programs, where the selection rule is fixed as leftmost selection rule. Following [2], we call the SLD-derivations and SLD-refutations under Prolog's selection rule, LD-derivations and LD-refutations respectively.

The class of recursion bounded programs is defined using the concept of modes and linear inequalities.

**Definition 9.** A *mode*  $m$  of an  $n$ -ary predicate  $p$  is a function from  $\{1, \dots, n\}$  to the set  $\{in, out\}$ . The set  $in(p) = \{i \leq n \mid m(i) = in\}$  is the set of input positions of  $p$  and  $out(p) = \{o \leq n \mid m(o) = out\}$  is the set of output positions of  $p$ .

A moded program is a logic program with each predicate having a unique mode associated with it.

**Notation:** In the following,  $p(\mathbf{s}; \mathbf{t})$  denotes an atom with a sequence  $\mathbf{s}$  of input terms and a sequence  $\mathbf{t}$  of output terms. Without loss of generality, we assume that input positions of a predicate precede its output positions. The expressions  $invar(A)$  and  $outvar(A)$  denote the sets of variables occurring in the input and output positions respectively of the atom  $A$ .

**Definition 10.** A Prolog clause  $H \leftarrow B_1, \dots, B_k$  ( $k \geq 0$ ) is *well-moded* if  
 (a)  $outvar(H) \subseteq invar(H) \cup outvar(B_1) \cup \dots \cup outvar(B_k)$  and  
 (b) for every  $i \in [1, k]$ ,  $invar(B_i) \subseteq invar(H) \cup outvar(B_1) \cup \dots \cup outvar(B_{i-1})$ .  
 A Prolog program  $P$  is *well-moded* if every clause in it is well-moded. A *well-moded clause without head* is a well-moded clause without head.

In this paper, we only consider well-moded programs. The following results for well-moded programs are well-known [9].

**Lemma 1.** If  $\leftarrow B_1, \dots, B_n$  is a well-moded Prolog query then  $invar(B_1) = \phi$ .

**Lemma 2.** If  $P$  is a well-moded Prolog program and  $G$  is a well-moded goal then every goal in any LD-derivation (SLD-derivation under Prolog's selection rule) of  $P \cup \{G\}$  is well-moded.

### 3 Recursion Bounded Programs

In this section, we define the class of recursion bounded programs and illustrate the concept with a few examples. The definition is based on the concept of modes and linear predicate inequalities.



**Definition 11.** For a term  $t$ , the parametric size  $[t]$  of  $t$  is defined recursively as follows:

- if  $t$  is a variable  $x$  then  $[t]$  is a linear expression  $x$ ,
- if  $t$  is a constant then  $[t]$  is zero,
- if  $t = f(t_1, \dots, t_n)$  then  $[t]$  is a linear expression  $1 + [t_1] + \dots + [t_n]$ .

The parametric size of a sequence  $\mathbf{t}$  of terms  $t_1, \dots, t_n$  is the sum  $[t_1] + \dots + [t_n]$ .

**Example 1.** The parametric sizes of terms  $\mathbf{a}$ ,  $[\ ]$ ,  $\mathbf{X}$ ,  $[\mathbf{a}]$ ,  $[\mathbf{a}, \mathbf{b}, \mathbf{c}]$ ,  $[[\ ]]$ ,  $[[\ ], [\ ]]$ ,  $[[\mathbf{a}], [\mathbf{b}], [\mathbf{c}]]$  are 0, 0,  $X + 1$ , 1, 3, 3, 6 respectively.  $\square$

**Remark 1.** The above notion of parametric size is an improvement of the notion defined in [7]. In particular, all constants are treated uniformly by the above definition, while constants zero, the empty list  $[\ ]$  and the empty tree `void` are treated as special constants in [7].

The following definition introduces the notation  $LI(A, I, O)$ , which is central to our results. It captures the relation between the sizes of input and output terms of an atom.

**Definition 12.** Let  $P$  be a moded program and  $I$  and  $O$  be mappings from the set of predicates occurring in  $P$  to sets of input positions and output positions satisfying  $I(p) \subseteq in(p)$  and  $O(p) \subseteq out(p)$  for each predicate  $p$  in  $P$ . For an atom  $A = p(\mathbf{s}; \mathbf{t})$ , we denote the linear inequality

$$\sum_{i \in I(p)} [s_i] \geq \sum_{j \in O(p)} [t_j] \tag{1}$$

by  $LI(A, I, O)$ .

**Remark 2.** The validity of (linear) inequalities is traditionally defined as the follows:  $\mathbf{expression1} \geq \mathbf{expression2}$ . In the sequel, we only talk of sizes which are obviously non-negative and hence  $\mathbf{expression1} \geq \mathbf{expression2}$ . According to this,  $X + 1 > X$  is valid but  $X + Y > X$  is not valid because  $Y$  can take a zero value and  $X + 0$  is not greater than  $X$ . Similarly,  $2X > X$  is not valid because  $X$  can take a zero value. However, both  $X + Y \geq X$  and  $2X \geq X$  are valid.

The following lemma is a simple consequence of this notion of validity.

**Lemma 3.** The following holds for any inequality  $exp_1 \geq exp_2$ .

1.  $exp_1 \geq exp_2$  is valid if and only if  $exp_1\theta \geq exp_2\theta$  is valid for every substitution  $\theta$ , and
2.  $exp_1 \geq exp_2$  is valid if and only if the constant in  $exp_2$  is less than or equal to the constant in  $exp_1$  and the coefficient of each variable in  $exp_2$  is less than or equal to its coefficient in  $exp_1$ .

The following definition captures the call dependencies (and mutual recursion, if any) between predicates in a program.

**Definition 13.** Let  $P$  be a program,  $p$  and  $q$  be predicates. We say that predicate  $p$  depends on predicate  $q$  in  $P$  if there is a clause in  $P$  with  $p$  in the head and  $q$  in the body. We say that  $p$  depends on  $q$  and write  $p \succeq_P q$  if  $(p, q)$  is in the reflexive and transitive closure of the relation  $\text{depends}$ .

Now, we are in a position to define the class of recursion bounded programs. Intuitively, any atom  $p(\mathbf{s}; \mathbf{t})$  in the least Herbrand model of a recursion bounded program (w.r.t.  $I$  and  $O$ ) satisfies the property that the total size of output terms in positions  $O(p)$  is bounded by the total size of input terms in positions  $I(p)$ .

**Definition 14.** Let  $P$  be a well-moded program and  $I$  and  $O$  be mappings from the set of predicates occurring in  $P$  to sets of input positions and output positions satisfying  $I(p) \subseteq \text{in}(p)$  and  $O(p) \subseteq \text{out}(p)$  for each predicate  $p$  in  $P$ . We say  $P$  is recursion bounded w.r.t.  $I$  and  $O$  if each clause

$$p_0(\mathbf{s}_0; \mathbf{t}_0) \leftarrow p_1(\mathbf{s}_1; \mathbf{t}_1), \dots, p_k(\mathbf{s}_k; \mathbf{t}_k)$$

$k \geq 0$ , in  $P$  satisfies the following:

1.  $LI(A_1, I, O), \dots, LI(A_{j-1}, I, O)$  together imply

$$[Iterms(A_0, I)] > [Iterms(A_j, I)]$$

for each  $j \geq 1$  such that  $p_j \succeq_P p_0$ , and

2.  $LI(A_1, I, O), \dots, LI(A_k, I, O)$  together imply  $LI(A_0, I, O)$ .

where  $A_j$  is the atom  $p_j(\mathbf{s}_j; \mathbf{t}_j)$  for each  $j \geq 0$  and  $Iterms(A, I)$  is the sequence of terms occurring in atom  $A$  in positions specified by  $I$ .

A program  $P$  is recursion bounded w.r.t. some mappings  $I$  and  $O$ .

The main differences between the above definition and that of linear-moded programs in [7] are the following:

- In Definition 14, condition 1 is only applicable to the (mutually) recursive atoms in the body, while condition 1 of linear-moded programs in [7] is applicable to all the atoms in the body, and
- In Definition 14, condition 1 requires a strict inequality  $[Iterms(A_0, I)] > [Iterms(A_j, I)]$ , while condition 1 of linear-moded programs in [7] only requires  $[\mathbf{s}_0] \geq [\mathbf{s}_j]$ . The strict inequality ensures termination of recursion bounded programs, as shown by Theorem 6 below.

We illustrate different aspects of our definition through a sequence of examples.

**Example 2.** Consider the following quick-sort program.

moding: app (in, in, out); part (in, in, out, out) and  
qs (in, out)

app([], Ys, Ys) ←  
app([X|Xs], Ys, [X|Zs]) ← app(Xs, Ys, Zs)

part([], H, [], []) ←  
part([X|Xs], H, [X|Ls], Bs) ← X ≤ H, part(Xs, H, Ls, Bs)  
part([X|Xs], H, Ls, [X|Bs]) ← X > H, part(Xs, H, Ls, Bs)

qs([], []) ←  
qs([H|L], S) ← part(L, H, A, B), qs(A, A1), qs(B, B1), app(A1, [H|B1], S)

This program is  $\dots$  w.r.t. the mappings such that  $I(p) = in(p)$  and  $O(p) = out(p)$  for each predicate except that  $I(\mathbf{part}) = \{1\}$ . The first clause satisfies the requirements of Definition 14 as  $LI(\mathbf{app}([], Ys, Ys), I, O)$  is  $Ys \geq Ys$ , which obviously holds. Now consider the second clause.

$$LI(\mathbf{app}(Xs, Ys, Zs), I, O) := Xs + Ys \geq Zs \quad (2)$$

$$LI(\mathbf{app}([X|Xs], Ys, [X|Zs]), I, O) := 1 + X + Xs + Ys \geq 1 + X + Zs. \quad (3)$$

It is easy to see that inequality 2 implies inequality 3 satisfying the requirement 2 of Definition 14. The requirement 1 of Definition 14 obviously holds as  $1 + X + Xs + Ys > Xs + Ys$ .

It is easy to see that the third and the sixth clauses satisfy the requirements of Definition 14 (like the first clause). Let us now consider the fourth clause. For the recursive atom  $\mathbf{part}(Xs, H, Ls, Bs)$ , the requirement 1 of Definition 14 holds as  $1 + X + Xs > Xs$ . We now prove that requirement 2 also holds for this clause.

$$LI(X \leq H, I, O) := X + H \geq 0, \quad (4)$$

$$LI(\mathbf{part}(Xs, H, Ls, Bs), I, O) := Xs \geq Ls + Bs \quad (5)$$

$$LI(\mathbf{part}([X|Xs], H, [X|Ls], Bs), I, O) := 1 + X + Xs \geq 1 + X + Ls + Bs. \quad (6)$$

It is easy to see that inequality 5 implies inequality 6 satisfying the requirement 2 of Definition. 14. It can be similarly proved that the fifth clause satisfies the requirements.

Now consider the last clause.

$$LI(\mathbf{part}(L, H, A, B), I, O) := L \geq A + B, \quad (7)$$

$$LI(\mathbf{qs}(A, A1), I, O) := A \geq A1, \quad (8)$$

$$LI(\mathbf{qs}(B, B1), I, O) := B \geq B1, \quad (9)$$

$$LI(\mathbf{app}(A1, [H|B1], S), I, O) := A1 + 1 + H + B1 \geq S \quad (10)$$

and for the head  $qs([H|L], S)$  of the clause,  $LI(qs([H|L], S), I, O)$  is

$$1 + H + L \geq S. \tag{11}$$

It is easy to see that inequalities 7, 8, 9 and 10 together imply inequality 11 satisfying requirement 2 of Definition 14. The requirement 1 of Definition 14 holds for recursive atoms  $qs(A, A1)$  and  $qs(B, B1)$  as inequality 7 implies  $1 + H + L > A$  and  $1 + H + L > B$ . Therefore, **quick-sort** is a recursion bounded program.  $\square$

The above proof is very similar to the proof given in [7] that **quick-sort** is a linear-moded program. We presented it for pedagogical reasons — for the sake of completeness and to illustrate later that every proof of linear-modedness does not go as a proof of recursion boundedness.

The following example shows that the class of recursion bounded programs is rich enough to include programs with non-linear relationships between the input and output.

**Example 3.** Consider the following **nthpower** program.

moding: `add(in,in, out); mult(in,in, out) and nthpower(in,in, out)`

```

add(0, Y, Y) ←
add(s(X), Y, s(Z)) ← add(X, Y, Z)

mult(0, Y, 0) ←
mult(s(X), Y, Z) ← mult(X, Y, Z1), add(Y, Z1, Z)

nthpower(0, Y, 1) ←
nthpower(s(X), Y, Z) ← nthpower(X, Y, Z1), mult(Y, Z1, Z)

```

To prove that this program is  $\dots$ , take  $I$  and  $O$  as the mappings  $I(p) = in(p)$  for each predicate  $p \in \{\text{add}, \text{mult}, \text{nthpower}\}$ ,  $O(\text{add}) = out(\text{add})$  and  $O(\text{mult}) = O(\text{nthpower}) = \emptyset$ . It is easy to verify that the first, third and fifth clauses satisfy the requirements of Definition 14.

Let us now consider the second clause.

$$LI(\text{add}(X, Y, Z), I, O) := X + Y \geq Z \tag{12}$$

$$LI(\text{add}(s(X), Y, s(Z)), I, O) := 1 + X + Y \geq 1 + Z. \tag{13}$$

It is easy to see that inequality 12 implies inequality 13 satisfying the requirement 2 of Definition 14. The requirement 1 of Definition 14 obviously holds as  $1 + X + Y > X + Y$ .

Let us now consider the fourth clause.

$$LI(\text{mult}(X, Y, Z1), I, O) := X + Y \geq 0 \tag{14}$$

$$LI(\text{add}(Y, Z1, Z), I, O) := Y + Z1 \geq Z. \tag{15}$$

and for the head  $\text{mult}(s(X), Y, Z)$  of the clause,

$$LI(\text{mult}(s(X), Y, Z), I, O) := 1 + X + Y \geq 0. \tag{16}$$

It is easy to see that inequalities 14 and 15 together imply inequality 16 satisfying the requirement 2 of Definition 14 (in fact, inequality 16 is vacuously true, no need to use inequalities 14 and 15). The requirement 1 of Definition 14 obviously holds as  $1 + X + Y > X + Y$ .

Proving that the sixth clause satisfies the requirements of Definition 14 is very similar. Therefore, `nthpower` is a recursion bounded program.  $\square$

The above two examples may give an impression that the class of recursion bounded programs properly includes the class of linear-moded programs. The following example shows that it is not the case.

**Example 4.** Consider the following `merge-sort` program.

```

moding: split (in, out, out); merge (in, in, out) and
        ms (in, out)

split([], [], []) ←
split([X|Xs], [X|As], Bs) ← split(Xs, Bs, As)

merge([], Ys, Ys) ←
merge(Xs, [], Xs) ←
merge([X|Xs], [Y|Ys], [X|Zs]) ← X ≤ Y, merge(Xs, [Y|Ys], Zs)
merge([X|Xs], [Y|Ys], [Y|Zs]) ← X > Y, merge([X|Xs], Ys, Zs)

ms([], []) ←
ms([H|L], S) ← split([H|L], A, B), ms(A, A1), ms(B, B1),
                merge(A1, B1, S)
    
```

This program is shown in [7] to be, w.r.t. a mapping  $I$  such that  $I(p) = in(p)$  for each predicate. However, we cannot prove that `ms` is recursion bounded w.r.t.  $I$  and  $O$  such that  $I(p) = in(p)$  and  $O(p) = out(p)$  for each predicate, because  $1 + H + L \geq A + B$  does not imply  $1 + H + L > A$  and  $1 + H + L > B$  for recursive atoms  $ms(A, A1)$ ,  $ms(B, B1)$ .  $\square$

It may be noted that the above Prolog program `merge-sort` does not terminate for the well-moded query  $\leftarrow ms([1], S)$ . On the other hand, every recursion bounded program terminates for all well-moded queries, as shown in the next section. The main reason for the non-termination of the above `merge-sort` program is the instantiation of variable `A` in the last clause to `[1]` for the well-moded query  $\leftarrow ms([1], S)$ , as `split([1], A, B)` binds `A` to `[1]`. This is also the reason why the above `merge-sort` program is not recursion bounded.

The following terminating Prolog program for `merge-sort` is recursion bounded.

**Example 5.** Consider the program obtained by replacing the last clause in the above program with the following two clauses.

$$\begin{aligned} \text{ms}([X], [X]) &\leftarrow \\ \text{ms}([X1, X2|L], S) &\leftarrow \text{split}(L, A, B), \text{ms}([X1|A], A1), \text{ms}([X2|B], B1), \\ &\quad \text{merge}(A1, B1, S) \end{aligned}$$

This program is recursion bounded as the inequality  $LI(\text{split}(L, A, B), I, O)$ , i.e.,  $L \geq A + B$  implies  $2 + X1 + X2 + L > 1 + X1 + A$  and  $2 + X1 + X2 + L > 1 + X2 + B$  for the two recursive atoms  $\text{ms}(A, A1)$ ,  $\text{ms}(B, B1)$  —it is easy to see that the other clauses are recursion bounded.  $\square$

## 4 Some Properties of Recursion Bounded Programs

In this section, we prove some properties of recursion bounded programs.

A nice property of the class of recursion bounded programs is that it is decidable whether a given well-moded program  $P$  is recursion bounded. We prove this result by first proving that it is decidable whether a given moded program  $P$  is recursion bounded w.r.t. a given pair of mappings  $I$  and  $O$ .

**Theorem 2.** It is decidable whether a well-moded program  $P$  is recursion bounded w.r.t. a given pair of mappings  $I$  and  $O$  satisfying  $I(p) \subseteq in(p)$  and  $O(p) \subseteq out(p)$  for each predicate  $p$  in  $P$ .

*Proof:* Follows from the fact that this problem can be reduced to the satisfiability problem of linear inequalities.  $\square$

**Theorem 3.** It is decidable whether a well-moded program  $P$  is recursion bounded or not.

*Proof:* Since only finitely many choices are possible for  $I$  and  $O$ , we can check if  $P$  is recursion bounded w.r.t. at least one such pair of mappings  $I$  and  $O$ .  $\square$

The following theorem states the central idea behind the concept of recursion bounded programs.

**Theorem 4.** Let  $P$  be a recursion bounded program w.r.t. a pair of mappings  $I$  and  $O$ , and  $\leftarrow A$  be a well-moded query. If there is an LD-refutation  $G$  of  $P \cup \{\leftarrow A\}$  with computed answer substitution  $\sigma$ , then  $LI(A\sigma, I, O) \blacktriangleright \dots$

*Proof:* See full paper.

The following theorem shows that the recursive calls get smaller and smaller in LD-derivations of recursion bounded programs.

**Theorem 5.** Let  $P$  be a recursion bounded program w.r.t. a pair of mappings  $I$  and  $O$ , and  $\leftarrow A$  be a well-moded query. If  $G = Q_0, Q_1, \dots, Q_n$  is an LD-derivation of  $P \cup \{\leftarrow A\}$  with partial computed answer substitution  $\sigma$  such that  $Q_n = \leftarrow A', \dots$  and  $A'$  is the first selected atom in  $G$  satisfying  $rel(A') \succeq_P rel(A)$ , then  $[Items(A, I)] > [Items(A', I)]$

*Proof:* See full paper.

Termination of recursion bounded programs follows from this theorem.

**Theorem 6.** If  $P$  is a recursion bounded program and  $\leftarrow A$  is a well-moded query, every LD-derivation of  $P \cup \{\leftarrow A\}$  is of finite length.

*Proof:* Follows from the above theorem by noetherian induction. □

Decidability of  $A \in M(P)$  for recursion bounded programs follows from this theorem.

**Theorem 7.** For any recursion bounded program  $P$  and ground atom  $A$ , it is decidable whether  $A \in M(P)$  or not.

*Proof:* Follows from the termination of recursion bounded programs. □

**Remark 3.** The main reason for using the strict inequality  $[Iterms(A_0, I)] > [Iterms(A_j, I)]$  in condition 1 of Definition 14 is to ensure termination and hence decidability of  $A \in M(P)$ .

Since the size of input terms in non-recursive atoms in LD-derivations is not bounded by the size of input terms of the initial query for recursion bounded programs (unlike the case with linear-moded programs), the proof techniques used in [7] do not work here.

## 5 Inferability of Recursion Bounded Programs from Positive Data

In this section, we establish inductive inferability of recursion bounded programs from positive data.

**Definition 15.** Let  $RB_k$  be the set of all recursion bounded clauses of size at most  $k$  and  $M_c$  be a semantic mapping such that  $M_c(P)$  is the set of all atoms of the target predicate  $c$  in the least Herbrand model of  $P$ . The concept defining framework  $\langle B(c), RB_k, M_c \rangle$  is denoted by **RB<sub>k</sub>**.

**Lemma 4.**  $\forall k \geq 1, \forall \langle B(c), RB_k, M_c \rangle \in \mathbf{RB}_k, \exists P \in \mathbf{RB}_k$  such that  $M_c(P) = M_c(B(c))$ .

*Proof:* By Theorem 7, it is decidable whether a ground atom  $A \in M(P)$  for any recursion bounded program  $P$ . For a given alphabet, there are at most finitely many choices of mode declarations and the mappings  $I$  and  $O$ . Hence, by Theorem 3, we can effectively enumerate all the recursion bounded programs in  $RB_k$ . This completes the proof. □

The following theorem plays the predominant role in proving our main result.

**Theorem 8.**  $\forall k \geq 1, \forall \langle B(c), RB_k, M_c \rangle \in \mathbf{RB}_k, \exists P \in \mathbf{RB}_k$  such that  $M_c(P) = M_c(B(c))$ .

*Proof*: Consider a finite set  $S \subseteq B(c)$  and a program  $P \subseteq RB_k$  be a recursion bounded program w.r.t. some  $I$  and  $O$  containing at most  $m \geq 1$  clauses such that  $P$  is reduced w.r.t.  $S$ . Let  $n$  be an integer such that  $n \geq [Iterms(c(\mathbf{u}; \mathbf{v}), I)]$  for every atom  $c(\mathbf{u}; \mathbf{v}) \in S$ . Let  $S' = \{c(\mathbf{w}_1; \mathbf{w}_2) \mid c(\mathbf{w}_1; \mathbf{w}_2) \text{ is a selected atom of the (target) predicate } c \text{ in an LD-refutation of an atom } A \text{ in } S\}$ . By Theorem 5,  $n \geq [Iterms(c(\mathbf{w}_1; \mathbf{w}_2), I)]$  for every atom  $c(\mathbf{w}_1; \mathbf{w}_2) \in S'$ . Since  $P$  is reduced w.r.t.  $S$ , every clause in  $P$  is used in LD-refutations of atoms in  $S$ . Hence,  $n \geq [Iterms(c(\mathbf{s}; \mathbf{t}), I)]$  for every atom  $c(\mathbf{s}; \mathbf{t})$  in any clause in  $P$ .

Since  $\Pi$  and  $\Sigma$  are finite, there are only finitely many recursion bounded programs containing at most  $m$  clauses of size at most  $k$  with  $n \geq [Iterms(c(\mathbf{s}; \mathbf{t}), I)]$  for every atom  $c(\mathbf{s}; \mathbf{t})$  of the target predicate  $c$  in them (except for the renaming of variables). Therefore, the set  $\{M_c(P) \mid P \subseteq RB_k, P \text{ is reduced w.r.t. } S \text{ and contains at most } m \text{ clauses}\}$  is finite. It is obvious that  $M_c$  is monotonic. Hence,  $\mathbf{RB}_k$  has bounded finite thickness.  $\square$

From this Theorem, Lemma 4 and Theorem 1, we obtain our main result.

**Theorem 9.**  $\forall m \geq 1, \forall c, \mathbf{RB}_k^m = \{M_c(P) \mid P \subseteq RB_k, |P| \leq m\}$

**Remark 4.** Note that the restriction that the size of each clause in  $RB_k$  is less than or equal to  $k$  is crucial for the above two theorems. The restriction on body-length used in [4, 7] for linear covering and linear-moded programs is not enough for recursion bounded programs as condition 1 of Definition 14 do not place any restriction on non-recursive atoms in a clause. The size of non-recursive atoms in recursion bounded clauses is indirectly restricted by the size of the clause ( $k$  in  $RB_k$ ). Essentially, bounded finite thickness will be violated if no upper bound is placed on the size of recursion bounded clauses.

**Example 6.** By the above Theorem, the concepts of **quick-sort**, **merge-sort** and **nthpower** are inferable from their positive presentations as (1) **quick-sort**  $\in \mathbf{RB}_{22}^7$ , (2) **merge-sort**  $\in \mathbf{RB}_{25}^9$  and (3) **nthpower**  $\in \mathbf{RB}_{13}^6$ .  $\square$

## 6 Comparison with Related Works

In this section, we compare the class of recursion bounded programs with other classes known to be inferable from positive data, namely, linear [14, 15], linear-covering [4], linear-moded [7] and safe programs [12].

1. Krishna Rao [7] proved that the class of linear-moded programs includes both the classes of linear and linear-covering programs. The program **nthpower** is recursion bounded but not linear-moded (because of nonlinear output), while program **merge-sort** given in Example 4 is linear-moded but not recursion bounded. Therefore, the classes of linear-moded and recursion bounded programs are incomparable. However, all the known **terminating** linear-moded programs are recursion bounded.



2. Martin and Sharma [12] studied inferability of Prolog programs from positive data using very different concepts –bound kits rather than modes. The class of safe programs is the largest syntactic class given in [12]. The program `nthpower` given above is recursion bounded but not safe<sup>1</sup> [11], while program `merge-sort` given in Example 4 is safe but not recursion bounded. Therefore, the classes of safe and recursion bounded programs are incomparable.

## 7 Conclusion

In this paper, we study inductive inference of Prolog programs from positive data and present a class of Prolog programs that can capture nonlinear relationships between inputs and outputs, and yet inferable from positive data. This class of Prolog programs is rich enough to contain many programs from Sterling and Shapiro’s book [17] including `append`, `reverse`, `permute`, `delete`, `insert`, `merge`, `split`, `count`, `listsum`, `listproduct`, `merge-sort`, `quick-sort`, `insertion-sort` on lists, `preorder` and `inorder` traversal of binary trees, polynomial recognition, derivatives and `addition`, `multiplication`, `factorial`, `power` on natural numbers.

**Acknowledgements.** The author would like to thank Eric Martin for his clarifications on safe programs, and King Fahd University of Petroleum and Minerals for the generous support provided by it in conducting this research.

## References

- [1] D. Angluin (1980), *Inductive inference of formal languages from positive data*, Information and Control **45**, pp. 117-135.
- [2] K.R. Apt and D. Pedreschi (1993), *Reasoning about termination of pure Prolog programs*, Information and Computation **106**, pp. 109-157.
- [3] H. Arimura, T. Shinohara and S. Otsuki (1994), *Finding Minimal Generalizations for Unions of Pattern Languages and Its Application to Inductive Inference from Positive Data*, Proc. of STACS’94, LNCS **775**, pp. 649-660.
- [4] H. Arimura and T. Shinohara (1994), *Inductive inference of Prolog programs with linear data dependency from positive data*, Proc. Information Modelling and Knowledge Bases V, pp. 365-375, IOS press.
- [5] L. Blum and M. Blum (1975), *Towards a mathematical theory of inductive inference*, Information and Control **28**, pp. 125-155.
- [6] E.M. Gold (1967), *Language identification in the limit*, Information and Control **10**, pp. 447-474.
- [7] M. R. K. Krishna Rao (2000), *Some classes of Prolog programs inferable from positive data*, Theor. Comput. Sci. **241**, pp. 211-234.
- [8] M.R.K. Krishna Rao (2004), *Inductive inference of term rewriting systems from positive data*, Proc. of Algorithmic Learning Theory, ALT’2004, Lecture Notes in Artificial Intelligence **3244**, pp. 69-82.

---

<sup>1</sup> A slightly different program for `nthpower` given in [12] is safe. It uses `adminus1` predicate (with semantics  $Z=X+Y-1$  whenever `adminus1(X, Y, Z)` is true) instead of the natural `addition` predicate.

- [9] M.R.K. Krishna Rao and R. K. Shyamasundar (1995), *Unification-free execution of well-moded Prolog programs*, Proc. of International Static Analysis Symposium, SAS'95, Lecture Notes in Computer Science **983**, pp. 243-260, Springer-Verlag.
- [10] J. W. Lloyd (1987), *Foundations of Logic Programming*, Springer-Verlag.
- [11] E. Martin (2005), *Personal communication*.
- [12] E. Martin and A. Sharma (1999), *On Sufficient Conditions for Learnability of Logic Programs from Positive Data*, Proc. of Inductive Logic Programming, ILP'1999, Lecture Notes in Computer Science **1634**, pp. 198-209.
- [13] E. Shapiro (1981), *Inductive inference of theories from facts*, Tech. Rep., Yale Univ.
- [14] E. Shapiro (1983), *Algorithmic Program Debugging*, MIT Press.
- [15] T. Shinohara (1991), *Inductive inference of monotonic formal systems from positive data*, New Generation Computing **8**, pp. 371-384.
- [16] T. Shinohara, H. Arimura (2000), *Inductive inference of unbounded unions of pattern languages from positive data*, Theor. Comput. Sci. **241**, pp. 191-209.
- [17] L. Sterling and E. Shapiro (1994), *The Art of Prolog*, MIT Press.

# Absolute Versus Probabilistic Classification in a Logical Setting

Sanjay Jain<sup>1,\*</sup>, Eric Martin<sup>2,3,\*\*</sup>, and Frank Stephan<sup>4,\*</sup>

<sup>1</sup> School of Computing, National University of Singapore,  
Singapore 117543, Republic of Singapore  
[sanjay@comp.nus.edu.sg](mailto:sanjay@comp.nus.edu.sg)

<sup>2</sup> School of Computer Science and Engineering, The University of New South Wales,  
UNSW Sydney NSW 2052, Australia  
[emartin@cse.unsw.edu.au](mailto:emartin@cse.unsw.edu.au)

<sup>3</sup> National ICT Australia, UNSW Sydney NSW 2052, Australia

<sup>4</sup> School of Computing and Department of Mathematics,  
National University of Singapore, Singapore 117543, Republic of Singapore  
[fstephan@comp.nus.edu.sg](mailto:fstephan@comp.nus.edu.sg)

**Abstract.** Given a set  $\mathcal{W}$  of logical structures, or *possible worlds*, a set  $\mathcal{D}$  of logical formulas, or *possible data*, and a logical formula  $\varphi$ , we consider the classification problem of determining in the limit and *almost always correctly* whether a possible world  $\mathfrak{M}$  satisfies  $\varphi$ , from a complete enumeration of the possible data that are true in  $\mathfrak{M}$ . One interpretation of *almost always correctly* is that the classification might be wrong on a set of possible worlds of measure 0, w.r.t. some natural probability distribution over the set of possible worlds. Another interpretation is that the classifier is only required to classify a set  $\mathcal{W}'$  of possible worlds of measure 1, without having to produce any claim in the limit on the truth of  $\varphi$  in the members of the complement of  $\mathcal{W}'$  in  $\mathcal{W}$ . We compare these notions with absolute classification of  $\mathcal{W}$  w.r.t. a formula that is almost always equivalent to  $\varphi$  in  $\mathcal{W}$ , hence investigate whether the set of possible worlds on which the classification is correct is definable. Finally, in the spirit of the kind of computations considered in Logic programming, we address the issue of computing *almost correctly* in the limit witness to leading existentially quantified variables in existential formulas.

## 1 Introduction

Paradigms of inductive inference are often highly idealized, even for those that impose very tight restrictions on the learning scenario. One of the reasons for the

---

\* Sanjay Jain is supported by the NUS research grant R252-000-127-112; Frank Stephan is supported by the NUS research grant R252-000-212-112.

\*\* National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence Program.

idealization is that learners have to be correct w.r.t. all possible realities (languages in the numerical setting, structures in the logical setting) of the paradigm. Allowing the learning process to succeed w.r.t. *all* realities—intuitively, successfully learning with probability 1—appears as a requirement that deserves to be investigated.

Probabilistic elements have already been considered in inductive inference, but they relate more to the learning process than to the class of languages learnt by a machine (sample references are [6, 7, 11, 12, 14]). For example, learning functions in the limit with probability  $1/n$  turns out to be equivalent to having  $n$  nonprobabilistic learners such that at least one of them succeeds [12, 14]. Furthermore, most concepts have a break-even point at some probability  $c < 1$  in the sense that whenever such concepts are learnable with probability  $c$ , they are already learnable by a deterministic machine [1]. Meyer [10] showed that exact monotonic and exact conservative learning with any probability  $c < 1$  is more powerful than deterministic learning; still in case  $c = 1$ , the probabilistic and deterministic variants are again the same.

An example for a setting in inductive inference where learning with probability 1 is more powerful than deterministic learning is the following. Assign to each set  $A$  to be learnt the distribution  $p_A$  with  $p_A(x) = 2^{-1-x}$  for  $x \in A$ ,  $p_A(x) = 0$  for  $x \notin A$  and  $p_A(\#) = \sum_{x \notin A} 2^{-1-x}$ . Then any class of sets that is learnable from informant is also learnable from text with probability 1, provided that for every member  $A$  of the class, the elements of a text for  $A$  are drawn with probability  $p_A$ . As some classes are learnable from informant but not from text, these classes witness that learning from almost all texts is more powerful than learning from all texts.

The main reason why probabilistic elements are restricted to the learning process and not to the class of realities being considered is that in a countable domain, ‘almost all objects’ would normally mean ‘cofinitely many objects’ and finite exceptions can often be handled by suitably patching the machine. Therefore it is much more appropriate to consider classification where one deals with a continuum of possible realities which can be identified with the Cantor-Space. Then ‘almost all’ can be interpreted in two major ways: ‘of second category’ as defined in topology, or ‘of measure 1’ as defined in measure theory.

Classification was already implicitly considered by recursion theorists when they investigated computation in the limit relative to an oracle, which subsumes classification [13]. Ben-David [3] characterized classification in the limit in topological terms. Subsequent work then established a connection between classification on one side and logic on the other side [2, 8]. In [9] the relationship between classification and topology was brought one step further, by casting the classification in a logical setting that considered arbitrary sets of data, each set determining a particular topology, and arbitrary sets of structures. Of particular importance in [9] are (usually axiomatized) classes of structures consisting of Henkin structures only, where every individual of the domain is denoted by a term in the underlying language. The set of atomic sentences (closed atomic formulas) true in such structures uniquely determines the structure, and can be

identified with a point in the Cantor space. A probability distribution can then be defined on the set of structures which represent the possible realities, that generalizes the classical probability distribution on the Cantor space.

The setting chosen for this paper is a particular instance of the logical framework investigated in [9]. It conceives of a logical paradigm as a vocabulary  $\mathcal{V}$ , a set  $\mathcal{W}$  of  $\mathcal{V}$ -structures, or  $\langle \mathcal{V}, \mathcal{W} \rangle$ , and a set  $\mathcal{D}$  of closed  $\mathcal{V}$ -formulas, or  $\langle \mathcal{V}, \mathcal{W}, \mathcal{D} \rangle$ . Important cases of possible data are obtained by taking for  $\mathcal{D}$  the set of atomic sentences or the set of basic sentences (atomic sentences or their negations). Both choices determine counterparts to the numerical notions of text and informant, in the form of enumerations of all possible data that are true in an underlying possible world  $\mathfrak{M}$ , yielding an  $\langle \mathcal{V}, \mathcal{W}, \mathcal{D} \rangle$  for  $\mathfrak{M}$ . Given a formula  $\varphi$ , we consider the task of determining in the limit, from an environment for a possible world  $\mathfrak{M}$ , whether  $\mathfrak{M}$  satisfies  $\varphi$ . In other words, the task is to classify a possible world as a member of one of two classes: the class of structures that satisfy  $\varphi$ , and the class of structures that don't. But we allow the classification to fail on a set of environments for a small set of possible worlds—either of first category or of measure 0.

It is important to distinguish between failing to converge to some answer and misclassifying. In the case of misclassification w.r.t  $\varphi$ , an interesting question is whether perfect classification of  $\mathcal{W}$  is achieved on the basis of another formula  $\psi$ , whose set of models in  $\mathcal{W}$  is of course equal to the set of models of  $\varphi$  up to a set of measure 0. Whether the failure to classify correctly is due to nonconvergence or to genuine misclassification, we only measure on which class of possible worlds  $\mathfrak{M}$  a correct classification is achieved from all environments for  $\mathfrak{M}$ . We do not assume that the possible data are generated following some underlying probability distribution, nor do we impose any condition on the speed of convergence. In other words, we remain in the realm of inductive inference and our use of probabilities is essentially different to its use in the PAC framework.

We now proceed as follows. In Section 2 we introduce the basic notions, that we apply to the classification task in Sections 3 to 5. In Section 6 we particularize the framework to  $\varphi$  being of the form  $\exists x\psi(x)$ , with the aim of not only classifying  $\varphi$ , but of computing in the limit a witness to the existentially quantified variable  $x$ . Provided that  $\mathcal{W}$  is axiomatizable by a logic program, this correspond to ‘error tolerant’ computations in Logic programming, where  $\psi$  is assumed to be quantifier free or to only contain bounded quantifiers [16]. When  $\psi$  is universal, and also with some assumptions on  $\mathcal{W}$ , this corresponds to ‘error tolerant’ computations in Limiting resolution [5]. We conclude in Section 7.

## 2 Absolute and Probabilistic Classification

Let a class  $X$  be given. The class of finite sequences of members of  $X$ , including the empty sequence  $()$ , is represented by  $X^*$ . The length of  $\sigma \in X^*$  is denoted  $lt(\sigma)$ . The class of sequences of members of  $X$  of length  $\omega$  is represented by  $X^\omega$ .

Given a nonempty vocabulary  $V, \mathfrak{F}$ , a set of (possibly nullary) predicate and function symbols, we shall consider both  $\langle V, \mathfrak{F} \rangle$  and

...  $V$ , built from the symbols in  $V$ , equality, the usual Boolean operators, first-order variables and quantifiers over those, and in the case of monadic second-order formulas, unary predicate variables and quantifiers over those. A (first-order or monadic second-order) ...  $V$  refers to a closed (first-order or monadic second-order) formula over  $V$ . We fix:

- a vocabulary  $\mathcal{V}$  containing at least a constant  $\bar{0}$ , a unary function symbol  $s$  and a unary predicate symbol  $P$ ;
- a language  $\mathcal{L}$  equal either to the set of first-order sentences over  $\mathcal{V}$  or to the set of monadic second-order sentences over  $\mathcal{V}$ ;
- a set  $\mathcal{D}$  of first-order sentences over  $\mathcal{V}$ , referred to as ...

Given  $n \in \mathbb{N}$ , we denote by  $\bar{n}$  the term obtained from  $\bar{0}$  by  $n$  applications of  $s$  (hence  $\bar{n} + \bar{1} = s(\bar{n})$  for all  $n \in \mathbb{N}$ ). We say ... for  $\mathcal{V}$ -structure, ... for term over  $\mathcal{V}$ , and ... for member of  $\mathcal{L}$ . A structure  $\mathfrak{M}$  is said to be ... iff  $\mathfrak{M}$ 's individuals are the nonempty sets of closed terms that they interpret. Given  $T \subseteq \mathcal{L}$ ,  $\text{Mod}_{\mathcal{W}}(T)$  represents the set of models of  $T$  in  $\mathcal{W}$ . Given  $\varphi \in \mathcal{L}$ , we write  $\text{Mod}_{\mathcal{W}}(\varphi)$  for  $\text{Mod}_{\mathcal{W}}(\{\varphi\})$ .

**Definition 1.** Given  $X \subseteq \mathcal{L}$  and a structure  $\mathfrak{M}$ , we define the  $X$  ...  $\mathfrak{M}$ , denoted  $\text{Diag}_X(\mathfrak{M})$ , as the set of all members of  $X$  that are true in  $\mathfrak{M}$ .

We fix a class  $\mathcal{W}$  of structures, referred to as ... with the following property. Put  $X = \{P(\bar{n}) : n \in \mathbb{N}\}$ . Then every subset of  $X$  is the  $X$ -diagram of some member of  $\mathcal{W}$ .

We use ... to refer to an enumeration of a member of  $\mathcal{W}$ :

**Definition 2.** Given a possible world  $\mathfrak{M}$ , an ...  $\mathfrak{M}$  is any member  $e$  of  $(\mathcal{D} \cup \{\#\})^\omega$  such that for all  $\varphi \in \mathcal{D}$ ,  $\varphi$  occurs in  $e$  iff  $\varphi$  belongs to  $\text{Diag}_{\mathcal{D}}(\mathfrak{M})$ .

Put  $X = \{P(\bar{n}) : n \in \mathbb{N}\}$ . The  $X$ -diagram of a possible world can be identified with a point in the Cantor space. The next definition makes the relationship explicit and maps the natural measure on the Cantor space to a measure on  $\mathcal{W}$ .

**Definition 3.** Given a possible world  $\mathfrak{M}$ , the ...  $\mathfrak{M}$  is the (unique) member  $e$  of  $\{0, 1\}^\omega$  such that for all  $n \in \mathbb{N}$ ,  $e(n) = 1$  iff  $\mathfrak{M} \models P(\bar{n})$ .

Let a set  $W$  of possible worlds be given, and let  $X$  be the set of standard informants of the members of  $W$ .

- If  $X$  is Lebesgue measurable in the Cantor space and of measure  $m$  then we put  $\mu(W) = m$ .
- If  $X$  is of first, respect., second, category in the Cantor space then we say that  $W$  is of first, respect., second, category.

Note that in case  $\mathcal{D} = \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\}$ , the standard informant for  $\mathfrak{M}$  can be identified with an environment for  $\mathfrak{M}$ .

**Definition 4.** Two sentences  $\varphi$  and  $\psi$  are said to be ... iff  $\mu(\text{Mod}_{\mathcal{W}}(\varphi \leftrightarrow \neg\psi))$  is defined and null.

Given  $\sigma \in (\mathcal{D} \cup \{\#\})^*$ ,  $\text{cnt}(\sigma)$  denotes the set of members of  $\mathcal{D}$  that occur in  $\sigma$ . The proofs of many propositions will make use of the next technical definition.

**Definition 5.** We say that a member  $\sigma$  of  $(\mathcal{L} \cup \{\#\})^*$  is *classified* by  $\mathcal{W}$  just in case there exists a member  $\mathfrak{M}$  of  $\mathcal{W}$  such that  $\mathfrak{M} \models \text{cnt}(\sigma)$ .

The classification task will be performed by a classifier, defined next.

**Definition 6.** Given a set  $X$  of sentences, an *X-classifier* is a partial function from  $(X \cup \{\#\})^*$  into  $\{0, 1\}$ . We say *f* for *D*-classifier.

The following pair of definitions capture the absolute notion of classification.

**Definition 7.** Let a classifier  $f$  and a subset  $\mathcal{W}'$  of  $\mathcal{W}$  be given.

Given a subset  $X$  of  $\mathcal{W}$ , we say that  $f$  *classifies*  $\mathcal{W}'$  in the limit following  $X$  just in case for all  $\mathfrak{M} \in \mathcal{W}'$  and environments  $e$  for  $\mathfrak{M}$ :

- $\mathfrak{M} \in X$  iff  $\{\sigma \in (\mathcal{D} \cup \{\#\})^* : \sigma \subset e \text{ and } f(\sigma) = 1\}$  is cofinite;
- $\mathfrak{M} \notin X$  iff  $\{\sigma \in (\mathcal{D} \cup \{\#\})^* : \sigma \subset e \text{ and } f(\sigma) = 0\}$  is cofinite.

Given a sentence  $\varphi$ , we say that  $f$  *classifies*  $\mathcal{W}'$  in the limit following  $\varphi$  iff  $f$  classifies  $\mathcal{W}'$  in the limit following  $\text{Mod}_{\mathcal{W}}(\varphi)$ .

**Definition 8.** Given  $\varphi \in \mathcal{L}$  and  $\mathcal{W}' \subseteq \mathcal{W}$ , we say that  $\mathcal{W}'$  is *absolutely classified*, *classified*, *probabilistically classified*, *classified in the limit*, *classified in the limit following*  $\varphi$  iff some classifier, *respect.*, *computable classifier*, *classifies*  $\mathcal{W}'$  in the limit following  $\varphi$ .

We are interested in classifiers that classify all possible worlds, but misclassify a subset of  $\mathcal{W}$  of measure 0, as captured in the next pair of definitions.

**Definition 9.** Let a classifier  $f$  and a sentence  $\varphi$  be given. We say that  $f$  *almost classifies*  $\mathcal{W}$  in the limit following  $\varphi$  iff there exists a subset  $X$  of  $\mathcal{W}$  such that:

- $\mu(\text{Mod}_{\mathcal{W}}(\varphi) \Delta X)$  is defined and null;
- $f$  classifies  $\mathcal{W}$  in the limit following  $X$ .

**Definition 10.** Given  $\varphi \in \mathcal{L}$ , we say that  $\mathcal{W}$  is *absolutely classified almost everywhere*, *classified almost everywhere*, *probabilistically classified almost everywhere*, *classified almost everywhere in the limit*, *classified almost everywhere in the limit following*  $\varphi$  iff some classifier, *respect.*, *computable classifier*, *classifies*  $\mathcal{W}$  in the limit following  $\varphi$  almost everywhere.

### 3 Failing to Classify Versus Misclassifying

We start with the simple observation that in measure-theoretic terms, misclassification of a small set of possible worlds implies absolute classification of almost all possible worlds:

**Property 11.** Let  $\varphi \in \mathcal{L}$  and  $\mathcal{W}$  be given. Let  $\mathcal{W}' \subseteq \mathcal{W}$  be such that  $\mu(\mathcal{W}') = 1$  and  $\mathcal{W}'$  is *absolutely classified*, *classified*, *probabilistically classified*, *classified in the limit*, *classified in the limit following*  $\varphi$ .

Still the converse of Property 11 does not necessarily hold. Indeed, a classifier that correctly classifies a subset  $\mathcal{W}'$  of  $\mathcal{W}$  of measure 1 might be forced not to converge on some environments for some members of  $\mathcal{W} \setminus \mathcal{W}'$ . The next proposition shows that this might indeed happen.

**Proposition 12.** *Let  $\mathcal{V} = \langle \overline{0}, s, +, \langle \rangle \rangle$  be a Presburger arithmetic,  $\mathcal{D} = \{P(\overline{n}), \neg P(\overline{n}) : n \in \mathbb{N}\}$  a set of formulas,  $\varphi$  a formula, and  $\mathcal{W}' \subseteq \mathcal{W}$  a subset of environments with  $\mu(\mathcal{W}') = 1$ . Then there exists a classifier  $f$  such that  $f$  outputs 1 in response to cofinitely many initial segments of  $e$  for all  $e \in \mathcal{W}'$  and  $f$  outputs 0 in response to cofinitely many initial segments of  $e$  for all  $e \in \mathcal{W} \setminus \mathcal{W}'$ .*

**Proof.** Write  $x \leq y$  for  $\exists z(x + z = y)$  and  $x < y$  for  $x \leq y \wedge x \neq y$ , and define

- a formula  $\psi(x)$  whose meaning is that  $P(y)$  holds for all  $y \in (\frac{x}{2}, x)$ ;
- a sentence  $\varphi$  whose meaning is that  $\psi(x)$  holds for finitely many  $x$ 's only, and the maximum  $x$  such that  $\psi(x)$  holds is even. Formally:

$$\begin{aligned} \psi(x) &\equiv \forall y((x < y + y \wedge y < x) \rightarrow P(y)) \\ \varphi &\equiv \exists x(\psi(x + x) \wedge \forall y(\psi(y) \rightarrow y \leq x + x)) \end{aligned}$$

Note that for all Henkin models  $\mathfrak{M}$  of Presburger arithmetics, the reduct of  $\mathfrak{M}$  to  $\{\overline{0}, s, +, \langle \rangle\}$  is isomorphic to  $\mathbb{N}$  with the standard interpretation of  $\overline{0}$ ,  $s$ ,  $+$  and  $\langle \rangle$ . Also note that for all  $\mathfrak{M} \in \mathcal{W}$ ,  $\mathfrak{M} \models \varphi$  iff for all  $\mathfrak{N} \in \mathcal{W}$  with  $\text{Diag}_{\mathcal{D}}(\mathfrak{N}) = \text{Diag}_{\mathcal{D}}(\mathfrak{M})$ ,  $\mathfrak{N} \models \varphi$  iff  $\mathfrak{M} \models \varphi$ . Let a computable classifier  $f$  be defined as follows. Let a member  $\sigma$  of  $(\mathcal{D} \cup \{\#\})^*$  be given. Let  $n \in \mathbb{N}$  be maximal such that for all  $m < n$ , either  $P(\overline{m})$  or  $\neg P(\overline{m})$  occurs in  $\sigma$ , and all models of  $\text{cnt}(\sigma) \cap \{P(\overline{m}), \neg P(\overline{m}) : m < n\}$  in  $\mathcal{W}$  are models of  $\psi(\overline{n})$ . Put  $f(\sigma) = 1$  if  $n$  is even, and put  $f(\sigma) = 0$  otherwise. Let  $X$  be the set of all  $\mathfrak{M} \in \mathcal{W}$  for which there exists infinitely many  $n \in \mathbb{N}$  with  $\mathfrak{M} \models \psi(\overline{n})$ . It is immediately verified that  $\mu(X) = 0$  and:

- for all  $\mathfrak{M} \in \text{Mod}_{\mathcal{W}}(\varphi)$  and environments  $e$  for  $\mathfrak{M}$ ,  $f$  outputs 1 in response to cofinitely many initial segments of  $e$ ;
- for all  $\mathfrak{M} \in \text{Mod}_{\mathcal{W}}(\neg\varphi) \setminus X$  and environments  $e$  for  $\mathfrak{M}$ ,  $f$  outputs 0 in response to cofinitely many initial segments of  $e$ .

This shows that  $\mathcal{W} \setminus X$  is computably classifiable in the limit following  $\varphi$ .

Let a classifier  $g$  be given. Suppose for a contradiction that  $g$  classifies  $\mathcal{W}$  in the limit following some subset  $Y$  of  $\mathcal{W}$  with  $\mu(\text{Mod}_{\mathcal{W}}(\varphi) \Delta Y) = 0$ . Note that for all  $\sigma \in \mathcal{D}^*$  that are consistent in  $\mathcal{W}$ , neither  $\mu(\text{Mod}_{\mathcal{W}}(\text{cnt}(\sigma) \cup \{\varphi\}))$  nor  $\mu(\text{Mod}_{\mathcal{W}}(\text{cnt}(\sigma) \cup \{\neg\varphi\}))$  is equal to 0, hence that  $g$  has to output 1 in response to some extension  $\sigma_1$  of  $\sigma$  in  $\mathcal{D}^*$ , and 0 in response to another extension  $\sigma_2$  of  $\sigma$  in  $\mathcal{D}^*$ , with both  $\text{cnt}(\sigma_1)$  and  $\text{cnt}(\sigma_2)$  being consistent in  $\mathcal{W}$ . Also note that for all  $\sigma \in \mathcal{D}^*$  such that  $\text{cnt}(\sigma)$  is consistent in  $\mathcal{W}$  and for all  $m \in \mathbb{N}$ , there exists  $n > m$  such that  $\text{cnt}(\sigma) \cup \{\psi(\overline{n})\}$  is consistent in  $\mathcal{W}$ . Using these observations, it is easy to construct an environment  $e$  for a member of  $X$  such that  $g(\sigma)$  is



equal to 1 for infinitely many initial segments  $\sigma$  of  $e$ , and equal to 0 for infinitely many initial segments  $\sigma$  of  $e$ . Contradiction. ■

Considering only computable classification, as opposed to noncomputable classification, a similar result to Proposition 12 can be established using Peano arithmetics instead of Presburger arithmetics:

**Proposition 13.**  $\mathcal{V} \text{ is a } \bar{0} \text{ s.t. } P \text{ is a } \bar{0} \text{ s.t. } P$   
 $\mathcal{W} \text{ is a } \bar{0} \text{ s.t. } P \text{ is a } \bar{0} \text{ s.t. } P$   
 $\mathcal{D} = \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\}$   
 $\varphi$   
 $\mathcal{W}' \text{ is a } \bar{0} \text{ s.t. } P \text{ is a } \bar{0} \text{ s.t. } P$   
 $\mu(\mathcal{W}') = 1$   
 $\varphi$   
 $\varphi$

**Proof.** Let  $(\phi_e)_{e \in \mathbb{N}}$  denote an acceptable indexing of the unary partial recursive functions from  $\mathbb{N}$  into  $\{0, 1\}$ . Given  $e \in \mathbb{N}$ , let  $\psi(\bar{e})$  be a formula expressing that for all  $y \in \mathbb{N}$ , if  $\phi_e(y)$  is defined then  $\phi_e(y) = 1$  iff  $P(y)$  holds. Given  $e, x \in \mathbb{N}$ , let  $\chi(\bar{e}, \bar{x})$  be a formula which expresses that  $\phi_e(x)$  is undefined. Define  $\varphi$  as

$$\exists e \exists x (P(x) \wedge P(e) \wedge \chi(e, x) \wedge \psi(e) \wedge \forall y (y < e \rightarrow \neg P(y))).$$

Let a classifier  $f$  have the following properties. For all  $\sigma \in (\mathcal{D} \cup \{\#\})^*$ , if  $\sigma$  contains no possible datum of the form  $P(\bar{n})$  then  $f(\sigma) = 0$ . Let  $\sigma \in (\mathcal{D} \cup \{\#\})^*$  and  $n \in \mathbb{N}$  be such that  $P(\bar{n})$  occurs in  $\sigma$  and for all  $m < n$ ,  $\neg P(\bar{m})$  occurs in  $\sigma$ . If  $\{\varphi\} \cup \text{cnt}(\sigma)$  is consistent in  $\mathcal{W}$  and there exists  $m \in \mathbb{N}$  such that  $P(\bar{m}) \in \text{cnt}(\sigma)$  and  $\phi_n(m)$  is undefined then  $f(\sigma) = 1$ ; otherwise  $f(\sigma) = 0$ . It is immediately verified that  $f$  classifies  $\mathcal{W}$  in the limit following  $\varphi$ .

Let  $\mathcal{W}'$  be the class of all  $\mathfrak{M} \in \mathcal{W}$  such that:

- $\mathfrak{M} \models P(\bar{n})$  for some  $n \in \mathbb{N}$ ;
- if  $e$  is the least  $n \in \mathbb{N}$  with  $\mathfrak{M} \models P(\bar{n})$  then either  $\mathfrak{M} \models \varphi$  or  $\mathfrak{M} \not\models \psi(\bar{e})$ .

Note that  $\mathcal{W}'$  is cocountable, since its complement consists only of possible worlds  $\mathfrak{M}$  where the interpretation of  $P$  in  $\mathfrak{M}$  is recursively enumerable. Moreover, it is easy to exhibit a computable classifier that classifies  $\mathcal{W}'$  in the limit following  $\varphi$ .

Now, suppose for a contradiction that a (partial) computable function  $f$  classifies  $\mathcal{W}$  following  $\varphi$  almost everywhere. Then, by the recursion theorem, there exists  $e \in \mathbb{N}$  such that  $\phi_e$  can be defined as follows. Given a member  $\sigma$  of  $\{0, 1\}^*$ , let  $\hat{\sigma}$  be the sequence obtained from  $\sigma$  by replacing  $\sigma(n)$  by  $P(\bar{n})$  if  $\sigma(n) = 1$ , and by  $\neg P(\bar{n})$  otherwise, for all natural numbers  $n$  smaller than the length of  $\sigma$ . We will define  $\phi_e$  in stages. Before stage 0,  $\sigma_0$  is defined to be the sequence of  $e$  0's followed by a 1. and  $\phi_e$  is defined to be 0 on  $x < e$ , and 1 on  $e$ . Let  $x_s = e + 1$ . Intuitively,  $x_s$  is the least input on which  $\phi_e$  is not defined before stage  $s$ . Stage  $s$  consists of the following steps.

1. Dovetail steps 2 and 3, until step 2 succeeds, if ever. If and when step 2 succeeds, then stop step 3 also and go to step 4.

2. Search for a  $\tau \subseteq \sigma_s 00^\infty$  or  $\tau \subseteq \sigma_s 10^\infty$  such that  $f(\widehat{\sigma}_s) \neq f(\widehat{\tau})$ .
3. Let  $x = x_s + 1$ . **Loop** Let  $\phi_e(x) = 0$ . Let  $x = x + 1$ . **EndLoop**
4. Let  $y$  be the largest element in domain of  $\phi_e$  defined up to now, or the domain of  $\tau$  as found in step 2. Let  $\phi_e(x_s) = \tau(x_s)$ , and  $\phi_e(x) = 0$ , for  $x_s < x \leq y$ . Let  $\sigma_{s+1} = (\phi_e(0) \dots \phi_e(y))$ . Go to stage  $s + 1$ .

Now if there exist infinitely many stages, then  $f$  makes infinitely many mind changes on some possible world. On the other hand, if some stage  $s$  does not end then  $\phi_e$  is defined on all inputs except  $x_s$ , and for any  $P$  consistent with  $\phi_e$ ,  $f$  converges to  $f(\widehat{\sigma}_s)$ . However, the possible worlds which satisfy  $\varphi$  and are consistent with  $\phi_e$ , have measure exactly half (when  $P(\overline{x}_s) = 1$ ). Thus  $\mathcal{W}$  is not classifiable in the limit following  $\varphi$  almost everywhere. ■

## 4 Definability Versus Nondefinability of Misclassified Sets

The next fundamental result shows that a classifier who uses  $\varphi$  to partition the set of possible worlds might have to misclassify a subset of  $\mathcal{W}$  that is not only of measure 0, but also necessarily not definable. Hence almost correct classification is not equivalent to absolute classification w.r.t. a partition of the possible worlds given by another formula than  $\varphi$  (that formula would of course be almost equivalent to  $\varphi$ ). It is worth noting that the next proposition uses a set of possible data that is neither  $\{P(\overline{n}) : n \in \mathbb{N}\}$  nor  $\{P(\overline{n}), \neg P(\overline{n}) : n \in \mathbb{N}\}$ , henceforth exploiting the generality and flexibility allowed by the parameter  $\mathcal{D}$ .

**Proposition 14.** *Let  $\mathcal{V}$  be a  $\omega$ -sequence of strings of even length,  $\overline{0} \leq s < \omega$ ,  $P \in \mathcal{W}$ ,  $\mathcal{D} \subseteq \mathcal{L}$ ,  $\varphi \in \mathcal{L}$ .*

- $\mathcal{W}$  is partitioned into two sets  $\mathcal{W}_\varphi$  and  $\mathcal{W}_{\neg\varphi}$ .
- $\mathcal{W}_{\neg\varphi}$  is not definable by any member of  $\mathcal{D}$ .

**Proof.** Let  $\varphi$  be a sentence which expresses that the characteristic function of  $P$  is lexicographically greater than the characteristic function of  $2\mathbb{N} = \{0, 2, 4, \dots\}$ . Formally:  $\varphi \equiv P(0) \wedge \exists x(P(x) \wedge P(s(x)) \wedge \forall y < x (P(y) \leftrightarrow \neg P(s(y))))$ . Let  $B$  be the set whose characteristic function is the concatenation of all strings of even length, in lexicographic order and in increasing length. Hence the characteristic function of  $B$  can be represented by the  $\omega$ -sequence:

$$00\ 01\ 10\ 11\ 0000\ 0001\ 0010\ \dots\ 1111\ 000000\ 000001\ 000010\ \dots$$

Put  $\mathcal{D} = \{P(\overline{n}) : n \in B\} \cup \{\neg P(\overline{n}) : n \notin B\}$ . Note that  $B$  is not definable by a member of  $\mathcal{L}$  (this property is key to the proof of the proposition). Another essential property of  $B$  used in the proof is that

(†) for every  $\tau \in \{0, 1\}^*$  there are infinitely many even numbers  $x$  such that for all  $y < \text{lt}(\tau)$ ,  $\tau(y) = B(x + y)$ .

We first define a computable classifier  $f$  which classifies  $\mathcal{W}$  following  $\varphi$  almost everywhere; in other words,  $f$  converges on all environments for all members of  $\mathcal{W}$ , but  $f$ 's conjectures can be false in the limit on some environments for a set of possible worlds of measure 0. The classifier  $f$  outputs 0 until it is presented with a datum of the form  $P(\bar{n})$  for an odd  $n \in \mathbb{N}$ , or of the form  $\neg P(\bar{n})$  for an even  $n \in \mathbb{N}$ . Then  $f$  takes this  $n$  as a parameter and computes from now on for any stage  $s$  the set  $R_s$  consisting of all  $m \in \{0, 1, \dots, n\} \cap B$  such that  $P(\bar{m})$  has appeared in the first  $s$  elements of the input, and all members  $m$  of  $\{0, 1, \dots, n\} \setminus B$  such that  $\neg P(\bar{m})$  has not appeared in the first  $s$  elements of the input. Note that when  $s$  is large enough, the restriction of the characteristic function of  $R_s$  to  $\{0, 1, \dots, n\}$  is equal to the restriction of the characteristic function of the interpretation of  $P$  in the possible world one of whose environments is fed to  $f$ . Let  $f$  conjecture in the limit the value 0 if the characteristic function of  $R_s$  is lexicographically smaller than the characteristic function of  $2\mathbb{N}$ , and 1 if the characteristic function of  $2\mathbb{N}$  is lexicographically smaller than the characteristic function of  $R_s$ . Clearly,  $f$  has the desired properties.

Now let  $\psi$  be a member of  $\mathcal{L}(\mathcal{V})$ , a first-order sentence over  $\mathcal{V}$ , and assume for a contradiction that a classifier  $g$  classifies  $\mathcal{W}$  in the limit following  $\psi$ . Let  $\mathfrak{M}$  be the (unique) possible world such that the characteristic function of the interpretation of  $P$  in  $\mathfrak{M}$  is isomorphic to the characteristic function of  $2\mathbb{N}$ . Then there is an initial segment  $\sigma$  of  $\mathcal{D}^*$  such that  $\mathfrak{M}$  is a model of  $\text{cnt}(\sigma)$  and for all  $\tau \in \mathcal{D}^*$ , if  $\tau$  extends  $\sigma$  and  $\mathfrak{M} \models \text{cnt}(\tau)$  then  $g(\tau)$  is defined and equal to  $g(\sigma)$ . Since  $\psi$  contains only finitely many occurrences of  $\bar{0}$ ,  $s$  and variables, there exists  $k \in \mathbb{N}$  such that:

- for all  $n \in \mathbb{N}$ , if either  $P(\bar{n})$  or  $\neg P(\bar{n})$  occurs in  $\sigma$  then  $n < 2k$ ;
- for all members  $\tau_0, \tau_1, \dots$  of  $\{0, 1\}^*$  of even length and for all possible worlds  $\mathfrak{N}, \mathfrak{N}'$ , if the characteristic functions of  $P$  in  $\mathfrak{N}$  and  $\mathfrak{N}'$  are both of the form  $(01)^k(01)^*\tau_0(01)^k(01)^*\tau_1(01)^k(01)^*\tau_2 \dots$  then  $\mathfrak{N} \models \psi$  iff  $\mathfrak{N}' \models \psi$ .

It is known from the theory of randomness that the characteristic function of any random set coincides with the characteristic function of  $2\mathbb{N}$  on infinitely many even places for at least  $2k$  bits. Furthermore, the measure of all random sets starting with  $(01)^k$  equals  $2^{-2k}$ . Fix a random set  $R$  whose characteristic function extends  $(01)^k$ . Thus we can choose some members  $\tau_0, \tau_1, \dots$  of  $\{0, 1\}^*$  of even length such that the characteristic function of  $R$  is of the form

$$(01)^k\tau_0(01)^k\tau_1(01)^k\tau_2 \dots$$

Using (†) above, there exists  $a_0, a_1, \dots \in \mathbb{N}$  such that the set  $R'$  whose characteristic function is  $(01)^{k+a_0}\tau_0(01)^{k+a_1}\tau_1(01)^{k+a_2}\tau_2 \dots$  satisfies the following property. Given  $n \in \mathbb{N}$ , put  $x_n = \sum_{i \leq n} (2k + a_i) + \sum_{i < n} \text{lt}(\tau_i)$ . For all  $n \in \mathbb{N}$  and  $y < \text{lt}(\tau_n)$ , if  $y$  is odd and  $\tau_n(y) = 1$  then  $B(x_n + y) = 0$ , whereas if  $y$  is even and  $\tau_n(y) = 0$  then  $B(x_n + y) = 1$ . Let  $\mathfrak{N}$ , respect.  $\mathfrak{N}'$ , be the (unique) possible world such that the characteristic function of the interpretation of  $P$  in  $\mathfrak{N}$ , respect.  $\mathfrak{N}'$ , is isomorphic to the characteristic function of  $R$ , respect.  $R'$ . Note that  $\text{Diag}_{\mathcal{D}}(\mathfrak{N}') \subset \text{Diag}_{\mathcal{D}}(\mathfrak{M})$ . Moreover,  $\sigma$  is an initial segment of some environment for  $\mathfrak{N}'$ . As a consequence,  $g$  converges in the limit to  $g(\sigma)$  on all

environments for  $\mathfrak{N}'$  that extend  $\sigma$ . But since both  $\mathfrak{N}$  and  $\mathfrak{N}'$  agree on  $\psi$  and  $g$  is assumed to classify  $\mathcal{W}$  in the limit following  $\psi$ ,  $g$  converges in the limit to  $g(\sigma)$  on all environments for  $\mathfrak{N}$  that extend  $\sigma$ . It follows that all random sequences extending  $(01)^k$  are classified as  $g(\sigma)$ , and the measure of all extensions of  $(01)^k$  which are classified as  $g(\sigma)$  is  $2^{-2k}$ . On the other hand, those extensions of  $(01)^k$  which are identified with models of  $\varphi$  in  $\mathcal{W}$  have measure  $2^{-2k}/3$ . Therefore  $\varphi$  and  $\psi$  differ on a class of possible worlds of positive measure. Contradiction.  $\blacksquare$

Proposition 14 cannot be generalized to arbitrary paradigms. This can be proved by using results from automata theory. Remember that a subset of  $\{0, 1\}^\omega$  (or  $\omega$ -regular set) is *regular* if there exists a finite automaton  $A$  with a set  $F$  of accepting states such that for all  $w \in \{0, 1\}^\omega$ ,  $e$  belongs to  $S$  iff there is run of  $A$  on  $w$  which goes infinitely often through an accepting state. The next result plays a crucial role in the proof of Proposition 17 below.

**Lemma 15.** [4–Theorem 3.1]  $S \subseteq \{0, 1\}^\omega$  is regular if and only if there exists a regular set  $R$  and a sequence of regular sets  $Y_0, \dots, Y_n$  such that  $S = R \star Y_0 \star \dots \star Y_n$ .

**Corollary 16.** Let  $S, R, R' \subseteq \{0, 1\}^\omega$  be regular sets. Then:

- $\mu(R \star \{0, 1\}^\omega \cup R' \star \{0, 1\}^\omega) = 1$ .
- $\mu(R \star \{0, 1\}^\omega \cap R' \star \{0, 1\}^\omega) = \mu(R \star \{0, 1\}^\omega) \cdot \mu(R' \star \{0, 1\}^\omega)$ .
- $\mu(R \star \{0, 1\}^\omega \Delta S) = 0$

**Proposition 17.** Let  $\mathcal{L}$  be a countable set of sentences,  $\bar{0} \in \mathcal{L}$ ,  $P \in \mathcal{L}$ ,  $\mathcal{W}$  a countable set of worlds,  $\mathcal{D}$  a countable set of standard informants,  $\{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\} \subseteq \mathcal{L}$ ,  $\varphi \in \mathcal{L}$ ,  $\mathcal{W}$  a countable set of worlds,  $\varphi$  a sentence.

**Proof.** Let a sentence  $\varphi$  be given. Let  $S$  be the set of standard informants for the Henkin models of  $\varphi$ . By the choice of  $\mathcal{L}$ ,  $S$  is Büchi recognizable. Let  $R$  be defined from  $S$  as in Corollary 16. Write  $\widehat{S}$  for  $R \star \{0, 1\}^\omega$ . Since  $R$  is regular and prefix free, it is immediately verified that there exists a computable classifier  $f$  such that for all standard informants (identified with environments) for some possible world, if  $e \in \widehat{S}$  then  $f$  outputs 1 in response to cofinitely many finite initial segments of  $e$ , and if  $e \notin \widehat{S}$  then  $f$  outputs 0 in response to cofinitely many finite initial segments of  $e$ . Let  $W$  be the set of all possible worlds whose standard informants belong to  $\widehat{S}$ . Using [4–Theorem 3.1] again, we infer that  $W$  is the set of models of some member  $\psi$  of  $\mathcal{L}$ . Moreover, it follows immediately from the definition of  $\widehat{S}$  that  $\mu(\text{Mod}_{\mathcal{W}}(\varphi) \Delta W) = 0$ , hence  $\psi$  is almost equivalent to  $\varphi$ . Since  $\mathcal{W}$  is classifiable in the limit following  $\psi$ , we are done.  $\blacksquare$

Corollary 16 has other applications. For instance, considering classification from positive data only, and using a different language and set of possible worlds than in Proposition 12, one can strengthen the first statement of Proposition 12:

**Proposition 18.**  $\mathcal{V} = \{\bar{0}, s, P\}$   $\mathcal{W}$ ,  $\mathcal{D} = \{P(\bar{n}) : n \in \mathbb{N}\}$   
 $\mu(W') = 1$   $\psi$   
 $\varphi$   $\mathcal{W}$

**Proof.** Let a sentence  $\psi$  be given. Let  $S^+$  be the set of standard informants for the models of  $\psi$  in  $\mathcal{W}$ , and let  $S^-$  be the set of standard informants for the models of  $\neg\psi$  in  $\mathcal{W}$ . Let  $R^+$  be defined from  $S^+$  as  $R$  is defined from  $S$  in Corollary 16, and let  $R^-$  be defined from  $S^-$  as  $R$  is defined from  $S$  in Corollary 16. Put  $\widehat{S}^+ = R^+ \star \{0, 1\}^\omega$  and  $\widehat{S}^- = R^- \star \{0, 1\}^\omega$ . Note that the intersection of  $\widehat{S}^+$  with  $\widehat{S}^-$  is of measure 0. Since both  $\widehat{S}^+$  and  $\widehat{S}^-$  are open sets (w.r.t. the usual topology over the Cantor space), the intersection of  $\widehat{S}^+$  with  $\widehat{S}^-$  is actually empty. Hence  $R^+ \cup R^-$  is prefix free, which implies immediately that there exists a computable  $\{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\}$ -classifier such that for all standard informants  $e$  for some possible world, if  $e \in \widehat{S}^+$  then  $f$  outputs 1 in response to cofinitely many finite initial segments of  $e$ , and if  $e \in \widehat{S}^-$  then  $f$  outputs 0 in response to cofinitely many finite initial segments of  $e$ . Let  $W^+$ , respect.,  $W^-$ , be the set of all possible worlds  $\mathfrak{M}$  whose standard informants belong to  $\widehat{S}^+$ , respect.,  $\widehat{S}^-$ . It follows immediately from the definition of  $\widehat{S}^+$  and  $\widehat{S}^-$  that both  $\mu(\text{Mod}_{\mathcal{W}}(\psi) \Delta W^+)$  and  $\mu(\text{Mod}_{\mathcal{W}}(\neg\psi) \Delta W^-)$  are null. Hence  $\mu(W^+ \cup W^-) = 1$  and  $W^+ \cup W^-$  is classifiable in the limit following  $\psi$  almost everywhere. Since the number of sentences is countable, the first part of the proposition follows immediately.

For the second part, define  $\varphi$  as  $\exists x \forall y (y < s(s(x)) \rightarrow (P(y) \leftrightarrow x \neq y))$ . Suppose for a contradiction that a classifier  $f$  classifies  $\mathcal{W}$  almost everywhere in the limit following  $\varphi$ . Since  $f$  converges (possibly to 1) on all environments for the possible world whose  $\mathcal{D}$ -diagram is  $\{P(\bar{n}) : n \in \mathbb{N}\}$ , we can choose a member  $\sigma$  of  $\mathcal{D}^*$  such that for all  $\tau \in \mathcal{D}^*$  that extend  $\sigma$ ,  $f(\tau) = f(\sigma)$ . Let  $a \in \mathbb{N}$  be greater than all members of  $\text{cnt}(\sigma)$ . Put

$$\begin{aligned} W_1 &= \text{Mod}_{\mathcal{W}}(P(\bar{0}) \wedge \dots \wedge P(\bar{a}) \wedge \neg P(\overline{a+1}) \wedge P(\overline{a+2})) \\ W_2 &= \text{Mod}_{\mathcal{W}}(P(\bar{0}) \wedge \dots \wedge P(\bar{a}) \wedge \neg P(\overline{a+1}) \wedge \neg P(\overline{a+2})) \end{aligned}$$

Note that  $W_1 \subseteq \text{Mod}_{\mathcal{W}}(\varphi)$  and  $W_2 \subseteq \text{Mod}_{\mathcal{W}}(\neg\varphi)$ , and that both  $\mu(W_1)$  and  $\mu(W_2)$  are nonnull. But by the choice of  $\sigma$ , for all  $\mathfrak{M} \in W_1 \cup W_2$  and for all environments  $e$  for  $\mathfrak{M}$  that extend  $\sigma$ ,  $f$  outputs  $f(\sigma)$  in response to every initial segment of  $e$  that extends  $\sigma$ . Contradiction. ■

### 5 Positive Only Versus Positive and Negative Data

Being allowed to misclassify a set of possible worlds of measure 0 when classifying from positive data does not always make up for non-access to negative data:

**Proposition 19.**  $\forall \varphi \in \mathcal{L}, \exists \bar{0} \in \Sigma, \exists P \in \mathcal{L}, \exists \mathcal{D} \subseteq \mathcal{L}, \exists \mathcal{W} \subseteq \mathcal{L}$   
 $\mathcal{D} = \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\} \subseteq \mathcal{W}$   
 $\mathcal{D} = \{P(\bar{n}) : n \in \mathbb{N}\} \subseteq \mathcal{W}' \subseteq \mathcal{W}$   
 $\mathcal{W}' \models \varphi$

**Proof.** Define  $\varphi$  as  $\exists x \forall y ((x < y \wedge y < x + x + 3) \rightarrow \neg P(y))$ . It is immediately verified that if  $\mathcal{D} = \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\}$  then  $\mathcal{W}$  is computably classifiable in the limit following  $\varphi$  (with at most one mind change).

Now suppose that  $\mathcal{D} = \{P(\bar{n}) : n \in \mathbb{N}\}$ . Trivially,  $\mu(\text{Mod}_{\mathcal{W}}(\varphi)) > 0$ . Moreover,  $\mu(\text{Mod}_{\mathcal{W}}(\varphi)) \leq \sum_{n \in \mathbb{N}} 2^{-n-2} = 2^{-1}$ . For a contradiction, let a classifier  $f$  be such that  $f$  classifies  $\mathcal{W}$  in the limit following  $\varphi$  almost everywhere. For all  $\sigma \in \mathcal{D}^*$ , define  $U_{\sigma}$  as the set of all  $\mathfrak{M} \in \text{Mod}_{\mathcal{W}}(\neg\varphi)$  such that for all  $\tau \in \mathcal{D}^*$ , if  $\tau$  extends  $\sigma$  and  $\mathfrak{M} \models \text{cnt}(\tau)$  then  $f(\tau) = 0$ . Choose  $\sigma \in \mathcal{D}^*$  with  $\mu(U_{\sigma}) > 0$ . Denote by  $a$  the maximal number in  $\text{cnt}(\sigma)$ . Let  $U$  be the set of all  $\mathfrak{M} \in \mathcal{W}$  with:

- $\mathfrak{M} \models \neg P(\overline{a+1}) \wedge \dots \wedge \neg P(\overline{a+a+2})$ ;
- the  $\mathcal{D}$ -diagram of  $\mathfrak{M}$  agrees with the  $\mathcal{D}$ -diagram of some member of  $U_{\sigma}$ , except perhaps on  $\{P(\bar{a}), \dots, P(\overline{a+a+2})\}$ .

Note that all members of  $\text{Mod}_{\mathcal{W}}(U)$  are models of  $\varphi$ . Since  $\mu(\text{Mod}_{\mathcal{W}}(U))$  is at least equal to  $2^{-a-2}\mu(U_{\sigma})$ ,  $\mu(\text{Mod}_{\mathcal{W}}(U))$  is nonnull. Let  $\mathfrak{M} \in U$  be given. Since the  $\mathcal{D}$ -diagram of  $\mathfrak{M}$  is included in the  $\mathcal{D}$ -diagram of some member of  $U_{\sigma}$ , we infer that for all environments  $e$  for  $\text{Diag}_{\mathcal{D}}(\mathfrak{M})$  that extend  $\sigma$ ,  $f$  outputs 0 in response to all finite initial segments of  $e$  that extend  $\sigma$ . Contradiction. ■

Considering failing to classify a set of first category rather than misclassifying a set of measure 0, one can contrast Proposition 19 with the following.

**Proposition 20.**  $\forall \varphi \in \mathcal{L}, \exists \mathcal{D} = \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\} \subseteq \mathcal{W}$   
 $\mathcal{D} = \{P(\bar{n}) : n \in \mathbb{N}\} \subseteq \mathcal{W}' \subseteq \mathcal{W}$   
 $\mathcal{W}' \models \varphi$   
 $\mathcal{W} \setminus \mathcal{W}' \neq \emptyset$   
 $\mathcal{W}' \models \varphi$

**Proof.** Put  $D = \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\}$ . Consider a  $D$ -classifier  $f$  such that if  $\mathcal{D} = D$  then  $f$  classifies  $\mathcal{W}$  in the limit following  $\varphi$ . Without loss of generality we can assume that  $f$  is total and  $f$  depends only on the content of the input and not on the order and number of repetitions of symbols. Let  $S$  be the set of all

members  $\sigma$  of  $D^*$  such that  $f(\tau) = f(\sigma)$  for all consistent  $\tau \in D^*$  that extend  $\sigma$ . Note that if  $f$  is computable then  $S$  is co-r.e. Let  $\mathcal{W}$  be the set of possible worlds that are models of  $\text{cnt}(\sigma)$  for some  $\sigma \in S$ . By the choice of  $f$ , for all  $\sigma \in D^*$ , some member of  $S$  extends  $\sigma$ . This implies immediately that  $\mathcal{W} \setminus \mathcal{W}'$  is of first category. Fix an enumeration  $(\tau_i)_{i \in \mathbb{N}}$  of  $D^* \setminus S$  that in case  $f$  is computable, is itself computable. Assume that  $\mathcal{D} = \{P(\bar{n}) : n \in \mathbb{N}\}$ . Define a classifier  $g$  as follows. Let  $\sigma \in (\mathcal{D} \cup \{\#\})^*$  be given, and let  $p$  denote the length of  $\sigma$ . Then  $g(\sigma)$  is equal to  $f(\tau)$  for the shortest  $\tau \in D^* \setminus \{\tau_0, \dots, \tau_p\}$  such that for all  $n \leq p$ ,  $P(\bar{n})$  occurs in  $\tau$  iff  $P(\bar{n})$  occurs in  $\sigma$ . Obviously,  $g$  is computable if  $f$  is computable. Moreover, for all  $\mathfrak{M} \in \mathcal{W}$  and environments  $e$  for  $\mathfrak{M}$  (w.r.t. the choice of  $\mathcal{D}$ ), some member  $\tau$  of  $S$  satisfies  $\{n \in \mathbb{N} : P(\bar{n}) \in \text{cnt}(\tau)\} \subseteq \{n \in \mathbb{N} : P(\bar{n}) \in \text{cnt}(e)\}$  and  $\{n \in \mathbb{N} : \neg P(\bar{n}) \in \text{cnt}(\tau)\} \cap \{n \in \mathbb{N} : P(\bar{n}) \in \text{cnt}(e)\} = \emptyset$ ; hence  $g$  converges on  $e$  to  $f(\tau)$ . This shows that  $g$  classifies  $\mathcal{W}'$  in the limit following  $\varphi$ . ■

## 6 Learning Witnesses

In this section we focus on classifiability of existential sentences. When the class of possible worlds consists of Henkin structures only, such a sentence, of the form  $\exists x\psi(x)$ , is true in a member  $\mathfrak{M}$  of  $\mathcal{W}$  iff  $\psi(t)$  is true for some closed term  $t$ . It is then natural not only to determine that  $\exists x\psi(x)$  is true, but also to learn in the limit such a witness  $t$ . This amounts to a computation in the limit that generalizes the kind of computations done by Prolog systems. We now formalize the concepts that have just been introduced.

**Definition 21.** A  $f : (\mathcal{D} \cup \{\#\})^* \rightarrow \{0\} \cup \text{ClosedTerms}$  is a partial function from  $(\mathcal{D} \cup \{\#\})^*$  into the union of  $\{0\}$  with the set of closed terms.

**Definition 22.** We say that a classifier  $g$  is  $\varphi$ -witnessing, for a learner  $f$  iff for all  $\sigma \in (\mathcal{D} \cup \{\#\})^*$ ,  $g(\sigma)$  is defined iff  $f(\sigma)$  is defined and  $g(\sigma) = 0$  iff  $f(\sigma) = 0$ .

**Definition 23.** Let a learner  $f$ , a sentence of the form  $\exists x\psi(x)$ , and a subset  $\mathcal{W}'$  of  $\mathcal{W}$  be given.

We say that  $f$   $\varphi$ -witnesses  $\exists x\psi(x)$  in  $\mathcal{W}'$  just in case for all  $\mathfrak{M} \in \mathcal{W}'$  and environments  $e$  for  $\mathfrak{M}$ , the following holds.

- If  $\mathfrak{M} \models \exists x\psi(x)$  then there exists a closed term  $t$  such that  $\mathfrak{M} \models \psi(t)$  and  $\{\sigma \in (\mathcal{D} \cup \{\#\})^* : \sigma \subseteq e \text{ and } f(\sigma) = t\}$  is cofinite.
- If  $\mathfrak{M} \not\models \exists x\psi(x)$  then  $\{\sigma \in (\mathcal{D} \cup \{\#\})^* : \sigma \subseteq e \text{ and } f(\sigma) = 0\}$  is cofinite.

**Definition 24.** Let a learner  $f$  and an existential sentence  $\varphi$  be given. We say that  $f$   $\varphi$ -witnesses in  $\mathcal{W}$  just in case:

- the classifier associated with  $f$  classifies  $\mathcal{W}$  in the limit following  $\varphi$  almost everywhere;
- there exists  $\mathcal{W}' \subseteq \mathcal{W}$  with  $\mu(\mathcal{W}') = 1$  such that  $f$  learns  $\varphi$  in the limit in  $\mathcal{W}'$ .

**Definition 25.** We say that an existential sentence  $\varphi$  is  $\mu$ -witnessed, respectively,  $\mu$ -witnessed by a learner, respectively, a computable learner, in  $\mathcal{W}$  almost correctly, iff some learner, respectively, computable learner, learns  $\varphi$  in the limit in  $\mathcal{W}$  almost correctly.

Adapting the proof of Proposition 17, we obtain an important particular case where limiting computation of witnesses for existentially quantified sentences is always possible:

**Proposition 26.**  $\forall \bar{0} s \dots P \mathcal{W}, \dots \mathcal{L}, \dots \mathcal{D}, \dots \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\} \dots \varphi \varphi, \dots \mathcal{W}$

**Proof.** Let a sentence of the form  $\exists x\psi(x)$  be given. For all  $n \in \mathbb{N}$ , define  $\psi_n$  as  $\neg\psi(\bar{0}) \wedge \dots \wedge \neg\psi(\overline{n-1}) \wedge \psi(\bar{n})$ . For all  $n \in \mathbb{N}$ , denote by  $S^n$  and  $\widehat{S}^n$  the sets  $S$  and  $\widehat{S}$  defined from  $\varphi$  in the proof of Proposition 17, taking  $\varphi = \psi_n$ . Denote by  $S^-$  and  $\widehat{S}^-$  the sets  $S$  and  $\widehat{S}$  defined from  $\varphi$  in the proof of Proposition 17, taking  $\varphi = \neg\exists x\psi(x)$ . A variation on the argument used in Proposition 18 proves the claim of the proposition. ■

When investigating the connections between classifiability and learnability of sentences of the form  $\exists x\psi(x)$ , it is reasonable to assume that  $\mathcal{W}$  is classifiable in the limit following any of the sentences  $\psi(\bar{n})$ ; otherwise one could obtain a separation by taking a sentence  $\xi$  such that  $\mathcal{W}$  is not almost everywhere classifiable in the limit following  $\xi$  and then consider  $\exists x ((x = \bar{0} \wedge \neg\xi) \vee (x = \bar{1} \wedge \xi))$ . Furthermore, if  $\mathcal{W}$  is classifiable in the limit following  $\exists x\psi(x)$  as well as any of the sentences  $\psi(\bar{n})$ , the parameter  $x$  is also learnable, though computability might be lost, for example with  $\exists x (\phi_x \text{ is total} \wedge \phi_x(0) = \min\{z : Pz\} \wedge \forall y < x (\phi_y \neq \phi_x))$ . This example can be formalized in full arithmetic. So it is natural to ask what happens if one only postulates that  $\mathcal{W}$  is almost everywhere classifiable in the limit following  $\exists x\psi(x)$  and the other conditions are kept as such. The next result shows that learnability is then lost.

**Proposition 27.**  $\forall \bar{0} s \dots P \dots + \mathcal{W}, \dots \mathcal{L}, \dots \mathcal{D} = \{P(\bar{n}), \neg P(\bar{n}) : n \in \mathbb{N}\} \dots \psi(x) \dots \forall y \exists u \exists v \exists w (x + y + v = u \wedge u + w = x + x + y + y \wedge P(u)) \dots \exists x\psi(x), \mathcal{W}, \dots n \in \mathbb{N} \mathcal{W}, \dots \psi(\bar{n}), \dots \exists x\psi(x), \mathcal{W}$

A further question would be whether classification can be parametrized. That is, given a sentence  $\varphi$  such that  $\mathcal{W}$  can be classified in the limit following  $\varphi$ , is  $\varphi$  equivalent to a sentence of the form  $\exists x\psi(x)$  such that  $\mathcal{W}$  is classifiable with a constant number of mind changes, following any sentence  $\psi(\bar{n})$ ? The answer is yes for computable classification in full arithmetic since one can transform every learner into a formula monitoring its behaviour and then quantify over the last time where the classifier makes a wrong conjecture.



## 7 Conclusion

In this paper the relationship between classifying all possible worlds and classifying almost all possible worlds has been investigated. The main results include the following. In the case of monadic second order logic with one successor, which is well known to be decidable thanks to its connections with Büchi automata, one can classify  $\mathcal{W}$  following a formula  $\varphi$  almost everywhere from both positive and negative data. But with positive data only, convergence might fail on a set of measure 0. For other choices of possible data,  $\mathcal{W}$  might even be classifiable in the limit following a given sentence  $\varphi$  almost everywhere, though no sentence differing from  $\varphi$  only on a null set of worlds allows for absolute classification. With Presburger arithmetic, there is a sentence  $\varphi$  such that some set of worlds of measure 1 can be classified in the limit following  $\varphi$ , though no classifier which is correct on a set of measure 1 converges on all environments for all possible worlds. Furthermore, some connections were made between classification and learning of an essential parameter, namely, the value of the first quantified variable in an existential sentence. It turns out that this can be achieved on a set of worlds of measure 1 in monadic second order logic with one successor, demonstrating that Prolog style inferences are almost everywhere possible in this case.

**Acknowledgments.** We would like to thank Wolfgang Merkle for very helpful discussions. We also thank the referees for their comments.

## References

- [1] A. Ambainis: *Probabilistic Inductive Inference: a Survey*. Theoretical Computer Science 264, pp. 155–167 (2001).
- [2] J. Bārzdīņš, R. Freivalds, C. Smith: *Learning formulae from elementary facts*. Proceedings of the Third European Conference on Computational Learning Theory (EuroCOLT 1997), LNCS 1208, pp. 272–285 (1997).
- [3] Sh. Ben-David: *Can Finite Samples Detect Singularities of Real-Valued Functions?* Proceedings of the 24th Annual ACM Symposium on the Theory of Computer Science, Victoria, B.C., pp. 390–399 (1992).
- [4] J. Büchi: *On a decision method in restricted second order arithmetic*. In Proc. of the International Congress on Logic, Methodology and Philosophy of Science, Stanford Univ. Press, pp. 1–11 (1960)
- [5] P. Caldon, E. Martin: *Limiting Resolution: from Foundations to Implementation*. In B. Dörmann, V. Lifschitz: Proceedings of the 20th International Conference on Logic Programming, Springer-Verlag, LNCS 3132, pp. 149–164 (2004)
- [6] R. Daley: *Inductive Inference Hierarchies: Probabilistic vs. Pluralistic*. Mathematical Methods of Specification and Synthesis of Software Systems, LNCS 215, pp. 73–82 (1985).
- [7] R. Freivalds: *Finite Identification of General Recursive Functions by Probabilistic Strategies*. Proceedings of the Conference on Fundamentals of Computation Theory, Akademie Verlag, Berlin, pp. 138–145 (1979).
- [8] W. Gasarch, M. Pleszkoch, F. Stephan, M. Velauthapillai: *Classification Using Information*. Annals of Math. and Artificial Intelligence 23, pp. 147–168 (1998).

- [9] E. Martin, A. Sharma, F. Stephan: *Logic, Learning, and Topology in a Common Framework*. In N. Cesa-Bianchi, M. Numao, R. Reischuk: Proc. of the 13th Intern. Conf. on Alg. Learning Theory. Springer-Verlag, LNAI 2533 pp. 248-262 (2002)
- [10] L. Meyer: *Probabilistic language learning under monotonicity constraints*. Theoretical Computer Science 185, pp. 81-128 (1997).
- [11] L. Pitt: *Probabilistic Inductive Inference*. Journal of the Association for Computing Machinery 36, pp. 383-333 (1989).
- [12] L. Pitt, C. Smith: *Probability and plurality for aggregations of learning machines*. Information and Computation 77, pp. 77-92 (1988).
- [13] H. Jr. Rogers: *Theory of recursive functions and effective computability*. McGraw-Hill Book Company, New York (1967).
- [14] C. Smith: *The power of pluralism for automatic program synthesis*. Journal of the Association for Computing Machinery 29, pp. 1144-1165 (1982).
- [15] I. Takeuti: *The measure of an omega regular language is rational*. Available at <http://www.sato.kuis.kyoto-u.ac.jp/~takeuti/art/regular.ps>
- [16] A. Voronkov: *Logic programming with bounded quantifiers*. In Proc. LPAR, Springer-Verlag, LNCS 592, pp. 486-514 (1992)

# Online Allocation with Risk Information

Shigeaki Harada, Eiji Takimoto\*, and Akira Maruoka

Graduate School of Information Sciences,  
Tohoku University,  
Sendai 980-8579, Japan  
{sig, t2, maruoka}@maruoka.ecei.tohoku.ac.jp

**Abstract.** We consider the problem of dynamically apportioning resources among a set of options in a worst-case online framework. The model we investigate is a generalization of the well studied online learning model. In particular, we allow the learner to see as additional information how high the risk of each option is. This assumption is natural in many applications like horse-race betting, where gamblers know odds for all options before placing bets. We apply the Aggregating Algorithm to this problem and give a tight performance bound. The results support our intuition that we should bet more on low-risk options. Surprisingly, however, the Hedge Algorithm without seeing risk information performs nearly as well as the Aggregating Algorithm. So the risk information does not help much. Moreover, the loss bound does not depend on the values of relatively small risks.

## 1 Introduction

Consider the following classical online allocation problem. At each trial  $t = 1, 2, \dots, T$ , the learner (an algorithm  $A$ ) chooses a probability distribution  $\mathbf{v}_t = (v_{1,t}, \dots, v_{N,t})$  over the set  $\{1, \dots, N\}$  of experts and then the environment determines experts' losses  $\mathbf{l}_t = (l_{1,t}, \dots, l_{N,t})$ . In this trial, the learner  $A$  suffers loss  $\mathbf{v}_t \cdot \mathbf{l}_t = \sum_{i=1}^N v_{i,t} l_{i,t}$ . This loss  $\mathbf{l}_t$  can be interpreted as the loss if the learner apportioned the wager among the experts according to the distribution  $\mathbf{v}_t$  and let the experts actually make decisions against the environment on behalf of the learner. Let  $L_{A,T} = \sum_{t=1}^T \mathbf{v}_t \cdot \mathbf{l}_t$  denote the total loss of the learner  $A$  and  $L_{i,T} = \sum_{t=1}^T l_{i,t}$  denote the total loss of expert  $i$ . The goal of the learner is to minimize the . . .  $L_{A,T} - \min_{i \in \{1, \dots, N\}} L_{i,T}$ .

This problem has been extensively studied in the case where the losses  $l_{i,t}$  are uniformly bounded between  $a$  and  $b$  for some fixed real numbers  $a < b$ . (Note that we may assume without loss of generality that  $l_{i,t} \in [0, 1]$ .) The Hedge Algorithm [2] (which is a reformulation of the Weighted Majority Algorithm [6]) assigns  $v_{i,t}$  proportional to  $\beta^{\sum_{s=1}^{t-1} l_{i,s}}$  and has the following loss bound

$$L_{\text{Hedge},T} \leq \frac{\ln(1/\beta)}{1-\beta} \left( \min_i L_{i,T} + \frac{\ln N}{\ln(1/\beta)} \right), \quad (1)$$

---

\* Supported by MEXT Grant-in-Aid for Scientific Research on Priority Area "New Horizon in Computing."

where  $\beta \in (0, 1)$  is a fixed learning rate. Vovk’s Aggregating Algorithm (AA, for short) produces a more sophisticated predictions [8] and gives a slightly better bound

$$L_{AA,T} \leq c(\beta) \left( \min_i L_{i,T} + \frac{\ln N}{\ln(1/\beta)} \right), \tag{2}$$

where

$$c(\beta) = \frac{\ln(1/\beta)}{N \ln \frac{N}{N+\beta-1}} < \frac{\ln(1/\beta)}{1-\beta}.$$

Cesa-Bianchi and Lugosi [1] give the following regret bound for the Hedge Algorithm

$$L_{\text{Hedge},T} - \min_{i \in \{1, \dots, N\}} L_{i,T} \leq \frac{\ln N}{\ln(1/\beta)} + \frac{T \ln(1/\beta)}{8} = \sqrt{(T/2) \ln N} \tag{3}$$

with an appropriate choice of  $\beta = 1 - O(\sqrt{(\ln N)/T})$ . (For this choice, the horizon  $T$  of the game needs to be known in advance). Kalai and Vempala [5] propose a family of algorithms based on Hannan’s “Follow the Perturbed Leader” approach [3] and its regret for this problem is shown to be of the same form as (3) with an additional small constant factor [4]. For all these results, the experts’ losses are assumed to be uniformly bounded to  $[0, 1]$ .

In this paper, we consider a more general setting in the following two ways. Firstly, we allow the learner to see experts’ decisions to make its own decisions about how to apportion the wager among the underlying options, rather than among the experts. Note that in the model described above, the learner is only allowed to interact with experts who actually make decisions against the environment, whereas the interactions between experts and the environment are hidden and just summarized as the loss vectors  $\mathbf{l}_t$ . Secondly, we do not require experts’ losses to be uniformly bounded but allow the learner to see upper and lower bounds on the  $l_{i,t}$  of the options before making its decisions. These assumptions are natural in many applications such as horse-race betting, where the learner knows  $l_{i,t}$  for all horses (options) before placing bets. Clearly, odds provide information about the maximum return for each option.

To be more specific, we assume that there are  $K$  options to be bet in every trial and the environment determines costs for the options, by which the losses for the learner and the experts are defined. At each trial  $t$ , the following happens.

1. Each expert  $i$  gives a  $K$ -dimensional probability vector  $\mathbf{x}_{i,t}$  that indicates how to apportion the wager among options.
2. The learner observes the  $a_{j,t}, b_{j,t}, \dots, a_{j,t}$  and  $b_{j,t}$  that are lower and upper bounds on the costs for all options  $j$ , respectively.
3. The learner chooses a  $K$ -dimensional probability vector  $\mathbf{p}_t$ .
4. The costs  $\mathbf{y}_t = (y_{1,t}, \dots, y_{K,t})$  for the options are revealed, where each cost correctly falls in the given range, i.e.,  $y_{j,t} \in [a_{j,t}, b_{j,t}]$  for all  $j$ .
5. The learner suffers loss  $\mathbf{p}_t \cdot \mathbf{y}_t$  and each expert  $i$  suffers loss  $l_{i,t} = \mathbf{x}_{i,t} \cdot \mathbf{y}_t$ .

Note that if the learner’s decision  $\mathbf{p}_t$  is given by  $\mathbf{p}_t = \sum_{i=1}^N v_{i,t} \mathbf{x}_{i,t}$  for a distribution  $\mathbf{v}_t$  over the experts, then we have  $\mathbf{p}_t \cdot \mathbf{y}_t = \mathbf{v}_t \cdot \mathbf{l}_t$ , which coincides with the learner’s loss in the previous model. Therefore, if we assume the uniform risk (i.e.,  $a_{j,t} = 0$  and  $b_{j,t} = 1$ ), then all the previous results remain to hold in the new model, regardless of the number  $K$  of options.

Vovk [8] also considers this model with the uniform risk and shows that the performance of the Aggregating Algorithm is again given by (2) but now the leading factor is

$$c(\beta) = \frac{\ln(1/\beta)}{K \ln \frac{K}{K+\beta-1}}, \tag{4}$$

which is monotonically increasing in  $K$ . So, if  $K < N$ , this gives an even better bound. Unfortunately, however, it seems to be hard to estimate a loss bound of the Aggregating Algorithm for the general case where the risk is not fixed.

In this paper, we assume a fixed but non-uniform upper risk, namely,  $a_{j,t} = 0$  and  $b_{j,t} = b_j$ , where  $\mathbf{b} = (b_1, \dots, b_K)$  is a fixed vector. In this setting, we analyze the performance of the Aggregating Algorithm and give a leading factor  $c(\beta)$  in the loss bound in terms of the risk vector  $\mathbf{b}$ . Surprisingly, we show that the Hedge Algorithm, which works without knowledge of risk information, performs nearly as well as the Aggregating Algorithm. So, the risk information does not help to improve the performance of algorithms in our model. We discuss some interesting behaviors of the loss bound. In particular, the factor  $c(\beta)$  does not depend on the  $b_j$ ’s that are smaller than some threshold value.

## 2 Aggregating Algorithm

The Aggregating Algorithm is a very general strategy that works for various games. In this section, we describe the algorithm with its performance bound in a generic form. Vovk shows that under some mild assumptions, the bound cannot be essentially improved and thus the Aggregating Algorithm is optimal [8].

First we describe a game that involves the learner,  $N$  experts, and the environment. A game is specified by a triple  $(\Gamma, \Omega, \lambda)$ , where  $\Gamma$  is a fixed prediction space,  $\Omega$  is a fixed outcome space, and  $\lambda : \Omega \times \Gamma \rightarrow [0, \infty]$  is a fixed loss function. At each trial  $t = 1, 2, \dots$ , the following happens.

1. Each expert  $i$  makes a prediction  $\gamma_{i,t} \in \Gamma$ .
2. The learner, who allowed to see all  $\gamma_{i,t}$ , makes his own prediction  $\gamma_t \in \Gamma$ .
3. The environment chooses some outcome  $\omega_t \in \Omega$ .
4. Each expert  $i$  suffers loss  $\lambda(\omega_t, \gamma_{i,t})$  and the learner suffers loss  $\lambda(\omega_t, \gamma_t)$ .

Next we give the assumptions about the game that makes the Aggregating Algorithm well-defined and perform optimally.

**Assumption 1** Let  $(\Gamma, \Omega, \lambda)$  be a game with  $N$  experts and a learner. Assume that

- $\Gamma$  is a finite set
- $\omega \in \Omega$  and  $\gamma \in \Gamma \mapsto \lambda(\omega, \gamma)$  is a real-valued function
- $\lambda(\omega, \gamma) < \infty$  for all  $\omega \in \Omega$  and  $\gamma \in \Gamma$
- $\lambda(\omega, \gamma) = 0$  for all  $\omega \in \Omega$  and  $\gamma \in \Gamma$

It is known that lots of games considered in the literature satisfies the assumptions.

We define a *simple game* in  $\Gamma$  to be a function  $Q$  that assigns to each element  $\gamma$  of its finite domain  $\text{dom } Q \subseteq \Gamma$  a positive weight  $Q(\gamma)$  so that  $\sum_{\gamma} Q(\gamma) = 1$  ( $\gamma$  ranging over  $\text{dom } Q$ ). Let  $\beta \in (0, 1)$ . A *simple game* (with respect to  $Q$ ) is a function from  $\Omega$  to the set of real numbers given by

$$g^Q(\omega) = \log_{\beta} \left( \sum_{\gamma \in \text{dom } Q} Q(\gamma) \beta^{\lambda(\omega, \gamma)} \right).$$

We will omit the superscript  $Q$  when it is clear from context. Let

$$c(\beta) = \sup_Q \inf_{\gamma \in \Gamma} \sup_{\omega \in \Omega} \frac{\lambda(\omega, \gamma)}{g^Q(\omega)}. \tag{5}$$

**Lemma 1 ([8]).**

Let  $\Sigma_{\beta}$  be a substitution function  $\Sigma_{\beta}: \Gamma \rightarrow \Gamma$  and  $Q$  be a simple game in  $\Gamma$  with  $\omega \in \Omega$

$$\lambda(\omega, \gamma) \leq c(\beta) g^Q(\omega),$$

then  $\gamma = \Sigma_{\beta}(g^Q)$

$$\Sigma_{\beta}(g^Q) = \arg \inf_{\gamma \in \Gamma} \sup_{\omega \in \Omega} \frac{\lambda(\omega, \gamma)}{g^Q(\omega)}. \tag{6}$$

Now we show how the Aggregating Algorithm behaves. It maintains a weight  $v_{i,t}$  for each expert  $i$  so that  $\mathbf{v}_t = (v_{1,t}, \dots, v_{N,t})$  is a probability vector. When the experts make predictions  $\gamma_{i,t}$ , we consider  $\mathbf{v}_t$  as a simple distribution  $Q_t$  such that  $Q_t(\gamma_{i,t}) = v_{i,t}$ . Then, the Aggregating Algorithm predicts with  $\gamma_t = \Sigma_{\beta}(g^{Q_t})$  with some substitution function  $\Sigma_{\beta}$ . When an outcome  $\omega_t$  is given, the weights are updated according to  $v_{i,t+1} = v_{i,t} \beta^{\lambda(\omega_t, \gamma_{i,t})} / Z$ , where  $Z$  is for normalization. We give a pseudocode in Figure 1.

The loss bound of the Aggregating Algorithm is represented by  $c(\beta)$ .

**Theorem 1 ([8]).**

Let  $T$  be a positive integer and  $\omega_1, \dots, \omega_T$  be a sequence of outcomes

$$L_{AA(\beta), T} \leq c(\beta) \left( \min_i L_{i,T} + \frac{\ln N}{\ln(1/\beta)} \right).$$

where  $c(\beta) = c(\beta) / \ln(1/\beta)$ . For any  $a < c(\beta)$ , we have  $L_{AA(\beta), T} \leq c \min_i L_{i,T} + a \ln N$ .

**Algorithm AA**( $\beta$ )  
**begin**  
 $\mathbf{v}_1 = (1/N, \dots, 1/N)$ ;  
**for**  $t = 1$  **to**  $T$  **do begin**  
 receive experts' predictions  $(\gamma_{1,t}, \dots, \gamma_{N,t})$ ;  
 let  $g_t : \omega \mapsto \log_\beta \sum_{i=1}^N v_{i,t} \beta^{\lambda(\omega, \gamma_{i,t})}$ ;  
 output  $\gamma_t = \Sigma_\beta(g_t)$ ;  
 observe an outcome  $\omega_t$  and suffer loss  $\lambda(\omega_t, \gamma_t)$ ;  
**for**  $i = 1$  **to**  $N$  **do**  

$$v_{i,t+1} = \frac{v_{i,t} \beta^{\lambda(\omega_t, \gamma_{i,t})}}{\sum_{j=1}^N v_{j,t} \beta^{\lambda(\omega_t, \gamma_{j,t})}}$$
;  
**end**  
**end**

**Fig. 1.** Aggregating Algorithm

### 3 The Aggregating Algorithm for Our Game

It is easy to see that the following game  $(\Gamma, \Omega, \lambda)$  corresponds to our online allocation problem: The prediction space  $\Gamma$  is the  $K$ -dimensional probability simplex, the outcome space is  $\Omega = [0, 1]^K$ , and the loss function is given by

$$\lambda(\boldsymbol{\omega}, \mathbf{p}) = \sum_{j=1}^K b_j \omega_j p_j,$$

where  $\mathbf{b} = (b_1, \dots, b_K)$  is a fixed risk vector. Note that the losses for options at trial  $t$  are given by  $y_{j,t} = b_j \omega_{j,t}$  for some  $\boldsymbol{\omega}_t \in \Omega$  so that  $y_{j,t} \in [0, b_j]$  and  $\lambda(\boldsymbol{\omega}_t, \mathbf{p}_t) = \mathbf{p}_t \cdot \mathbf{y}_t$ .

We can show that the assumptions in Assumption 1 are satisfied for this game. So the Aggregating Algorithm and the loss bound given in Theorem 1 apply. Vovk analyzes only the case where  $\mathbf{b} = \mathbf{1}^K = (1, \dots, 1)$  and show that  $c(\beta)$  is given by (4). In what follows, we assume, without loss of generality, that  $b_1 \leq \dots \leq b_K$ .

First we claim that it suffices to consider (5) for  $Q$  concentrated on the extreme points  $\mathbf{e}_j$  ( $j = 1, \dots, K$ ) of the simplex  $\Gamma$ , where the  $m$ -th component of  $\mathbf{e}_j$  is 1 if  $m = j$  and 0 otherwise.

**Lemma 2.**

$$\hat{\mathbf{p}} = (\hat{p}_1, \dots, \hat{p}_K) = \sum_{\mathbf{x} \in \text{dom } Q} Q(\mathbf{x}) \mathbf{x}.$$

$$Q'(\mathbf{e}_j) = \hat{p}_j \quad \boldsymbol{\omega} \in \Omega$$

$$g^Q(\boldsymbol{\omega}) \geq g^{Q'}(\boldsymbol{\omega}).$$

Note that each  $\mathbf{x} = (x_1, \dots, x_K) \in \text{dom } Q$  is a probability vector. The convexity of the function  $\zeta \mapsto \beta^\zeta$  implies

$$\begin{aligned} \sum_{\mathbf{x}} Q(\mathbf{x})\beta^{\lambda(\boldsymbol{\omega}, \mathbf{x})} &= \sum_{\mathbf{x}} Q(\mathbf{x})\beta^{\sum_j b_j \omega_j x_j} \leq \sum_{\mathbf{x}} Q(\mathbf{x}) \sum_j x_j \beta^{b_j \omega_j} \\ &= \sum_j \sum_{\mathbf{x}} x_j Q(\mathbf{x})\beta^{\lambda(\boldsymbol{\omega}, \mathbf{e}_j)} = \sum_j \hat{p}_j \beta^{\lambda(\boldsymbol{\omega}, \mathbf{e}_j)} = \sum_{\mathbf{x}} Q'(\mathbf{x})\beta^{\lambda(\boldsymbol{\omega}, \mathbf{x})}, \end{aligned}$$

which completes the lemma. □

Now we give a prediction  $\mathbf{p}_t$  of the learner of a closed form. Let  $g_t$  be the pseudoprediction defined at trial  $t$  in the Aggregating Algorithm. That is,

$$g_t(\boldsymbol{\omega}) = \log_{\beta} \sum_{i=1}^N v_{i,t} \beta^{\lambda(\boldsymbol{\omega}, \mathbf{x}_{i,t})}$$

where  $\mathbf{x}_{i,t}$  is the suggested prediction from expert  $i$ . By Lemma 2, we have another pseudoprediction

$$g'_t(\boldsymbol{\omega}) = \log_{\beta} \sum_{j=1}^K \hat{p}_{j,t} \beta^{\lambda(\boldsymbol{\omega}, \mathbf{e}_j)}$$

such that  $g_t(\boldsymbol{\omega}) \geq g'_t(\boldsymbol{\omega})$ , where  $\hat{\mathbf{p}}_t = (\hat{p}_{1,t}, \dots, \hat{p}_{K,t}) = \sum_{i=1}^N v_{i,t} \mathbf{x}_{i,t}$ . So, we compute  $\mathbf{p}_t = \arg \inf_{\mathbf{p}} \sup_{\boldsymbol{\omega}} \lambda(\boldsymbol{\omega}, \mathbf{p}) / g'_t(\boldsymbol{\omega})$  since Lemma 1 implies  $\lambda(\boldsymbol{\omega}, \mathbf{p}_t) \leq c(\beta)g'_t(\boldsymbol{\omega}) \leq c(\beta)g_t(\boldsymbol{\omega})$  as desired.

**Theorem 2.** *Let  $\mathbf{p}_t = \arg \inf_{\mathbf{p}} \sup_{\boldsymbol{\omega}} \lambda(\boldsymbol{\omega}, \mathbf{p}) / g'_t(\boldsymbol{\omega})$ .*

$$p_{j,t} = \frac{\frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}{\sum_{j=1}^K \frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})} \tag{7}$$

*Then,  $\lambda(\boldsymbol{\omega}, \mathbf{p}_t) \leq c(\beta) \sup_{\boldsymbol{\omega}} \lambda(\boldsymbol{\omega}, \mathbf{p}_t) / g'_t(\boldsymbol{\omega})$ .*

Since  $g'_t(\boldsymbol{\omega})$  is concave, it suffices to consider only for  $\boldsymbol{\omega}$  that are extreme points of the cube  $[0, 1]^K$ . Using the notation of  $I = \{j \mid \omega_j = 1, 1 \leq j \leq K\}$ , we write

$$\begin{aligned} g'_t(\boldsymbol{\omega}) &= \log_{\beta} \sum_{j=1}^K \hat{p}_{j,t} \beta^{b_j \omega_j} \\ &= \log_{\beta} \left( \sum_{j \in I} \beta^{b_j} \hat{p}_{j,t} + \sum_{j \notin I} \hat{p}_{j,t} \right) \\ &= \log_{\beta} \left( 1 - \sum_{j \in I} (1 - \beta^{b_j}) \hat{p}_{j,t} \right). \end{aligned}$$



So,

$$\inf_{\mathbf{p}} \sup_{\boldsymbol{\omega}} \frac{\lambda(\boldsymbol{\omega}, \mathbf{p})}{g'_t(\boldsymbol{\omega})} = \inf_{\mathbf{p}} \max_I \frac{\sum_{j \in I} b_j p_j}{\log_{\beta} \left( 1 - \sum_{j \in I} (1 - \beta^{b_j}) \hat{p}_{j,t} \right)}. \quad (8)$$

Using the inequality  $\sum_j \log_{\beta}(1 - a_j) = \log_{\beta} \prod_j (1 - a_j) \leq \log_{\beta}(1 - \sum_j a_j)$  for  $a_j > 0$ , we have

$$\begin{aligned} \frac{\sum_{j \in I} b_j p_j}{\log_{\beta} \left( 1 - \sum_{j \in I} (1 - \beta^{b_j}) \hat{p}_{j,t} \right)} &\leq \frac{\sum_{j \in I} b_j p_j}{\sum_{j \in I} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})} \\ &\leq \max_{j \in I} \frac{b_j p_j}{\log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}. \end{aligned}$$

Hence

$$\inf_{\mathbf{p}} \sup_{\boldsymbol{\omega}} \frac{\lambda(\boldsymbol{\omega}, \mathbf{p})}{g'_t(\boldsymbol{\omega})} = \inf_{\mathbf{p}} \max_{j \in \{1, \dots, K\}} \frac{p_j}{\frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}. \quad (9)$$

Note that for any  $\mathbf{p} \in \Gamma$ , it must hold that

$$\max_{j \in \{1, \dots, N\}} \frac{p_j}{\frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})} \geq \frac{1}{\sum_{j=1}^K \frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}$$

because otherwise we would have some  $\mathbf{p} \in \Gamma$  such that

$$p_j < \frac{\frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}{\sum_{j=1}^K \frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}$$

for all  $1 \leq j \leq K$ , which implies  $\sum_{j=1}^N p_j < 1$ , a contradiction. Therefore, our choice of

$$p_{j,t} = \frac{\frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}{\sum_{j=1}^K \frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) \hat{p}_{j,t})}$$

attains the infimum.  $\square$

Next we estimate the value of  $c(\beta)$ . Since  $c(\beta)$  depends on the risk vector  $\mathbf{b}$  as well, we write it as  $c(\beta, \mathbf{b})$  to explicitly specify  $\mathbf{b}$ . By virtue of Lemma 2 we can only consider a distribution  $Q$  on the extreme points. Let  $\mathbf{q} = (q_1, \dots, q_K)$  be the probability vector induced by  $Q$ , i.e.,  $q_j = Q(\mathbf{e}_j)$ . From the proof above, it follows that

$$c(\beta, \mathbf{b}) = \sup_Q \inf_{\mathbf{p}} \sup_{\boldsymbol{\omega}} \frac{\lambda(\boldsymbol{\omega}, \mathbf{p})}{g^Q(\boldsymbol{\omega})} = \sup_{\mathbf{q}} \frac{1}{\sum_{j=1}^K \frac{1}{b_j} \log_{\beta} (1 - (1 - \beta^{b_j}) q_j)}. \quad (10)$$

In other words, we need to solve the following optimization problem:

$$\text{minimize } - \sum_{j=1}^K \frac{1}{b_j} \ln (1 - (1 - \beta^{b_j}) q_j)$$

subject to  $\sum_j q_j = 1$  and  $q_j \geq 0$  for all  $j$ . Since the objective function and the domain are both convex, it is well known that  $\mathbf{q}$  is an optimal point if and only if the following Kursh-Kuhn-Tucker (KKT) conditions are satisfied:

$$\frac{1 - \beta^{b_j}}{b_j (1 - (1 - \beta^{b_j})q_j)} - s_j + t = 0, \quad (j = 1, \dots, K) \tag{11}$$

$$\sum_{j=1}^K q_j - 1 = 0, \tag{12}$$

$$s_j \cdot (-q_j) = 0, \quad (j = 1, \dots, K) \tag{13}$$

$$-q_j \leq 0, \quad (j = 1, \dots, K) \tag{14}$$

$$s_j \geq 0, \quad (j = 1, \dots, K) \tag{15}$$

for some  $t$  and  $s_j$  for  $j = 1, \dots, K$ . Note that the first condition (11) is derived from  $\nabla_{\mathbf{q}}L(\mathbf{q}, t, \mathbf{s}) = 0$  where  $L$  is the Lagrangian function.

Solving  $\mathbf{q}$  that satisfies these conditions, we get  $c(\beta, \mathbf{b})$  which is given in the next theorem.

**Theorem 3.** *Let  $b_1 \leq \dots \leq b_K$ .*

$$c(\beta, \mathbf{b}) = \frac{\ln(1/\beta)}{\sum_{j=z^*}^K \frac{1}{b_j} \ln\left(\frac{b_j}{1-\beta^{b_j}} s(z^*)\right)} \tag{16}$$

$$s(z) = \frac{\sum_{j=z}^K \frac{1}{b_j}}{\sum_{j=z}^K \frac{1}{1-\beta^{b_j}} - 1} \tag{17}$$

$$z^* = \arg \min_{z \in \{1, \dots, K\}} s(z). \tag{18}$$

*where  $z^*$  is a threshold index*

Let  $\mathbf{q}$  be a solution that satisfies the KKT conditions. By conditions (13) and (14), if  $q_j > 0$  then  $s_j = 0$ . So, conditions (11) and (15) imply that

$$q_j > 0 \Rightarrow q_j = \frac{1}{t(1 - \beta^{b_j})} \left( t + \frac{1 - \beta^{b_j}}{b_j} \right), \tag{19}$$

$$q_j = 0 \Rightarrow s_j = t + \frac{1 - \beta^{b_j}}{b_j} \geq 0. \tag{20}$$

Plugging (19) into condition (12), we get

$$t = -\frac{\sum_{j:q_j>0} \frac{1}{b_j}}{\sum_{j:q_j>0} \frac{1}{1-\beta^{b_j}} - 1}, \tag{21}$$

which is negative. So, we have that  $q_j > 0 \Leftrightarrow (1 - \beta^{b_j})/b_j < -t$ . Since the function  $(1 - \beta^\zeta)/\zeta$  is monotonically decreasing and we assume  $b_1 \leq \dots \leq b_K$ ,

it follows that there exists  $z^* \in \{1, \dots, K\}$  such that  $q_j = 0$  for all  $j < z^*$  and  $q_j > 0$  for all  $j \geq z^*$ . Equivalently,  $z^*$  must satisfy

$$\frac{1 - \beta^{b_{z^*}}}{b_{z^*}} < -t \leq \frac{1 - \beta^{b_{z^*-1}}}{b_{z^*-1}},$$

where

$$t = -s(z^*). \tag{22}$$

It is straightforward to confirm that  $z^* = \operatorname{argmin}_z s(z)$  actually satisfies the above inequalities. Now all the KKT conditions are satisfied. Plugging (19) with (22) into (10) we get the theorem.  $\square$

### 4 Risk Information does not Help Much

Recall that the prediction of the Aggregating Algorithm is given by

$$p_{j,t} \sim -\frac{1}{b_j} \ln(1 - (1 - \beta^{b_j})\hat{p}_{j,t})$$

where  $\hat{p}_{j,t} = \sum_{i=1}^N v_{i,t} \mathbf{x}_{i,t}$  is the prediction of the Hedge Algorithm that does not use the risk information. First, we observe that the prediction  $p_{j,t}$  is shifted to large values when the corresponding risk  $b_j$  is small. This supports our intuition that it would be safe to bet more on low-risk options. Figure 2 illustrates the shift caused by non-uniform risk.

Nevertheless, we show in the next theorem that the Hedge prediction  $\hat{p}_{j,t}$  approximates  $p_{j,t}$  by a constant factor that does not depend on the risk vector  $\mathbf{b}$ . This means that the Hedge Algorithm performs nearly as well as the Aggregating Algorithm without knowledge of risk information.

**Theorem 4.** For all  $\beta \in (0, 1)$  and  $1 \leq j \leq N$

$$\hat{p}_{j,t} \leq \frac{\ln(1/\beta)}{1 - \beta} p_{j,t}.$$

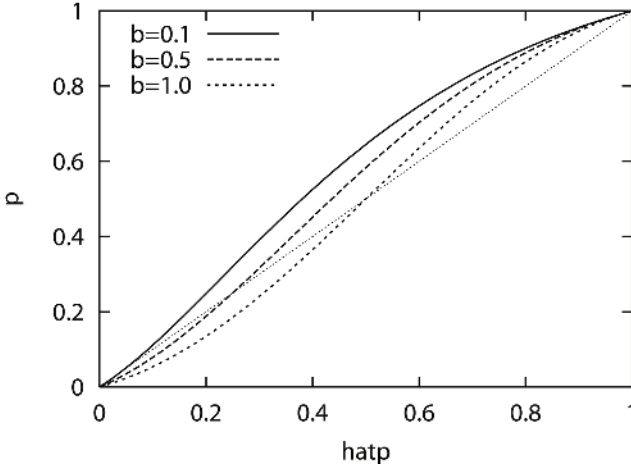
$$L_{\text{Hedge},T} \leq \frac{\ln(1/\beta)}{1 - \beta} L_{\text{AA},T}.$$

Using the inequalities  $(1 - c)x \leq -\ln(1 - (1 - c)x) \leq (\ln(1/c))x$ , we have

$$\frac{1 - \beta^{b_j}}{b_j} \hat{p}_{j,t} \leq -\frac{1}{b_j} \ln(1 - (1 - \beta^{b_j})\hat{p}_{j,t}) \leq (\ln(1/\beta)) \hat{p}_{j,t}.$$

Since the function  $(1 - \beta^x)/x$  is monotonically decreasing, the left hand side of the above formula is lower bounded by  $(1 - \beta)p_{j,t}$ , which implies the theorem.  $\square$

Note that the argument in the proof above shows that the smaller  $b_j$  is, the better  $\hat{p}_{j,t}$  approximates  $p_{j,t}$ , since  $\lim_{b_j \rightarrow 0} (1 - \beta^{b_j})/b_j = \ln(1/\beta)$ . This suggests that the gap of the losses of the both algorithms would be maximized when the risk is uniform. Moreover, in many natural situations, the optimal value of  $\beta$  is close to 1 and thus the approximation is tight.



**Fig. 2.** The prediction  $\mathbf{p} = (p, 1 - p)$  of the Aggregating Algorithm as the function of Hedge prediction  $\hat{\mathbf{p}} = (\hat{p}, 1 - \hat{p})$  for  $K = 2$ ,  $\beta = 0.1$ , and for various risk vectors of the form of  $\mathbf{b} = (b, 1)$  (i.e., the risk for the second option is fixed to be one). The curve for  $b = 1$  is the prediction of the normal Aggregating Algorithm that assumes the uniform risk.

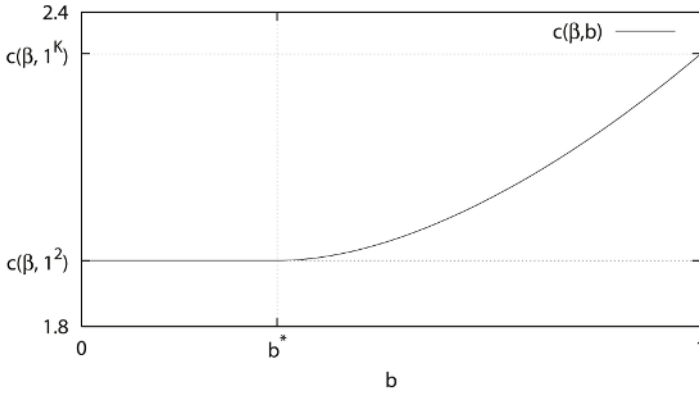
### 5 Behaviors of $c(\beta, \mathbf{b})$

Here we observe some interesting properties of  $c(\beta, \mathbf{b})$ . First we confirm that  $c(\beta, \mathbf{b})$  with the uniform risk vector  $\mathbf{b} = 1^K$  coincides with the classical value (4). (In this case, the threshold index is  $z^* = 1$ .) Second, it is easy to verify that  $s(K - 1) < s(K)$  where the function  $s(z)$  is defined in (17) and by (18) the threshold index  $z^*$  must satisfy  $1 \leq z^* \leq K - 1$ . This implies that  $c(\beta, \mathbf{b}) > 1$  for all  $\mathbf{b}$ . Moreover, since  $s(z)$  does not depend on  $b_1, \dots, b_{z-1}$  and  $z^*$  is the largest index  $z$  that satisfies  $s(z - 1) > s(z)$ , we can conclude that  $z^*$  is determined independently of the values  $b_1, \dots, b_{z^*-2}$ . Below we examine this insensitivity to small risks in more detail.

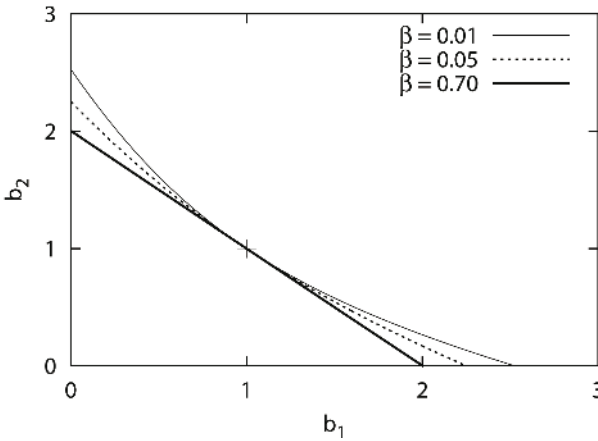
Recall that  $c(\beta, \mathbf{b})$  given in (16) does not depend on small  $b_j$ 's for  $j < z^*$ . In other words,  $c(\beta, \mathbf{b})$  is the same as  $c(\beta, \mathbf{b}')$  for  $\mathbf{b}' = (b_{z^*}, b_{z^*+1}, \dots, b_K)$  which corresponds to a game with  $K - z^* + 1$  options. It is curious that the prediction of the Aggregating Algorithm strongly depends on small risks but the loss bound does not depend on them at all. Let us look at this more closely. Consider an extreme case where  $b_{K-1} = b_K = 1$ . Then, we can show that if

$$\frac{b_{K-2}}{1 - \beta^{b_{K-2}}} \leq \frac{1}{1 - \beta} - \frac{1}{2},$$

then  $z^* = K - 1$  and hence  $c(\beta, \mathbf{b}) = c(\beta, 1^2)$  regardless of the number  $K$  of options. In Figure 3 we show the curve of  $c(\beta, \mathbf{b})$  for  $\mathbf{b} = (b, b, b, 1, 1)$  ( $K = 5$ )



**Fig. 3.** The curve of  $c(\beta, \mathbf{b})$  with  $b_1 = \dots = b_{K-2} = b$  and  $b_{K-1} = b_K = 1$ . The threshold  $b^*$  is given by the solution of  $\frac{b^*}{1-\beta b^*} = \frac{1}{1-\beta} - \frac{1}{2}$ . Here we set  $K = 5$  and  $\beta = 0.1$ .



**Fig. 4.** Curves of  $\mathbf{b} = (b_1, b_2)$  that satisfies  $c(\beta, \mathbf{b}) = c(\beta, 1^2)$  for various  $\beta$

as the function of  $b$ . From this we can confirm that  $c(\beta, \mathbf{b})$  decreases as  $b$  gets smaller and converges to  $c(\beta, 1^2)$  when  $b < b^*$  for some threshold  $b^*$ .

Next we investigate a condition that makes  $c(\beta, \mathbf{b}) \leq c(\beta, 1^K)$ . Note that  $\mathbf{b} \leq 1^K$  is a trivial condition. Interestingly, if some of the risks  $b_j$  are much larger than 1,  $c(\beta, \mathbf{b})$  can be smaller than  $c(\beta, 1^K)$ . Figure 4 describes the curves of  $\mathbf{b} = (b_1, b_2)$  such that  $c(\beta, \mathbf{b}) = c(\beta, 1^K)$  with  $K = 2$  for various values of  $\beta$ . If a point  $\mathbf{b}$  lays in the lower left part of the curve for  $\beta$ , then  $c(\beta, \mathbf{b}) < c(\beta, 1^K)$ . Intuitively, the curves show that the algorithm performs better when faced with options of various risks than when faced with options of the same risk.

## 6 Concluding Remarks

We generalize the online allocation model so that the learner is allowed to see the risk information about the options. We apply the Aggregating Algorithm and give a tight loss bound. Although the results we give are only for a restricted case where the risks are all lower bounded by zero and time invariant, the model has a nice application, e.g., the online shortest path problem [7]: When given a network, the learner tries to choose a routing path in every trial in an attempt to minimize the total expected time delay. In this case, each path corresponds to an option and the number of nodes in the path can be regarded as its risk. (The more nodes it has, the more time is expected to take for sending a packet through that path). Since the Hedge Algorithm combined with the technique of the path kernel efficiently solves this problem, our results give a refined loss bound.

We can view our results as a starting point to investigate the role of non-uniform risks in prediction problems. For example, it is interesting to generalize the portfolio game to the one with risk information, where the game has the nonlinear loss function  $-\ln \mathbf{p} \cdot \mathbf{y}$ .

We remark that the conditions for which our results hold can be generalized to some extent. First we can show that all the arguments developed in this paper remain hold even if the risk vector is permuted in each trial, i.e., we allow the risk vector  $\mathbf{b}_t$  to be a permutation on a fixed set  $\{b_1, \dots, b_K\}$ .

Moreover, we can treat a dual case where the risks are all upper bounded by zero, i.e., the costs  $y_{j,t}$  fall in the range  $[-a_j, 0]$  where  $(a_1, \dots, a_K)$  is a fixed gain vector (now  $a_j$  is interpreted as the largest possible gain for the  $j$ -th option). For this we develop the gain version of the Aggregating Algorithm in a similar way and the dual arguments say that the prediction

$$p_{j,t} = \frac{\frac{1}{a_j} \log_\alpha (1 + (\alpha^{a_j} - 1) \hat{p}_{j,t})}{\sum_{j=1}^K \frac{1}{a_j} \log_\alpha (1 + (\alpha^{a_j} - 1) \hat{p}_{j,t})}$$

with some learning rate  $\alpha > 1$  guarantees

$$G_{AA,T} \geq c(\alpha) \left( \max_i G_{i,T} - \frac{\ln N}{\ln \alpha} \right),$$

where  $G_{AA,T}$  and  $G_{i,T}$  are the total gain of the AA and that of expert  $i$ , respectively. The factor  $c(\alpha)$  is now given by

$$c(\alpha) = \frac{\ln \alpha}{\sum_{j=z^*}^K \frac{1}{a_j} \ln \left( \frac{\alpha^{a_j} - 1}{a_j s(z^*)} \right)},$$

where

$$z^* = \arg \max_z s(z), \quad s(z) = \frac{\sum_{j=z}^K \frac{1}{a_j}}{1 + \sum_{j=1}^K \frac{1}{\alpha^{a_j} - 1}}.$$

It also holds that the gain version of the Hedge Algorithm is a good approximation of the AA so that

$$G_{\text{Hedge},T} \geq \frac{\ln \alpha}{\alpha - 1} G_{\text{AA},T}.$$

So far we do not know when the risk information helps. Even in the restricted case, we showed that there is a small but non-negligible gap between the Aggregating and the Hedge Algorithms. But the gap already exists under the uniform risk. So it is interesting to investigate whether the gap becomes large in a non-uniform risk setting.

## References

- [1] N. Cesa-Bianchi and G. Lugosi. On prediction of individual sequences. *Annals of Statistics*, 27(6):1865–1895, 1999.
- [2] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS*, 55(1):119–139, 1997.
- [3] J. Hannan. *Approximation to Bayes risk in repeated play*, volume III. Princeton University Press, 1957.
- [4] M. Hutter and J. Poland. Prediction with expert advice by following the perturbed leader for general weights. In *Proc. ALT 2004*, volume 3244 of *LNAI*, 279–293, 2004.
- [5] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. In *Proc. COLT 2003*, volume 2777 of *LNAI*, 26–40, 2003.
- [6] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Inform. Comput.*, 108(2):212–261, 1994.
- [7] E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003.
- [8] V. Vovk. A game of prediction with expert advice. *JCSS*, 56(2):153–173, 1998.

# Defensive Universal Learning with Experts

Jan Poland<sup>1</sup> and Marcus Hutter<sup>2</sup>

<sup>1</sup> Grad. School of Inf. Sci. and Tech., Hokkaido University, Japan  
[jan@ist.hokudai.ac.jp](mailto:jan@ist.hokudai.ac.jp)

<http://www-alg.ist.hokudai.ac.jp/~jan>  
<sup>2</sup> IDSIA, Galleria 2, CH-6928 Manno (TI), Switzerland  
[marcus@idsia.ch](mailto:marcus@idsia.ch)  
<http://www.idsia.ch/~marcus>

**Abstract.** This paper shows how universal learning can be achieved with expert advice. To this aim, we specify an experts algorithm with the following characteristics: (a) it uses only feedback from the actions actually chosen (bandit setup), (b) it can be applied with countably infinite expert classes, and (c) it copes with losses that may grow in time appropriately slowly. We prove loss bounds against an adaptive adversary. From this, we obtain a master algorithm for “reactive” experts problems, which means that the master’s actions may influence the behavior of the adversary. Our algorithm can significantly outperform standard experts algorithms on such problems. Finally, we combine it with a universal expert class. The resulting universal learner performs – in a certain sense – almost as well as any computable strategy, for any online decision problem. We also specify the (worst-case) convergence speed, which is very slow.

## 1 Introduction

Expert advice has become a well-established paradigm of machine learning in the last decade, in particular for prediction. It is very appealing from a theoretical point of view, as performance guarantees usually hold in the worst case, without any (statistical) assumption on the data. Such assumptions are generally required for other statistical learning methods, often however not resulting in stronger guarantees.

Using expert advice in the standard way seems a rather bad idea in some cases where the decisions of the learner or master algorithm influence the behavior of the environment or adversary. One example is the repeated prisoner’s dilemma when the opponent plays “tit for tat” (see Section 4). This was noted and resolved by [1], who introduced a “strategic expert algorithm” for so-called reactive environments. Their algorithm works with a finite class of experts and attains asymptotically optimal behavior. No convergence speed is asserted, and the analysis is quite different from that of standard experts algorithms.

In this paper, we show how the more general task with a countably infinite expert class can be accomplished, building on standard experts algorithms, and simultaneously also bounding the convergence rate ( $t^{-\frac{1}{10}}$ , which can be actually improved to  $t^{-\frac{1}{3}+\epsilon}$ ). To this aim, we will combine techniques from [2, 3, 4, 5] and obtain a master algorithm which performs well on loss functions that may



Then this is applied to (possibly) reactive problems by yielding the control to the selected expert for an increasing period of time steps. Using a universal expert class defined by the countable set of all programs on some fixed universal Turing machine, we obtain an algorithm which is in a sense asymptotically optimal with respect to any computable strategy. An easy additional construction guarantees that our algorithm is computable, in contrast to other universal approaches which are non-computable [6]. To our knowledge, we also propose the first algorithm for non-stochastic bandit problems with countably many arms.

The paper is structured as follows. Section 2 introduces the problem setup, the notation, and the algorithm. In Sections 3, we give the (worst-case) analysis of the master algorithm. The implications to active experts problems and a universal master algorithms are given in Section 4. We discuss our results in Section 5.

## 2 The Master Algorithm

**Setup.** We are acting in an online decision problem. “We” is here an abbreviation for the master algorithm which is to be designed. An “online decision problem” is to be understood in a very general sense, it is just a sequence of decisions each of which results in some loss. This could be e.g. a prediction task, a repeated game, etc. In each round, that is at each time step  $t$ , we have access to the recommendations of countably infinitely many “experts” or strategies. (For simplicity, we restrict our notation to a countably infinite expert class, all results also hold for finite classes.) We do not specify what exactly a “recommendation” is – we just follow the advice of one expert. At each time step  $t$ , we reveal our move, the adversary has to assign losses  $\ell_t^i \geq 0$  to each expert  $i$ . There is an upper bound  $B_t \geq \|\ell_t\|_\infty$  on the maximum loss the adversary may use. This quantity may depend on  $t$  and is not controlled by the adversary. After the move, only the loss of the selected expert  $i$  is revealed. Our goal is to perform nearly as well as the best available strategy (expert) in terms of cumulative loss, after any number  $T$  of time steps which is not known in advance. The difference between our loss and the loss of some expert is also termed *regret*. We consider the general case of an *adversary*, which may assign losses depending on our past decisions.

If there are only finitely many experts or strategies, then it is common to give no prior preferences to any of them. Formally, this is realized by defining uniform prior weights  $w^i = \frac{1}{n}$  for each expert  $i$ . This is not possible for countably infinite expert classes, as there is no uniform distribution on the natural numbers. In this case, we need some non-uniform prior  $(w^i)_{i \in \mathbb{N}}$  and require  $w^i > 0$  for all experts  $i$  and  $\sum_i w^i \leq 1$ . We also define the complexity of expert  $i$  as  $k^i = -\ln w^i$ . This quantity is important since in the full observation game (i.e. after our decision we get to know the losses of all experts), the regret can usually be bounded by some function of the best expert’s complexity.

Our algorithm “Follow or Explore” (Fig. 1) builds on McMahan and Blum’s online geometric optimization algorithm [4]. It is a bandit version

For  $t = 1, 2, 3, \dots$   
 set  $\hat{\ell}_t^i = \hat{B}_t$  for  $i \notin \{t \geq \tau\}$  (see (2))  
 sample  $r_t \in \{0, 1\}$  independently s.t.  $P[r_t = 1] = \gamma_t$   
 If  $r_t = 0$  Then  
     invoke  $FPL(t)$  and play its decision  
     set  $\hat{\ell}_t^i = 0$  for  $i \in \{t \geq \tau\}$   
 Else  
     sample  $I_t^{FoE} := u_t$  “uniformly”, see (1), and play  $I := I_t^{FoE}$   
     set  $\hat{\ell}_t^I = \ell_t^I / (u_t^I \gamma_t)$  and  $\hat{\ell}_t^i = 0$  for  $i \in \{t \geq \tau\} \setminus \{I\}$

**Fig. 1.** The algorithm  $FoE$ . The exploration rate  $\gamma_t$  will be specified in Corollary 8.

Sample  $q_t^i \stackrel{d}{\sim} Exp$  (i.e.  $\mathbf{P}(q_t^i \geq x) = e^{-x}$  for  $x \geq 0$ ) indep.  $\forall i \in \{t \geq \tau\}$   
 select and play  $I_t^{FPL} = \arg \min_{i:t \geq \tau} \{\eta_t \hat{\ell}_{<t}^i + k^i - q_t^i\}$

**Fig. 2.** The algorithm  $FPL$ . The learning rate  $\eta_t$  will be specified in Corollary 8.

of a “Follow the Perturbed Leader” experts algorithm. This approach to online prediction and playing repeated games has been pioneered by Hannan [2]. For the full observation game and uniform prior, [3] gave a very elegant analysis which is clearly different from the standard analysis of exponential weighting schemes. It has one advantage over other aggregating algorithms such as exponential weighting schemes: the analysis is not complicated if the learning rate is dynamic rather than fixed in advance. A dynamic learning rate is necessary if there is no target time  $T$  known in advance. For non-uniform prior, an analysis was given in [5]. The following issues are important for  $\text{FoE}$ ’s design.

**Exploration.** Since we are playing the bandit game (as opposed to the full information game), we need to explore sufficiently [7, 4]. At each time step  $t$ , we decide randomly according to some exploration rate  $\gamma_t \in (0, 1)$  whether to explore or not. If so, we would like to choose an expert according to the prior distribution. There is a caveat: In order to make the analysis go through, we have to assure that we are working with unbiased estimates of the losses. This is achieved by dividing the observed loss by the probability of choosing the expert. But this quantity could become arbitrarily large if we admit arbitrarily small weights. We address this problem by restricting the expert pool at each time  $t$ . For each expert  $i$ , we define an active time  $\tau^i$ , that is, expert  $i$  is active only for  $t \geq \tau^i$ . We denote the set of active experts at time  $t$  by  $\{t \geq \tau\} = \{i : t \geq \tau^i\}$ . For exploration, the prior is then replaced by the finitized prior distribution  $u_t$ ,

$$\mathbf{P}(u_t = i) = \frac{w^i \mathbb{1}_{t \geq \tau^i}}{\sum_j w^j \mathbb{1}_{t \geq \tau^j}}. \tag{1}$$

Consequently, the maximum unbiasedly estimated instantaneous loss is (note that the exploration probability also scales with the exploration rate  $\gamma_t$ )

$$\hat{B}_t = \frac{B_t}{\gamma_t \min\{w^i : t \geq \tau^i\}}. \tag{2}$$

It is convenient for the analysis to assign estimated loss of  $\hat{B}_t$  to all currently inactive experts. Observe finally that in this way, our master algorithm  $\bar{\gamma}$  always deals with a finite expert class and is thus computable.

**Follow the Perturbed Leader.** ( $\bar{\gamma}$ , specified in Fig. 2) is invoked if  $\bar{\gamma}$  does not explore. Just following the “leader” (the best expert so far) may not be a good strategy [3]. Instead we subtract an exponentially distributed perturbation  $q_t$  from the current score (the complexity penalized past loss) of the experts. An important detail of the  $\bar{\gamma}$  subroutine is the  $\eta_t > 0$ , which should be adaptive if the total number of steps  $T$  is not known in advance. Please see e.g. [3, 5] for more details. Also the variant of  $\bar{\gamma}$  we use (specified in Fig. 2) works on the finitized expert pool.

Note that each time randomness is used, it is assumed to be  $\mathcal{F}_{t-1}$  of the past randomness. Performance is evaluated in terms of true or estimated cumulative loss, this is specified in the notation. E.g. for the true loss of  $\bar{\gamma}$  up to and including time  $T$  we write  $\ell_{1:T}^{FPL}$ , while the estimated loss of  $\bar{\gamma}$  and not including time  $T$  is  $\hat{\ell}_{<T}^{FoE}$ .

### 3 Analysis on the Master’s Time Scale

The following analysis uses McMahan and Blum’s trick [4] in order to prove bounds against adaptive adversary. With a different argument, it is possible to circumvent Lemma 6, thus achieving better bounds [8]. This will be briefly discussed in the last section.

Let  $B_t \geq 0$  be some sequence of upper bounds on the instantaneous losses,  $\gamma_t \in (0, 1)$  be a sequence of exploration rates, and  $\eta_t > 0$  be a sequence of learning rates. The analysis proceeds according to the following diagram (where  $L$  is an informal abbreviation for the loss and always refers to cumulative loss, but sometimes additionally to instantaneous loss).

$$L^{FoE} \lesssim \mathbf{E}L^{FoE} \lesssim \mathbf{E}L^{FPL} \lesssim \mathbf{E}\hat{L}^{FPL} \lesssim \mathbf{E}\hat{L}^{IFPL} \lesssim \mathbf{E}\hat{L}^{best} \lesssim L^{best} \tag{3}$$

Each “ $\lesssim$ ” means that we bound the quantity on the left by the quantity on the right plus some additive term. The first and the last expressions are the losses of the  $\bar{\gamma}$  algorithm and the best expert, respectively. The intermediate quantities belong to different algorithms, namely  $\bar{\gamma}$ ,  $\bar{\gamma}$ , and a third one called  $\bar{\gamma}$  for “infeasible” FPL [3].  $\bar{\gamma}$ , as specified in Fig. 3, is the same as  $\bar{\gamma}$  except that it has access to an oracle providing the current estimated loss vector  $\hat{\ell}_t$  (hence infeasible). Then it assigns scores of  $\eta_t \hat{\ell}_{1:t}^i + k^i - q_t^i$  instead of  $\eta_t \ell_{<t}^i + k^i - q_t^i$ .

The randomization of  $\bar{\gamma}$  and  $\bar{\gamma}$  gives rise to two filtrations of  $\sigma$ -algebras. By  $\mathcal{A}_t$  we denote the  $\sigma$ -algebra generated by the  $\bar{\gamma}$ ’s randomness up to time  $t$ , meaning  $\mathcal{A}_t$  the random variables  $\{u_{1:t}, r_{1:t}\}$ . Then  $(\mathcal{A}_t)_{t \geq 0}$  is a filtration ( $\mathcal{A}_0$  is the trivial  $\sigma$ -algebra). We may also write  $\mathcal{A} = \bigcup_{t \geq 0} \mathcal{A}_t$ . Similarly,  $\mathcal{B}_t$  is

Sample  $q_t^i \stackrel{d}{\sim} \text{Exp}$  independently for all  $i \in \{t \geq \tau\}$   
 select and play  $I_t^{FPL} = \arg \min_{i:t \geq \tau} \{\eta_t \hat{\ell}_{1:t}^i + k^i - q_t^i\}$

**Fig. 3.** The algorithm *IFPL*. The learning rate  $\eta_t$  will be specified in Corollary 8.

the  $\sigma$ -algebra generated by the  $\tau_{i,t}$ 's  $\tau_{i,t} > \tau$ 's randomness up to time  $t$  (i.e.  $\mathcal{B}_t \hat{=} \{u_{1:t}, r_{1:t}, q_{1:t}\}$ ). Then clearly  $\mathcal{A}_t \subset \mathcal{B}_t$  for each  $t$ .

The reader should think of the expectations in (3) as of both ordinary and conditional expectations. Conditional expectations are mostly with respect to  $\tau_{i,t}$ 's past randomness  $\mathcal{A}_{t-1}$ . These conditional expectations of some random variable  $X$  are abbreviated by

$$\mathbf{E}_t[X] := \mathbf{E}[X | \mathcal{A}_{t-1}].$$

Then  $\mathbf{E}_t[X]$  is an  $\mathcal{A}_{t-1}$ -measurable random variable, meaning that its value is determined for fixed past randomness  $\mathcal{A}_{t-1}$ . Note in particular that the estimated loss vectors  $\hat{\ell}_t^i$  are random vectors which depend on  $\tau_{i,t}$ 's randomness  $\mathcal{A}_t$  up to time  $t$ . In this way,  $\tau_{i,t}$ 's (and  $\tau_{i,t} > \tau$ 's and  $\tau_{i,t} < \tau$ 's) actions depend on  $\tau_{i,t}$ 's past randomness. Note, however, that they do not depend on  $\tau_{i,t} > \tau$ 's randomness  $q_{1:t}$ .

We now start with proving the diagram (3). In order to understand the analysis, it is important to consider each intermediate algorithm as a stand-alone procedure which is actually executed (with an oracle if necessary) on the specified inputs (e.g. on the estimated losses) and has the asserted performance guarantees (e.g. again on the estimated losses).

**Lemma 1.**  $[L^{FoE} \lesssim \mathbf{E}L^{FoE}]_{\tau_{i,t} > \tau, T \geq 1, \delta_T \in (0, 1)}$   $\bullet \dots \bullet$   
 $\dots \dots 1 - \frac{\delta_T}{2} \bullet \dots \bullet$

$$\ell_{1:T}^{FoE} \leq \sum_{t=1}^T \mathbf{E}_t \ell_t^{FoE} + \sqrt{(2 \ln \frac{4}{\delta_T}) \sum_{t=1}^T B_t^2}.$$

The sequence of random variables  $X_T = \sum_{t=1}^T [\ell_t^{FoE} - \mathbf{E}_t \ell_t^{FoE}]$  is a martingale with respect to the filtration  $\mathcal{B}_t$  (not  $\mathcal{A}_t$ !). In order to see this, observe  $\mathbf{E}[\ell_T^{FoE} | \mathcal{B}_{T-1}] = \mathbf{E}(\mathbf{E}[\ell_T^{FoE} | \mathcal{A}_{T-1}] | \mathcal{B}_{T-1})$  and  $\mathbf{E}[\ell_t^{FoE} | \mathcal{B}_{T-1}] = \ell_t^{FoE}$  for  $t < T$ , which implies

$$\begin{aligned} \mathbf{E}[X_T | \mathcal{B}_{T-1}] &= \sum_{t=1}^T (\mathbf{E}[\ell_t^{FoE} | \mathcal{B}_{T-1}] - \mathbf{E}[\mathbf{E}[\ell_t^{FoE} | \mathcal{A}_{t-1}] | \mathcal{B}_{T-1}]) \\ &= \sum_{t=1}^{T-1} (\ell_t^{FoE} - \mathbf{E}[\ell_t^{FoE} | \mathcal{A}_{t-1}]) = X_{T-1}. \end{aligned}$$

Its differences are bounded:  $|X_t - X_{t-1}| \leq B_t$ . Hence, it follows from Azuma's inequality (see e.g. [9]) that the probability that  $X_T$  exceeds some  $\lambda > 0$  is bounded by  $p = 2 \exp(-\frac{\lambda^2}{2 \sum_t B_t^2})$ . Requesting  $\frac{\delta_T}{2} = p$  and solving for  $\lambda$  gives the assertion. □

**Lemma 2.**  $[\mathbf{E}\ell^{FoE} \lesssim \mathbf{E}\ell^{FPL}] \mathbf{E}_t \ell_t^{FoE} \leq (1 - \gamma_t)\mathbf{E}_t \ell_t^{FPL} + \gamma_t B_t, \forall t \geq 1$

This follows immediately from the specification of  $\ell_t^{FoE}$ . Clearly, a corresponding assertion for the ordinary expectations holds by just taking expectations on both sides. This is the case for all subsequent lemmas, except for Lemma 6.

The next lemma relating  $\mathbf{E}L^{FPL}$  and  $\mathbf{E}\hat{L}^{FPL}$  is technical but intuitively clear. It states that in expectation, the real loss suffered by  $\mathcal{A}_t$  is the same as the estimated loss. This is simply because the loss estimate is unbiased. A combination of this and the previous lemma was shown in [4]. Note that  $\hat{\ell}_t^{FPL}$  is the loss  $\hat{\ell}_t^I$  estimated by  $\mathcal{A}_t$ , but for the expert  $I = I_t^{FPL}$  chosen by  $\mathcal{A}_t$ .

**Lemma 3.**  $[\mathbf{E}L^{FPL} \lesssim \mathbf{E}\hat{L}^{FPL}] \mathbf{E}_t \ell_t^{FPL} = \mathbf{E}_t \hat{\ell}_t^{FPL}, t \geq 1$

Let  $f_t^i = \mathbf{P}[I_t^{FPL} = i | \mathcal{A}_{t-1}]$  be the probability distribution over actions  $i$  which  $\mathcal{A}_t$  uses at time  $t$ , depending on the past randomness  $\mathcal{A}_{t-1}$ . Let  $u_t$  be the finitized prior distribution (1) at time  $t$ . Then

$$\mathbf{E}_t[\hat{\ell}_t^{FPL}] = (1 - \gamma_t) \cdot 0 + \gamma_t \sum_{i=1}^{\infty} f_t^i [(1 - u_t^i) \cdot 0 + u_t^i \hat{\ell}_t^i |_{r_t=1 \wedge I_t^{FoE}=i}] = \sum_{i=1}^{\infty} f_t^i \ell_t^i = \mathbf{E}_t[\ell_t^{FPL}],$$

where  $\hat{\ell}_t^i |_{r_t=1 \wedge I_t^{FoE}=i} = \ell_t^i / (u_t^i \gamma_t)$  is the estimated loss under the condition that  $\mathcal{A}_t$  decided to explore ( $r_t = 1$ ) and chose action  $I_t^{FoE} = i$ .  $\square$

The following lemma relates the losses of  $\mathcal{A}_t$  and  $\hat{\mathcal{A}}_t$ . It is proven in [3] and [5]. We give the full proof, since it is the only step in the analysis where we have to be careful with the upper loss bound  $B_t$ . Let  $\hat{B}_t$  be the upper bound on the estimated loss (2). (We remark that also for losses which grow sufficiently slowly do not cause any problem in the analysis. In this way, it is straightforward to modify the algorithm by Auer et al. [10] for reactive tasks with a finite expert class.)

**Lemma 4.**  $[\mathbf{E}\hat{L}^{FPL} \lesssim \mathbf{E}L^{IFPL}] \mathbf{E}_t \hat{\ell}_t^{FPL} \leq \mathbf{E}_t \hat{\ell}_t^{IFPL} + \gamma_t \eta_t \hat{B}_t^2, t \geq 1$

If  $r_t = 0$ ,  $\hat{\ell}_t = 0$  and thus  $\hat{\ell}_t^{FPL} = \hat{\ell}_t^{IFPL}$  holds. This happens with probability  $1 - \gamma_t$ . Otherwise we have

$$\mathbf{E}_t \hat{\ell}_t^{FPL} = \sum_{i=1}^{\infty} \int \mathbb{I}_{I_t^{FPL}=i} \hat{\ell}_t^i d\mu(x), \tag{4}$$

where  $\mu$  denotes the (exponential) distribution of the perturbations, i.e.  $x_i := q_t^i$  and density  $\mu(x) := e^{-\|x\|_{\infty}}$ . The idea is now that if action  $i$  was selected by  $\mathcal{A}_t$ , it is – because of the exponentially distributed perturbation – with high probability also selected by  $\hat{\mathcal{A}}_t$ . Formally, we write  $u^+ = \max(u, 0)$  for  $u \in \mathbb{R}$ , abbreviate  $\lambda = \hat{\ell}_{<t} + k/\eta_t$ , and denote by  $\int \dots d\mu(x_{\neq i})$  the integration leaving

out the  $i$ th action. Then, using  $\eta_t \lambda_i - x_i \leq \eta_t \lambda_j - x_j \forall j$  if  $I_t^{FPL} = i$  in the first equation, and  $\hat{B}_t \geq \hat{\ell}_t^i - \hat{\ell}_t^j$  in the last line, we get

$$\begin{aligned} \int \mathbb{I}_{I_t^{FPL}=i} \hat{\ell}_t^i d\mu(x) &= \int \int_{x_i \geq \max_{j \neq i} \{\eta_t(\lambda_i - \lambda_j) + x_j\}} \hat{\ell}_t^i d\mu(x_i) d\mu(x_{\neq i}) = \int \hat{\ell}_t^i e^{-\max_{j \neq i} \{\eta_t(\lambda_i - \lambda_j) + x_j\}^+} d\mu(x_{\neq i}) \\ &\leq \int \hat{\ell}_t^i e^{\eta_t \hat{B}_t} e^{-\max_{j \neq i} \{\eta_t(\lambda_i - \lambda_j) + x_j\} + \eta_t \hat{B}_t} d\mu(x_{\neq i}) \\ &\leq e^{\eta_t \hat{B}_t} \int \hat{\ell}_t^i e^{-\max_{j \neq i} \{\eta_t(\lambda_i + \hat{\ell}_t^i - \lambda_j - \hat{\ell}_t^j) + x_j\}^+} d\mu(x_{\neq i}) \\ &= e^{\eta_t \hat{B}_t} \int \mathbb{I}_{I_t^{FPL}=i} \hat{\ell}_t^i d\mu(x). \end{aligned}$$

Summing over  $i$  and using the analog of (4) for  $\hat{\ell}_t^{IFPL}$ , we see that if  $r_t = 1$ , then  $\mathbf{E}_t \hat{\ell}_t^{FPL} \leq e^{\eta_t \hat{B}_t} \mathbf{E}_t \hat{\ell}_t^{IFPL}$  holds. Thus  $\mathbf{E}_t \hat{\ell}_t^{IFPL} \geq e^{-\eta_t \hat{B}_t} \mathbf{E}_t \hat{\ell}_t^{FPL} \geq (1 - \eta_t \hat{B}_t) \mathbf{E}_t \hat{\ell}_t^{FPL} \geq \mathbf{E}_t \hat{\ell}_t^{FPL} - \eta_t \hat{B}_t^2$ . The assertion now follows by taking expectations w.r.t.  $r_t$ .  $\square$

The next lemma relates the losses of  $\hat{\ell}_t^{IFPL}$  and the best action in hindsight. For an oblivious adversary (which means that the adversary’s decisions do not depend on our past actions), the proof is quite simple [3]. An additional step is necessary for an adaptive adversary [11].

**Lemma 5.**  $[\mathbf{E} \hat{L}^{IFPL} \lesssim \mathbf{E} \hat{L}^{best}]$ .  $\dots \sum_i e^{-k^i} \leq 1 \dots \tau^i \dots k^i \geq k^j \dots T \geq 1 \dots i \geq 1$

$$\sum_{t=1}^T \mathbf{E}_t \hat{\ell}_t^{IFPL} \leq \hat{\ell}_{1:T}^i + \frac{k^i + 1}{\eta_T}.$$

This is a modification of the corresponding proofs in [3] and [5]. We may fix the randomization  $\mathcal{A}$  and suppress it in the notation. Then we only need to show

$$\mathbf{E} \hat{\ell}_{1:T}^{IFPL} \leq \min_{i \geq 1} \{ \hat{\ell}_{1:T}^i + \frac{k^i + 1}{\eta_T} \}, \tag{5}$$

where the expectation is with respect to  $\hat{\ell}_t$ ’s randomness  $q_{1:T}$ .

Assume first that the adversary is oblivious. We define an algorithm  $A$  as a variant of  $\hat{\ell}_t^{IFPL}$  which samples only one perturbation vector  $q$  in the beginning and uses this in each time step, i.e.  $q_t \equiv q$ . Since the adversary is oblivious,  $A$  is equivalent to  $\hat{\ell}_t^{IFPL}$  in terms of expected performance. This is all we need to show (5). Let  $\eta_0 = \infty$  and  $\lambda_t = \hat{\ell}_t + (k - q)(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}})$ , then  $\lambda_{1:t} = \hat{\ell}_{1:t} + \frac{k-q}{\eta_t}$ . Recall  $\{t \geq \tau\} = \{i : t \geq \tau^i\}$ . We argue by induction that for all  $T \geq 1$ ,

$$\sum_{t=1}^T \lambda_t^A \leq \min_{T \geq \tau} \lambda_{1:T}^i + \max_{T \geq \tau} \left\{ \frac{q^i - k^i}{\eta_T} \right\}. \tag{6}$$

This clearly holds for  $T = 0$ . For the induction step, we have to show

$$\begin{aligned} \min_{T \geq \tau} \lambda_{1:T}^i + \max_{T \geq \tau} \left\{ \frac{q^i - k^i}{\eta_T} \right\} + \lambda_{T+1}^A &\leq \lambda_{1:T}^{I_{T+1}^A} + \max_{T+1 \geq \tau} \left\{ \frac{q^i - k^i}{\eta_{T+1}} \right\} + \lambda_{T+1}^{I_{T+1}^A} \quad (7) \\ &= \min_{T+1 \geq \tau} \lambda_{1:T+1}^i + \max_{T+1 \geq \tau} \left\{ \frac{q^i - k^i}{\eta_{T+1}} \right\}. \end{aligned}$$

The inequality is obvious if  $I_{T+1}^A \in \{T \geq \tau\}$ . Otherwise, let  $J = \arg \max \{q^i - k^i : i \in \{T \geq \tau\}\}$ . Then

$$\begin{aligned} \min_{T \geq \tau} \lambda_{1:T}^i + \max_{T \geq \tau} \left\{ \frac{q^i - k^i}{\eta_T} \right\} &\leq \lambda_{1:T}^J + \frac{q^J - k^J}{\eta_T} = \sum_{t=1}^T \hat{\ell}_t^J \leq \sum_{t=1}^T \hat{B}_t \\ &= \sum_{t=1}^T \hat{\ell}_t^{I_{T+1}^A} \leq \lambda_{1:T}^{I_{T+1}^A} + \max_{T+1 \geq \tau} \left\{ \frac{q^i - k^i}{\eta_{T+1}} \right\} \end{aligned}$$

shows (7). Rearranging terms in (6), we see

$$\sum_{t=1}^T \hat{\ell}_t^A \leq \min_{T \geq \tau} \lambda_{1:T}^i + \max_{T \geq \tau^i} \left\{ \frac{q^i - k^i}{\eta_T} \right\} + \sum_{t=1}^T (q - k)^{I_t^A} \left( \frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right)$$

The assertion (5) – still for oblivious adversary and  $q_t \equiv q$  – then follows by taking expectations and using

$$\mathbf{E} \min_{T \geq \tau} \lambda_{1:T}^i \leq \min_{T \geq \tau} \{ \hat{\ell}_{1:T}^i + \frac{k^i}{\eta_T} - \mathbf{E} \frac{q^i}{\eta_T} \} \leq \min_{i \geq 1} \{ \hat{\ell}_{1:T}^i + \frac{k^i - 1}{\eta_T} \} \text{ and} \quad (8)$$

$$\mathbf{E} \sum_{t=1}^T (q - k)^{I_t^A} \left( \frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) \leq \mathbf{E} \max_{T \geq \tau} \left\{ \frac{q^i - k^i}{\eta_T} \right\} \leq \frac{1}{\eta_T}. \quad (9)$$

The second inequality of (8) holds because  $\tau^i$  depends monotonically on  $k^i$ , and  $\mathbf{E} q^i = 1$ , and maximality of  $\hat{\ell}_{1:T}^i$  for  $T < \tau_i$ . The second inequality of (9) can be proven by a simple application of the union bound, see e.g. [5–Lem.1].

Sampling the perturbations  $q_t$  independently is equivalent under expectation to sampling  $q$  only once. So assume that  $q_t$  are sampled independently, i.e. that  $\dots$  is played against an oblivious adversary: (5) remains valid. In the last step, we argue that then (5) also holds for an  $\dots$  adversary. This is true because the future actions of  $\dots$  do not depend on its past actions, and therefore the adversary cannot gain from deciding after having seen  $\dots$ 's decisions. This argument can be made formal, as shown in [11–Lemma 12]. (Note the subtlety that the future actions of  $\dots$  would depend on its past actions.)  $\square$

Finally, we give a relation between the estimated and true losses (adapted from [4]).

**Lemma 6.**  $[\mathbf{E} \hat{L}^{best} \lesssim L^{best}] \dots T \geq 1 \quad \delta_T \in (0, 1) \dots i \geq 1 \dots$

- (i)  $\hat{\ell}_{1:T}^i \leq \ell_{1:T}^i + \sqrt{(2 \ln \frac{4}{\delta_T}) \sum_{t=1}^T \hat{B}_t^2} + \sum_{t=1}^{\tau^i - 1} \hat{B}_t \dots 1 - \frac{\delta_T}{2} \dots$
- (ii)  $\mathbf{E} \hat{\ell}_{1:T}^i \leq \ell_{1:T}^i + \sqrt{(2 \ln \frac{4}{\delta_T}) \sum_{t=1}^T \hat{B}_t^2} + \frac{\delta_T}{2} \sum_{t=1}^T \hat{B}_t + \sum_{t=1}^{\tau^i - 1} \hat{B}_t.$

For  $t \geq \tau^i$ ,  $X_t = \hat{\ell}_{1:t}^i - \ell_{1:t}^i$  is a martingale, since

$$\mathbf{E}[X_t | \mathcal{A}_{t-1}] = \mathbf{E}[\hat{\ell}_{1:t}^i | \mathcal{A}_{t-1}] - \ell_{1:t}^i = X_{t-1} + \mathbf{E}[\hat{\ell}_t^i | \mathcal{A}_{t-1}] - \ell_t^i = X_{t-1}.$$

It is clear that  $X_{\tau^i-1} \leq \sum_{t=1}^{\tau^i-1} \hat{B}_t$ . Moreover,  $|X_t - X_{t-1}| \leq \hat{B}_t$  for  $t \geq \tau^i$ , i.e. we have bounded differences. By Azuma’s inequality, the actual value  $X_T - X_{\tau^i-1}$  does not exceed  $\sqrt{(2 \ln \frac{4}{\delta_T}) \sum_{t=1}^T \hat{B}_t^2}$  with probability  $1 - \frac{\delta_T}{2}$ . This proves (i). To arrive at (ii), take expectations and observe that (i) fails with probability at most  $\frac{\delta_T}{2}$ , in which case  $\hat{\ell}_{1:T}^i \leq \sum_{t=1}^T \hat{B}_t$  holds.  $\square$

We now combine the above results and derive an upper bound on the expected regret of  $\gamma$  against an adaptive adversary.

**Theorem 7.** [ $\gamma$  against an adaptive adversary]  $\sum_i e^{-k^i} \leq 1$ ,  $\eta_t \leq \ell_t$ ,  $\|\ell_t\|_\infty \leq B_t$ ,  $\gamma_t \leq 1 - \delta_T$

$$\ell_{1:T}^{FoE} \leq \ell_{1:T}^i + \frac{k^i+1}{\eta_T} + \sum_{t=1}^{\tau^i-1} \hat{B}_t + \sum_{t=1}^T \gamma_t \eta_t \hat{B}_t^2 + \sum_{t=1}^T \gamma_t B_t + \sqrt{(2 \ln \frac{4}{\delta_T}) \left( \sqrt{\sum_{t=1}^T \hat{B}_t} + \sqrt{\sum_{t=1}^T B_t^2} \right)}.$$

$$\mathbf{E} \ell_{1:T}^{FoE} \leq \ell_{1:T}^i + \frac{k^i+1}{\eta_T} + \sum_{t=1}^{\tau^i-1} \hat{B}_t + \sum_{t=1}^T \gamma_t \eta_t \hat{B}_t^2 + \sum_{t=1}^T \gamma_t B_t + \sqrt{(2 \ln \frac{4}{\delta_T}) \sum_{t=1}^T \hat{B}_t + \frac{\delta_T}{2} \sum_{t=1}^T \hat{B}_t}.$$

This follows by summing up all excess terms in the above lemmas. Recall that we only need to take expectations on both sides of the assertions of Lemmas 2–5 in order to obtain the second bound on the expectation (and we don’t need Lemma 1 there).  $\square$

**Corollary 8.**  $\eta_t = t^{-\frac{3}{4}}$ ,  $\gamma_t = t^{-\frac{1}{4}}$

- (i)  $B_t \equiv 1, \tau^i = \lceil (w^i)^{-8} \rceil \Rightarrow \mathbf{E} \ell_{1:T}^{FoE} \leq \ell_{1:T}^i + O\left(\left(\frac{1}{w^i}\right)^{11} + k^i T^{\frac{7}{8}} \sqrt{\ln T}\right)$ ,
- (ii)  $B_t \equiv 1, \tau^i = \lceil (w^i)^{-8} \rceil \Rightarrow \ell_{1:T}^{FoE} \leq \ell_{1:T}^i + O\left(\left(\frac{1}{w^i}\right)^{11} + k^i T^{\frac{7}{8}} \sqrt{\ln T}\right)$ ,  $1 - \frac{1}{T^2}$ ,
- (iii)  $B_t = t^{\frac{1}{16}}, \tau^i = \lceil (w^i)^{-16} \rceil \Rightarrow \mathbf{E} \ell_{1:T}^{FoE} \leq \ell_{1:T}^i + O\left(\left(\frac{1}{w^i}\right)^{22} + k^i T^{\frac{7}{8}} \sqrt{\ln T}\right)$ ,
- (iv)  $B_t = t^{\frac{1}{16}}, \tau^i = \lceil (w^i)^{-16} \rceil \Rightarrow \ell_{1:T}^{FoE} \leq \ell_{1:T}^i + O\left(\left(\frac{1}{w^i}\right)^{22} + k^i T^{\frac{7}{8}} \sqrt{\ln T}\right)$ ,  $1 - \frac{1}{T^2}$ ,

$i \rightarrow \infty, T \geq 1, k^i = -\ln w^i, \hat{B}_t \rightarrow \dots$

$$\limsup_{T \rightarrow \infty} \frac{\ell_{1:T}^{FoE} - \ell_{1:T}^i}{T} \leq 0$$



The asymptotic optimality is sometimes termed *asymptotic optimality*, in particular if the limit equals zero. We only show the upper bound.

Assertions (i)-(iv) follow from the previous theorem: Set  $\delta_T = T^{-2}$ , abbreviate  $w_T^{\min} = \min\{w^i : t \geq \tau^i\}$ , and observe that for  $\tau^i = \lceil (w^i)^{-\alpha} \rceil$  and  $B_t = t^\beta$ , we have

$$w_T^{\min} = \min\{w^i : T \geq \lceil (w^i)^{-\alpha} \rceil\} \geq \min\{w^i : T^{-\frac{1}{\alpha}} \leq w^i\} \geq T^{-\frac{1}{\alpha}} \quad \text{and}$$

$$\sum_{t=1}^{\tau^i-1} \hat{B}_t \leq (\tau^i - 1)\hat{B}_{\tau^i-1} \leq \frac{(w^i)^{-\alpha} B_{\tau^i-1}}{\gamma_{\tau^i-1} w_{\tau^i-1}^{\min}} \leq \frac{(w^i)^{-\alpha} (w^i)^{-\alpha\beta}}{(w^i)^{\frac{\alpha}{4}} w^i}$$

(note  $w_{\tau^i-1}^{\min} \geq (\tau^i - 1)^{-\frac{1}{\alpha}} \geq (w^i)^{(-\alpha)(-\frac{1}{\alpha})}$ ). Then (i) and (ii) follow from  $\alpha = 8$ ,  $\beta = 0$ , and (iii) and (iv) follow from  $\alpha = 16$ ,  $\beta = \frac{1}{16}$ . The asymptotic optimality finally follows from the Borel-Cantelli Lemma, since according to (ii) and (iv),

$$\mathbf{P} \left[ \frac{\ell_{1:T}^{FoE} - \min_i \ell_{1:T}^i}{T} > CT^{-\frac{1}{8}} \sqrt{\ln T} \right] \leq \frac{1}{T^2}$$

for an appropriate  $C > 0$ . □

As mentioned in the first paragraph of this section, it is possible to avoid Lemma 6, thus arriving at better bounds. E.g. in (i), choosing  $\tau^i = \lceil (\frac{1}{w^i})^8 \rceil$ ,  $\gamma_t = t^{-\frac{1}{4}}$ , and  $\eta_t = t^{-\frac{3}{4}}$ , a regret bound of  $O((\frac{1}{w^i})^{11} + k^i T^{\frac{3}{4}})$  can be shown. Of course, also a corresponding high probability bound like (ii) holds. Likewise, for a similar statement as (iii), we may set  $\tau^i = \lceil (\frac{1}{w^i})^{16} \rceil$ ,  $B_t = t^{\frac{1}{8}}$ ,  $\gamma_t = t^{-\frac{1}{4}}$ , and  $\eta_t = t^{-\frac{3}{4}}$ , arriving at a regret bound of  $O((\frac{1}{w^i})^{23} + k^i T^{\frac{3}{4}})$ . Generally, in this way any regret bound  $O((\frac{1}{w^i})^c + k^i T^{\frac{2}{3}+\epsilon})$  is possible, at the cost of increasing  $c$  where  $\epsilon \rightarrow 0$ .

## 4 Reactive Environments and a Universal Master Algorithm

can become a quite subtle notion if we start considering reactive environments, i.e. care for future consequences of a decision. An extreme case is the “heaven-hell” example: We have two experts, one always playing 0 (“saying a prayer”), the other one always playing 1 (“cursing”). If we always follow the first expert, we stay in heaven and get no loss in each step. As soon as we “curse” only once, we get into hell and receive maximum loss in all subsequent time steps. Clearly, algorithm without prior knowledge must “fail” in this situation.

One way to get around this problem is taking into account the (realization of the) game we are playing. For instance, after “cursing” once, also the praying expert goes to hell together with us and subsequently has maximum loss. Hence, we are interested in a regret defined as  $\mathbf{E}\ell_{1:T} - \ell_{1:T}^i$  as in the previous section. So what is missing? This becomes clear in the following example.

Consider the repeated “prisoner’s dilemma” against the tit-for-tat<sup>1</sup> strategy [1]. If we use two strategies as experts, namely “always cooperate” and “always defect”, then it is clear that always cooperating will have the best long-term reward. However standard expert advice or bandit master algorithm will not discover this, since it compares only the losses in one step, which are always lower for the defecting expert. To put it differently, minimizing short-term regret is not at all a good idea here. E.g. always defecting has no regret, while for always cooperating the regret grows  $\log t$ . But this is only the case for short-term regret, i.e. if we restrict to time intervals of length one.

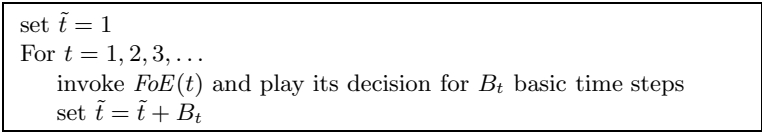


Fig. 4. The algorithm  $\widetilde{FoE}$ , where  $B_t$  is specified in Corollary 9

We therefore give the control to a selected expert for  $B_t$  basic time steps. Precisely, we introduce a new time scale  $\tilde{t}$  (the  $\tilde{t}$ -time scale) at which we have single games with losses  $\tilde{\ell}_{\tilde{t}}$ . The master’s time scale  $t$  does not coincide with  $\tilde{t}$ . Instead, at each  $t$ , the master gives control to the selected expert  $i$  for  $B_t \geq 1$  single games and receives loss  $\ell_t^i = \sum_{\tilde{t}=\tilde{t}(t)}^{\tilde{t}(t)+B_t-1} \tilde{\ell}_{\tilde{t}}^i$ . (The points  $\tilde{t}(t)$  in basic time are defined recursively, see Fig. 4.) Assume that the game has bounded instantaneous losses  $\tilde{\ell}_{\tilde{t}}^i \in [0, 1]$ . Then the master algorithm’s instantaneous losses are bounded by  $B_t$ . We denote the algorithm, which is completely specified in Fig. 4, by  $\widetilde{FoE}$ . Then the following assertion is an easy consequence of the previous results.

**Corollary 9.** Let  $\tilde{\ell}_{\tilde{t}}^i \in [0, 1]$ ,  $\gamma_t = t^{-\frac{1}{4}}$ ,  $\eta_t = t^{-\frac{3}{4}}$ ,  $B_t = \lceil t^{\frac{1}{16}} \rceil$ ,  $\tau^i = \lceil (w^i)^{-16} \rceil$ ,  $\tilde{T} \geq 1$

$$\begin{aligned} \tilde{\ell}_{1:\tilde{T}}^{\widetilde{FoE}} &\leq \tilde{\ell}_{1:\tilde{T}}^i + O\left(\left(\frac{1}{w^i}\right)^{22} + k^i \tilde{T}^{\frac{9}{10}}\right), \quad 1 - \tilde{T}^{-\frac{32}{17}} \\ \mathbf{E} \tilde{\ell}_{1:\tilde{T}}^{\widetilde{FoE}} &\leq \tilde{\ell}_{1:\tilde{T}}^i + O\left(\left(\frac{1}{w^i}\right)^{22} + k^i \tilde{T}^{\frac{9}{10}}\right). \end{aligned}$$

$$\limsup_{T \rightarrow \infty} \frac{\tilde{\ell}_{1:\tilde{T}}^{\widetilde{FoE}} - \tilde{\ell}_{1:\tilde{T}}^i}{\tilde{T}} \leq 0 \quad \text{with probability } 1 - \frac{1}{w^i}$$

<sup>1</sup> In the prisoner’s dilemma, two players both decide independently if they are *cooperating* ( $C$ ) or *defecting* ( $D$ ). If both play  $C$ , they get both a small loss, if both play  $D$ , they get a large loss. However, if one plays  $C$  and one  $D$ , the cooperating player gets a very large loss and the defecting player no loss at all. Thus defecting is a *dominant* strategy. Tit-for-tat plays  $C$  in the first move and afterwards the opponent’s respective preceding move.

This follows from changing the time scale from  $t$  to  $\tilde{t}$  in Corollary 8:  $\tilde{t}$  is of order  $t^{1+\frac{1}{10}}$ . Consequently, the regret bound is  $O((\frac{1}{w^i})^{22} + k^i \tilde{T}^{\frac{15}{17}} \sqrt{\ln \tilde{T}}) \leq O((\frac{1}{w^i})^{22} + k^i \tilde{T}^{\frac{9}{10}})$ .  $\square$

Broadly spoken, this means that  $\tilde{\gamma}$  performs asymptotically as well as the best expert. Asymptotic performance guarantees for the Strategic Experts Algorithm have been derived in [1]. Our results improve upon this by providing a rate of convergence. One can give further corollaries, e.g. in terms of flexibility as defined in [1].

Since we can handle countably infinite expert classes, we may specify a universal experts algorithm. To this aim, let expert  $i$  be derived from the  $i$ th (valid) program  $p^i$  of some fixed universal Turing machine. The  $i$ th program can be well-defined, e.g. by representing programs as binary strings and lexicographically ordering them [6]. Before the expert is consulted, the relevant input is written to the input tape of the corresponding program. If the program halts, an appropriate part of the output is interpreted as the expert's recommendation. E.g. if the decision is binary, then the first bit suffices. (If the program does not halt, we may for well-definedness just fill its output tape with zeros.) Each expert is assigned a prior weight by  $w^i = 2^{-\text{length}(p^i)}$ , where  $\text{length}(p^i)$  is the length of the corresponding program and we assume the program tape to be binary. This construction parallels the definition of Solomonoff's universal prior [12].

**Corollary 10.**

Let  $\eta_t, \gamma_t, B_t, \delta_T$  be any computable expert  $i$  with  $k^i \leq t^{-\frac{1}{3}+\epsilon}$ .

The universal prior has been used to define a universal agent AIXI in a quite different way [13, 6]. Note that like the universal prior and the AIXI agent, our universal experts algorithm is not computable, since we cannot check if a the computation of an expert halts. On the other hand, if used with computable experts, the algorithm is computationally feasible (at each time  $t$  we need to consider only finitely many experts). Moreover, it is easy to impose an additional constraint on the computation time of each expert and abort the expert's computation after  $C_t$  operations on the Turing machine. We may choose some (possibly rapidly) growing function  $C_t$ , e.g.  $C_t = 2^t$ . The resulting master algorithm is fully computable and has small regret with respect to all resource bounded strategies.

It is important to keep in mind that Corollaries 9 and 10 give assertions relative to the experts' performance merely on the action-observation sequence. In other words, if we wish to assess how well  $\tilde{\gamma}$  does, we have to evaluate the actual performance of the best expert [14]. Note that the whole point of our increasing time construction is to cause this actual value to coincide with the value under  $\tilde{\gamma}$  conditions. For passive tasks, this coincidence always holds with any experts algorithm. With  $\tilde{\gamma}$ , the actual and the ideal value of an

expert coincide in many further situations, such as “finitely controllable tasks”. By this we mean cases where the best expert can drive the environment into some optimal state in a fixed finite number of time steps. An instance is the prisoner’s dilemma with tit-for-tat being the opponent. The following is an example for a formalization of this statement.

**Proposition 11.**

This statement may be generalized to cases where only a close to optimal state sequence is reached with high probability. However, we need assumptions on the closeness to optimality for a given target probability, which are compatible with the sampling behavior of  $\tau_{\gamma}$ .

Not all environments have this or similar nice properties. As mentioned above, any version of  $\tau_{\gamma}$  would not perform well in the “heaven-hell” example. The following is a slightly more interesting variant of the heaven-hell task, where we might wish to learn optimal behavior, however  $\tau_{\gamma}$  will not. Consider the heaven-hell example from the beginning of this section, but assume that if at time  $t$  I am in hell and I “pray” for  $t$  consecutive time steps, I will get back into heaven. Then it is not hard to see that  $\tau_{\gamma}$ ’s exploration is so dominant that almost surely,  $\tau_{\gamma}$  will eventually stay in hell.

Simulations with some  $2 \times 2$  matrix games show similar effects, depending on the opponent. We briefly discuss the repeated game of “chicken”<sup>2</sup>. In this game, it is desirable for the learner to become the “dominant defector”, i.e. to defect in the majority of the cases while the opponent cooperates. Let’s call an opponent “primitive” if he agrees to cooperate after a fixed number of consecutive defecting moves of  $\tau_{\gamma}$ , and let’s call him “stubborn” if this number is high. Then  $\tau_{\gamma}$  learns to be the dominant defector against any primitive opponent, however stubborn. On the other hand, if the opponent is some learning strategy which also tries to be the dominant defector and learns faster (we conducted the experiment with AIXI [6]), then  $\tau_{\gamma}$  settles for cooperating, and the opponent will be the dominant defector. Interestingly however, AIXI would not learn to defect against a stubborn primitive opponent. Under this point of view, it seems questionable that there is something like a universally optimal balance of exploration vs. exploitation in active learning at all.

<sup>2</sup> This game, also known as “Hawk and Dove”, can be interpreted as follows. Two coauthors write a paper, but each tries to spend as little effort as possible. If one succeeds to let the other do the whole work, he has a high reward. On the other hand, if no one does anything, there will be no paper and thus no reward. Finally, if both decide to cooperate, both get some reward. We choose the loss matrix as  $\begin{pmatrix} 1 & 0 \\ 0.8 & 0.5 \end{pmatrix}$ , the learner is the column player, the opponent’s loss matrix is the transpose, choosing the first column means to defect, the second to cooperate. Hence, in the repeated game, it is socially optimal to take turns cooperating and defecting.

## 5 Discussion

**An Alternative Argument for Adaptive Adversary.** As mentioned in the beginning of Section 3, the analysis we gave uses a trick from [4]. Such a trick seems necessary, as the basic FPL analysis only works for oblivious adversary. The simple argument from [11] which we used in the last paragraph of the proof of Lemma 5 works only for full observation games (note that considering the estimated losses, we were actually dealing with full observations there). In order to obtain a similar result in the partial observation case, we may argue as follows. We let the game proceed for  $T$  time steps with independent randomization against an adaptive adversary. Then we analyze  $\mathcal{A}_T$ 's performance,  $\mathbb{E}[\sum_{t=1}^T \ell_{i_t} - \min_{i \in \mathcal{I}} \sum_{t=1}^T \ell_i]$ . In particular, we note that for the losses assigned by the adversary,  $\mathcal{A}_T$ 's expected regret coincides with the regret of another, virtual algorithm, which uses (in its FPL subroutine) identical perturbations  $q_t \equiv q$ . Performing the analysis for this virtual algorithm, we arrive at the desired assertion, however without needing Lemma 6. This results in tighter bounds as stated above. The argument is formally elaborated in [8].

**Actual Learning Speed and Lower Bounds.** In practice, the bounds we have proven seem irrelevant except for small expert classes, although asserting almost sure optimality and even a convergence rate. The exponential of the complexity  $\frac{1}{w^t}$  may be huge. Imagine for instance a moderately complex task and some good strategy, which can be coded with mere 500 bits. Then its prior weight is  $2^{-500}$ , a constant which is not distinguishable from zero in all practical situations. Thus, it seems that the bounds can be relevant at most for small expert classes with uniform prior. This is a general shortcoming of bandit style experts algorithms: For uniform prior a lower bound on the expected loss which scales with  $\sqrt{n}$  (where  $n$  is the size of the expert class) has been proven [10].

In order to get a lower bound on  $\mathcal{A}_T$ 's regret in the time  $T$ , observe that  $\mathcal{A}_T$  is a  $\gamma$ - $\epsilon$ -learner [15, 16]: According to the definition in [16], we may assume that in each exploration step, we incur maximal loss  $B_t$ . It is immediate that the same analysis then still holds. For label-efficient prediction, Cesa-Bianchi et al. [16] have shown a lower regret bound of  $O(T^{\frac{2}{3}})$ . Since according to the remark at the end of Section 3, we have an upper bound of  $O((\frac{1}{w^t})^c + k^j T^{\frac{2}{3} + \epsilon})$ , this is almost tight except for the additive  $(\frac{1}{w^t})^c$  term. It is an open problem to state a lower bound simultaneously tight in both  $\frac{1}{w^t}$  and  $T$ .

Even if the bounds, in particular  $\frac{1}{w^t}$ , seem not practical, maybe  $\mathcal{A}_T$  would learn sufficiently quickly in practice anyway? We believe that this is not so in most cases: The design of  $\mathcal{A}_T$  is too much tailored towards worst-case environments,  $\mathcal{A}_T$  is too  $\epsilon$ -greedy. Assume that we have a "good" and a "bad" expert, and  $\mathcal{A}_T$  learns this fact after some time. Then it still would spend a relatively huge fraction of  $\gamma_t = t^{-\frac{1}{4}}$  to exploring the bad expert. Such defensive behavior seems only acceptable if we are already starting with a class of good experts.

**Acknowledgment.** This work was supported by SNF grant 2100-67712.02 and JSPS 21st century COE program C01.

## References

- [1] de Farias, D.P., Megiddo, N.: How to combine expert (and novice) advice when actions impact the environment? In: *Advances in Neural Information Processing Systems (NIPS) 16*. MIT Press, Cambridge, MA (2004)
- [2] Hannan, J.: Approximation to Bayes risk in repeated plays. In Dresher, M., Tucker, A.W., Wolfe, P., eds.: *Contributions to the Theory of Games 3*. Princeton University Press (1957) 97–139
- [3] Kalai, A., Vempala, S.: Efficient algorithms for online decision. In: *Proc. 16th Annual Conference on Learning Theory (COLT)*. Springer (2003) 506–521
- [4] McMahan, H.B., Blum, A.: Online geometric optimization in the bandit setting against an adaptive adversary. In: *17th Annual Conference on Learning Theory (COLT)*. Volume 3120 of *Lecture Notes in Computer Science.*, Springer (2004) 109–123
- [5] Hutter, M., Poland, J.: Prediction with expert advice by following the perturbed leader for general weights. In: *International Conference on Algorithmic Learning Theory (ALT)*. (2004) 279–293
- [6] Hutter, M.: *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin (2004)
- [7] Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: Gambling in a rigged casino: The adversarial multi-armed bandit problem. In: *Proc. 36th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE (1995) 322–331
- [8] Poland, J.: FPL analysis for adaptive bandits. *3rd Symposium on Stochastic Algorithms, Foundations and Applications (SAGA)*, to appear (2005)
- [9] Motwani, R., Raghavan, P.: *Randomized Algorithms*. Cambridge University Press, Cambridge, England (1995)
- [10] Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multi-armed bandit problem. *SIAM Journal on Computing* **32** (2002) 48–77
- [11] Hutter, M., Poland, J.: Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research* **6** (2005) 639–660
- [12] Solomonoff, R.J.: Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Inform. Theory* **24** (1978) 422–432
- [13] Hutter, M.: Towards a universal theory of artificial intelligence based on algorithmic probability and sequential decisions. *Proc. 12<sup>th</sup> European Conference on Machine Learning (ECML-2001)* (2001) 226–238
- [14] de Farias, D.P., Megiddo, N.: Exploration-exploitation tradeoffs for experts algorithms in reactive environments. In: *Advances in Neural Information Processing Systems 17*. (2005)
- [15] Cesa-Bianchi, N., Lugosi, G., Stoltz, G.: Minimizing regret with label efficient prediction. In: *17th Annual Conference on Learning Theory (COLT)*. Volume 3120 of *Lecture Notes in Computer Science.*, Springer (2004) 77–92
- [16] Cesa-Bianchi, N., Lugosi, G., Stoltz, G.: Regret minimization under partial monitoring. Technical report (2004)

# On Following the Perturbed Leader in the Bandit Setting

Jussi Kujala and Tapio Elomaa

Institute of Software Systems,  
Tampere University of Technology,  
P.O. Box 553, FI-33101 Tampere, Finland  
jussi.kujala@tut.fi, elomaa@cs.tut.fi

**Abstract.** In an online decision problem an algorithm is at each time step required to choose one of the feasible points without knowing the cost associated with it. An adversary assigns the cost to possible decisions either obliviously or adaptively. The online algorithm, naturally, attempts to collect as little cost as possible. The cost difference of the online algorithm and the best static decision in hindsight is called the *regret* of the algorithm.

Kalai and Vempala [1] showed that it is possible to have efficient solutions to some problems with a linear cost function by following the perturbed leader. Their solution requires the costs of all decisions to be known. Recently there has also been some progress in the *bandit setting*, where only the cost of the selected decision is observed. A bound of  $O(T^{2/3})$  on  $T$  rounds was first shown by Awerbuch and Kleinberg [2] for the regret against an oblivious adversary and later McMahan and Blum [3] showed that a bound of  $O(\sqrt{\ln T} T^{3/4})$  is obtainable against an adaptive adversary.

In this paper we study Kalai and Vempala's model from the viewpoint of bandit algorithms. We show that the algorithm of McMahan and Blum attains a regret of  $O(T^{2/3})$  against an oblivious adversary. Moreover, we show a tighter  $O(\sqrt{m \ln m} \sqrt{T})$  bound for the expert setting using  $m$  experts.

## 1 Introduction

An  $\mathcal{A}$  performs a sequence of tasks without any information on the future conditions, while an  $\mathcal{B}$  has the full information on them at its disposal. Online algorithms are often analyzed against an adversary, which gets to pick the next task (with the intention to hurt the algorithm as much as possible) after observing the preceding actions of the algorithm. Moreover, their performance is compared to that of offline algorithms. There are also differences in how much information does the online algorithm receive. Typically a cost is assigned to the decisions taken by the algorithm.

One method of ranking online algorithms is to compare their cost to the best static object, i.e., to a situation where the offline algorithm consistently chooses

a single static output. Of course, the applicability of this model depends on the specific problem. For example, a binary search tree algorithm could compare its cost to that of the best static tree for the access sequence [4]. It can be argued that a static distribution models well some real world situations.

Kalai and Vempala [1] showed that many online decision problems have efficient solutions, given an oracle for the offline version of the problem. The problems they consider are those where the decisions can be thought of as vectors  $x \in \mathbb{R}^n$  and the cost is a linear function on decision vectors. We will adopt their model and notation, which are as follows.

Throughout this article we denote the dimension of the decision and cost vector space by  $n$ . At each time step  $t \in \{1, \dots, T\}$  a decision vector  $x_t$  is chosen from a set  $S_D \subset \mathbb{R}^n$ . Then a cost vector  $c_t$  from a set  $S_C \subset \mathbb{R}^n$  is revealed, and the algorithm suffers a cost  $c_t \cdot x_t$ . The cost is an Euclidean dot product of the cost vector and the decision vector. Thus, the cost function captures redundancy in problems where the costs for different decisions can be linearly related.

Kalai and Vempala [1] (re)introduced the *Follow the Perturbed Leader* (FPL) algorithm, which selects a decision based on past cost vectors. Originally this method was proposed by Hannan [5] already in 1957. In short, the FPL selects the best decision for the past cost vectors which are perturbed by a random vector (hence the name). Note that if the set of decisions is discrete then a randomized algorithm must be used, because a deterministic algorithm can always be misguided. For example, if there are two experts  $A$  and  $B$  that incur a cost in sequential turns  $A, B, A, B, A, B, \dots$  then choosing the expert with the lower cost results in a high regret. In a convex decision space deterministic algorithms can be competitive. In deed, the algorithm of Zinkevich [6] deterministically solves a convex optimization problem, where both the decision space and cost function are convex.

A substantial amount of research has been devoted to *Follow the Leader* algorithms [7], where no assumption on the linear nature of the cost function is made. One can rather assume the experts (and their costs) to be independent of each other. In the expert setting one needs to have an explicit representation for the decision vector space  $S_D$ , while in online decision problems one only requires access to an efficient oracle for choosing the best decision in hindsight. For example, the *Follow the Perturbed Leader* algorithm of Littlestone and Warmuth [8] can be used to solve the same problems as FPL if the set of decision vectors is finite.

However, a more specialized algorithm has advantages for example in the *Shortest Online Routing* problem. In this problem there is a directed acyclic graph with a start node and an end node. At each turn an algorithm selects a route from the start node to the end node based on past observations of the costs on edges. The weighted majority algorithm may be applied to the shortest online routing by considering each route as an individual expert. However, in the worst case the number of routes is exponential in the number of edges and, thus, the computational complexity becomes intractable. Furthermore, one can expect that the internal structure of the shortest online routing results in a lower regret, because different routes may share edges and their costs are related. In deed, the



weighted majority algorithm can be applied in an efficient manner to a problem that is very similar to the shortest online routing [9].

The shortest online routing problem can be solved by the FPL algorithm by considering each edge as an element in a vector. An element is either zero or one depending on whether the edge was used on the route or not. Thus, the cost vector consists of the edge weights and only these need to be saved for future use, in contrast to the weighted majority algorithm, which stores a weight for each route.

Let us introduce some further notation. There is an oracle  $M: S_C \rightarrow S_D$  such that  $M(c)$  is the optimal decision for a cost vector  $c$ . We abbreviate the sum of costs up to time step  $t$  as  $c_{1:t} = \sum_{i=1}^t c_i$ . Hence, if  $T$  is the total number of rounds, then  $M(c_{1:T})$  is the best static decision and  $M(c_{1:T}) \cdot c_{1:T}$  is its cost. The regret of an algorithm is the cost difference between the best static decision  $M(c_{1:T})$  and the algorithm. The sets of decision and cost vectors have to be restricted, because otherwise it is possible to suffer an unbounded amount of regret during a single turn. There is a bound on the diameter of the decision set  $\|x_t - x'_t\|_1 \leq D$  for all  $x_t, x'_t \in S_D$ , the cost set  $\|c_t\|_1 \leq A$  for any  $c_t \in S_C$ , and the maximum cost  $|c_t \cdot x_t| \leq R$ .

With the above restrictions the FPL algorithm achieves a cost of

$$(1 + \epsilon)M(c_{1:T}) \cdot c_{1:T} + \frac{D \ln n}{\epsilon}$$

for a parameter  $\epsilon$ . Thus, the regret of the FPL algorithm is  $O(\sqrt{T})$  when a suitable  $\epsilon$  is plugged into the above formula. This means that the decisions selected by the FPL algorithm are in the long run almost as good as the best decision chosen consistently.

Recall that the FPL requires all past information to be known, i.e., all previous cost vectors. In a natural generalization of the problem the whole cost vector is not observed. For example, in the shortest online routing with end-to-end feedback [2] only the cost on the chosen route is given. Similar restrictions have been studied in the expert setting, where one expert is selected at each turn from a pool of experts and the cost  $0 \leq c \leq 1$  of the selected expert is suffered. In a restricted setting only the cost of the selected expert is observed. In this situation the algorithm of Auer et al. [10] obtains a regret of  $O(\sqrt{Tm \ln m})$  for  $m$  experts. Note that we use  $m$  to indicate the number of the experts (or decision vectors) and  $n$  to indicate the dimension of some particular problem as described before. We are especially interested in the case where  $n \ll m$ .

Awerbuch and Kleinberg [2] consider the online decision situation, where only the cost  $c_t \cdot x_t$  is observed after a decision  $x_t$ . Their algorithm uses the FPL algorithm as a black-box and estimates the cost vector from independent samples. This setting can be split into two separate problems, first how to estimate the cost vector, and second how to select and analyze the decision based on the estimates of the cost vectors. This situation is the  $(R, n, T)$ -bandit problem. The algorithm of Awerbuch and Kleinberg has a regret of the order  $O(Rn^{5/3} \ln^{1/3} n T^{2/3})$  against an  $(R, n, T)$ -adversary, which decides the cost vectors in advance without knowledge of the decisions made by the algorithm. McMahan and Blum [3] obtained

**Table 1.** Summary of earlier bounds for the regret in terms of the length of the sequence  $T$  and our results

Paper	Adversary Model	Setting	Regret Bound
Auer et al. [10]	Adaptive	Bandit, Expert	$O(\sqrt{Tm \ln m})$
Kalai and Vempala [1]	Adaptive	Transparent	$O(\sqrt{DRAT})$
Awerbuch and Kleinberg [2]	Oblivious	Bandit	$O(Rn^{5/3} \ln^{1/3} n T^{2/3})$
McMahan and Blum [3]	Adaptive	Bandit	$O(\sqrt{\ln T} T^{3/4})^a$
This paper	Oblivious	Bandit, Expert	$O(\sqrt{Tm \ln m})$
	Oblivious	Bandit	$O(Rn(DA \ln n)^{1/3} T^{2/3})$

<sup>a</sup> Polynomial dependence on parameters is not shown

a regret of  $O(\sqrt{\ln T} T^{3/4})$  for the bandit FPL against an  $\epsilon$ -adversary, which may use the past decisions  $x_1, \dots, x_{t-1}$  of the algorithm when choosing the cost vector  $c_t$ .

Flaxman, Kalai, and McMahan [11] analyze a bandit algorithm for the Zinkevich’s [6] convex optimization algorithm, proving a regret of  $O(T^{5/6})$  against an oblivious adversary. Hutter and Poland [12] study an  $\epsilon$ -adversary, i.e., a version of FPL in which the total number of rounds  $T$  is not known. Additionally, they show that many of the results obtained in the expert setting can be elegantly proved also in the FPL setting, albeit with worse constants.

Table 1 summarizes the relevant results from previous work and puts the results derived in this paper into the map.

In this paper we analyze one way of implementing a bandit algorithm for the FPL. Section 2 gives a general condition for unbiased estimates for the cost vector  $c_t$  when a decision  $x_t$  has been made and the cost  $c_t \cdot x_t$  has been observed. In Sections 3–4 we analyze this estimate. Our results apply against an oblivious adversary. In particular, Section 3 shows how to use the framework of Kalai and Vempala to derive an upper bound for the bandit FPL. In Section 3.1 we proceed to derive a  $O(\sqrt{Tm \ln m})$  regret in the expert setting. This is to the best of our knowledge the first time that a  $O(\sqrt{T})$  regret has been demonstrated in a bandit FPL. Finally, in Section 4 we show an upper bound  $O(T^{2/3})$  that holds in the general case. This upper bound can be applied to the algorithm analyzed by McMahan and Blum [3], thus improving their bound  $O(\sqrt{\ln T} T^{3/4})$  against an adaptive adversary to  $O(T^{2/3})$  against an oblivious adversary (which, of course, is not necessarily an improvement).

## 2 An Unbiased Estimate for a Cost Vector

Awerbuch and Kleinberg [2] as well as McMahan and Blum [3] mention one of problems of the bandit FPL to be a lack of a good way of estimating the cost vector. In this section we will study a one kind of unbiased estimate for the cost vector in a restricted setting. We consider the case where the number of decision vectors  $m$  is finite. Let  $x_1, \dots, x_m$  be the decision vectors of the problem and define a matrix  $X = [x_1, \dots, x_m]$  to contain the decision vectors as columns.

At time  $t$  we choose a decision  $x_t$  from a distribution given by probabilities  $p(x_t) > 0$  (recall that randomization is necessary) and then observe the cost  $c_t \cdot x_t$ . We limit ourselves to the situation in which the cost vector  $c_t$  is estimated independently at each turn with some unbiased  $\hat{c}_t$  and the cumulative estimate is simply the additive  $\hat{c}_{1:t}$ . Thus an estimate  $\hat{c}$  of  $c$  can be based only on the selected decision  $x_t$ , the probability distribution on decisions, and the cost  $c_t \cdot x_t$ . We assume that  $\hat{c} = (c \cdot x/p(x)) v_x$ , where  $v_x \in \mathbb{R}^n$  is the direction vector of the estimate. Let the directions  $v_x$  make up a matrix  $V = [v_1, \dots, v_m]$ . Then

$$\mathbb{E}(\hat{c}) = \sum_{x \in X} (c \cdot x) v_x = \sum_{x \in X} v_x x^T c = VX^T c. \tag{1}$$

This implies that ideally we should choose  $V$  so that  $VX^T = I_{n \times n}$ . However, the matrix  $X$  does not necessarily have a rank of  $n$ . Then

$$VX^T = P_{R(X^T)}, \tag{2}$$

where  $P_{R(X^T)}$  is a projection matrix to the column space of  $X$ , is a sufficient condition to obtain an unbiased  $\hat{c}$ . Such an estimate is unbiased in the sense that the part of the cost vector  $c$ , which is in the null space of the matrix  $X^T$ , does not matter when calculating the actual costs. Another possibility is to first make the matrix  $X$  full rank by projecting it to a suitable subspace [3]. Both approaches require  $R(X^T)$  to be known.

Kalai and Vempala’s [1] approach resulted in an efficient solution given  $M(c)$ . To have an efficient procedure for estimation, we need to assume that the probabilities  $p(x_t)$  can be obtained, and that an solution for the matrix  $V$  is found. Previous work [3, 2] has estimated the cost vector from independent samples on a certain subset of decision vectors, so the probabilities have been uniform, and hence there has not been a need for more complex estimation of the probabilities on decisions. Our estimates may be applied similarly, or in any other way as long as it is possible to obtain the values  $p(x_t)$ .

Choosing the matrix  $V$  is a separate problem from estimating the probabilities. We show one method for selecting a matrix  $V$  efficiently in the routing case. Although  $X \in R^{n \times m}$ , it is possible to calculate the matrix  $XX^T \in R^{n \times n}$  in time polynomial in  $n$ . This is because  $(XX^T)_{ij}$  is the number of routes that share both edges  $i$  and  $j$ . Now if  $X$  has a rank of  $n$  then  $V = (XX^T)^{-1}X$  is one possible solution to equation (2). In this case  $v_x = (XX^T)^{-1}x$ . On the other hand, if  $X$  is not full rank, we may select  $V = (XX^T)^+ X$ , where  $(XX^T)^+$  is the Moore-Penrose matrix inverse of the matrix  $XX^T$ , which can be calculated efficiently using the singular value decomposition. The Moore-Penrose matrix inverse [13, 14] for a  $n \times m$  matrix  $A$  is a unique matrix pseudoinverse  $A^+$  such that:

1.  $AA^+A = A$ ,
2.  $A^+AA^+ = A^+$ ,
3.  $(AA^+)^T = AA^+$ , and
4.  $(A^+A)^T = A^+A$ .

**Table 2.** The bandit FPL algorithm considered in this paper

Set  $v_x$  to be the column corresponding to the decision  $x$  in the matrix  $V$  satisfying a formula  $VX^T = P_{R(X^T)}$  (bounds will depend on the particular choice). For example,  $v_x = (XX^T)^+x$  or a choice based on barycentric spanner [2].

On time step  $t$

1. **Sampling step:** With probability  $\gamma$  obtain a decision independent of  $\hat{c}_{1:t-1}$  such that the probability of sampling a decision  $x_t$  is proportional to  $\|v_{x_t}\|_1$ .  
**Exploitation step:** With probability  $1 - \gamma$  the decision  $M(\hat{c}_{1:t-1} + r_t)$  is used.
2. Set

$$\hat{c}_t = \frac{c_t \cdot x_t}{p(x_t)} v_{x_t}$$

to be the estimate of the  $c_t$ , where  $p(x_t)$  is the probability of choosing the decision  $x_t$  (both sampling and exploitation steps could be taken into consideration).

It is well known that these conditions imply that  $AA^+$  is the projection matrix to the column space of a matrix  $A$ . Thus, because the column spaces of  $X$  and  $XX^T$  are the same, and  $XX^T$  is a symmetric matrix,

$$V^T X = (XX^T)^+ XX^T = XX^T (XX^T)^+ = P_{R(X^T)}$$

as was required. Actually,  $(XX^T)^{-1} = (XX^T)^+$  if  $X$  is full rank, so we would not have needed to handle separately the case when  $X$  is full rank.

In general, equation (2) gives us some flexibility on choosing the estimate  $\hat{c}_t$ . In the following chapters we give conditions when an estimate based on a matrix  $V$  results in a low regret.

### 3 An Upper Bound for a Bandit FPL Algorithm

Let us first describe the algorithm that we use. At time  $t$  do a  $\gamma$ -step with probability  $\gamma$  and an  $(1-\gamma)$ -step with probability  $1-\gamma$ . On a sampling step an independent estimation of the cost vector should be obtained, in case of independent experts this could be drawn from a uniform distribution. On an exploitation step choose a decision  $M(\hat{c}_{1:t-1} + r_t)$ , where  $r_t$  is a random perturbation vector. The probabilities for decisions must be such that the possible decisions span the column space of  $X$ , otherwise it is not possible to estimate an unbiased cost vector  $c_t$ . Table 2 summarizes the algorithm that will be analyzed in the following.

We derive an upper bound for the regret of the above algorithm against an oblivious adversary. The bound holds under expected value. Recall that  $R$  is the maximum cost. For the sampling steps the trivial  $\gamma RT$  bound is enough. For the exploitation steps we use the following two lemmas that Kalai and Vempala [1] proved.

**Lemma 1.**  $\dots, c_1, \dots, c_T, \dots, p_0 = 0, p_1, \dots, p_T, \dots$

$$\sum_{t=1}^T M(c_{1:t} + p_t) \cdot c_t \leq M(c_{1:T}) \cdot c_{1:T} + \sum_{t=1}^T (M(c_{1:T}) - M(c_{1:t} + p_t)) \cdot (p_t - p_{t-1}).$$

**Lemma 2.**  $\dots, A, \dots$   
 $\dots \epsilon/2 \exp(-\epsilon|x|)$

$$\sum_{t=1}^T M(c_{1:t-1} + p_t) \cdot c_t \leq \sum_{t=1}^T M(c_{1:t} + p_t) \cdot c_t + \epsilon RAT.$$

The intuitive picture is that Lemma 1 can be used to estimate how much the difference  $\hat{c}_{1:t} - c_{1:t}$  matters. Thus, by setting  $p_t$  to be  $\hat{c}_{1:t-1} - c_{1:t-1} + r_t$ , which is the estimation error plus the random perturbation, we obtain by Lemma 1 the following inequality.

$$\begin{aligned} & \sum_{t=1}^T M(\hat{c}_{1:t-1} + c_t + r_t) \cdot c_t \\ & \leq M(c_{1:T}) \cdot c_{1:T} \\ & \quad + \sum_{t=1}^T (M(c_{1:T}) - M(\hat{c}_{1:t-1} + c_t + r_t)) \cdot (\hat{c}_{t-1} - c_{t-1} + r_t - r_{t-1}). \end{aligned} \tag{3}$$

There are random variables  $r_t$  and  $\hat{c}_{1:t-1}$  in the equation. Note that  $\hat{c}_{1:t-1}$  depends on the variables  $r_1, \dots, r_{t-1}$ . As we are going to take expected value over the equation we can replace the variable  $r_t$  with a variable  $s_t$  that is similarly distributed. We do not replace the variables  $r_1, \dots, r_{t-1}$  within the variable  $\hat{c}_{1:t-1}$ . This is possible by linearity of expectation. We can now apply the trick used by Kalai and Vempala [1]: as expected value is linear, we can correlate the values  $s_t$  by setting  $s_t = s_1$  for all  $t$ , which eliminates some terms.

Now we proceed to the following chain of equations for the cost of the exploitation steps. Recall that  $D$  is the diameter of the decision space.

$$\begin{aligned} & \mathbb{E} \left( \sum_{t=1}^T M(\hat{c}_{1:t-1} + s_t) \cdot c_t \right) \\ & \leq \mathbb{E} \left( \sum_{t=1}^T M(\hat{c}_{1:t-1} + c_t + s_t) \cdot c_t \right) + \epsilon RAT \end{aligned} \tag{4}$$

$$\begin{aligned} & \leq M(c_{1:T}) \cdot c_{1:T} + \epsilon RAT \\ & \quad + \mathbb{E} \left( \sum_{t=1}^T (M(c_{1:T}) - M(\hat{c}_{1:t-1} + c_t + s_t)) \cdot (\hat{c}_{t-1} - c_{t-1} + s_t - s_{t-1}) \right) \end{aligned} \tag{5}$$

$$\begin{aligned} &\leq M(c_{1:T}) \cdot c_{1:T} + \frac{D(1 + \ln n)}{\epsilon} + \epsilon RAT \\ &\quad + \mathbb{E} \left( \sum_{t=1}^T (M(c_{1:T}) - M(\hat{c}_{1:t-1} + c_t + s_t)) \cdot (\hat{c}_{t-1} - c_{t-1}) \right). \end{aligned} \tag{6}$$

Equation (4) follows from Lemma 2 and equation (5) follows by inequality (3). Equation (6) again follows from the arguments of Kalai and Vempala [1]: as  $s_t = s_1$ , we only need to worry of the first term of the sum, which they upper bounded by  $\mathbb{E}(D\|s_1\|_\infty) \leq D(1 + \ln n)/\epsilon$ .

Now the problem has reduced to solving the expected value on the right-hand side of equation (6). As we are using the oblivious adversary model, the vectors  $c_t$  are not random variables and thus we simply calculate

$$\mathbb{E}(M(c_{1:T}) \cdot (\hat{c}_{t-1} - c_{t-1})) = M(c_{1:T}) \cdot (\mathbb{E}(\hat{c}_{t-1}) - c_{t-1}) = 0.$$

We are finally left with

$$\sum_{t=1}^T M(\hat{c}_{1:t-1} + c_t + s_t) \cdot (c_{t-1} - \hat{c}_{t-1}). \tag{7}$$

This term cannot be bounded elementarily, because both sides of the dot product are dependent random variables.

### 3.1 Restriction to the Experts Setting

The simplest application of the bandit FPL is the expert setting, where there are  $m$  independent experts. Let us examine this framework for a while. Similar techniques as applied here are subsequently used to prove the more general Theorem 2. The expert situation can be described in the general setting by selecting the possible decisions to be the standard basis, i.e., vectors  $e_i$  which form the identity matrix  $I$ . Naturally, the number of dimensions  $n$  equals the number of experts  $m$  in this case. Thus,  $\hat{c}_t$  now reduces to  $(c_t/p(x_t)) M(\hat{c}_{1:t-1} + s_t)$ , as can be expected if one is familiar with prior work [10].

**Theorem 1.** *Let  $c_t$  be a cost vector and  $s_t$  be a subgradient of  $M$  at  $c_t$ . Let  $p_t(i)$  be the probability that the  $i$ th expert is chosen at time  $t$  and let  $p_t(i | j)$  be the probability that it is chosen at time  $t$  given that the  $j$ th expert was chosen at time  $t - 1$ . Let  $c_t^i$  be the  $i$ th element of the cost vector  $c_t$ . Under expectation one term of the sum (7) simplifies in the expert setting to*

$$\mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot (c_{t-1} - \hat{c}_{t-1})) \leq \epsilon (2(\|c_{t-1}\|_1 + 1) + \|c_{t-1}\|_2^2).$$

Let  $p_t(i)$  be the probability that the  $i$ th expert is chosen at time  $t$  and let  $p_t(i | j)$  be the probability that it is chosen at time  $t$  given that the  $j$ th expert was chosen at time  $t - 1$ . Let  $c_t^i$  be the  $i$ th element of the cost vector  $c_t$ . Under expectation one term of the sum (7) simplifies in the expert setting to

$$\begin{aligned} &\mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot (c_{t-1} - \hat{c}_{t-1})) \\ &= \sum_{i=1}^m \sum_{j=1}^m c_{t-1}^j p_t(i | j) p_{t-1}(j) - \sum_{i=1}^m c_{t-1}^i p_t(i | i). \end{aligned} \tag{8}$$

In order to upper bound equation (8) we use a technique similar to the one employed by Kalai and Vempala [1]. The probability of selecting expert  $i$  during an exploitation step is the probability that the used cost vector plus random perturbation vector will lie in the area of space where the  $i$ th coordinate is smaller than other coordinates. Let us denote this area by  $S_i \subseteq \mathbb{R}^n$ . For example, in case of two experts there are two areas  $S_1$  and  $S_2$  which are separated in  $\mathbb{R}^2$  by the diagonal line  $x = y$ . The probability  $p_t(i | j)$  in terms of the change  $\delta = \hat{c}_{t-1}$  in the cost vector  $b = \hat{c}_{1:t-2}$  is

$$\begin{aligned} p_t(i | j) &= \frac{\gamma}{m} + (1 - \gamma) \int_{S_i} \left(\frac{\epsilon}{2}\right)^n \exp(-\epsilon\|x - (b - \delta)\|_1) dx \\ &\leq \frac{\gamma}{m} + (1 - \gamma) \exp(\epsilon\|\delta\|_1) \int_{S_i} \left(\frac{\epsilon}{2}\right)^n \exp(-\epsilon\|x - b\|_1) dx \\ &\leq \exp(\epsilon\|\delta\|_1) p_{t-1}(i), \end{aligned} \tag{9}$$

where the inequality follows by the triangle inequality. Similarly  $p_t(i | j)$  may be lower bounded:

$$p_t(i | j) \geq \exp(-\epsilon\|\delta\|_1) p_{t-1}(i).$$

We assume that  $\epsilon c_i / p_i \leq 1$ , which implies that the sampling probability  $\gamma$  should be chosen such that  $\epsilon n c_i \leq \gamma$ . Then

$$\begin{aligned} p_t(i | j) &\leq \exp\left(\epsilon\left(\frac{c_{t-1}^i}{p_{t-1}(i)} + \|c_{t-1}\|\right)\right) p_{t-1}(i) \\ &\leq 1 + 2\epsilon\left(\frac{c_{t-1}^i}{p_{t-1}(i)} + \|c_{t-1}\|\right) p_{t-1}(i). \end{aligned} \tag{10}$$

Now, the left-hand term of equation (8) can be upper bounded as

$$\begin{aligned} &\sum_{i=1}^m \sum_{j=1}^m c_{t-1}^i p_t(i | j) p_{t-1}(j) \\ &\leq \sum_{i=1}^m \sum_{j=1}^m c_{t-1}^i \exp\left(\epsilon c_{t-1}^j \left(1 + \frac{1}{p_{t-1}(j)}\right)\right) p_{t-1}(i) p_{t-1}(j) \\ &\leq \sum_{i=1}^m c_{t-1}^i p_{t-1}(i) \left(\sum_{j=1}^m p_{t-1}(j) + 2\epsilon c_{t-1}^j (1 + p_{t-1}(j))\right) \\ &= \sum_{i=1}^m c_{t-1}^i p_{t-1}(i) + 2\epsilon (\|c\|_{t-1} + 1). \end{aligned}$$

Similarly, the right-hand term of equation (8) can be lower bounded as

$$\sum_{i=1}^n c_{t-1}^i p_t(i | i) \geq \sum_{i=1}^n c_{t-1}^i p_{t-1}(i) - \epsilon \|c_{t-1}\|_2^2.$$

Hence, the estimation regret obtained is  $\epsilon (2 (\|c_{t-1}\|_1 + 1) + \|c_{t-1}\|_2^2)$  as claimed, thus, completing the proof.

Theorem 1 can be interpreted to give the regret we suffer from not knowing the true  $c_t$  at each turn. All in all, the total regret in the expert setting can be upper bounded by

$$\frac{D(1 + \ln n)}{\epsilon} + \epsilon RAT + \gamma RT + \epsilon \left( 2(\|c_t\|_1 + 1) + \sum_{t=1}^T \|c_t\|_2^2 \right).$$

We were analyzing the expert setting so  $A = m$ ,  $R = 1$ ,  $D = 2$ ,  $\|c_t\|_1 \leq m$ , and  $\|c_t\|_2^2 \leq m$ . Plugging in a good  $\epsilon$  and  $\gamma$  we get an upper bound of

$$4\sqrt{5}\sqrt{(1 + \ln m)mT} + \frac{2\sqrt{1 + \ln m}}{\sqrt{5mT}},$$

which is  $O(\sqrt{T})$ . As far as we know, this is the first time that a  $O(\sqrt{T})$  bound has been demonstrated in a bandit FPL. As a side note, we failed to analyze the expert setting when the perturbation distribution was uniform, because if the probabilities are in the order of  $\sqrt{\epsilon}$ , a high regret occurs when this kind of analysis is used.

### 4 The General Case

Now we assume that the decision vectors may be arbitrary and prove a weaker bound for the regret than in the expert setting. Equation (2) gave a necessary condition for an unbiased estimate for  $c_t$ . However, an unbiased estimate as such does not guarantee a low regret, because the obtained estimates may be far from the true cost vector; i.e., the variance of the estimate is high. To analyze the regret we set a bound for the column vectors  $v_x$  of the matrix  $V$  that is  $\sum_{y \in X} \|v_y\|_1 \leq E$ . Intuitively, this restriction ensures that  $\mathbb{E}(\|\hat{c}_t\|_1)$  is at most proportional to  $E$ . In the next paragraphs it is argued that we can assume  $E$  to be polynomial in  $n$  (in fact linear, if we do a certain change in the basis).

Recall the definition of the matrix  $V$ :  $VX^T = P_{R(X^T)}$ . If  $v_i$  is the  $i$ th row of the matrix  $V$ , then  $Xv_i = P_{R(X^T)}^i$ , where  $P_{R(X^T)}^i$  is the  $i$ th column of the symmetric projection matrix  $P_{R(X^T)}$ . Now, if the vector  $P_{R(X^T)}^i$  belongs to the convex closure spanned by the column vectors of  $X$  (which are the decision vectors), then there are solutions in which  $\|v_i\|_1 = 1$ . Of course, this does not necessarily happen, but we can conclude that the norm  $\|v_i\|_1$  is a property of the relative locations of the decision vectors, not a function on the number of decisions  $m$  (if  $m$  is large relative to  $n$ ). Even better, the decision vectors can be transformed to a base where we can select  $E = n$ .

Recall that the barycentric spanner introduced by Awerbuch and Kleinberg [2] demonstrates that any decision vector can be presented as a linear combination of  $n$  decision vectors, where the coefficients are in  $[-1, 1]$ . The  $n$  decision vectors are called the barycentric spanner, and we choose columns of a matrix  $X_B$  to be these decision vectors. If the decision vectors are transformed to a base, where  $e_i$  corresponds to the  $i$ th decision vector in the barycentric spanner, then



we can choose  $v_i$  so that  $\|v_i\|_1 = 1$ , because we can choose  $v_i$  to be one for the  $i$ th decision in the barycentric spanner and zero otherwise. Thus,  $E = n$  in this case. If we are forced to use this trick with the barycentric spanner to bound  $E$ , then we estimate only using the vectors in the barycentric spanner. Additionally, we need to be sure that the bounds  $D$ ,  $A$ , and  $R$  do not change too much when moving to the base formed with the barycentric spanner. If we denote by  $D'$ ,  $R'$  and  $A'$  the new bounds when the decision vectors are transformed by  $X_B$ , then  $D' \leq 2n$ ,  $R' = R$ , and  $A' = \max_{c'} \|c'\|_1 = \max_c \|X_B c\|_1 \leq DA$ .

Finally, note that even though the vectors are projected to a base spanned by the barycentric spanner, the problem can be solved without projecting the decision vectors to this space by transforming the perturbation vector  $r_t$  to  $X_B^{-1}r_t$ . Then the oracle  $M(c)$  selects always the same decisions and thus the regret of both the projected and the original algorithm is the same. Thus, we can analyze an algorithm in the barycentric spanner, but in practice we do not need to project all decision vectors to the space spanned by the barycentric spanner.

The bound in the following theorem depends on the fact that  $\epsilon \|v_x\|_1 \leq p(x)$ , because then  $\exp(\epsilon \|\hat{c}_t\|_1)$  can be upper bounded additively. The assumption  $\epsilon \|v_x\|_1 \leq p(x)$  can be satisfied by lower bounding  $p(x)$  by choosing our sampling probability for a vector  $x$  to be  $\gamma \|v_x\|_1 / \sum_{y \in X} \|v_y\|_1$ . The following theorem gives the bound that results.

**Theorem 2.**  $\mathbb{E} (M(\hat{c}_{1:t-1} + r_t) \cdot \hat{c}_t(x_t) - (c_t \cdot x_t / p(x_t)) v_{x_t}) \leq \epsilon \sum_{t=1}^T \sum_{x \in X} \|v_x\|_1 / \sum_{y \in X} \|v_y\|_1 \cdot \exp(\epsilon \|\hat{c}_t\|_1)$

$$\mathbb{E} (M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot (c_{t-1} - \hat{c}_{t-1})) \leq 2\epsilon(R^2E + RA) + \frac{2\epsilon}{\gamma}(R^2AE^2 + RAE).$$

The proof of this theorem is given in Appendix A. The idea behind the proof is similar to the one given for the expert setting. The total regret over  $T$  time steps is:

$$\frac{D(1 + \ln n)}{\epsilon} + \epsilon RAT + \gamma RT + 2\epsilon R^2 ET + \frac{2\epsilon}{\gamma} R^2 AE^2 T.$$

Selecting

$$\epsilon = 2^{-1/3}(1 + \ln n)^{2/3} D^{2/3} A^{-1/3} R^{-1} E^{-2/3} T^{-2/3}$$

and

$$\gamma = 2^{1/3}(1 + \ln n)^{1/3} D^{1/3} A^{-1/3} R^{-1} E^{2/3} T^{-1/3}$$

leads to a regret in which the highest power of  $T$  is in the term

$$3 \cdot 2^{1/3} R (D(1 + \ln n) AE^2)^{1/3} T^{2/3}.$$

If the decision vectors are transformed to a basis spanned by a barycentric spanner, then the regret is

$$3 \cdot 4^{1/3} R (D(1 + \ln n) A)^{1/3} n T^{2/3}. \tag{11}$$

This bound can be applied to the algorithm of McMahan and Blum [3], which they analyzed to have a regret of  $O(\sqrt{\ln T} T^{3/4})$  against an adaptive adversary. However, some caution is required. Their estimation is based on sampling the decisions in the barycentric spanner and setting the estimate  $\hat{c}_t$  to  $n(\text{loss occurred})/\gamma X_B^{-1}e_i$  if the  $i$ th decision in the barycentric spanner was chosen on a sampling step. On exploitation steps they set the estimate  $\hat{c}_t$  to zero. Theorem 2 applies if the denominator in the estimate is the actual probability  $p(x_t)$  of choosing the decision in the barycentric spanner, not  $\gamma/n$ . Additionally, the perturbation vector  $r_t$  has to be transformed to  $X_B^{-1}r_t$ , as argued before. Now, a closer investigation to the proof of Theorem 2 reveals that if the first term  $2\epsilon(R^2E + RA)$  in the bound is divided by  $\gamma/n$ , then the bound applies. This modification to the bound has a small effect of increasing the constant in bound (11). Hence, the regret is  $O(T^{2/3})$  against an oblivious adversary.

## 5 Conclusions and Discussion

We have studied the performance of the bandit FPL algorithm, which can be used to solve an interesting class of problems, for example the shortest online routing with end-to-end feedback. Although this paper did not give a full solution to this problem, we were able to make some progress. In the standard multi-armed bandits problem  $O(\sqrt{T})$  regret bounds are attainable, and they are also asymptotically tight [10]. Our work implies that this is also true when the FPL algorithm is applied to the expert setting, although this is not very surprising. The general upper bound we proved implies that the algorithm of McMahan and Blum [3] has a regret of the order  $O(T^{2/3})$  against an oblivious adversary. These results can be seen as progress towards better understanding of the relationship between linear dependence of the cost function on decisions and the regret that can be achieved using the bandit FPL algorithms.

Several questions remain to be solved in future work. First, can our work be applied against an adaptive adversary? Second, what kinds of regrets are actually achievable in the setting considered in this paper?

## Acknowledgments

This work was supported by Academy of Finland project “INTENTS: Intelligent Online Data Structures”. Moreover, the work of J. Kujala is financially supported by Tampere Graduate School in Information Science and Engineering (TISE).

## References

- [1] Kalai, A., Vempala, S.: Efficient algorithms for online decision problems. In Schölkopf, B., Warmuth, M.K., eds.: Proceeding of the Sixteenth Annual Conference on Computational Learning Theory. Volume 2777 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer (2003) 26–40

- [2] Awerbuch, B., Kleinberg, R.D.: Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In: Proceeding of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, New York, NY, ACM Press (2004) 45–53
- [3] McMahan, H.B., Blum, A.: Geometric optimization in the bandit setting against an adaptive adversary. In Shawe-Taylor, J., Singer, Y., eds.: Proceeding of the Seventeenth Annual Conference on Learning Theory. Volume 3120 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer (2004) 109–123
- [4] Sleator, D., Tarjan, R.: Self-adjusting binary search trees. *Journal of the ACM* **32** (1985) 652–686
- [5] Hannan, J.: Approximation to Bayes risk in repeated plays. In Dresner, M., Tucker, A., Wolfe, P., eds.: Contributions to the Theory of Games. Volume 3. Princeton University Press, Princeton, NJ (1957) 97–139
- [6] Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In Fawcett, T., Mishra, N., eds.: Proceeding of the Twentieth International Conference on Machine Learning, Menlo Park, CA, AAAI Press (2003) 928–936
- [7] Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D., Schapire, R.E., Warmuth, M.K.: How to use expert advice. *Journal of the ACM* **44** (1997) 427–485
- [8] Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. *Information and Computation* **108** (1994) 212–261
- [9] Takimoto, E., Warmuth, M.K.: Path kernels and multiplicative updates. *Journal of Machine Learning Research* **4** (2003) 773–818
- [10] Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing* **32** (2002) 48–77
- [11] Flaxman, A.D., Kalai, A.T., McMahan, H.B.: Online convex optimization in the bandit setting: Gradient descent without a gradient. In: Proceeding of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New York, NY, ACM Press (2005) 385–394
- [12] Hutter, M., Poland, J.: Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research* **6** (2005) 639–660
- [13] Moore, E.H.: On the reciprocal of the general algebraic matrix (abstract). *Bulletin of the American Mathematical Society* **26** (1920) 394–395
- [14] Penrose, R.A.: A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society* **51** (1955) 406–413

## A Proof of Theorem 2

◊ ◊ ◊ We proceed with a straightforward but technically complex evaluation of the expected value.

Similarly as in the expert setting, the probability  $p_t(x_t | x_{t-1})$  can be upper bounded by

$$\exp(\epsilon \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1) p_{t-1}(x_t)$$

and, respectively, lower bounded by

$$\exp(-\epsilon \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1) p_{t-1}(x_t).$$

With the assumption that  $\epsilon \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1 \leq 1$  and using the properties of the function  $\exp$ , the bounds above imply the following additive bounds:

$$\begin{aligned} & p_{t-1}(x_t)(1 - \epsilon \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1) \\ & \leq p_t(x_t \mid x_{t-1}) \\ & \leq p_{t-1}(x_t)(1 + 2\epsilon \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1). \end{aligned} \tag{12}$$

Now we bound the estimation regret  $\mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot (c_{t-1} - \hat{c}_{t-1}))$ . First we upper bound  $\mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot c_{t-1})$ .

$$\begin{aligned} & \mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot c_{t-1}) \\ & = \sum_{x_{t-1} \in X} \mathbb{E}(M(\hat{c}_{1:t-2} + \hat{c}_{t-1}(x_{t-1}) + c_{t-1} + s_t) \cdot c_{t-1}) p_{t-1}(x_{t-1}) \end{aligned} \tag{13}$$

$$= \sum_{x_{t-1} \in X} p_{t-1}(x_{t-1}) \sum_{x_t \in X} ((c_{t-1} \cdot x_t) p_t(x_t \mid x_{t-1})) \tag{14}$$

$$\begin{aligned} & \leq \sum_{x_{t-1} \in X} p_{t-1}(x_{t-1}) \sum_{x_t \in X} ((x_t \cdot c_{t-1}) p_{t-1}(x_t)) \\ & \quad + 2\epsilon \sum_{x_{t-1} \in X} p_{t-1}(x_{t-1}) \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1 \sum_{x_t \in X} (|c_{t-1} \cdot x_t| p_{t-1}(x_t)) \end{aligned} \tag{15}$$

$$\leq c_{t-1} \cdot \mathbb{E}(M(\hat{c}_{1:t-2} + s_{t-1})) + 2\epsilon (R^2 E + RA). \tag{16}$$

Equations (13) and (14) result from the definition of the expected value that we are taking, inequality (15) follows from inequalities (12), and inequality (16) follows from  $|c_{t-1} \cdot x_t| \leq R$ , the definition of the bound  $E$ , and the triangle inequality.

Then we lower bound  $\mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot \hat{c}_{t-1})$ .

$$\begin{aligned} & \mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot \hat{c}_{t-1}) \\ & = \sum_{x_{t-1} \in X} \left( p_{t-1}(x_{t-1}) \hat{c}_{t-1}(x_{t-1}) \cdot \left( \sum_{x_t \in X} x_t p_t(x_t \mid x_{t-1}) \right) \right) \end{aligned} \tag{17}$$

$$\begin{aligned} & \geq \sum_{x_{t-1} \in X} (c_{t-1} \cdot x_{t-1}) v_{x_{t-1}} \\ & \quad \cdot \sum_{x_t \in X} (x_t (p_{t-1}(x_t) \pm 2\epsilon \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1 p_{t-1}(x_t))) \end{aligned} \tag{18}$$

$$\begin{aligned} & = c_{t-1} \cdot \mathbb{E}(M(\hat{c}_{1:t-1} + s_{t-1})) \\ & \quad - 2\epsilon \sum_{x_{t-1} \in X} |(c_{t-1} \cdot x_{t-1}) v_{x_{t-1}} \cdot \mathbb{E}(M(\hat{c}_{1:t-2} + s_{t-1}))| \\ & \quad \|\hat{c}_{t-1}(x_{t-1}) + c_{t-1}\|_1 \end{aligned} \tag{19}$$

$$\geq c_{t-1} \cdot \mathbb{E}(M(\hat{c}_{1:t-2} + s_{t-1})) - \frac{2\epsilon}{\gamma} (R^2 AE^2 + RAE). \tag{20}$$

Equation (17) is a definition of the expected value taken, inequality (18) follows from inequalities (12) if the  $\pm$  is interpreted as always choosing the sign that

results in the worst regret. Equality (19) has its terms calculated open and  $\pm$  is transformed to a minus by taking an absolute value of the terms within the sum. Finally, inequality (20) follows from the definitions of different bounds  $R$ ,  $A$ ,  $E$ , and the fact that  $p(x)$  is lower bounded by a sampling probability.

Thus,

$$\begin{aligned} & \mathbb{E}(M(\hat{c}_{1:t-1} + c_{t-1} + s_t) \cdot (c_{t-1} - \hat{c}_{t-1})) \\ & \leq 2\epsilon (R^2 E + RA) + \frac{2\epsilon}{\gamma} (R^2 AE^2 + RAE) \end{aligned}$$

as claimed.

# Mixture of Vector Experts

Matthew Henderson<sup>1</sup>, John Shawe-Taylor<sup>1</sup>, and Janez Žerovnik<sup>2</sup>

<sup>1</sup> School of Electronics and Computer Science,  
University of Southampton, Southampton SO17 1BJ, England

<sup>2</sup> Institute of Mathematics, Physics and Mechanics,  
Jadranska 19, Ljubljana 1111, Slovenia

**Abstract.** We describe and analyze an algorithm for predicting a sequence of  $n$ -dimensional binary vectors based on a set of experts making vector predictions in  $[0, 1]^n$ . We measure the loss of individual predictions by the 2-norm between the actual outcome vector and the prediction. The loss of an expert is then the sum of the losses experienced on individual trials. We obtain bounds for the loss of our expert algorithm in terms of the loss of the best expert analogous to the well-known results for scalar experts making real-valued predictions of a binary outcome.

## 1 Introduction

The first paper to consider the problem of predicting from expert advice with no assumptions made that the data is generated according to a probability distribution was Vovk's [3]. The assumption is that nature can be adversarial in its choice of outcomes forcing the learner to mix experts predictions in order to hedge against the worst case. Remarkably one is able to bound the cumulative loss experienced by the learner by the loss of the best expert plus an additional regret term that involves the logarithm of the number of experts.

Perhaps the definitive paper analysing this scenario is Cesa-Bianchi et al. [1]. The outcomes considered are in all cases binary, but the predictions themselves of both the experts and the learner can be real values in the interval  $[0, 1]$ . The main loss considered is the  $\ell_1$  between the prediction and target. This can be considered as the expected loss if a random prediction is made by viewing the real value as a probability.

Cesa-Bianchi et al. are able to define very general conditions under which the loss of the learner can be bounded. They also extend consideration to log loss. Haussler et al. have undertaken an extensive study of different loss functions [2].

The aim of the current paper is to extend the analysis of Cesa-Bianchi et al. to the case where the outcomes are binary vectors, the experts make real vector predictions and the loss is the 2-norm between outcome and prediction. We develop an algorithm and associated analysis that shows that as with the scalar case the learner can approximate the loss of the best expert to within a factor proportional to the logarithm of the number of experts.

It is natural to ask which norm should be used in the case of vector experts since both 1- and 2-norms reduce to the standard absolute difference in the scalar case. The 2-norm

$$\|\mathbf{x}\| = \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \tag{1}$$

is a natural loss measure to use for vector estimation and we use that norm in our analysis.

If, however, we were playing a 0-1 vector prediction game then the 1-norm would correspond to  $1/n$  times the expected Hamming distance when randomly setting each coordinate independently to 1 with probability equal to its prediction. The 2-norm of equation (1) provides an upper bound on the 1-norm since using the Cauchy-Schwarz inequality we have

$$\|\mathbf{x}\|_1 = \frac{1}{n} \sum_{i=1}^n x_i = \sum_{i=1}^n x_i (1/n) \leq \sqrt{\sum_{i=1}^n x_i^2} \sqrt{\frac{1}{n}} = \|\mathbf{x}\|.$$

Hence, our results also provide loss bounds for the 0-1 vector prediction game. We leave open the question of whether tighter bounds could not be obtained by working directly with the 1-norm loss.

A scenario where such a situation might arise is the labelling of images on sets of web pages. When crawling the web the types of images encountered are likely to behave in a manner that is unpredictable as different communities are encountered. It therefore does not make sense to probabilistic assumptions about the distributions of images that are encountered. Suppose now that we have a set of algorithms that are able to deliver labellings of images in the form of a vector indicating the probabilities of different words appearing in the caption. These algorithms can be considered as experts with the binary vector of words giving the true labelling as the outcome vector.

In the final section we describe a set of experiments for testing the performance of the approach against the strategy of using scalar experts for each component.

## 2 Mixture of Vector Experts Algorithm

We first define the mixture of vector experts algorithm.

**Algorithm 1.** Mixture of Vector Experts Algorithm

- $l$  experts, each returning a vector  $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_l\}$ ,  $\mathbf{y}_i \in \{0, 1\}^n$ ,  $1 \leq i \leq l$ .
- $N$  scalar experts, each returning a scalar  $\xi = \{\xi_1, \dots, \xi_N\}$ ,  $\xi_i \in [0, 1]$ ,  $1 \leq i \leq N$ .
- $t$  rounds of prediction,  $t < j$ ,  $1 \leq j \leq l$ .
- $l$  weight vectors,  $\mathbf{p} = \{\mathbf{p}_1, \dots, \mathbf{p}_l\}$ ,  $\mathbf{p}_i \in [0, 1]^n$ ,  $1 \leq i \leq l$ .

*Initialise weight vector*

$$w_{i,1} = 1, \dots, 1 \leq i \leq N.$$

for  $t \leq l$

**Compute prediction vector**

$$\mathbf{p}_t = \tilde{F}_\beta(\mathbf{r}_t) \in [0, 1]^n$$

$$\mathbf{r}_t = \sum_{i=1}^N \hat{w}_{i,t} \boldsymbol{\xi}_{i,t} \quad \hat{w}_{i,t} = w_{i,t} / \sum_{i=1}^N w_{i,t}$$

**Update weight vector**

$$w_{i,t+1} = w_{i,t} U_\beta(\|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\|), \quad 1 \leq i \leq n$$

$$\|\mathbf{x}\| = \sqrt{\frac{1}{n} \sum_{i=1}^N x_i^2}$$

### 2.1 Definition of $\tilde{F}_\beta$ and $U_\beta$

The function

$$\tilde{F}_\beta : [0, 1]^n \rightarrow [0, 1]^n$$

is defined by  $\tilde{F}_\beta(\mathbf{v}) = (F_\beta(v_1), F_\beta(v_2), \dots, F_\beta(v_n))$ , for some function  $F_\beta : [0, 1] \rightarrow [0, 1]$ . There is some flexibility in the choice of  $F_\beta$ . Any function which satisfies

$$1 + \frac{\ln((1-r)\beta + r)}{2 \ln(2/(1+\beta))} \leq F_\beta(r) \leq \frac{-\ln(1-r+r\beta)}{2 \ln(2/(1+\beta))}$$

for all  $0 \leq q \leq 1$  suffices for the bound given in Theorem 2.

There is a similar flexibility in the choice of the update function  $U_\beta$ . Any function which satisfies

$$\beta^q \leq U_\beta(q) \leq 1 - (1-\beta)q$$

for all  $0 \leq q \leq 1$  suffices for the bound in Theorem 2.

## 3 Performance of Algorithm 1

This section now considers bounding the performance of the mixture of vector experts algorithm (Algorithm 1).

**Theorem 2.** Let  $0 \leq \beta < 1$ ,  $\Xi = \{\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_N\}$ ,  $N$  experts,  $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_l\}$ ,  $\mathbf{y}_i \in \{0, 1\}^n$ ,  $1 \leq i \leq l$ ,  $\mathbf{p} = \{\mathbf{p}_1, \dots, \mathbf{p}_l\}$ ,  $\mathbf{p}_i \in [0, 1]^n$ ,  $1 \leq i \leq l$ . Then  $L(\mathbf{y}) = \sum_{t=1}^l \|\mathbf{y}_t - \mathbf{p}_t\|$

$$L(\mathbf{y}) \leq \frac{\ln N - \min_{1 \leq i \leq N} L_i(\mathbf{y}) \ln \beta}{2 \ln(2/(1+\beta))}$$

$$L_i(\mathbf{y}) = \sum_{t=1}^l \|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\|$$

The proof of Theorem 2 is based on the following lemma.



**Lemma 3.**

Let  $w_{i,j} \geq 0$ ,  $1 \leq i \leq N$ ,  $1 \leq j \leq l+1$ .

$$L(\mathbf{y}) \leq \frac{1}{2 \ln(2/(1 + \beta))} \ln \left( \frac{\sum_{i=1}^N w_{i,1}}{\sum_{i=1}^N w_{i,l+1}} \right)$$

We prove Lemma 3 in the Appendix. In the next subsection we prove Theorem 2 based on Lemma 3

**3.1 Proof of Theorem 2**

Consider the  $j$ th expert  $\xi_j \in \Xi$ . By assumption  $U_\beta(q) \geq \beta^q$  for all  $0 \leq q \leq 1$  therefore

$$\begin{aligned} \sum_{i=1}^N w_{i,l+1} &\geq w_{j,l+1} \\ &= w_{j,1} \prod_{t=1}^l U_\beta(\|\xi_{j,t} - \mathbf{y}_t\|) \\ &\geq \prod_{t=1}^l \beta^{\|\xi_{j,t} - \mathbf{y}_t\|} \\ &= \beta^{\sum_{t=1}^l \|\xi_{j,t} - \mathbf{y}_t\|} \\ &= \beta^{L_j(\mathbf{y})}. \end{aligned}$$

Now, from Lemma 3

$$\begin{aligned} L(\mathbf{y}) &\leq \frac{1}{2 \ln(2/(1 + \beta))} \ln \left( \frac{\sum_{i=1}^N w_{i,1}}{\beta^{L_j(\mathbf{y})}} \right) \\ &= \frac{1}{2 \ln(2/(1 + \beta))} \ln \left( \frac{N}{\beta^{L_j(\mathbf{y})}} \right) \end{aligned}$$

Therefore,

$$L(\mathbf{y}) \leq \frac{1}{2 \ln(2/(1 + \beta))} \ln \left( \frac{N}{\beta^{L_j(\mathbf{y})}} \right)$$

and so

$$L(\mathbf{y}) \leq \frac{\ln N + L_j(\mathbf{y}) \ln(1/\beta)}{2 \ln(2/(1 + \beta))}$$

Since the choice of the  $j$ th expert was arbitrary, it follows that

$$L(\mathbf{y}) \leq \frac{\ln N + \min_{1 \leq j \leq N} L_j(\mathbf{y}) \ln(1/\beta)}{2 \ln(2/(1 + \beta))}$$

which is precisely the statement of Theorem 2.

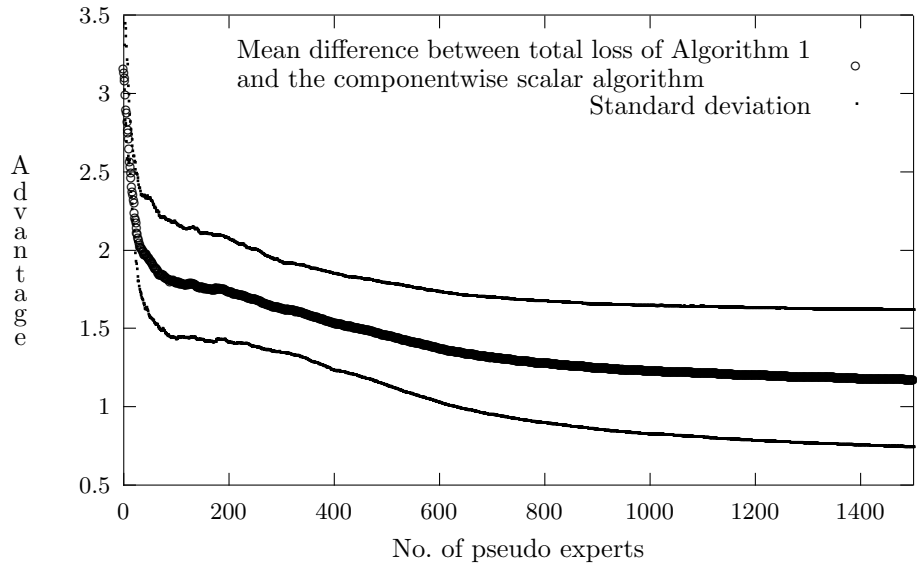
## 4 Computer Simulations

### 4.1 Motivation and Simulation Details

Some might argue that Algorithm 1 is redundant because of the existence of the mixture of scalar experts from [1]. After all, a perfectly feasible strategy for predicting a vector outcome sequence  $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_l\}$ , where  $\mathbf{y}_i \in \{0, 1\}^n$  for all  $1 \leq i \leq l$ , using the advice of a set  $\Xi$  of vector experts, is to take the prediction sequence  $(\xi_i)_j$  of each expert  $\xi_i \in \Xi$  on component  $j$  as the prediction sequence of a set of scalar experts and then to apply the mixture of scalar experts algorithm to this set of experts in order to predict the sequence  $\{\mathbf{y}_{1,j}, \mathbf{y}_{2,j} \dots \mathbf{y}_{l,j}\}$ . This sequence can be interpreted as the prediction sequence for component  $j$  and the same strategy can then be applied to the remaining components. (We call this strategy the *componentwise scalar algorithm*.) In this section we give some evidence which points towards a definite advantage of Algorithm 1 over the componentwise scalar algorithm.

Figure 1 is the result of a computer simulation in which Algorithm 1 is pitted against the componentwise scalar algorithm. In this simulation a noisy version of the outcome sequence is included as the prediction sequence of one of the vector experts and the objective of both algorithms is to track this expert to the point where the only loss suffered is just noise.

The zeroth data point of the middle line in Figure 1 is obtained in the following way. Both Algorithm 1 and the componentwise scalar algorithm are run with the same outcome sequence  $\mathbf{y} \in \{0, 1\}^4$  and the same set  $\Xi$  of 5 vector



**Fig. 1.** Plot showing the advantage of Algorithm 1 over the componentwise scalar algorithm

experts (each  $\xi \in \Xi$  predicting a sequence from  $[0, 1]^4$ ) as input. Algorithm 1 optimises according to the two-norm while the componentwise scalar algorithm optimises by the absolute difference and both algorithms use the same learning rate of  $\beta = 0.9$ . Both algorithms use the suitable update function

$$U_\beta(q) = 1 - (1 - \beta)q$$

and prediction function

$$F_\beta(r) = \begin{cases} 0 & \text{if } r < 1/2 - c \\ 1/2 - (1 - 2r)/4c & \text{if } 1/2 - c \leq r \leq 1/2 + c \\ 1 & \text{if } r > 1/2 + c \end{cases}$$

where

$$c = \frac{(1 + \beta) \ln(2/(1 + \beta))}{2(1 - \beta)}$$

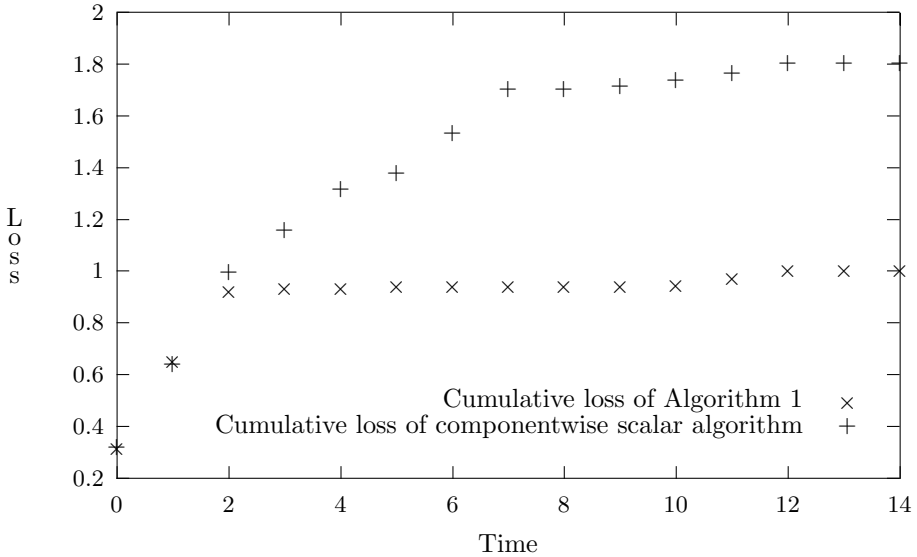
both of which are suggested in [1].

Such an individual trial for the zeroth data point is repeated a further 9 times, with 9 different random outcome sequences and 9 different sets of 5 vector experts. Then the mean difference between the total loss of the componentwise scalar algorithm and the total loss of Algorithm 1 is plotted on the vertical axis as the advantage of the componentwise scalar algorithm over Algorithm 1. The result of a typical simulation for the zeroth data point is illustrated in Figure 2. This shows a plot averaged over 10 trials of the cumulative loss against time of both Algorithm 1 and the componentwise scalar algorithm when given the same sequence (in this case of length 15) of outcome vectors (in this case each an element of  $\{0, 1\}^4$ ) and the same sequence of vector experts predictions (in this case each a member of  $[0, 1]^4$ ). The point plotted of the middle line in Figure 1 is the difference in height between the two curves at time  $t = 14$ . The point directly above and directly below show the standard deviation from the mean.

The simulation develops by introducing to the set of  $i_1, \dots, i_n$  experts an expanding set of randomly selected  $j_1, \dots, j_x$ . If  $\xi_i \in \Xi$  is a typical genuine vector expert whose prediction at time  $t$  on component  $j$  is  $(\xi_{i,t})_j \in [0, 1]$  then a typical pseudo-expert is defined by a random  $n$ -tuple of the indices of the genuine experts  $\mathbf{i} = (i_1, \dots, i_n)$  and predicts the sequence of vectors

$$\{\xi_{i_1,t}, \dots, \xi_{i_n,t}\} \quad \text{where} \quad (\xi_{\mathbf{i},t})_j = (\xi_{i_j,t})_j.$$

Introducing a pseudo-expert does not affect the componentwise scalar algorithm since it only remixes the same scalar experts. A pseudo-expert can, however, make tracking more difficult for Algorithm 1. Therefore, as the number of pseudo-experts grows the componentwise scalar algorithm is likely to become more competitive. This is illustrated by the plot in Figure 1. The point  $(x, y)$  on the middle plot is the mean, averaged over 10 trials, of the difference between the total loss of the two algorithm when  $x$  pseudo-experts are involved. The other two plots show the standard deviation. As the number of pseudo-experts increases Figure 1 illustrates the improved competitiveness of the componentwise scalar mixture algorithm.



**Fig. 2.** Plot of the cumulative losses of both Algorithm 1 and the componentwise scalar algorithm on the same set of experts and sequence of outcomes

## 4.2 Implementation Details

To produce Figure 1 both Algorithm 1 and the componentwise scalar algorithm were implemented in Python using the Numeric extension module. The entire simulation took approximately 28 minutes on a 1750 MHz Sempron.

## 4.3 Simulation Conclusions

In the case where a noisy version of the outcome sequence is included as the predictions of one of the vector experts Figure 1 suggests that it is a better strategy, albeit marginally so, in terms of minimizing total loss, to apply Algorithm 1 than to consider the components of the output sequence separately and run the algorithm from [1] on each component. This can be explained because Algorithm 1 has the ability to take less seriously the predictions for every component made by those experts who predict badly on any single component. This is a reasonable strategy under these circumstances because we know that a noisy version outcome is the prediction sequence made by at least one expert and so we expect the best expert to do well on all components. On the other hand the componentwise scalar algorithm is not able to punish a vector expert across all its component predictions and so if a vector expert does well on some of its components the componentwise scalar algorithm will take that expert's advice seriously despite the fact that it may be performing poorly elsewhere. This advantage is negated by introducing pseudo-experts and, as Figure 1 shows, the two algorithms become more competitive.

## 5 Conclusions

We have extended the experts algorithm and analysis to the case of predicting binary vectors. Our analysis bounds the loss of the algorithm in terms of the loss of the best vector expert. The loss used in the analysis is the 2-norm but this provides an upper bound on the 1-norm loss that would correspond to the expected Hamming distance if we performed a 0-1 vector prediction game.

Experimental evidence is given showing that the approach does outperform applying individual scalar experts to each component for the case where one of the experts is a good predictor.

We leave open the extension of these results to different norms, in particular to tuning them to provide better bounds for the case of a 1-norm loss.

## References

- [1] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, 1997.
- [2] D. Haussler, J. Kivinen, and M. K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Trans. on Information Theory*, 44(5):1906–1925, 1998.
- [3] V. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

## 6 Appendix

### 6.1 Proof of Lemma 3

We will show that for  $1 \leq t \leq l$ ,

$$\|\mathbf{p}_t - \mathbf{y}_t\| \leq \frac{1}{2 \ln(2/(1 + \beta))} \left( \frac{\sum_{i=1}^N w_{i,t}}{\sum_{i=1}^N w_{i,t+1}} \right) \quad (2)$$

The result then follows from summing (2) for  $t = 1, 2, \dots, l$ . To show that (2) holds for  $1 \leq t \leq l$  we show, eventually, that

$$\ln \left( \frac{\sum_{i=1}^N w_{i,t}}{\sum_{i=1}^N w_{i,t+1}} \right) \geq -\ln(1 - (1 - \beta)\|\mathbf{r}_t - \mathbf{y}_t\|) \quad (3)$$

holds for all  $1 \leq t \leq l$ . Before this we show that (2) follows from (3). Indeed, suppose (3) holds. Then (2) follows immediately if

$$\|\mathbf{p}_t - \mathbf{y}_t\| \leq \frac{-\ln(1 - (1 - \beta)\|\mathbf{r}_t - \mathbf{y}_t\|)}{2 \ln(2/(1 + \beta))}. \quad (4)$$

To show that (4) holds we first show that

$$|(\mathbf{p}_t)_j - (\mathbf{y}_t)_j| \leq \frac{-\ln(1 - (1 - \beta)|(\mathbf{r}_t)_j - (\mathbf{y}_t)_j|)}{2 \ln(2/(1 + \beta))} \tag{5}$$

and then apply Jensen’s Inequality to a certain real-valued function on  $\mathbb{R}$  to show that (4) follows from (5).

To see that (5) holds we first observe that  $(\mathbf{y}_t)_j \in \{0, 1\}$ . If  $(\mathbf{y}_t)_j = 0$  then (5) holds if and only if

$$|(\mathbf{p}_t)_j| \leq \frac{-\ln(1 - (1 - \beta)|(\mathbf{r}_t)_j|)}{2 \ln(2/(1 + \beta))} \tag{6}$$

and if  $(\mathbf{y}_t)_j = 1$  then (5) holds if and only if

$$|(\mathbf{p}_t)_j - 1| \leq \frac{-\ln(1 - (1 - \beta)|(\mathbf{r}_t)_j - 1|)}{2 \ln(2/(1 + \beta))}. \tag{7}$$

As  $(\mathbf{p}_t)_j \in [0, 1]$  and  $(\mathbf{r}_t)_j \in [0, 1]$  the latter inequality (7) holds if and only if

$$1 - (\mathbf{p}_t)_j \leq \frac{-\ln(1 - (1 - \beta)(1 - (\mathbf{r}_t)_j))}{2 \ln(2/(1 + \beta))}$$

which in turn holds if and only if

$$1 + \frac{\ln(r + \beta(1 - r))}{2 \ln(2/(1 + \beta))} \leq (\mathbf{p}_t)_j. \tag{8}$$

Inequalities (6) and (8) hold because  $(\mathbf{p}_t)_j = F_\beta((\mathbf{r}_t)_j)$  for some function  $F_\beta$  which, by choice, satisfies

$$1 + \frac{\ln(r + \beta(1 - r))}{2 \ln(2/(1 + \beta))} \leq F_\beta(r) \leq \frac{-\ln(1 - r + r\beta)}{2 \ln(2/(1 + \beta))}$$

for all  $0 \leq r \leq 1$ .

Now that we have proved (5) we prove that (4) follows. If we expand the left hand side of (4) we see immediately that (4) holds if and only if

$$\sqrt{\frac{1}{n} \sum_{j=1}^n ((\mathbf{p}_t)_j - (\mathbf{y}_t)_j)^2} \leq \frac{-\ln(1 - (1 - \beta)\sqrt{\frac{1}{n} \sum_{j=1}^n ((\mathbf{r}_t)_j - (\mathbf{y}_t)_j))}}{2 \ln(2/(1 + \beta))}$$

and from (5) we know that

$$\sqrt{\frac{1}{n} \sum_{j=1}^n ((\mathbf{p}_t)_j - (\mathbf{y}_t)_j)^2} \leq \sqrt{\frac{1}{n} \sum_{j=1}^n \left( \frac{-\ln(1 - (1 - \beta)|(\mathbf{r}_t)_j - (\mathbf{y}_t)_j|)}{2 \ln(2/(1 + \beta))} \right)^2}.$$

Therefore, (4) holds if

$$\sqrt{\frac{1}{n} \sum_{j=1}^n \left( \frac{-\ln(1 - (1 - \beta)|(\mathbf{r}_t)_j - (\mathbf{y}_t)_j|)}{2 \ln(2/(1 + \beta))} \right)^2} \leq \frac{-\ln(1 - (1 - \beta)\sqrt{\frac{1}{n} \sum_{j=1}^n ((\mathbf{r}_t)_j - (\mathbf{y}_t)_j)})}{2 \ln(2/(1 + \beta))}. \tag{9}$$

We prove (9) by defining the function  $h : \mathbb{R} \rightarrow \mathbb{R}$  given by the map

$$h : x \mapsto \left( \frac{-\ln(1 - (1 - \beta)\sqrt{x})}{2 \ln(2/(1 + \beta))} \right)^2.$$

This function is concave and so if we let  $x_j = ((\mathbf{r}_t)_j - (\mathbf{y}_t)_j)^2$  we conclude from Jensen’s Inequality that

$$\frac{1}{n} \sum_{j=1}^n \left( \frac{-\ln(1 - (1 - \beta)|(\mathbf{r}_t)_j - (\mathbf{y}_t)_j|)}{2 \ln(2/(1 + \beta))} \right)^2 \leq \left( \frac{-\ln(1 - (1 - \beta)\sqrt{\frac{1}{n} \sum_{j=1}^n ((\mathbf{r}_t)_j - (\mathbf{y}_t)_j)^2})}{2 \ln(2/(1 + \beta))} \right)^2 \tag{10}$$

which implies (9) and, consequently, (4).

So now that we have proved (4) we can prove Lemma 3 by showing that (3) holds. To show this we first observe that

$$\ln \left( \frac{\sum_{i=1}^N w_{i,t}}{\sum_{i=1}^N w_{i,t+1}} \right) = -\ln \left( \frac{\sum_{i=1}^N w_{i,t+1}}{\sum_{i=1}^N w_{i,t}} \right) \tag{11}$$

$$= -\ln \left( \frac{\sum_{i=1}^N w_{i,t} U_\beta(\|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\|)}{\sum_{i=1}^N w_{i,t}} \right) \tag{12}$$

$$\geq -\ln \left( \frac{\sum_{i=1}^N w_{i,t} (1 - (1 - \beta)\|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\|)}{\sum_{i=1}^N w_{i,t}} \right) \tag{13}$$

where (13) follows directly from the assumption that  $\beta^q \leq U_\beta(q) \leq 1 - (1 - \beta)q$ .

Therefore, (3) holds if

$$-\ln \left( \frac{\sum_{i=1}^N w_{i,t} (1 - (1 - \beta)\|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\|)}{\sum_{i=1}^N w_{i,t}} \right) \geq -\ln(1 - (1 - \beta)\|\mathbf{r}_t - \mathbf{y}_t\|). \tag{14}$$

To prove (14) it suffices to show that

$$\sum_{i=1}^N \hat{w}_{i,t} \|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\| \geq \|\mathbf{r}_t - \mathbf{y}_t\| \tag{15}$$

where  $\hat{w}_{i,t} = w_{i,t}/(\sum_{i=1}^N w_{i,t})$ . Indeed, suppose (15) holds. Then, because  $\beta \leq 1$  and so  $1 - \beta \geq 0$ , it follows that

$$(1 - \beta) \sum_{i=1}^N \hat{w}_{i,t} \|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\| \geq (1 - \beta) \|\mathbf{r}_t - \mathbf{y}_t\|. \tag{16}$$

Now, because both sides are  $\leq 1$  it follows that

$$1 - (1 - \beta) \sum_{i=1}^N \hat{w}_{i,t} \|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\| \leq 1 - (1 - \beta) \|\mathbf{r}_t - \mathbf{y}_t\| \tag{17}$$

and therefore, as  $\sum_{i=1}^N \hat{w}_{i,t} = 1$ , that

$$\sum_{i=1}^N \hat{w}_{i,t} (1 - (1 - \beta) \|\boldsymbol{\xi}_{i,t} - \mathbf{y}_t\|) \leq 1 - (1 - \beta) \|\mathbf{r}_t - \mathbf{y}_t\|$$

from which, because  $\hat{w}_{i,t} = w_{i,t}/(\sum_{i=1}^N w_{i,t})$ , (14) follows immediately.

Now, to prove (15) we define a certain real-valued concave function on  $\{0, 1\}^N$  and apply Jensen’s Inequality. The function in question is given by the map

$$h : \mathbf{x} \mapsto \left( \sum_{i=1}^N \hat{w}_{i,t} \sqrt{x_i} \right)^2.$$

Because the set  $\{0, 1\}^N$  is convex it is sufficient to show that the Hessian at  $x$ ,

$$D^2h(x) := \left( \frac{\partial^2 h}{\partial x_i \partial x_j} (x) \right)_{1 \leq i, j \leq N}$$

is a negative semidefinite matrix for every  $x \in \{0, 1\}^N$ . To prove this we first write  $x^2 = (x_1^2, x_2^2, \dots, x_N^2)$  for all  $x \in \{0, 1\}^N$  and then observe that

$$\frac{\partial h(x^2)}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_j} h_i(x^2) 2x_i = \begin{cases} 4h_{ij}(x^2)x_i x_j & \text{if } i \neq j \\ 4h_{ij}(x^2)x_i^2 + 2h_i(x^2) & \text{if } i = j \end{cases} \tag{18}$$

where

$$h_i = \frac{\partial h}{\partial x_i} \text{ and } h_{ij} = \frac{\partial h}{\partial x_i \partial x_j}.$$

Now, because  $h(x^2) = \left( \sum_{i=1}^N \hat{w}_{i,t} x_i \right)^2$  it follows that

$$\frac{\partial^2 h(x^2)}{\partial x_i \partial x_j} = \frac{\partial^2}{\partial x_i \partial x_j} \left( \sum_{k=1}^N \hat{w}_{k,t} x_k \right)^2 = 2\hat{w}_{i,t} \hat{w}_{j,t}$$

and because

$$\frac{\partial h}{\partial x_i} (x^2) = 2h_i(x^2)x_i = 2\sqrt{h(x^2)}\hat{w}_{i,t}$$

it follows that



$$h_{ij}(x^2) = \begin{cases} \hat{w}_{i,t}\hat{w}_{j,t}/2x_i x_j & \text{if } i \neq j \\ \hat{w}_{i,t}^2/2x_i^2 - \hat{w}_{i,t}\sqrt{h(x^2)}/2x_i^3 & \text{if } i = j. \end{cases} \tag{19}$$

The Hessian at  $x$ ,  $D^2h(x)$  is negative semidefinite, by definition, if and only if  $\alpha^T H \alpha \leq 0$  for all  $\alpha \in \mathbb{R}^N$  or equivalently,

$$\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j h_{ij}(x^2) \leq 0 \tag{20}$$

for all  $\alpha_i, \alpha_j \in \mathbb{R}$ , where without loss of generality we can take the argument to be  $x^2$ .

From (19),

$$\begin{aligned} \sum_{j=1}^N \alpha_i \alpha_j h_{ij}(x^2) &= \sum_{j \in [N] \setminus \{i\}} \alpha_i \alpha_j h_{ij}(x^2) + \alpha_i h_{ii}(x^2) \\ &= \sum_{j \in [N] \setminus \{i\}} \frac{\alpha_i \alpha_j \hat{w}_{i,t} \hat{w}_{j,t}}{2x_i x_j} + \frac{\alpha_i^2 \hat{w}_{i,t}^2}{2x_i^2} - \frac{\alpha_i^2 \hat{w}_{i,t}}{2x_i^3} \sqrt{h(x^2)} \\ &= \sum_{j=1}^N \frac{\alpha_i \alpha_j \hat{w}_{i,t} \hat{w}_{j,t}}{2x_i x_j} + \frac{\alpha_i^2 \hat{w}_{i,t}}{2x_i^3} \sqrt{h(x^2)} \end{aligned} \tag{21}$$

So (20) implies that  $D^2h(x)$  is negative semidefinite if and only if

$$\sum_{i=1}^N \left\{ \sum_{j=1}^N \frac{\alpha_i \alpha_j \hat{w}_{i,t} \hat{w}_{j,t}}{2x_i x_j} - \frac{\alpha_i^2 \hat{w}_{i,t}}{2x_i^3} \sqrt{h(x^2)} \right\} \leq 0. \tag{22}$$

To prove (22) we observe that the Cauchy-Schwarz inequality implies that

$$\left( \sum_{i=1}^N \frac{\alpha_i \hat{w}_{i,t}}{x_i} \right)^2 \leq \left( \sum_{i=1}^N \frac{\alpha_i^2 \hat{w}_{i,t}}{x_i^3} \right) \left( \sum_{i=1}^N \hat{w}_{i,t} x_i \right)$$

and so, expanding the left-hand side and dividing throughout by 2, we get

$$\sum_{i=1}^N \sum_{j=1}^N \frac{\alpha_i \alpha_j \hat{w}_{i,t} \hat{w}_{j,t}}{2x_i x_j} \leq \left( \sum_{i=1}^N \frac{\alpha_i^2 \hat{w}_{i,t}}{2x_i^3} \right) \left( \sum_{i=1}^N \hat{w}_{i,t} x_i \right)$$

which, because  $\sqrt{h(x^2)} = \sum_{i=1}^N w_i x_i$ , is the same inequality as (22). Therefore  $D^2h$  is negative semidefinite and so the function  $h$  is concave.

Now we can apply the function  $h$  to a particular vector and use Jensen's Inequality to deduce (14). The vector in question is

$$\mathbf{x}_j = \left( ((\boldsymbol{\xi}_{1,t})_j - (\mathbf{y}_t)_j)^2, ((\boldsymbol{\xi}_{2,t})_j - (\mathbf{y}_t)_j)^2, \dots, ((\boldsymbol{\xi}_{N,t})_j - (\mathbf{y}_t)_j)^2 \right).$$

By Jensen’s Inequality,

$$h\left(\frac{1}{n}\sum_{j=1}^n \mathbf{x}_j\right) \geq \frac{1}{n}\sum_{j=1}^n h(\mathbf{x}_j) \tag{23}$$

which, as we show now, implies (14).

To see that (23) implies (14) suppose that

$$h(\mathbf{x}_j) = ((\mathbf{r}_t)_j - (\mathbf{y}_t)_j)^2. \tag{24}$$

Then, (23) implies that

$$\left(\sum_{i=1}^N \hat{w}_{i,t} \sqrt{\frac{1}{n}\sum_{j=1}^n ((\boldsymbol{\xi}_{i,t})_j - (\mathbf{y}_t)_j)^2}\right)^2 \geq \sqrt{\frac{1}{n}\sum_{j=1}^n ((\mathbf{r}_t)_j - (\mathbf{y}_t)_j)^2}$$

which, if we take square roots of both sides and observe that  $\|\mathbf{x}\| = \sqrt{\frac{1}{n}\sum_{i=1}^N x_i^2}$ , in turn implies (14).

So to prove (14), which completes the proof of Lemma 3, it remains to show that (24) holds. To show that (24) holds we first observe that, from the definition of  $h(\mathbf{x}_j)$ , (24) can be written as

$$\left(\sum_{i=1}^N \hat{w}_{i,t} |(\boldsymbol{\xi}_{i,t})_j - (\mathbf{y}_t)_j|\right)^2 = ((\mathbf{r}_t)_j - (\mathbf{y}_t)_j)^2. \tag{25}$$

To show that (25) holds we observe that, by definition,  $(\mathbf{y}_t)_j \in \{0, 1\}$ . If  $(\mathbf{y}_t)_j = 0$  then (25) holds if and only if

$$\sum_{i=1}^N \hat{w}_{i,t} |(\boldsymbol{\xi}_{i,t})_j| = (\mathbf{r}_t)_j$$

which in this case follows directly from the definition of  $\mathbf{r}$ . If, on the other hand,  $(\mathbf{y}_t)_j = 1$  then we observe that  $(\boldsymbol{\xi}_{i,t})_j \in \{0, 1\}$ . If  $(\boldsymbol{\xi}_{i,t})_j = 0$  then (25) holds if and only if  $\left(\sum_{i=1}^N \hat{w}_{i,t}\right)^2 = ((\mathbf{r}_t)_j - 1)^2$  which follows because, from the definition of  $\mathbf{r}$ ,  $(\mathbf{r}_t)_j = \sum_{i=1}^N \hat{w}_{i,t} (\boldsymbol{\xi}_{i,t})_j$  and so in this case  $(\mathbf{r}_t)_j = 0$  and because, by definition,  $\sum_{i=1}^N \hat{w}_{i,t} = 1$ . Finally, if  $(\boldsymbol{\xi}_{i,t})_j = 1$  then (25) holds if and only if  $((\mathbf{r}_t)_j - 1)^2 = 0$ . This follows because  $(\mathbf{r}_t)_j = \sum_{i=1}^N \hat{w}_{i,t} (\boldsymbol{\xi}_{i,t})_j$  by definition and, in this case,  $\sum_{i=1}^N \hat{w}_{i,t} (\boldsymbol{\xi}_{i,t})_j = \sum_{i=1}^N \hat{w}_{i,t} = 1$ .

# On-line Learning with Delayed Label Feedback

Chris Mesterharm

Rutgers Computer Science Department,  
110 Frelinghuysen Road, Piscataway, NJ 08854  
mesterha@cs.rutgers.edu

**Abstract.** We generalize on-line learning to handle delays in receiving labels for instances. After receiving an instance  $x$ , the algorithm may need to make predictions on several new instances before the label for  $x$  is returned by the environment. We give two simple techniques for converting a traditional on-line algorithm into an algorithm for solving a delayed on-line problem. One technique is for instances generated by an adversary; the other is for instances generated by a distribution. We show how these techniques effect the original on-line mistake bounds by giving upper-bounds and restricted lower-bounds on the number of mistakes.

## 1 Introduction

In this paper, we consider the problem of label feedback in on-line learning. On-line learning is composed of trials. Each trial  $t$  can be broken up into three steps. First, the algorithm receives an instance from a set  $X$ . Second, the algorithm predicts a label from a finite set  $Y$ . Last, the algorithm receives the correct label from the environment. The goal of the algorithm is to minimize the number of mistakes. [1] We are interested in the last step. For many practical problems, the algorithm may not receive the label feedback in a timely matter.

Delayed labels are a realistic assumption for many potential on-line learning problems. Consider spam email filtering. The filtering algorithm often allows the user to train the algorithm using labeled emails. [2] In between training, many emails may arrive that need to be classified. Anytime successive predictions need to be made without receiving a label, it is a delayed learning problem. Another example is webpage prefetching. This is useful for speeding up the performance of low bandwidth Internet connections. Learning which links to preload is a useful optimization[3], however, the label feedback might be delayed until it is determined that a prefetched webpage will not be used. As a final example, a doctor may want to predict health problems in a patient in order to start treatment as soon as possible. A more definitive test may be prescribed to confirm the diagnosis; this test provides delayed feedback.

To solve this problem, we propose the delayed model of on-line learning. This model is identical to the traditional on-line learning except that the environment can return the label feedback any number of trials after the arrival of the instance. This amounts to changing the last step of on-line learning to

receive possibly multiple labels from the current or previous trials. For every on-line learning problem, there are matching delayed learning problems that receive delayed labels.

The main contribution of this paper is to give two ways to transform a traditional on-line algorithm to an algorithm that works with delayed labels. We give upper-bounds on the number of mistakes of these algorithms, where the bounds are given as a function of the bounds for the original on-line algorithm. We assume two different techniques for instance generation. First, we assume the instances are generated by an adversary. This is a common assumption used when analyzing many on-line algorithms. [1] Second, we assume the instances are generated by a distribution. While this assumption is less common [4], it can give tighter bounds for a large class of practical problems. In both cases, the bounds are robust; the bounds allow noisy instances that do not correspond to a target function [4], and the bounds allow tracking a target function that is allowed to change over the trials. [5, 6] We also give lower-bounds on restricted forms of these two instance generation techniques. We show these lower-bounds are close to our upper-bounds for these restricted problems.

## 2 Notation

All of our transformations take an existing traditional on-line algorithm and convert it to handle delayed instances. Let algorithm  $B$  be our traditional on-line algorithm. The pseudo-code for algorithm  $B$  is given in Fig. 1. On trial  $t$ , the algorithm accepts instance  $x_t \in X$  and returns a distribution  $\hat{y}_t \in [0, 1]^{|Y|}$  over the possible output labels. The algorithm predicts by sampling from this distribution.<sup>1</sup> The algorithm then receives feedback on the correct label in  $y_t \in Y$ . It can use this information to update the current state of the algorithm to improve performance on future instances.

### Initialization

$t \leftarrow 0$  is the trial number.  
Initialize algorithm state  $s \leftarrow s_0$ .

### Trials

$t \leftarrow t + 1$ .  
**Instance:**  $\mathbf{x}_t$ .  
**Prediction:**  $\hat{y}_t \leftarrow \text{Pred}(s, \mathbf{x}_t)$ .  
**Update:** Let  $y_t$  be the correct label.  
 $s \leftarrow \text{Update}(s, \mathbf{x}_t, y_t, \hat{y}_t)$ .

**Fig. 1.** On-line algorithm  $B$

The transformed algorithms use the same procedures as algorithm  $B$  for updates and predictions. The prediction procedure of algorithm  $B$  accepts two

<sup>1</sup> While it is common to just let  $\hat{y}_t$  be the predicted label, our notation is needed to help describe one of the algorithm transformations.

parameters: the instance  $\mathbf{x}_t$  for prediction and the current state of the algorithm,  $s$ . The state of the algorithm encodes the value of all the memory used by the algorithm that can have an effect on future predictions. The initial state is represented as  $s_0$ . The prediction procedure returns a probability distribution for the label. We use the notation  $\hat{y}_t[i]$  to determine the probability that the predicted label is  $i$  on trial  $t$ . For a deterministic algorithm, a single label will have value 1. The update procedure accepts four parameter: the state of the algorithm, the instance, the label returned by the environment, and the predicted label distribution. The update procedure returns two outputs: the new algorithm state and boolean variable **change** that is **TRUE** if the algorithm state has changed because of the update and **FALSE** otherwise. We ignore the **change** variable if it is not used by a particular algorithm.

The only difference in delayed on-line learning is that the label for instance  $x_t$  may not be returned at the end of trial  $t$ . Therefore, we need notation to represent when the label feedback is returned by the environment. We use  $y_{a,b}$  to refer to the label of an instance where the attributes arrive on trial  $a$  and the label arrives right before the start of trial  $b$ . Each instance arrives at a unique trial, but labels may arrive during the same trial. Therefore, when we want to specify an instance, we will refer to the trial where the attributes arrive. We define the delay of a particular instance, with label  $y_{a,b}$ , as  $b - a$ . Let  $k$  be the maximum delay over all instances. In traditional on-line learning, all instances have a delay of 1 and have labels of the form  $y_{t,t+1}$ . In delayed on-line learning, each instance may have an arbitrary positive delay.

In the rest of the chapter, we use the following notation. Let  $E[\text{Mist}(B, s)]$  be the expected number of mistakes  $B$  makes on  $s$ , a sequence of instances, and let  $E[\text{Change}(B, s)]$  be the expected number of times  $B$  changes its state on sequence  $s$ . The expectation is taken with respect to any randomization used by the algorithm. Let  $E[\text{Mist}(B)]$  be the maximum expected mistakes made by algorithm  $B$  over a set of instance sequences when instances are generated by an adversary. Also let  $E[\text{Mist}(B)]$  be the expected number of mistakes when instances are generated by a distribution. It should be clear from context whether we are dealing with an adversary or a distribution. In the case of a distribution, the expectation is taken with respect to the generation of instances and any randomization of the algorithm. Let  $\cdot_{\text{opt}}$  be the algorithm that minimizes  $E[\text{Mist}(B)]$ . Since we are interested in taking existing on-line algorithms and converting them to the delayed on-line model, the restrictions on instances will come from the on-line algorithm. If the  $B$  algorithm is deterministic we can drop the expectation from the notation in the adversarial case.

### 3 Instances Generated by an Adversary

In this section, we give algorithms and upper-bounds on mistakes when instances are generated by an adversary. First, we need to define what we mean by an adversary generating instances. Later we will allow the adversary to set delays for these instances. Let  $\mathcal{A}$  be a nonempty set of instances  $(x, y)$  where  $x \in X$

and  $y \in Y$ . These are the instances that an adversary can pick during a trial. We also define a function  $\eta(x, y)$  that measures the amount of noise in an instance. This is used to control what instances the adversary can pick but is not revealed to the learning algorithm.

The adversary can pick any instance for the current trial that has zero noise. These non-noisy instances correspond to the target function. The adversary must have restrictions on the number of noisy instances it can generate otherwise learning would be impossible. A common bound on the noisy instances is to allow only a fixed total amount of noise, where the total noise is computed by summing the  $\eta()$  amounts from the generated instances. However, our result generalizes any traditional adversarial on-line algorithm to the delayed setting and inherits whatever noise assumption is made by the original on-line algorithm including the values for the  $\eta()$  function.

In this paper, we allow the adversary to track a moving target function. Let  $\Phi = [(\mathcal{A}_1, \eta_1()), (\mathcal{A}_2, \eta_2()), \dots]$  be a sequence of instance sets and noise functions. We continually allow the adversary two choices; the adversary can either generate a trial by selecting an instance, or the adversary can increment the instance selection to the next element of  $\Phi$ . Instance generation starts at  $(\mathcal{A}_1, \eta_1())$ , and the adversary is not allowed to go backwards in the sequence. We are interested in the worst-case performance of the algorithm over a set of possible  $\Phi$ . This is a general model of instance generation that includes fixed concepts [1], by having only a single element in  $\Phi$ , and concept tracking [5, 6], by using instance sets and noise functions that correspond to different target functions.

For delayed on-line learning, we need to let the adversary delay the label feedback. To make this as general as possible, we will let the adversary pick the delay for an instance from a multi-set  $\mathcal{D}$  of positive numbers. More formally let  $d_i \in R \cup \infty$  be the maximum number of instances that have a delay of  $i$  trials;  $\mathcal{D} = \cup_{i=1}^{\infty} (\cup_{j=1}^{d_i} i)$ . For example,  $\mathcal{D}$  may only contain the number 5 an infinite amount of times. In this case, the adversary must give each instance a delay of 5. Our bound will be based on values of the various  $d_i$ ; this allows us to model a wide range of problems. For example, in the medical problem explained earlier, each patient may take a different amount of time to get the lab test needed for the label; a few patients may never take the lab test and have an infinite delay.

Before we give our main bound, we need a lemma to help us work with the delay multi-set  $\mathcal{D}$ . We want to place as many elements from  $\mathcal{D}$  into a list  $L$  with the restriction that the first  $C$  numbers must be at least 1, the next  $C$  numbers must be at least 2, and so on where the  $m$ th block of  $C$  numbers must be at least  $m$ . We call this the ordered class selection problem. If one uses the greedy algorithm of always placing the smallest number remaining in  $\mathcal{D}$  into the next position in the list then the total number of elements of value  $i$  that are in this greedy list is  $r_i = \min(d_i, iC - \sum_{j=1}^{i-1} r_j)$ .

**Lemma 1.**  $F(\mathcal{D}, C) = \sum_{j=1}^{\infty} r_j$

We break the proof into two cases. First, assume that  $F(\mathcal{D}, C) = \infty$ . Based on the definition of the ordered class problem, there must be no upper-bound on the elements in  $D$ . Therefore, the greedy algorithm will also generate an infinite list.

Second, assume that  $F(\mathcal{D}, C)$  is finite. Let  $l_o$  be a list of elements that satisfy the ordered class selection problem with the number of elements in  $l_o$  equal to  $F(\mathcal{D}, C)$ . Let  $l_g$  be the list generated by the greedy algorithm. We will compare each element of  $l_g$  with  $l_o$  and show that the lists must have the same length.

Start at the beginning of each list and compare elements. If  $l_o(0) = l_g(0)$  then go to the next element. If  $l_o(0) > l_g(0)$  then find the next index,  $i$ , in list  $l_o$  such that  $l_o(i) = l_g(0)$ . If index  $i$  exists then, in list  $l_o$ , swap values  $l_o(0)$  and  $l_o(i)$ . This still gives a legal list. If there is no such index then the number of elements with value  $l_g(0)$  used in list  $l_o$  must be less than  $d_{l_g(0)}$ . Therefore we can just assign  $l_o(0)$  to value  $l_g(0)$ . The new  $l_o$  list is still a valid list and still has length  $F(\mathcal{D}, C)$ .

We can repeat this procedure for each pair of elements from list  $l_o$  and  $l_g$ . Let  $i_e$  be the last element in list  $l_g$ . At this point both lists are identical up to index  $i_e$ . Any additional elements in  $l_o$  must have a value of at least  $l_g(i_e)$  since otherwise the  $l_g$  list would not be greedy. However, if the additional elements have values of at least  $l_g(i_e)$  then  $l_g$  would not end at index  $i_e$ . Therefore the new  $l_o$  list and  $l_g$  list must be the same length. Since the length of the modified  $l_o$  has not changed from the original length, the length of  $l_g$  is  $F(\mathcal{D}, C)$ . Based on the greedy algorithm, the length of  $l_g$  is also equal to  $\sum_{j=1}^{\infty} r_j$ . This proves the lemma.  $\square$

### 3.1 Algorithm

We call the first transformed algorithm  $1_{\downarrow}$ . This algorithm is similar to algorithm  $B$  except that it potentially skips some of the updates. The pseudocode for  $1_{\downarrow}$  is in Fig. 2. We use a stack  $U$  to store the instances that are ready for updates. We can store other instances in a hash table.

$1_{\downarrow}$  keeps track of a **last** trial and only performs updates using instances that are more recent than this trial. After the algorithm performs an update, if the update either changes the state of the algorithm, or if the update is based on an instance that could have caused a mistake, given the state used for the update, then the algorithm increases **last** to the trial of the instance used for the update. This ensures that the changes to the algorithm occur in the same order as the instance arrival times. Changes occurring out of order can cause problems when the concept is shifting.

The computational cost of the  $1_{\downarrow}$  algorithm is similar to the cost of the  $B$  algorithm. The number of updates is at most the same as algorithm  $B$  and the number of predictions is at most double. There is an extra cost based on sorting the instances in  $U$ . This cost depends on the number of labels returned per trial. Let  $\gamma$  be the maximum number of label returned during a trial. Using merge sort gives an amortized cost of at most  $O(\ln(\gamma))$  per trial. Because  $F(\mathcal{D}, 1)$  is the maximum number of instances that have arrived but have not yet received labels,

**Initialization**

$t \leftarrow 0$  is the trial number.  
 $\mathbf{last} \leftarrow 0$  is the last instance used for an update.  
 $U \leftarrow \mathbf{null}$  is a stack that stores instances that are ready for updates.  
 Initialize algorithm to state  $s \leftarrow s_0$ .

**Trials**

$t \leftarrow t + 1$ .  
**Instance:** Store  $\mathbf{x}_t$  using  $t$  as the key.  
**Prediction:**  $\hat{y}_t \leftarrow \text{Pred}(s, \mathbf{x}_t)$ .  
**Update:**  
 For all returned labels  $y_{a,t}$   
     If  $a > \mathbf{last}$  then  
         add instance  $(a, x_a, y_{a,t})$  to  $U$  in sorted order based on  $a$ .  
 For  $i = 1$  to  $|U|$   
      $(a, x_a, y_{a,t}) \leftarrow \text{pop}(U)$   
      $\hat{y} \leftarrow \text{Pred}(s, x_a)$   
      $(s, \mathbf{change}) \leftarrow \text{Update}(s, x_a, y_{a,t}, \hat{y})$   
     If  $\mathbf{change} = 1$  or  $\hat{y}[y_{a,t}] \neq 1$  then  
          $\mathbf{last} \leftarrow a$ .  
     Remove  $x_a$ .  
 Periodically remove all instances older than  $\mathbf{last}$ .

**Fig. 2.** Delayed on-line algorithm  $OD1-B$

we have that  $\gamma \leq F(\mathcal{D}, 1) \leq k$ . The  $\frac{1}{4}$ -algorithm needs extra storage for at most  $F(\mathcal{D}, 1)$  instances. By keeping space for  $2F(\mathcal{D}, 1)$  instances, the algorithm can periodically remove any old instances that missed an update with only a constant amortized increase in the cost per trial.

**3.2 Upper-Bound on Mistakes**

Here is our main result for learning against an adversary. Recall that we defined  $\text{Change}(B, s)$  as the number of times algorithm  $B$  changes its state on instance sequence  $s$ .

**Theorem 1.** *Let  $B$  be an algorithm that takes as input a sequence of instances  $s = (x_1, y_1, \dots, x_n, y_n)$  and returns a prediction  $\hat{y}_t$  for each instance  $x_t$ . Let  $\text{Mist}(B)$  be the number of mistakes made by  $B$  on  $s$ . Then  $E[\text{Mist}(B)] \leq E[\text{Change}(B)] + E[F(\mathcal{D}, 1)(B, s)] - E[\text{Change}(B, s)]$ .*

Consider all the instances that change the variable  $\mathbf{last}$  in algorithm  $\frac{1}{4}$ . The updates on these instances are in trial order. Call this sequence of instances  $u$ . If we pass these instances to algorithm  $B$ , giving each instance a delay of 1, we can expect at most  $E[\text{Mist}(B)]$  mistakes on algorithm  $B$ , since sequence  $u$  corresponds to a sequence that could be generated by the adversary. Notice that the algorithm states for  $\frac{1}{4}$  on subsequence  $u$  of  $s$  is identical to the states of algorithm  $B$  on  $u$ . For algorithm  $\frac{1}{4}$ , all the other trials in  $s$  just copy the state from the previous trial.



Next, it is useful to partition the sequence  $s$  into two sets. Let  $Q_1$  be the set of instances  $\mathbf{x}_t$  such that, when  $\mathcal{A}$  updates the label  $y_{t,t+k}$ , the state at trial  $t+k$  has not changed since trial  $t$ . Let  $Q_2$  be all other instances. We can divide the instances from  $Q_1$  into two groups. Let  $g_1$  be the instances from  $Q_1$  that are in  $u$ . Let  $g_2$  be all other instances in  $Q_1$ . Assume that  $x$  is an instance from  $g_1$ . The probability of a mistake by  $\mathcal{A}$  on  $x$  must equal the probability of a mistake in the equivalent instance from  $u$  on  $B$  since the state when the label arrived is the same as the state when the instance arrived. For the instances in  $g_2$ , the  $\mathcal{A}$  algorithm must have a zero probability of making a mistake otherwise the instance would be in  $u$ . Therefore the expected number of mistakes by algorithm  $\mathcal{A}$  for instances from  $Q_1$  must be at most  $E[\text{Mist}(B)]$ .

Next consider  $Q_2$ . There is a limit on the number of instances in  $Q_2$  based on the number of times the state changes and the number of instances with specific delay values. This number is primarily determined by the solution to the multi-set problem in lemma 1. However, for each state change at least one of the delayed instances from the multi-set solution must cause the update that changes the state. Therefore the expected number of elements in  $Q_2$  is at most  $E[F(\mathcal{D}, \text{Change}(B, s))] - E[\text{Change}(B, s)]$ . Since each instance in  $Q_2$  can cause at most one mistake, this proves the theorem.  $\square$

At this point, the bound is not very intuitive because of the complexity of the  $F$  function from Lemma 1. It is interesting to see how this function varies for different  $\mathcal{D}$  multi-sets. If all instances have delays of at most  $k$  then  $E[F(\mathcal{D}, \text{Change}(B, s))] \leq kE[\text{Change}(B, s)]$ . In addition, if we add  $m$  delays to  $\mathcal{D}$  of any value then the value of  $E[F(\mathcal{D}, \text{Change}(B, s))]$  can increase by at most  $m$ . This give a rough idea of how the bound depends on the delays of the instances. A more precise analysis will depend on a specific multi-set  $\mathcal{D}$ . In the remainder of the paper, we do not want to dwell on different choices for the  $\mathcal{D}$  multi-set. Therefore, we use the fact that  $E[F(\mathcal{D}, \text{Change}(B, s))] \leq kE[\text{Change}(B, s)]$  to simply our results. However, it is possible to generalize using the  $F$  function.

**Corollary 1.**  $E[\text{Mist}(B, s)] \leq E[\text{Change}(B, s)] + k(E[F(\mathcal{D}, \text{Change}(B, s))] - E[\text{Change}(B, s)])$

Combine  $E[F(\mathcal{D}, \text{Change}(B, s))] \leq kE[\text{Change}(B, s)]$  with Theorem 1.  $\square$

In order to get a good bound, we need to use a  $B$  algorithm that makes few mistakes and that changes its state few times. Fortunately, deterministic mistake-driven algorithms fulfill these criteria. A mistake-driven algorithm is an algorithm that only updates its state when it makes a mistake. [1] In Sect. 5, we will show that that converting a deterministic algorithm  $B$  to a mistake-driven form,  $\mathcal{A}$ , does not increase the mistake bound for a class of adversaries that includes the adversary used in this section.

To handle randomized algorithms, we use the fact that any randomized learning algorithm can be converted to a deterministic learning algorithm with a

similar mistake bound. On every trial, this new deterministic algorithm just predicts the highest probability label from the randomized algorithm. The deterministic algorithm makes at most double the expected number of mistakes of the randomized algorithm. [7] Given a learning algorithm  $B$ , we call  $\text{Derand}(B)$  the derandomized learning algorithm.

**Theorem 2.** *Let  $B$  be a randomized algorithm with  $E[\text{Mist}(B)] \leq M$ . Then  $E[\text{Mist}(\text{Derand}(B))] \leq 2M$ .*

Assume  $E[\text{Mist}(B)] = M$ . Using the derandomized algorithm, we get  $\text{Mist}(\text{Derand}(B)) \leq 2M$ . Next, we make the algorithm mistake-driven. Theorem 5 shows that  $\text{Mist}(\text{Mist-Driven}(\text{Derand}(B))) \leq 2M$ . Last, we use Corollary 1, and the fact that for any sequence of instances  $s$ ,  $E[\text{Change}(B, s)] \leq E[\text{Mist}(B)]$  for a mistake-driven algorithm.  $\square$

### 3.3 Lower-Bound on Mistakes

A natural question is whether a different transformations can make fewer mistakes. Assume that  $\mathcal{D}$  has an infinite number of delays of value  $k$  or greater. To help with the lower bound, we use another algorithm transformation. This transformation converts a delayed on-line learning algorithm  $C$  into a traditional on-line algorithm  $\text{Copy}(C, k)$ . We use the  $k$  notation because we have a different transformation for each value of  $k$ .

The  $\text{Copy}(C, k)$  algorithm solves a traditional on-line problem. The  $\text{Copy}(C, k)$  algorithm receives instance  $\mathbf{x}_1$  and creates  $k$  copies of the instance,  $(\mathbf{x}'_1 = \mathbf{x}_1, \dots, \mathbf{x}'_k = \mathbf{x}_1)$ . These  $k$  copies are used as the first  $k$  instances of algorithm  $C$ . When the label  $y_1$  is received, it is copied to  $k$  labels. The labels are spaced to give a delay of at least  $k$  to each instance. This continues for every trial of  $\text{Copy}(C, k)$  algorithm, creating  $k$  instances and labels for input into the  $C$  algorithm. The prediction for  $\mathbf{x}_t$  by the  $\text{Copy}(C, k)$  algorithm is just the random majority prediction of the  $C$  algorithm over the  $k$  identical instances. By random, we mean the algorithm predicts 1 with a probability equal to the ratio of the  $k$  instances that predict 1.

**Lemma 2.** *Let  $C$  be a delayed on-line learning algorithm with  $E[\text{Mist}(C)] \leq M$ . Then  $E[\text{Mist}(\text{Copy}(C, k))] \leq E[\text{Mist}(C)]/k$ .*

Consider of sequence of instances,  $s$ , for the on-line problem under consideration, where each instance has a delay of 1. Copy each instance  $k$  times and give each instance a delay of  $k$  or greater. Call this new sequence  $s'$ . Running algorithm  $\text{Copy}(C, k)$  on  $s$  is related to running algorithm  $C$  on  $s'$ . Every instance from  $s'$  that is predicted incorrectly increases the probability that  $\text{Copy}(C, k)$  will make a mistake on that instance by  $1/k$  because of the random majority algorithm. Therefore, for all sequences  $s$  generated by the adversary,  $E[\text{Mist}(\text{Copy}(C, k), s)] = E[\text{Mist}(C, s')]/k \leq E[\text{Mist}(C)]/k$ . Since this is true for all legal sequences  $s$ ,  $E[\text{Mist}(\text{Copy}(C, k))] \leq E[\text{Mist}(C)]/k$ .  $\square$

The previous lemma implies how the bound for a delayed on-line learning algorithm must grow with  $k$ .

**Theorem 3.**  $E[\text{Mist}(C)] = M$ .  $E[\text{Mist}(C)] \geq kM$

If  $E[\text{Mist}(C)] < kM$  then we can use Lemma 2 to create a traditional on-line algorithm,  $\text{Mist}(k)$ , where  $E[\text{Mist}(k)] \leq E[\text{Mist}(C)]/k < M$ . This contradicts the definition of  $M$ .  $\square$

There are learning problems that show this lower bound is tight. In addition, we can also give forms of the lower-bound that work for general delay multi-sets. This general form involves the  $F(\mathcal{D}, C)$  function. Because of space constraints, we will save these results for a later full version of this paper.

### 4 Instances Generated by a Distribution

In this section, we give an algorithm transformation for delayed on-line learning when the instances are generated by a shifting distribution. This shifting includes both the target function and the probability of a particular instance. A shifting distribution is a realistic model for many on-line learning problems. Often the learning environment is not trying to maximize the number of mistakes, instead the instances are generated by a distribution that is infrequently or slowly changing. For example, in our hypothetical medical learning problem, the population may be slowly changing dietary habits which could effect the target function.

Let  $\Psi = (\mathcal{W}_1, \eta_1()), (\mathcal{W}_2, \eta_2()), \dots$  be a sequence of distributions and noise functions over  $X \times Y$ . This model is similar to the adversary model in Sect. 3. The noise function  $\eta_i()$  maps the instances to a measure of noise, and the function may change for different algorithms. When using a shifting distribution for the traditional on-line model, we repeatedly allow the environment to either pick an instance from the current distribution or advance to the next distribution. While this is still partly adversarial, it does not give the environment as much freedom since the instance is picked from a distribution.

For the delayed on-line model, the environment selects the delays of the instances from a multi-set  $\mathcal{D}$ . We assume the environment must select a delay before picking an instance from the distribution. Allowing the environment to select the delays is a worst-case assumption, but it is possible to refine the analysis to allow a distribution to generate the delays.

A key component of the bound is the total amount the distribution changes over the trials. We use variational distance to measure the change between two distributions. [8] Given discrete distributions  $\mathcal{W}_1$  and  $\mathcal{W}_2$  over sample space  $H$ , let the probability of an element  $x$  be  $p_1(x)$  for  $\mathcal{W}_1$  and  $p_2(x)$  for  $\mathcal{W}_2$ . The variational distance is  $V(\mathcal{W}_1, \mathcal{W}_2) = \frac{1}{2} \sum_{x \in H} |p_1(x) - p_2(x)|$ . The total variational distance over all trials is  $\Psi = \sum_{i=1}^{\infty} V(\mathcal{W}_{i+1}, \mathcal{W}_i)$ . This definition generalizes to arbitrary probability measures.

### 4.1 Algorithm

In this section, we transform an on-line algorithm  $B$  to perform well in the delayed on-line model when instances are generated by a shifting distribution. We call the transformed algorithm  $\overline{\text{OD3}}_A(k')$ . The  $\overline{\text{OD3}}_A(k')$  algorithm does an update with every instance, and it does the updates in trial order, the same order as  $B$ . In other words,  $\overline{\text{OD3}}_A(k')$  can only update the instance from trial  $t + 1$  after the update for trial  $t$  occurs. Also only a single update is allowed per trial, so if multiple labels arrive at the start of the trial, only one can be used for the update. The remaining labels must wait for another trial to perform their update. Therefore,  $\overline{\text{OD3}}_A(k')$  computes the same hypotheses as algorithm  $B$ , but it will use them in different trials. There is an exception to the above scheme based on the single parameter  $k'$ . This parameter controls the maximum delay  $\overline{\text{OD3}}_A(k')$  will allow for any instance. If an instance has not received its label after  $k' + 1$  trials then the algorithm pretends the instance does not exist for the purpose of updates. For some problems, this technique is important since otherwise a single early instance with an infinite delay can prevent all later updates. The pseudo-code for  $\overline{\text{OD3}}_A(k')$  is given in Fig. 3. We use a heap  $U$  to store the instances that are ready for updates.

**Initialization**

$t \leftarrow 0$  is the trial number.  
**current**  $\leftarrow 0$  is the next instance for an update.  
 $U \leftarrow \text{null}$  is a heap that stores instances that are ready for updates.  
 Initialize algorithm to state  $s \leftarrow s_0$ .

**Trials**

$t \leftarrow t + 1$ .  
**Instance:** Store  $\mathbf{x}_t$  using  $t$  as the key.  
**Prediction:**  $\hat{y}_t \leftarrow \text{Pred}(s, \mathbf{x}_t)$ .  
**Update:**  
 If  $t - \text{current} = k' + 1$  then  
     **current**  $\leftarrow \text{current} + 1$ .  
 For all returned labels  $y_{a,t}$   
     If  $a \geq \text{current}$  then  
         add instance  $(a, x_a, y_{a,t})$  to  $U$ .  
 $a \leftarrow \text{top}(U)$ .  
 If  $a = \text{current}$   
      $(a, x_a, y_{a,t}) \leftarrow \text{extract-min}(U)$   
      $\hat{y} \leftarrow \text{Pred}(s, x_a)$   
      $s \leftarrow \text{Update}(s, x_a, y_{a,t}, \hat{y})$   
     **current**  $\leftarrow \text{current} + 1$ .  
     Remove  $x_a$ .  
 Periodically remove all instances older than **current**.

**Fig. 3.** Delayed on-line algorithm  $\overline{\text{OD3}}_A(k')$

The computational cost of the  $\overline{\text{OD3}}_A(k')$  algorithm is similar to the cost of the  $B$  and  $\text{OD3}_A$  algorithms. The number of updates is at most the same

as algorithm  $B$  and the number of predictions is at most double. There is an extra cost based on using the heap. The cost to insert and remove instances from the heap adds an amortized cost of  $O(\ln(k'))$  per trial. The  $\overline{3}_A(k')$  algorithm needs space for at most  $\min(k', k)$  instances. By keeping space for  $2 \min(k', k)$  instances, it can periodically remove any old instances with only a constant amortized increase in the cost per trial.

### 4.2 Upper-Bound on Mistakes

First, we prove a lemma that bounds the number of trials that do not perform an update. We need the following notation; let  $\mu$  be the maximum number of instances with a delay greater than  $k'$ .

**Lemma 3.** *Let  $t$  be the  $(\mu + \min(k', k) + 1)$ th trial that does not perform an update.*

Assume that trial  $t$  is the  $\min(k', k) + \mu + 1$  trial that does not perform an update. Therefore only  $t - \min(k', k) - \mu - 1$  labels have been used for updates. Looking at instance  $x_1$  to instance  $x_{t - \min(k', k)}$ , all of the labels from these instances that have a delay of at most  $\min(k', k)$  must have been returned by trial  $t$ . Therefore the minimum number of labels from these instances that have been received is  $t - \min(k', k) - \mu$ . This means there must be at least one label from this sequence of instances that has not been used for an update. By trial  $t$ , this label has been placed on the heap for an update. This is a contradiction since trial  $t$  does not perform an update.  $\square$

**Theorem 4.** *Let  $M$  be the maximum number of updates in any sequence of length  $\Psi$  on algorithm  $B$ . Let  $\mu$  be the maximum number of instances with a delay greater than  $k'$ . Then the maximum number of trials that do not perform an update for algorithm  $\overline{3}_A(k')$  is  $M + (\mu + \min(k', k) - 1)(\Psi + 1)$ .*

Let  $u$  be the sequence of instances that cause updates in algorithm  $\overline{3}_A(k')$ . If we use sequence  $u$  on algorithm  $B$  with a delay of 1 given to each instance then we expect to make at most  $M$  mistakes. This is because the distribution of sequence  $u$  can be generated by  $\Psi$ .

For algorithm  $B$  on sequence  $u$ , let  $h_i$  be the hypothesis that is used for prediction in trial  $i$ , and let  $X_i$  be a random variable that is 1 if algorithm  $B$  makes a mistake on trial  $i$  and 0 otherwise. Let  $Y_i$  be a random variable that is 1 if algorithm  $\overline{3}_A(k')$  makes a mistake on trial  $i$  and 0 otherwise.

The hypotheses used by  $\overline{3}_A(k')$  are the same as the hypotheses used by algorithm  $B$ . The only difference is that hypotheses are shifted a certain positive number of trials since the instances have to wait for their labels. This will force the  $\overline{3}_A(k')$  algorithm to occasionally use the same hypothesis for multiple trials as the algorithm waits for a label. Based on Lemma 3, the maximum number of trials that do not perform an update for algorithm  $\overline{3}_A(k')$  is  $\mu + \min(k', k)$ . Since the first trial never performs an update, this means that  $\mu + \min(k', k) - 1$  of the hypothesis from algorithm  $B$  are reused. Let  $r = \mu + \min(k', k) - 1$ .

Consider the shifted hypothesis of algorithm  $\overline{3}_r(k')$ . When the same hypothesis is used on two related distributions, the accuracies will be similar. The difference in accuracy comes from the amount the distribution changes between the trials. Let  $v_t = V(D_{t+1}, D_t)$ . Based on this metric, the error-rate of hypothesis  $h_t$  may, in the worst case, increase by  $v_t$  if used during trial  $t + 1$ . Since each hypothesis is shifted by at most  $r$  trials, the error-rate of hypothesis  $h_t$  can increase by at most  $\sum_{i=t}^{t+r-1} v_i$ . We can use this to bound the expected number of mistakes.

$$E[Mist(\overline{3}_r(k'))] = \sum_{i=1}^{\infty} E[Y_i] .$$

Assuming mistakes on repeated hypotheses and taking into account the number of trials the hypothesis from  $B$  are shifted, the above is

$$\leq r + \sum_{i=1}^{\infty} \left( E[X_i] + \sum_{j=i}^{i+r-1} v_j \right) \leq r + E \left[ \sum_{i=1}^{\infty} X_i \right] + r \sum_{i=1}^{\infty} v_i \leq r + M + r\Psi .$$

This proves the result. □

A possible modification to algorithm  $\overline{3}_r(k')$  is to predict with a random coin flip on any repeated hypothesis. This will lower the upper-bound on mistakes to  $M + (\mu + \min(k', k) - 1)(\Psi + 1/2)$ . In practice, one may want to restrict a coin flip prediction to repeat hypothesis near the start of the trials, since later repeated hypotheses may have a high accuracy.

### 4.3 Lower-Bound on Mistakes

For a shifting adversary, a trivial lower bound, for the delayed learning problem, is the bound of the optimal algorithm on the traditional on-line learning problem. This lower-bound bound is good when  $(\mu + \min(k', k) - 1)(\Psi + 1)$  is small.

Take for example the case where  $\Psi = 0$  forcing the distribution to be constant. Here the bound for  $\overline{3}_r(k')$  is  $Mist(\cdot) + \mu + \min(k', k) - 1$ . The term  $\mu + \min(k', k) - 1$  comes from the assumption that  $\overline{3}_r(k')$  makes a mistake on all the repeated hypotheses. In the worst-case, these repeated hypothesis can be forced to be the beginning trials of the sequence. Since the algorithm will have no information about the labels of these beginning trials, the probability of a mistake will depend on the algorithm being able to select a initial hypothesis that will guarantee good performance no matter what learning problem the adversary selects. For many learning problems, this will not be possible. In the full version of the paper, we give an algorithm for learning fixed distributions that slightly improves the bound and removes the parameter  $k'$ . This new algorithm's upper-bound on mistakes does not depend on  $k$  but instead depends on  $q$ , the maximum number of starting trials that do not receive a label. It has an expected bound of  $Mist(\cdot) + (q - 1)/2$  which, as explained, for many problems is optimal.

We do not have a good lower bound when  $\Psi > 0$ , however notice that a shifting distribution can duplicate an adversary by using distributions that place all

the weight on particular instances. Therefore, the bounds for shifting distributions also covers adversaries. Unfortunately, the distribution based bounds get considerably weaker when the distribution changes frequently. Therefore, when dealing with a problem that is more adversarial, this bound will be quite poor. The distribution bound is most relevant for problems where  $\Psi$  is small.

## 5 Mistake-Driven Algorithms

In this section, we restrict ourselves to deterministic traditional on-line algorithms. We prove that a simple transformation for converting any on-line algorithm into a mistake-driven algorithm does not effect the mistake bound for a wide range of adversaries. A mistake-driven algorithm is an algorithm that only changes its internal state when it makes a prediction mistake on an instance.

In a paper by Littlestone [1], a transformation is given to convert an on-line learning algorithm to a mistake-driven algorithm with the same mistake-bound. However, this transformation only applies to learning fixed concepts without noise. We consider a simpler transformation that skips any instances that are correctly classified. The only instances that can effect the state of the algorithm are instances that cause mistakes. The technique was originally used for converting Bayesian algorithms into algorithms that perform well against adversaries.[9, 10] Notationally, we will add the prefix  $\text{MD}$  to any algorithm to show that it has been converted into the mistake-driven form.

We prove that this simple transformation retains the existing mistake bound for a specific type of adversary. We call these special adversaries subset adversaries.

**Definition 1.** A subset adversary is an adversary that only generates instances that cause a mistake for algorithm  $A$ . Let  $s$  be a sequence of instances that cause a mistake for algorithm  $A$ .

The next theorem is used in Sect. 3 to help give a bound on algorithm  $\text{MD-}A$  when instances are generated by an adversary.

**Theorem 5.** Let  $B$  be an algorithm with mistake bound  $M(B)$ . Let  $A$  be an algorithm with mistake bound  $M(A)$ . Then the mistake bound of  $\text{MD-}A$  is at most  $M(A) + M(B)$ .

Since the instances are generated by a subset adversary, there must exist a sequence of instances  $s$  that maximizes the number of mistakes for algorithm  $A$  where all the mistakes occur at the beginning of the sequence. Up to a certain trial  $m$ , both algorithm  $B$  and  $A$  must make identical predictions, updates, and mistakes on instances sequence  $s$ . Since  $A$  makes no further mistakes past trial  $m$ ,  $Mist(\text{MD-}A) = Mist(\text{MD-}A, s) \leq Mist(B, s) \leq Mist(B)$ .  $\square$

To understand the limits of a subset adversary, we need a general definition for an adversary. A definition used in many papers is to define an adversary with

the set of sequences it is allowed to generate. [7] Call this set of sequences  $\mathcal{S}$ . Anytime  $\mathcal{S}$  is not closed with respect to subsets then the adversary is not a subset adversary. A non-subset adversary must generate correctly classified instances for some algorithm/problem combinations in order to maximize the mistake bound. If these instances contain new information about the target function, they can help lower the mistake bound.

As an example of a non-subset adversary, consider a shifting concept where the concept is forced to shift at specific trials. This is not a subset adversary since all possible subsequences of instances are not allowed. This may force the adversary to generate an instance that will be predicted correctly. However, if we assume that there are default instances that the adversary can always generate that will not give much information about the target concept<sup>2</sup> then mistake-driven algorithms will still give close to the best bounds.

It is an open question as to whether any types of non-subset adversaries are useful for modeling learning problems. One purpose of an adversarial analysis is to show that an algorithm performs well even given the worst-case assumption of an adversary. Since an adversary can always be extended to a subset adversary by adding instance sequences, a subset adversary extends this notion of worst-case. In addition, being a subset adversary is only a sufficient condition for Theorem 5. For many problems,  $\text{Mist}(\cdot) \leq \text{Mist}(B)$  is true even if the adversary is not a subset adversary.

We want to stress that many practical on-line problems will not have an adversary generating the instances. In these cases, a more aggressive algorithm that sometimes updates on correct predictions can improve performance. [11, 12] Still the algorithm must be careful to avoid extra updates that increase the number of mistakes.

## 6 Conclusion

In this paper, we give algorithms and mistake bounds for delayed on-line learning. In general, when dealing with an adversary generating the instances, the new bounds can be poor. If the instances all have a delay of  $k$  trials then the mistake bound can grow by a factor of  $2k$  over the normal on-line learning bounds. We show this bound is within a factor of 2 from optimal. We also give a more general analysis for problems where the adversary must select the delay of instances from a multi-set  $\mathcal{D}$ . This upper-bound depends on a particular combinatorial property of  $\mathcal{D}$  and gives insight to how the algorithm behaves with a range of instance delays. Things are more hopeful when dealing with a distribution generating the instances. In this case, if all the instances have a delay  $k$  then the expected mistake bound only increases by  $k - 1$ . Slightly shifting the distribution also performs well with a penalty based on the amount of shifting.

---

<sup>2</sup> This instance could be a previously given instance or an instance that has a known value based on the set of target functions, such as the instance of all zeros for monotone disjunctions.



In the full version of the paper, we will include two additional algorithms and experiments to show how the delayed algorithms perform on a shifting concept using a form of the Winnow algorithm. [13] These additional algorithms fill in the gaps with the numbering convention used in naming our delayed on-line algorithm transformations.

## References

- [1] Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* **2** (1988) 285–318
- [2] Androutsopoulos, I., Koutsias, J., Chandrinos, K., Paliouras, G., Spyropoulos, C.: An evaluation of naive bayesian anti-spam filtering (2000)
- [3] Padmanabhan, V.N., Mogul, J.C.: Using predictive prefetching to improve world wide web latency. *ACM SIGCOMM Computer Communication Review* **26** (1996) 22–36
- [4] Littlestone, N.: Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In: *Proceedings of the Third Annual Conference on Computational Learning Theory*. (1991) 147–156
- [5] Helmbold, D.P., Long, P.M.: Tracking drifting concepts using random examples. In: *Proceedings of the Third Annual Conference on Computational Learning Theory*. (1991) 13–23
- [6] Kuh, A., Petsche, T., Rivest, R.L.: Learning time-varying concepts. In: *Neural Information Processing Systems Three*, Morgan Kaufmann Publishers, Inc. (1991) 183–189
- [7] Auer, P., Warmuth, M.K.: Tracking the best disjunction. In: *Proceedings of the 36th annual symposium on foundations of computer science*, IEEE Computer Society Press (1995) 312–321
- [8] Devroye, L., Györfi, L., Lugosi, G.: *A Probabilistic Theory of Pattern Recognition*. Springer, New York (1991)
- [9] Littlestone, N.: Comparing several linear-threshold learning algorithms on tasks involving superfluous attributes. In: *Proceeding of the Twelve International Conference on Machine Learning*. (1995) 353–361
- [10] Littlestone, N., Mesterharm, C.: An apobayesian relative of winnow. In: *Neural Information Processing Systems Nine*, MIT Press (1997) 204–210
- [11] Minsky, M.L., Papert, S.A.: *Perceptrons*. MIT Press, Cambridge, MA (1969)
- [12] Li, Y., Long, P.: The relaxed online maximum margin algorithm. In: *Neural Information Processing Systems Twelve*, MIT Press (2000) 498–504
- [13] Mesterharm, C.: Tracking linear-threshold concepts with winnow. *Journal of Machine Learning Research* **4** (2003) 819–838

# Monotone Conditional Complexity Bounds on Future Prediction Errors\*

Alexey Chernov and Marcus Hutter

IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland  
{alexey, marcus}@idsia.ch  
<http://www.idsia.ch/~{alexey,marcus}>

**Abstract.** We bound the future loss when predicting any (computably) stochastic sequence online. Solomonoff finitely bounded the total deviation of his universal predictor  $M$  from the true distribution  $\mu$  by the algorithmic complexity of  $\mu$ . Here we assume we are at a time  $t > 1$  and already observed  $x = x_1 \dots x_t$ . We bound the future prediction performance on  $x_{t+1}x_{t+2} \dots$  by a new variant of algorithmic complexity of  $\mu$  given  $x$ , plus the complexity of the randomness deficiency of  $x$ . The new complexity is monotone in its condition in the sense that this complexity can only decrease if the condition is prolonged. We also briefly discuss potential generalizations to Bayesian model classes and to classification problems.

## 1 Introduction

We consider the problem of online=sequential predictions. We assume that the sequences  $x = x_1x_2x_3 \dots$  are drawn from some “true” but unknown probability distribution  $\mu$ . Bayesians proceed by considering a class  $\mathcal{M}$  of models=hypotheses=distributions, sufficiently large such that  $\mu \in \mathcal{M}$ , and a prior over  $\mathcal{M}$ . Solomonoff considered the truly large class that contains all computable probability distributions [Sol64]. He showed that his universal distribution  $M$  converges rapidly to  $\mu$  [Sol78], i.e. predicts well in any environment as long as it is computable or can be modeled by a computable probability distribution (all physical theories are of this sort).  $M(x)$  is roughly  $2^{-K(x)}$ , where  $K(x)$  is the length of the shortest description of  $x$ , called Kolmogorov complexity of  $x$ . Since  $K$  and  $M$  are incomputable, they have to be approximated in practice. See e.g. [Sch02b, Hut04, LV97, CV05] and references therein. The universality of  $M$  also precludes useful statements of the prediction quality at particular time instances  $n$  [Hut04–p62], as opposed to simple classes like i.i.d. sequences (data) of size  $n$ , where accuracy is typically  $O(n^{-1/2})$ . Luckily, bounds on the expected  $\sum_{t=1}^n \ell_t$ =cumulative loss (e.g. number of prediction errors) for  $M$  can be derived [Sol78, Hut03a, Hut03b], which is often sufficient in an online setting. The bounds are in terms of the (Kolmogorov) complexity of  $\mu$ . For instance, for

---

\* This work was supported by SNF grants 200020-107590/1 (to Jürgen Schmidhuber), 2100-67712 and 200020-107616.

deterministic  $\mu$ , the number of errors is (in a sense tightly) bounded by  $K(\mu)$  which measures in this case the information (in bits) in the observed infinite sequence  $x$ .

**What’s New.** In this paper we assume we are at a time  $t > 1$  and already observed  $x = x_1 \dots x_t$ . Hence we are interested in the future prediction performance on  $x_{t+1}x_{t+2}\dots$ , since typically we don’t care about past errors. If the total loss is finite, the future loss must necessarily be small for large  $t$ . In a sense the paper intends to quantify this apparent triviality. If the complexity of  $\mu$  bounds the total loss, a natural guess is that something like the conditional complexity of  $\mu$  given  $x$  bounds the future loss. (If  $x$  contains a lot of (or even all) information about  $\mu$ , we should make fewer (no) errors anymore.) Indeed, we prove two bounds of this kind but with additional terms describing structural properties of  $x$ . These additional terms appear since the total loss is bounded only in expectation, and hence the future loss is small only for “most”  $x_1 \dots x_t$ . In the first bound (Theorem 1), the additional term is the complexity of the length of  $x$  (a kind of worst-case estimation). The second bound (Theorem 7) is finer: the additional term is the complexity of the randomness deficiency of  $x$ . The advantage is that the deficiency is small for “typical”  $x$  and bounded on average (in contrast to the length). But in this case the conventional conditional complexity turned out to be unsuitable. So we introduce a new natural modification of conditional Kolmogorov complexity, which is monotone as a function of condition. Informally speaking, we require programs (=descriptions) to be consistent in the sense that if a program generates some  $\mu$  given  $x$ , then it must generate the same  $\mu$  given any prolongation of  $x$ . The new posterior bounds also significantly improve the previous total bounds.

**Contents.** The paper is organized as follows. Some basic notation and definitions are given in Sections 2 and 3. In Section 4 we prove and discuss the length-based bound Theorem 1. In Section 5 we show why a new definition of complexity is necessary and formulate the deficiency-based bound Theorem 7. We discuss the definition and basic properties of the new complexity in Section 6, and prove Theorem 7 in Section 7. We briefly discuss potential generalizations to general model classes  $\mathcal{M}$  and classification in the concluding Section 8.

## 2 Notation and Definitions

We essentially follow the notation of [LV97, Hut04].

**Strings and Natural Numbers.** We write  $\mathcal{X}^*$  for the set of finite strings over a finite alphabet  $\mathcal{X}$ , and  $\mathcal{X}^\infty$  for the set of infinite sequences. The cardinality of a set  $\mathcal{S}$  is denoted by  $|\mathcal{S}|$ . We use letters  $i, k, l, n, t$  for natural numbers,  $u, v, x, y, z$  for finite strings,  $\epsilon$  for the empty string, and  $\alpha = \alpha_{1:\infty}$  etc. for infinite sequences. For a string  $x$  of length  $\ell(x) = n$  we write  $x_1x_2\dots x_n$  with  $x_t \in \mathcal{X}$  and further abbreviate  $x_{k:n} := x_kx_{k+1}\dots x_{n-1}x_n$  and  $x_{<n} := x_1\dots x_{n-1}$ . For  $x_t \in \mathcal{X}$ , denote by  $\bar{x}_t$  an arbitrary element from  $\mathcal{X}$  such that  $\bar{x}_t \neq x_t$ . For binary alphabet  $\mathcal{X} = \{0,1\}$ , the  $\bar{x}_t$  is uniquely defined. We occasionally identify strings with natural numbers.

**Prefix Sets.** A string  $x$  is called a (proper) prefix of  $y$  if there is a  $z(\neq \epsilon)$  such that  $xz = y$ ;  $y$  is called a prolongation of  $x$ . We write  $x* = y$  in this case, where  $*$  is a wildcard for a string, and similarly for infinite sequences. A set of strings is called prefix free if no element is a proper prefix of another. Any prefix set  $\mathcal{P}$  has the important property of satisfying Kraft’s inequality  $\sum_{x \in \mathcal{P}} |\mathcal{X}|^{-\ell(x)} \leq 1$ .

**Asymptotic Notation.** We write  $f(x) \overset{\leq}{\asymp} g(x)$  for  $f(x) = O(g(x))$  and  $f(x) \overset{\geq}{\asymp} g(x)$  for  $f(x) \leq g(x) + O(1)$ . Equalities  $\overset{\approx}{\asymp}$ ,  $\overset{\pm}{\asymp}$  are defined similarly: they hold if the corresponding inequalities hold in both directions.

**(Semi)measures.** We call  $\rho : \mathcal{X}^* \rightarrow [0,1]$  a (semi)measure  $\nu$ ,  $\sum_{x_n \in \mathcal{X}} \rho(x_{1:n}) \overset{(\leq)}{\leq} \rho(x_{<n})$  and  $\rho(\epsilon) \overset{(\leq)}{\leq} 1$ .  $\rho(x)$  is interpreted as the  $\rho$ -probability of sampling a sequence which starts with  $x$ . The conditional probability (posterior)  $\rho(y|x) := \frac{\rho(xy)}{\rho(x)}$  is the  $\rho$ -probability that a string  $x$  is followed by (continued with)  $y$ . We call  $\rho$  deterministic if  $\exists \alpha : \rho(\alpha_{1:n}) = 1 \ \forall n$ . In this case we identify  $\rho$  with  $\alpha$ .

**Random Events and Expectations.** We assume that sequence  $\omega = \omega_{1:\infty}$  is sampled from the “true” measure  $\mu$ , i.e.  $\mathbf{P}[\omega_{1:n} = x_{1:n}] = \mu(x_{1:n})$ . We denote expectations w.r.t.  $\mu$  by  $\mathbf{E}$ , i.e. for a function  $f : \mathcal{X}^n \rightarrow \mathbb{R}$ ,  $\mathbf{E}[f] = \mathbf{E}[f(\omega_{1:n})] = \sum_{x_{1:n}} \mu(x_{1:n}) f(x_{1:n})$ . We abbreviate  $\mu_t := \mu(x_t | \omega_{<t})$ .

**Enumerable Sets and Functions.** A set of strings (or naturals, or other constructive objects) is called enumerable if it is the range of some computable function. A function  $f : \mathcal{X}^* \rightarrow \mathbb{R}$  is called enumerable if the set of pairs  $\{(x, \frac{k}{n}) \mid f(x) \overset{(\leq)}{\leq} \frac{k}{n}\}$  is enumerable. A measure  $\mu$  is called enumerable if it is enumerable and co-enumerable and the set  $\{x \mid \mu(x) = 0\}$  is decidable (i. e. enumerable and co-enumerable).

**Prefix Kolmogorov Complexity.** The conditional prefix complexity  $K(y|x) := \min\{\ell(p) : U(p,x) = y\}$  is the length of the shortest binary (self-delimiting) program  $p \in \{0,1\}^*$  on a universal prefix Turing machine  $U$  with output  $y \in \mathcal{X}^*$  and input  $x \in \mathcal{X}^*$  [LV97].  $K(x) := K(x|\epsilon)$ . For non-string objects  $o$  we define  $K(o) := K(\langle o \rangle)$ , where  $\langle o \rangle \in \mathcal{X}^*$  is some standard code for  $o$ . In particular, if  $(f_i)_{i=1}^\infty$  is an enumeration of all (co-)enumerable functions, we define  $K(f_i) := K(i)$ . We need the following properties: The co-enumerability of  $K$ , the upper bounds  $K(x|\ell(x)) \overset{\leq}{\leq} \ell(x) \log_2 |\mathcal{X}|$  and  $K(n) \overset{\leq}{\leq} 2 \log_2 n$ , Kraft’s inequality  $\sum_x 2^{-K(x)} \leq 1$ , the lower bound  $K(x) \geq l(x)$  for “most”  $x$  (which implies  $K(n) \overset{n \rightarrow \infty}{\rightarrow} \infty$ ), extra information bounds  $K(x|y) \overset{\leq}{\leq} K(x) \overset{\leq}{\leq} K(x,y)$ , sub-additivity  $K(xy) \overset{\leq}{\leq} K(x,y) \overset{\leq}{\leq} K(y) + K(x|y)$ , information non-increase  $K(f(x)) \overset{\leq}{\leq} K(x) + K(f)$  for computable  $f : \mathcal{X}^* \rightarrow \mathcal{X}^*$ , and coding relative to a probability distribution (MDL): if  $P : \mathcal{X}^* \rightarrow [0,1]$  is enumerable and  $\sum_x P(x) \leq 1$ , then  $K(x) \overset{\leq}{\leq} -\log_2 P(x) + K(P)$ .

**Monotone and Solomonoff Complexity.** The monotone complexity  $Km(x) := \min\{\ell(p) : U(p) = x*\}$  is the length of the shortest binary (possibly non-halting) program  $p \in \{0,1\}^*$  on a universal monotone Turing machine  $U$  which outputs a string starting with  $x$ . Solomonoff’s prior  $M(x) := \sum_{p:U(p)=x*} 2^{-\ell(p)} =: 2^{-Km(x)}$  is the probability that  $U$  outputs a string starting with  $x$  if provided

with fair coin flips on the input tape. Most complexities coincide within an additive term  $O(\log \ell(x))$ , e.g.  $K(x|\ell(x)) \stackrel{\pm}{\leq} KM(x) \leq Km(x) \leq K(x)$ , hence similar relations as for  $K$  hold.

### 3 Setup

**Convergent Predictors.** We assume that  $\mu$  is a “true”<sup>1</sup> sequence generating measure, also called environment. If we know the generating process  $\mu$ , and given past data  $x_{<t}$ , we can predict the probability  $\mu(x_t|x_{<t})$  of the next data item  $x_t$ . Usually we do not know  $\mu$ , but estimate it from  $x_{<t}$ . Let  $\rho(x_t|x_{<t})$  be an estimated probability<sup>2</sup> of  $x_t$ , given  $x_{<t}$ . Closeness of  $\rho(x_t|x_{<t})$  to  $\mu(x_t|x_{<t})$  is desirable as a goal in itself or when performing a Bayes decision  $y_t$  that has minimal  $\rho$ -expected loss  $l_t^\rho(x_{<t}) := \min_{y_t} \sum_{x_t} \text{Loss}(x_t, y_t) \rho(x_t|x_{<t})$ . Consider, for instance, a weather data sequence  $x_{1:n}$  with  $x_t = 1$  meaning rain and  $x_t = 0$  meaning sun at day  $t$ . Given  $x_{<t}$  the probability of rain tomorrow is  $\mu(1|x_{<t})$ . A weather forecaster may announce the probability of rain to be  $y_t := \rho(1|x_{<t})$ , which should be close to the true probability  $\mu(1|x_{<t})$ . To aim for

$$\rho(x'_t|x_{<t}) - \mu(x'_t|x_{<t}) \xrightarrow{(fast)} 0 \quad \text{for } t \rightarrow \infty$$

seems reasonable.

**Convergence in Mean Sum.** We can quantify the deviation of  $\rho_t$  from  $\mu_t$ , e.g. by the squared difference

$$s_t(\omega_{<t}) := \sum_{x_t \in \mathcal{X}} (\rho(x_t|\omega_{<t}) - \mu(x_t|\omega_{<t}))^2 \equiv \sum_{x_t} (\rho_t - \mu_t)^2$$

Alternatively one may also use the squared absolute distance  $s_t := \frac{1}{2}(\sum_{x_t} |\rho_t - \mu_t|)^2$ , the Hellinger distance  $s_t := \sum_{x_t} (\sqrt{\rho_t} - \sqrt{\mu_t})^2$ , the KL-divergence  $s_t := \sum_{x_t} \mu_t \ln \frac{\mu_t}{\rho_t}$ , or the squared Bayes regret  $s_t := \frac{1}{2}(l_t^\rho - l_t^\mu)^2$  for  $l_t \in [0, 1]$ . For all these distances one can show [Hut03a, Hut04] that their cumulative expectation from  $l$  to  $n$  is bounded as follows:

$$0 \leq \mathbf{E} \left[ \sum_{t=l}^n s_t | \omega_{<l} \right] \leq \mathbf{E} \left[ \ln \frac{\mu(\omega_{l:n} | \omega_{<l})}{\rho(\omega_{l:n} | \omega_{<l})} | \omega_{<l} \right] =: D_{l:n}(\omega_{<l}). \quad (1)$$

$D_{l:n}$  is increasing in  $n$ , hence  $D_{l:\infty} \in [0, \infty]$  exists [Hut01, Hut04]. A sequence of random variables like  $s_t$  is said to converge to zero with probability 1 if the set  $\{\omega : s_t(\omega) \xrightarrow{t \rightarrow \infty} 0\}$  has measure 1.  $s_t$  is said to converge to zero in mean sum if  $\sum_{t=1}^{\infty} \mathbf{E}[|s_t|] \leq c < \infty$ , which implies convergence with probability 1 (rapid if  $c$  is of reasonable size). Therefore a small finite bound on  $D_{1:\infty}$  would imply rapid convergence of the  $s_t$  defined above to zero, hence  $\rho_t \rightarrow \mu_t$  and  $l_t^\rho \rightarrow l_t^\mu$  fast. So the crucial quantities to consider and bound (in expectation) are  $\ln \frac{\mu(x)}{\rho(x)}$  if  $l=1$  and  $\ln \frac{\mu(y|x)}{\rho(y|x)}$  for  $l>1$ . For illustration we will sometimes loosely interpret  $D_{1:\infty}$

<sup>1</sup> Also called *objective* or *aleatory* probability or *chance*.

<sup>2</sup> Also called *subjective* or *belief* or *epistemic* probability.

and other quantities as the number of prediction errors, as for the error-loss they are closely related to it [Hut01].

**Bayes Mixtures.** A Bayesian considers a class of distributions  $\mathcal{M} := \{\nu_1, \nu_2, \dots\}$ , large enough to contain  $\mu$ , and uses the Bayes mixture

$$\xi(x) := \sum_{\nu \in \mathcal{M}} w_\nu \cdot \nu(x), \quad \sum_{\nu \in \mathcal{M}} w_\nu = 1, \quad w_\nu > 0. \tag{2}$$

for prediction, where  $w_\nu$  can be interpreted as the prior of (or initial belief in)  $\nu$ . The dominance

$$\xi(x) \geq w_\mu \cdot \mu(x) \quad \forall x \in \mathcal{X}^* \tag{3}$$

is its most important property. Using  $\rho = \xi$  for prediction, this implies  $D_{1:\infty} \leq \ln w_\mu^{-1} < \infty$ , hence  $\xi_t \rightarrow \mu_t$ . If  $\mathcal{M}$  is chosen sufficiently large, then  $\mu \in \mathcal{M}$  is not a serious constraint.

**Solomonoff Prior.** So we consider the largest (from a computational point of view) relevant class, the class  $\mathcal{M}_U$  of all enumerable semimeasures (which includes all computable probability distributions) and choose  $w_\nu = 2^{-K(\nu)}$  which is biased towards simple environments (Occam’s razor). This gives us Solomonoff-Levin’s prior  $M$  [Sol64, ZL70] (this definition coincides within an irrelevant multiplicative constant with the one in Section 2). In the following we assume  $\mathcal{M} = \mathcal{M}_U$ ,  $\rho = \xi = M$ ,  $w_\nu = 2^{-K(\nu)}$  and  $\mu \in \mathcal{M}_U$  being a computable (proper) measure, hence  $M(x) \geq 2^{-K(\mu)} \mu(x) \forall x$  by (3).

**Prediction of Deterministic Environments.** Consider a computable sequence  $\alpha = \alpha_{1:\infty}$  “sampled from  $\mu \in \mathcal{M}$ ” with  $\mu(\alpha) = 1$ , i.e.  $\mu$  is deterministic, then from (3) we get

$$\sum_{t=1}^{\infty} |1 - M(\alpha_t | \alpha_{<t})| \leq - \sum_{t=1}^{\infty} \ln M(\alpha_t | \alpha_{<t}) = - \ln M(\alpha_{1:\infty}) \leq K(\mu) \ln 2 < \infty, \tag{4}$$

which implies that  $M(\alpha_t | \alpha_{<t})$  converges rapidly to 1 and hence  $M(\bar{\alpha}_t | \alpha_{<t}) \rightarrow 0$ , i.e. asymptotically  $M$  correctly predicts the next symbol. The number of prediction errors is of the order of the complexity  $K(\mu) \pm Km(\alpha)$  of the sequence.

For binary alphabet this is the best we can expect, since at each time-step only a single bit can be learned about the environment, and only after we “know” the environment we can predict correctly. For non-binary alphabet,  $K(\mu)$  still measures the information in  $\mu$  in bits, but feedback per step can now be  $\log_2 |\mathcal{X}|$  bits, so we may expect a better bound  $K(\mu) / \log_2 |\mathcal{X}|$ . But in the worst case all  $\alpha_t \in \{0,1\} \subseteq \mathcal{X}$ . So without structural assumptions on  $\mu$  the bound cannot be improved even if  $\mathcal{X}$  is huge. We will see how our posterior bounds can help in this situation.

**Individual Randomness (Deficiency).** Let us now consider a general (not necessarily deterministic) computable measure  $\mu \in \mathcal{M}$ . The Shannon-Fano code of  $x$  w.r.t.  $\mu$  has code-length  $\lceil -\log_2 \mu(x) \rceil$ , which is “optimal” for “typical/random”  $x$  sampled from  $\mu$ . Further,  $-\log_2 M(x) \approx K(x)$  is the length of an “optimal” code

for  $x$ . Hence  $-\log_2 \mu(x) \approx -\log_2 M(x)$  for “ $\mu$ -typical/random”  $x$ . This motivates the definition of  $d_\mu(x)$

$$d_\mu(x) := \log_2 \frac{M(x)}{\mu(x)}$$

which is small for “typical/random”  $x$ . Formally, a sequence  $\alpha$  is called (Martin-Löf) random iff  $d_\mu(\alpha) := \sup_n d_\mu(\alpha_{1:n}) < \infty$ , i.e. iff its Shannon-Fano code is “optimal” (note that  $d_\mu(\alpha) \geq -K(\mu) > -\infty$  for all sequences), i.e. iff

$$\sup_n \left| \sum_{t=1}^n \log \frac{\mu(\alpha_t | \alpha_{<t})}{M(\alpha_t | \alpha_{<t})} \right| \equiv \sup_n \left| \log \frac{\mu(\alpha_{1:n})}{M(\alpha_{1:n})} \right| < \infty.$$

Unfortunately this does not imply  $M_t \rightarrow \mu_t$  on the  $\mu$ -random  $\alpha$ , since  $M_t$  may oscillate around  $\mu_t$ , which indeed can happen [HM04]. But if we take the expectation, Solomonoff [Sol78, Hut01, Hut04] showed

$$0 \leq \sum_{t=1}^{\infty} \mathbf{E} \sum_{x_t} (M_t - \mu_t)^2 \leq D_{1:\infty} = \lim_{n \rightarrow \infty} \mathbf{E}[-d_\mu(\omega_{1:n})] \ln 2 \leq K(\mu) \ln 2 < \infty \tag{5}$$

hence,  $M_t \rightarrow \mu_t$  with  $\mu$ -probability 1. So in any case,  $d_\mu(x)$  is an important quantity, since the smaller  $-d_\mu(x)$  (at least in expectation) the better  $M$  predicts.

### 4 Posterior Bounds

**Posterior Bounds.** Both bounds, (4) and (5) bound the total (cumulative) discrepancy (error) between  $M_t$  and  $\mu_t$ . Since the discrepancy sum  $D_{1:\infty}$  is finite, we know that after sufficiently long time  $t=l$ , we will make little further errors, i.e. the future error sum  $D_{l:\infty}$  is small. The main goal of this paper is to quantify this asymptotic statement. So we need bounds on  $\log_2 \frac{\mu(y|x)}{M(y|x)}$ , where  $x$  are past and  $y$  are future observations. Since  $\log_2 \frac{\mu(y)}{M(y)} \leq K(\mu)$  and  $\mu(y|x)/M(y|x)$  are conditional versions of true/universal distributions, it seems natural that the unconditional bound  $K(\mu)$  also simply conditionalizes to  $\log_2 \frac{\mu(y|x)}{M(y|x)} \stackrel{?}{\leq} K(\mu|x)$ . The more information the past observation  $x$  contains about  $\mu$ , the easier it is to code  $\mu$  i.e. the smaller is  $K(\mu|x)$ , and hence the less future predictions errors  $D_{l:\infty}$  we should make. Once  $x$  contains all information about  $\mu$ , i.e.  $K(\mu|x) \stackrel{\pm}{=} 0$ , we should make no errors anymore. More formally, optimally coding  $x$  then  $\mu|x$  and finally  $y|\mu,x$  by Shannon-Fano, gives a code for  $xy$ , hence  $K(xy) \lesssim K(x) + K(\mu|x) + \log_2 \mu(y|x)^{-1}$ . Since  $K(z) \approx -\log_2 M(z)$  this implies  $\log_2 \frac{\mu(y|x)}{M(y|x)} \lesssim K(\mu|x)$ , but with logarithmic fudge that tends to infinity for  $\ell(y) \rightarrow \infty$ , which is unacceptable. The  $y$ -independent bound we need was first stated in [Hut04-Prob.2.6(iii)]:

**Theorem 1.** For any  $\mu$  and  $x, y \in \mathcal{X}^*$ ,

$$\log_2 \frac{\mu(y|x)}{M(y|x)} \stackrel{\pm}{\leq} K(\mu|x) + K(\ell(x)).$$

**Proof.** For any fixed  $l$  we define the following function of  $z \in \mathcal{X}^*$ . For  $\ell(z) \geq l$ ,

$$\psi_l(z) := \sum_{\nu \in \mathcal{M}} 2^{-K(\nu|z_{1:l})} M(z_{1:l}) \nu(z_{l+1:\ell(z)}).$$

For  $\ell(z) < l$  we extend  $\psi_l$  by defining  $\psi_l(z) := \sum_{u:\ell(u)=l-\ell(z)} \psi(zu)$ . It is easy to see that  $\psi_l$  is an enumerable semimeasure. By definition of  $M$ , we have  $M(z) \geq 2^{-K(\psi_l)} \psi_l(z)$  for any  $l$  and  $z$ . Now let  $l = \ell(x)$  and  $z = xy$ . Let us define a semimeasure  $\mu_x(y) := \mu(y|x)$ . Then

$$M(xy) \geq 2^{-K(\psi_l)} \psi_l(xy) \geq 2^{-K(\psi_l)} 2^{-K(\mu_x|x)} M(x) \mu_x(y).$$

Taking the logarithm, after trivial transformations, we get  $\log_2 \frac{\mu(y|x)}{M(y|x)} \leq K(\mu_x|x) + K(\psi_l)$ . To complete the proof, let us note that  $K(\psi_l) \stackrel{\pm}{\leq} K(l)$  and  $K(\mu_x|x) \stackrel{\pm}{\leq} K(\mu|x)$ . □

**Corollary 2.**  $\dots \dots \dots M_t \dots \dots \dots \mu_t \dots \dots \dots$

- i)  $\sum_{t=l+1}^{\infty} \mathbf{E}[s_t|\omega_{1:l}] \leq D_{l+1:\infty}(\omega_{1:l}) \stackrel{\pm}{\leq} (K(\mu|\omega_{1:l}) + K(l)) \ln 2$
- ii)  $\sum_{t=1}^{\infty} \mathbf{E}[s_t] \stackrel{\pm}{\leq} \min_l \{ \mathbf{E}[K(\mu|\omega_{1:l}) + K(l)] \ln 2 + 2l \}$

**Proof.** (i) The first inequality is (1) and the second follows by taking the conditional expectation  $\mathbf{E}[\cdot|\omega_{1:l}]$  in Theorem 1. (ii) follows from (i) by taking the unconditional expectation and from  $\sum_{t=1}^l \mathbf{E}[s_t] \leq 2l$ , since  $s_t \leq 2$ . □

**Examples and More Motivation.** The bounds Theorem 1 and Corollary 2(i) prove and quantify the intuition that the more we know about the environment, the better our predictions. We show the usefulness of the new bounds for some deterministic environments  $\mu \hat{=} \alpha$ .

Assume all observations are identical, i.e.  $\alpha = x_1 x_1 x_1 \dots$ . Further assume that  $\mathcal{X}$  is huge and  $K(x_1) = \log_2 |\mathcal{X}|$ , i.e.  $x_1$  is a typical/random/complex element of  $\mathcal{X}$ . For instance if  $x_1$  is a  $256^3$  color  $512 \times 512$  pixel image, then  $|\mathcal{X}| = 256^{3 \times 512 \times 512}$ . Hence the standard bound (5) on the number of errors  $D_{1:\infty} / \ln 2 \leq K(\mu) \stackrel{\pm}{\leq} K(x_1) = 3 \cdot 2^{21}$  is huge. Of course, interesting pictures are not purely random, but their complexity is often only a factor 10..100 less, so still large. On the other hand, any reasonable prediction scheme observing a few (rather than several thousands) identical images, should predict that the next image will be the same. This is what our posterior bound gives,  $D_{2:\infty}(x_1) \stackrel{\pm}{\leq} K(\mu|x_1) + K(1) \stackrel{\pm}{\leq} 0$ , hence indeed  $M$  makes only  $\sum_{t=1}^{\infty} \mathbf{E}[s_t] = O(1)$  errors by Corollary 2(ii), significantly improving upon Solomonoff's bound  $K(\mu) \ln 2$ .

More generally, assume  $\alpha = x\omega$ , where the initial part  $x = x_{1:l}$  contains all information about the remainder, i.e.  $K(\mu|x) \stackrel{\pm}{\leq} K(\omega|x) \stackrel{\pm}{\leq} 0$ . For instance,  $x$  may be a binary program for  $\pi$  or  $e$  and  $\omega$  be its  $|\mathcal{X}|$ -ary expansion. Sure, given the algorithm for some number sequence, it should be perfectly predictable. Indeed, Theorem 1 implies  $D_{l+1:\infty} \stackrel{\pm}{\leq} K(l)$ , which can be exponentially smaller than Solomonoff's bound  $K(\mu)$  ( $\stackrel{\pm}{\leq} l$  if  $K(x) \stackrel{\pm}{\leq} \ell(x)$ ). On the other hand,  $K(l) \geq \log_2 l$  for most  $l$ , i.e. is larger than  $O(1)$  what one might hope for.



**Logarithmic Versus Constant Accuracy.** So there is one blemish in the bound. There is an additive correction of logarithmic size in the length of  $x$ . Many theorems in algorithmic information theory hold to within an additive constant, sometimes this is easily reached, sometimes hard, sometimes one needs a suitable complexity variant, and sometimes the logarithmic accuracy cannot be improved [LV97]. The latter is the case with Theorem 1:

**Lemma 3.**  $\mathcal{X} = \{0,1\}^*$ ,  $\mu$  is a computable measure,  $\alpha \in \{0,1\}^\infty$ ,  $l \in \mathbb{N}$

$$D_{l:\infty}(\alpha_{<l}) \geq D_{l:l}(\alpha_{<l}) \equiv \sum_{b \in \{0,1\}} \mu(b|\alpha_{<l}) \ln \frac{\mu(b|\alpha_{<l})}{M(b|\alpha_{<l})} \stackrel{\pm}{\geq} \frac{1}{3}K(l).$$

**Proof.** Let us construct a computable sequence  $\alpha \in \{0,1\}^\infty$  by induction. Assume that  $\alpha_{<l}$  is constructed. Since  $\mu$  is a measure, either  $\mu(0|\alpha_{<l}) > c$  or  $\mu(1|\alpha_{<l}) > c$  for  $c := [3\ln 2]^{-1} < \frac{1}{2}$ . Since  $\mu$  is computable, we can find (effectively)  $b \in \{0,1\}$  such that  $\mu(b|\alpha_{<l}) > c$ . Put  $\alpha_l = \bar{b}$ .

Let us estimate  $M(\bar{\alpha}_l|\alpha_{<l})$ . Since  $\alpha$  is computable,  $M(\alpha_{<l}) \stackrel{\pm}{\leq} 1$ . We claim that  $M(\alpha_{<l}\bar{\alpha}_l) \stackrel{\pm}{\leq} 2^{-K(l)}$ . Actually, consider the set  $\{\alpha_{<l}\bar{\alpha}_l \mid l > 0\}$ . This set is prefix free and decidable. Therefore  $P(l) = M(\alpha_{<l}\bar{\alpha}_l)$  is an enumerable function with  $\sum_l P(l) \leq 1$ , and the claim follows from the coding theorem. Thus, we have  $M(\bar{\alpha}_l|\alpha_{<l}) \stackrel{\pm}{\leq} 2^{-K(l)}$  for any  $l$ . Since  $\mu(\bar{\alpha}_l|\alpha_{<l}) > c$ , we get

$$\begin{aligned} \sum_{b \in \{0,1\}} \mu(b|\alpha_{<l}) \ln \frac{\mu(b|\alpha_{<l})}{M(b|\alpha_{<l})} &\stackrel{\pm}{\geq} \mu(\bar{\alpha}_l|\alpha_{<l}) \ln \frac{c}{2^{-K(l)}} + \min_{p \in [0,1-c]} p \ln \frac{p}{M(\alpha_l|\alpha_{<l})} \\ &\stackrel{\pm}{\geq} cK(l) \ln 2 \end{aligned} \quad \square$$

A constant fudge is generally preferable to a logarithmic one for quantitative and aesthetical reasons. It also often leads to particular insight and/or interesting new complexity variants (which will be the case here). Though most complexity variants coincide within logarithmic accuracy (see [Sch00, Sch02a] for exceptions), they can have very different other properties. For instance, Solomonoff complexity  $KM(x) = -\log_2 M(x)$  is an excellent predictor, but monotone complexity  $Km$  can be exponentially worse and prefix complexity  $K$  fails completely [Hut03c].

**Exponential Bounds.** Bayes is often approximated by MAP or MDL. In our context this means approximating  $KM$  by  $Km$  with exponentially worse bounds (in deterministic environments) [Hut03c]. (Intuitively, since an error with Bayes eliminates half of the environments, while MAP/MDL may eliminate only one.) Also for more complex “reinforcement” learning problems, bounds can be  $2^{K(\mu)}$  rather than  $K(\mu)$  due to sparser feedback. For instance, for a sequence  $x_1x_1x_1\dots$  if we do not observe  $x_1$  but only receive a reward if our prediction was correct, then the only way a universal predictor can find  $x_1$  is by trying out all  $|\mathcal{X}|$  possibilities and making (in the worst case)  $|\mathcal{X}|-1 \stackrel{\pm}{\leq} 2^{K(\mu)}$  errors. Posterization allows to boost such gross bounds to useful bounds  $2^{K(\mu|x_1)} = O(1)$ . But in

general, additive logarithmic corrections as in Theorem 1 also exponentiate and lead to bounds polynomial in  $l$  which may be quite sizeable. Here the advantage of a constant correction becomes even more apparent [Hut04–Problems 2.6, 3.13, 6.3 and Section 5.3.3].

### 5 More Bounds and New Complexity Measure

Lemma 3 shows that the bound in Theorem 1 is attained for some binary strings. But for other binary strings the bound may be very rough. (Similarly,  $K(x)$  is greater than  $\ell(x)$  infinitely often, but  $K(x) \ll \ell(x)$  for many ‘interesting’  $x$ .) Let us try to find a new bound, which does not depend on  $\ell(x)$ .

First observe that, in contrast to the unconditional case (5),  $K(\mu)$  is not an upper bound (again by Lemma 3). Informally speaking, the reason is that  $M$  can predict the future very badly if the past is not ‘typical’ for the environment (such past  $x$  have low  $\mu$ -probability, therefore in the unconditional case their contribution to the expected loss is small). So, it is natural to bound the loss in terms of randomness deficiency  $d_\mu(x)$ , which is a quantitative measure of ‘typicalness’.

**Theorem 4.** For any computable measure  $\mu$  and any  $x, y \in \{0,1\}^*$ ,

$$\log_2 \frac{\mu(y|x)}{M(y|x)} \equiv d_\mu(x) - d_\mu(xy) \stackrel{\pm}{\leq} K(\mu) + K(\lceil d_\mu(x) \rceil).$$

Theorem 4 is a variant of the ‘deficiency conservation theorem’ from [VSU05]. We do not know who was the first to discover this statement and whether it was published (the special case where  $\mu$  is the uniform measure was proved by An. Muchnik as an auxiliary lemma for one of his unpublished results; then A. Shen placed a generalized statement to the (unfinished) book [VSU05]).

Now, our goal is to replace  $K(\mu)$  in the last bound by a conditional complexity of  $\mu$ . Unfortunately, the conventional conditional prefix complexity is not suitable:

**Lemma 5.** For any computable measure  $\mu$  and any  $C_0, l \in \mathbb{N}$ ,

$$K(\mu|x) \leq C_0, \quad d_\mu(x) \leq C_0,$$

$$D_{l+1:l+1}(x) \equiv \sum_{b \in \{0,1\}} \mu(b|x) \ln \frac{\mu(b|x)}{M(b|x)} \stackrel{\pm}{\leq} K(l) \ln 2.$$

**Proof.** For  $l \in \mathbb{N}$ , define a deterministic measure  $\mu_l$  such that  $\mu_l$  is equal to 1 on the prefixes of  $0^l 1^\infty$  and is equal to 0 otherwise.

Let  $x = 0^l$ . Then  $\mu_l(x) = 1$ ,  $\mu_l(x0) = 0$ ,  $\mu_l(x1) = 1$ . Also  $1 \geq M(x) \geq M(x0) \geq M(0^\infty) \cong 1$  and (as in the proof of Lemma 3)  $M(x1) \stackrel{\leq}{\leq} 2^{-K(l)}$ . Trivially,  $d_{\mu_l}(x) = \log_2 M(x) \cong 1$ , and  $K(\mu_l|x) \stackrel{\pm}{\leq} K(\mu_l|l) \stackrel{\pm}{\leq} 0$ . Thus,  $K(\mu_l|x)$  and  $d_{\mu_l}(x)$  are bounded by a constant  $C_0$  independent of  $l$ . On the other hand,  $\sum_{b \in \{0,1\}} \mu(b|x) \ln \frac{\mu(b|x)}{M(b|x)} = \ln \frac{1}{M(1|x)} \stackrel{\pm}{\leq} K(l) \ln 2$ . (One can obtain the same result also for non-deterministic  $\mu$ , for example, taking  $\mu_l$  mixed with the uniform measure.)  $\square$

Informally speaking, in Lemma 5 we exploit the fact that  $K(y|x)$  can use the information about the length of the condition  $x$ . Hence  $K(y|x)$  can be small for a certain  $x$  and is large for some (actually almost all) prolongations of  $x$ . But in our case of sequence prediction, the length of  $x$  grows taking all intermediate values and cannot contain any relevant information. Thus we need a new kind of conditional complexity.

Consider a Turing machine  $T$  with two input tapes. Inputs are provided without delimiters, so the size of input is defined by the machine itself. Let us call such a machine  $\mathcal{M}$ . We write that  $T(x,y) = z$  if machine  $T$ , given a sequence beginning with  $x$  on the first tape and a sequence beginning with  $y$  on the second tape, halts after reading exactly  $x$  and  $y$  and prints  $z$  to the output tape. (Obviously, if  $T(x,y) = z$ , then the computation does not depend on the contents of the input tapes after  $x$  and  $y$ .) We define  $C_T(y|x) := \min\{\ell(p) \mid \exists k \leq \ell(x) : T(p, x_{1:k}) = y\}$ . Clearly,  $C_T(y|x)$  is an enumerable from above function of  $T$ ,  $x$ , and  $y$ . Using a standard argument [LV97], one can show that there exists an optimal twice prefix machine  $U$  in the sense that for any twice prefix machine  $T$  we have  $C_U(y|x) \leq C_T(y|x)$ .

**Definition 6.** Complexity monotone in conditions,  $U$

$$K_*(y|x^*) := C_U(y|x) = \min\{\ell(p) \mid \exists k \leq \ell(x) : U(p, x_{1:k}) = y\}.$$

\*  $x^*$   $K_*(y|x^*)$   $y$   $z = x^*$

**Theorem 7.**  $\mu$   $x, y \in \mathcal{X}^*$

$$\log_2 \frac{\mu(y|x)}{M(y|x)} \leq K_*(\mu|x^*) + K(\lceil d_\mu(x) \rceil).$$

One can get a slightly stronger variants of Theorems 1 and 7 by replacing the complexity of a standard code of  $\mu$  by more sophisticated values. First, in any effective encoding there are many codes for every  $\mu$ , and in all the upper bounds (including Solomonoff's one) one can take the minimum of the complexities of all the codes for  $\mu$ . Moreover, in Theorem 1 it is sufficient to take the complexity of  $\mu_x = \mu(\cdot|x)$  (and it is sufficient that  $\mu_x$  is enumerable, while  $\mu$  can be incomputable). For Theorem 7 one can prove a similar strengthening: The complexity of  $\mu$  is replaced by the complexity of any computable function that is equal to  $\mu$  on all prefixes and prolongations of  $x$ .

To demonstrate the usefulness of the new bound, let us again consider some deterministic environment  $\mu \hat{=} \alpha$ . For  $\mathcal{X} = \{0,1\}$  and  $\alpha = x^\infty$  with  $x = 0^n 1$ , Theorem 1 gives the bound  $K(\mu|n) + K(n) \leq K(n)$ . Consider the new bound  $K_*(\mu|x^*) + K(\lceil d_\mu(x) \rceil)$ . Since  $\mu$  is deterministic, we have  $d_\mu(x) = \log_2 M(x) \leq -K(n)$ , and  $K(\lceil d_\mu(x) \rceil) \leq K(K(n))$ . To estimate  $K_*(\mu|x^*)$ , let us consider a machine  $T$  that reads only its second tape and outputs the number of 0s before the first 1. Clearly,  $C_T(n|x) = 0$ , hence  $K_*(\mu|x^*) \leq 0$ . Finally,  $K_*(\mu|x^*) + K(\lceil d_\mu(x) \rceil) \leq K(K(n))$ , which is much smaller than  $K(n)$ .

## 6 Properties of the New Complexity

The above definition of  $K_*$  is based on computations of some Turing machine. Such definitions are quite visual, but are often not convenient for formal proofs. We will give an alternative definition in terms of enumerable sets (see [US96] for definitions of unconditional complexities in this style), which summarizes the properties we actually need for the proof of Theorem 7.

An enumerable set  $E$  of triples of strings is called  $K_*$ -correct if it satisfies the following requirements:

1. if  $\langle p,x,y_1 \rangle \in E$  and  $\langle p,x,y_2 \rangle \in E$ , then  $y_1 = y_2$ ;
2. if  $\langle p,x,y \rangle \in E$ , then  $\langle p',x',y \rangle \in E$  for all  $p'$  being prolongations of  $p$  and all  $x'$  being prolongations of  $x$ ;
3. if  $\langle p,x',y \rangle \in E$  and  $\langle p',x,y \rangle \in E$ , and  $p$  is a prefix of  $p'$  and  $x$  is a prefix of  $x'$ , then  $\langle p,x,y \rangle \in E$ .

A complexity of  $y$  under a condition  $x$  w.r.t. a set  $E$  is  $C_E(y|x) = \min\{\ell(p) \mid \langle p,x,y \rangle \in E\}$ . A  $K_*$ -correct set  $E$  is called  $K_*$ -optimal if  $C_E(y|x) \leq C_{E'}(y|x)$  for any  $K_*$ -correct set  $E'$ . One can easily construct an enumeration of all  $K_*$ -correct sets, and an optimal set exists by the standard argument.

It is easy to see that a twice prefix Turing machine  $T$  can be transformed to a set  $E$  such that  $C_T(y|x) = C_E(y|x)$ . The set  $E$  is constructed as follows:  $T$  is run on all possible inputs, and if  $T(p,x) = y$ , then pairs  $\langle p',x',y \rangle$  are added to  $E$  for all  $p'$  being prolongations of  $p$  and all  $x'$  being prolongations of  $x$ . Evidently,  $E$  is enumerable, and the second requirement of  $K_*$ -correctness is satisfied. To verify the other requirements, let us consider arbitrary  $\langle p'_1,x'_1,y_1 \rangle \in E$  and  $\langle p'_2,x'_2,y_2 \rangle \in E$  such that  $p'_1$  and  $p'_2$ ,  $x'_1$  and  $x'_2$  are comparable (one is a prefix of the other). Then, by construction of  $E$ , we have  $T(p_1,x_1) = y_1$  and  $T(p_2,x_2) = y_2$ , and  $p_1$  and  $p_2$ ,  $x_1$  and  $x_2$  are comparable too. Since replacing the unused part of the inputs does not affect the running of the machine  $T$  and comparable words have a common prolongation, we get  $p_1 = p_2$ ,  $x_1 = x_2$ , and  $y_1 = y_2$ . Thus  $E$  is a  $K_*$ -correct set.

The transformation in the other direction is impossible in some cases: the set  $E = \{\langle 0^{h(n)}p, 0^n 1q, 0 \rangle \mid n \in \mathbb{N}, p, q \in \{0,1\}^*\}$ , where  $h(n)$  is 0 if the  $n$ -th Turing machine halts and 1 otherwise, is  $K_*$ -correct, but does not have a corresponding machine  $T$ : using such a machine one could solve the halting problem. However, we conjecture that for every set  $E$  there exists a machine  $T$  such that  $C_T(x|y) \leq C_E(x|y)$ .

Probably, the requirements on  $E$  can be even weaker, namely, the third requirement can be superfluous. Let us notice that the first requirement of  $K_*$ -correctness allows us to consider the set  $E$  as a partial computable function:  $E(p,x) = y$  iff  $\langle p,x,y \rangle \in E$ . The second requirement says that  $E$  becomes a continuous function if we take the topology of prolongations (any neighborhood of  $\langle p,x \rangle$  contains the cone  $\{\langle p*,x* \rangle\}$ ) on the arguments and the discrete topology  $\{\{y\}\}$  is a neighborhood of  $y$  on values. It is known (see [US96] for references) that different complexities (plain, prefix, decision) can be naturally defined in a similar “topological” fashion. We conjecture the same is true in our case: an optimal enumerable set satisfying the requirements (1) and (2) (obviously, it

exists) specifies the same complexity (up to an additive constant) as an optimal twice prefix machine.

It follows immediately from the definition(s) that  $K_*(y|x^*)$  is monotone as a function of  $x$ :  $K_*(y|xz^*) \leq K_*(y|x^*)$  for all  $x, y, z$ .

The following lemma provides bounds for  $K_*(x|y^*)$  in terms of prefix complexity  $K$ . The lemma holds for all our definitions of  $K_*(x|y^*)$ .

**Lemma 8.**  $x, y \in \mathcal{X}^*, \ell(y) < \ell(x)$

$$K(x|y) \stackrel{\pm}{\leq} K_*(x|y^*) \stackrel{\pm}{\leq} \min_{l \leq \ell(y)} \{K(x|y_{1:l}) + K(l)\} \stackrel{\pm}{\leq} K(x).$$

*Proof.*  $K_*(x|y^*) \stackrel{\pm}{\leq} K(x|y^*) \stackrel{\pm}{\leq} K(x) + o(K(x))$ .  
 $K(x|y) \stackrel{\pm}{\leq} K(x|y_{1:l}) + K(l) \stackrel{\pm}{\leq} K(x|y^*) + K(l) \stackrel{\pm}{\leq} K(x) + o(K(x)) + K(l)$ .

$$K(x|y) \stackrel{\pm}{\leq} 0 \stackrel{\pm}{\leq} K_*(x|y^*) \stackrel{\pm}{\leq} K(x) \stackrel{\pm}{\leq} \min_{l \leq \ell(y)} \{K(x|y_{1:l}) + K(l)\},$$

*Proof.*  $x \in \mathcal{X}^*, y \in \mathcal{X}^*$

$$K(x|y) \stackrel{\pm}{\leq} K_*(x|y^*) \stackrel{\pm}{\leq} 0 \stackrel{\pm}{\leq} K(x) \stackrel{\pm}{\leq} \min_{l \leq \ell(y)} \{K(x|y_{1:l}) + K(l)\}.$$

**Corollary 9.**  $M_t \leq \mu_t, t \in \mathbb{N}$

$$\sum_{t=l+1}^{\infty} \mathbf{E}[s_t|\omega_{1:l}] \stackrel{\pm}{\leq} [\min_{i \leq l} \{K(\mu|\omega_{1:i}) + K(i)\} + K(d_\mu(\omega_{1:l}))] \ln 2.$$

Let us note that if  $\omega$  is  $\mu$ -random, then  $K(d_\mu(\omega_{1:l})) \stackrel{\pm}{\leq} K(d_\mu(\omega_{1:\infty})) + K(K(\mu))$ , and therefore we get the bound, which does not increase with  $l$ , in contrast to the bound (i) in Corollary 2.

## 7 Proof of Theorem 7

The plan is to get a statement of the form  $2^d \mu(y) \stackrel{\pm}{\leq} M(y)$ , where  $d \approx d_\mu(x) = \log_2 \frac{M(x)}{\mu(x)}$ . To this end, we define a new semimeasure  $\nu$ : we take the set  $S = \{z | d_\mu(z) > d\}$  and put  $\nu$  to be  $2^d \mu$  on prolongations of  $z \in S$ ; this is possible since  $S$  has  $\mu$ -measure  $2^{-d}$ . Then we have  $\nu(z) \leq C \cdot M(z)$  by universality of  $M$ . However, the constant  $C$  depends on  $\mu$  and also on  $d$ . To make the dependence explicit, we repeat the above construction for all numbers  $d$  and all semimeasures  $\mu^T$ , obtaining semimeasures  $\nu_{d,T}$ , and take  $\nu = \sum 2^{-K(d)} \cdot 2^{-K(T)} \nu_{d,T}$ . This construction would give us the term  $K(\mu)$  in the right-hand side of Theorem 7. To get  $K_*(\mu|x^*)$ , we need a more complicated strategy: instead of a sum of semimeasures  $\nu_{d,T}$ , for every fixed  $d$  we sum “pieces” of  $\nu_{d,T}$  at each point  $z$ , with coefficients depending on  $z$  and  $T$ .

Now proceed with the formal proof. Let  $\{\mu^T\}_{T \in \mathbb{N}}$  be any (effective) enumeration of all enumerable semimeasures. For any integer  $d$  and any  $T$ , put

$$S_{d,T} := \{z | \sum_{v \in \mathcal{X}^{\ell(z)} \setminus \{z\}} \mu^T(v) + 2^{-d} M(z) > 1\}.$$

The set  $S_{d,T}$  is enumerable given  $d$  and  $T$ .

Let  $E$  be the optimal  $K_*$ -correct set (satisfying all three requirements),  $E(p, z)$  is the corresponding partial computable function. For any  $z \in \mathcal{X}^*$  and  $T$ , put

$$\lambda(z, T) := \max\{2^{-\ell(p)} \mid \exists k \leq \ell(z): z_{1:k} \in S_{d,T} \text{ and } E(p, z_{1:k}) = T\}$$

(if there is no such  $p$ , then  $\lambda(z, T) = 0$ ). Put

$$\tilde{\nu}_d(z) := \sum_T \lambda(z, T) \cdot 2^d \mu^T(z).$$

Obviously, this value is enumerable. It is not a semimeasure, but it has the following property (we omit the proof).

••• For any prefix-free set  $A$ ,

$$\sum_{z \in A} \tilde{\nu}_d(z) \leq 1.$$

This implies that there exists an enumerable semimeasure  $\nu_d$  such that  $\nu_d(z) \geq \tilde{\nu}_d(z)$  for all  $z$ . Actually, to enumerate  $\nu_d$ , one enumerates  $\tilde{\nu}_d(z)$  for all  $z$  and at each step sets the current value of  $\nu_d(z)$  to the maximum of the current values of  $\tilde{\nu}_d(z)$  and  $\sum_{u \in \mathcal{X}} \nu_d(zu)$ . Trivially, this provides  $\nu_d(z) \geq \sum_{u \in \mathcal{X}} \nu_d(zu)$ . To show that  $\nu_d(\epsilon) \leq 1$ , let us note that at any step of enumeration the current value of  $\nu_d(\epsilon)$  is the sum of current values  $\tilde{\nu}_d(z)$  over some prefix-free set, and thus is bounded by 1. Put

$$\nu(z) := \sum_d 2^{-K(d)} \nu_d(z).$$

Clearly,  $\nu$  is an enumerable semimeasure, thus  $\nu(z) \stackrel{\times}{\leq} M(z)$ . Let  $\mu$  be an arbitrary computable measure, and  $x, y \in \mathcal{X}^*$ . Let  $p \in \{0,1\}^*$  be a string such that  $K_*(\mu|x^*) = \ell(p)$ ,  $E(p, x) = T$ , and  $\mu = \mu^T$ . Put  $d = \lceil d_\mu(x) \rceil - 1$ , i.e.,  $d_\mu(x) - 1 \leq d < d_\mu(x)$ . Hence  $\mu(x) < 2^{-d} M(x)$ . Since  $\mu = \mu^T$  is a measure, we have  $\sum_{v \in \mathcal{X}^{\ell(x)}} \mu^T(v) = 1$ , and therefore  $x \in S_{d,T}$ . By definition,  $\lambda(xy, T) \geq 2^{-\ell(p)}$ , thus  $\tilde{\nu}_d(xy) \geq 2^{-\ell(p)} 2^d \mu(xy)$ , and

$$2^{-K(d)} 2^{-\ell(p)} 2^d \mu(xy) \leq \nu(xy) \stackrel{\times}{\leq} M(xy).$$

After trivial transformations we get

$$\log_2 \frac{\mu(y|x)}{M(y|x)} \stackrel{\pm}{\leq} K_*(\mu|x^*) + K(d),$$

which completes the proof of Theorem 7.

## 8 Discussion

**Conclusion.** We evaluated the quality of predicting a stochastic sequence at an intermediate time, when some beginning of the sequence has been already

observed, estimating the future loss of the universal Solomonoff predictor  $M$ . We proved general upper bounds for the discrepancy between conditional values of the predictor  $M$  and the true environment  $\mu$ , and demonstrated a kind of tightness for these bounds. One of the bounds is based on a new variant of conditional algorithmic complexity  $K_*$ , which has interesting properties in its own. In contrast to standard prefix complexity  $K$ ,  $K_*$  is a monotone function of conditions:  $K_*(y|xyz^*) \leq K_*(y|x^*)$ .

**General Bayesian Posterior Bounds.** A natural question is whether posterior bounds for general Bayes mixtures based on general  $\mathcal{M} \ni \mu$  could also be derived. From the (obvious) posterior representation  $\xi(y|x) = \sum_{\nu \in \mathcal{M}} w_\nu(x) \nu(y|x) \geq w_\mu(x) \mu(y|x)$ , where  $w_\nu(x) := w_\nu \frac{\nu(x)}{\xi(x)}$  is the posterior belief in  $\nu$  after observing  $x$ , the bound  $D_{l:\infty} \leq \ln w_\mu(\omega_{<l})^{-1}$  immediately follows. Strangely enough, for  $\mathcal{M} = \mathcal{M}_U$ ,  $\log_2 w_\nu^{-1} := K(\nu)$  does not imply  $\log_2 w_\mu(x)^{-1} = K(\mu|x)$ , not even within logarithmic accuracy, so it was essential to consider  $D_{l:\infty}$ . It would be interesting to derive bounds on  $D_{l:\infty}$  or  $\ln w_\mu(x)^{-1}$  for general  $\mathcal{M}$  similar to the ones derived here for  $\mathcal{M} = \mathcal{M}_U$ .

**Online Classification.** All considered distributions  $\rho(x)$  (in particular  $\xi$ ,  $M$ , and  $\mu$ ), may be replaced everywhere by distributions  $\rho(x|z)$  additionally conditioned on some  $z$ . The  $z$ -conditions cause nowhere problems as they can essentially be thought of as fixed (or as oracles or spectators). An (i.i.d.) classification problem is a typical example: At time  $t$  one arranges an experiment  $z_t$  (or observes data  $z_t$ ), then tries to make a prediction, and finally observes the true outcome  $x_t$  with probability  $\mu(x_t|z_t)$ . In this case  $\mathcal{M} = \{\nu(x_{1:n}|z_{1:n}) = \nu(x_1|z_1) \cdot \dots \cdot \nu(x_n|z_n)\}$ . (Note that  $\xi$  is not i.i.d). Solomonoff's bound  $K(\mu) \ln 2$  (5) holds unchanged. Compared to the sequence prediction case we have extra information  $z$ , so we may wonder whether some improved bound  $K(\mu|z)$  or so, holds. For a fixed  $z$  this can be achieved by also replacing  $2^{-K(\mu)}$  in (2) by  $2^{-K(\mu|z)}$ . But if at time  $t$  only  $z_{1:t}$  is known like in the classification example, this leads to difficulties ( $\xi$  is no longer a (semi)measure, which sometimes can be corrected [PH04]). Alternatively we could keep definition (2) but apply it to the (chronologically correctly ordered) sequence  $z_1 x_1 z_2 x_2 \dots$ , condition to  $z_{1:t}$ , and try to derive improved bounds.

**More Open Problems.** Since  $D_{1:\infty}$  is finite, one may expect that the tails  $D_{l:\infty}$  tend to 0 as  $l \rightarrow \infty$ . However, as Lemma 3 implies, this holds only with probability 1: for some special  $\alpha$  we have even  $D_{l:\infty}(\alpha_{<l}) \stackrel{\pm}{\geq} \frac{1}{3} K(l) \xrightarrow{l \rightarrow \infty} \infty$ . It would be very interesting to find a wide class of  $\alpha$  such that  $D_{l:\infty}(\alpha_{<l}) \rightarrow 0$ . The natural conjecture is that one should take  $\mu$ -random  $\alpha$ . Another (probably, closely related) task is to study the asymptotic behavior of  $K_*(\mu|\alpha_{<l}^*)$ . It is natural to expect that  $K_*(\mu|\alpha_{<l}^*)$  is bounded by an absolute constant (independent of  $\mu$ ) for "most"  $\alpha$  and for sufficiently large  $l$ . Finally, (dis)proving equality of the various definitions of  $K_*$  we gave, would be useful.

## References

- [CV05] R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Trans. Information Theory*, 51(4):1523–1545, 2005.
- [HM04] M. Hutter and An. A. Muchnik. Universal convergence of semimeasures on individual random sequences. In *Proc. 15th International Conf. on Algorithmic Learning Theory (ALT'04)*, volume 3244 of *LNAI*, pages 234–248, Padova, 2004. Springer, Berlin.
- [Hut01] M. Hutter. Convergence and error bounds for universal prediction of non-binary sequences. *Proc. 12th European Conference on Machine Learning (ECML-2001)*, pages 239–250, December 2001.
- [Hut03a] M. Hutter. Convergence and loss bounds for Bayesian sequence prediction. *IEEE Trans. on Information Theory*, 49(8):2061–2067, 2003.
- [Hut03b] M. Hutter. Optimality of universal Bayesian prediction for general loss and alphabet. *Journal of Machine Learning Research*, 4:971–1000, 2003.
- [Hut03c] M. Hutter. Sequence prediction based on monotone complexity. In *Proc. 16th Annual Conference on Learning Theory (COLT'03)*, volume 2777 of *LNAI*, pages 506–521, Berlin, 2003. Springer.
- [Hut04] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2004. 300 pages, <http://www.idsia.ch/~marcus/ai/uaibook.htm>.
- [LV97] M. Li and P. M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2nd edition, 1997.
- [PH04] J. Poland and M. Hutter. Convergence of discrete MDL for sequential prediction. In *Proc. 17th Annual Conf. on Learning Theory (COLT'04)*, volume 3120 of *LNAI*, pages 300–314, Banff, 2004. Springer, Berlin.
- [Sch00] J. Schmidhuber. Algorithmic theories of everything. Report IDSIA-20-00, quant-ph/0011122, IDSIA, Manno (Lugano), Switzerland, 2000.
- [Sch02a] J. Schmidhuber. Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science*, 13(4):587–612, 2002.
- [Sch02b] J. Schmidhuber. The Speed Prior: a new simplicity measure yielding near-optimal computable predictions. In *Proc. 15th Annual Conference on Computational Learning Theory (COLT 2002)*, Lecture Notes in Artificial Intelligence, pages 216–228, Sydney, Australia, July 2002. Springer.
- [Sol64] R. J. Solomonoff. A formal theory of inductive inference: Part 1 and 2. *Inform. Control*, 7:1–22, 224–254, 1964.
- [Sol78] R. J. Solomonoff. Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Information Theory*, IT-24:422–432, 1978.
- [US96] V. A. Uspensky and A. Shen. Relations Between Varieties of Kolmogorov Complexities. *Math. Systems Theory*, 29:271–292, 1996.
- [VSU05] N. K. Vereshchagin, A. Shen, and V. A. Uspensky. Lecture Notes on Kolmogorov Complexity. Unpublished, <http://lpcs.math.msu.su/~ver/kolm-book>, 2005.
- [ZL70] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83–124, 1970.



# Non-asymptotic Calibration and Resolution

Vladimir Vovk

Computer Learning Research Centre,  
Department of Computer Science,  
Royal Holloway, University of London,  
Egham, Surrey TW20 0EX, England  
vovk@cs.rhul.ac.uk

**Abstract.** We analyze a new algorithm for probability forecasting of binary labels, without making any assumptions about the way the data is generated. The algorithm is shown to be well calibrated and to have high resolution for big enough data sets and for a suitable choice of its parameter, a kernel on the Cartesian product of the forecast space  $[0, 1]$  and the object space. Our results are non-asymptotic: we establish explicit inequalities for the performance of the algorithm.

## 1 Introduction

We consider the problem of predicting the label of a new object given a training set of labeled objects (or, as we will call them,  $(x_i, y_i)$ ). To make the process of prediction more vivid, we imagine that the examples are chosen by a player called Reality and the predictions are made by a player called Forecaster. To establish properties of prediction algorithms, the traditional theory of machine learning makes some assumptions about the way Reality generates the examples; e.g., statistical learning theory [14] assumes that the examples are generated independently from the same probability distribution. A more recent approach, prediction with expert advice (see, e.g., [1]), replaces the assumptions about Reality by a comparison class of prediction strategies; a typical result of this theory asserts that Forecaster can perform almost as well as the best strategies in the comparison class. This paper further explores a third possibility, suggested in [6], which requires neither assumptions about Reality nor a comparison class of Forecaster's strategies. It is shown in [6] that there exists a prediction strategy which is automatically well calibrated; this result has been further developed in several other papers. All the known calibration results, however, are asymptotic (see [10] for a critique of the standard asymptotic notion of calibration); the main results of this paper (Theorems 1 and 2) are first non-asymptotic results of this kind.

It should be noted that, although our approach was inspired by [6] and papers further developing [6], precise statements of our results and our proof techniques are completely different.

The non-asymptotic nature of our results makes it possible to derive new results in prediction with expert advice [15]. Such applications are, however, outside the scope of this paper.

## 2 The K29 and K29\* Algorithms

In this section we describe our learning protocol and the general prediction algorithms studied in this paper. The protocol is:

```

FOR  $n = 1, 2, \dots$ :
    Reality announces  $x_n \in \mathbf{X}$ .
    Forecaster announces  $p_n \in [0, 1]$ .
    Reality announces  $y_n \in \{0, 1\}$ .
END FOR.
    
```

On each round, Reality chooses an object  $x_n$ , then Forecaster gives his prediction  $p_n$  for this object's label, and finally Reality discloses the true label  $y_n \in \{0, 1\}$ . Reality chooses  $x_n$  from an arbitrary set  $\mathbf{X}$  and  $y_n$  from the two-element set  $\{0, 1\}$ ; intuitively, Forecaster's move  $p_n$  is the probability he attaches to the event  $y_n = 1$ . Forecaster's strategy in this protocol.

Our learning protocol is a perfect-information protocol; in particular, Reality may take into account the forecast  $p_n$  when deciding on her move  $y_n$ . (This feature is unusual for probability forecasting but it does extend the domain of applicability of our results.)

Next we describe the two general prediction algorithms that we study in this paper (one of them was derived informally in [17]). A function  $\mathbf{k} : Z^2 \rightarrow \mathbb{R}$ , where  $Z$  is an arbitrary set and  $\mathbb{R}$  is the set of real numbers, is a kernel on  $Z$  if it is symmetric ( $\mathbf{k}(z, z') = \mathbf{k}(z', z)$  for all  $z, z' \in Z$ ) and positive definite ( $\sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j \mathbf{k}(z_i, z_j) \geq 0$  for all  $(\lambda_1, \dots, \lambda_m) \in \mathbb{R}^m$  and all  $(z_1, \dots, z_m) \in Z^m$ ). The usual interpretation of a kernel  $\mathbf{k}(z, z')$  is as a measure of similarity between  $z$  and  $z'$  (see, e.g., [11], §1.1). The K29 and K29\* algorithms have one parameter, which is a kernel on the Cartesian product  $[0, 1] \times \mathbf{X}$ . The most standard way of constructing such kernels from kernels on  $[0, 1]$  and kernels on  $\mathbf{X}$  is the operation of tensor product. Let us say that a kernel  $\mathbf{k}$  on  $[0, 1] \times \mathbf{X}$  is a tensor product kernel if the function  $\mathbf{k}((p, x), (p', x'))$ , where  $p, p' \in [0, 1]$  and  $x, x' \in \mathbf{X}$ , is continuous in  $p$  for any fixed  $x, p', x'$ .

### THE K29 ALGORITHM

**Parameter:** admissible kernel  $\mathbf{k}$  on  $[0, 1] \times \mathbf{X}$

```

FOR  $n = 1, 2, \dots$ :
    Read  $x_n \in \mathbf{X}$ .
    Set  $S_n(p) := \sum_{i=1}^{n-1} \mathbf{k}((p, x_n), (p_i, x_i))(y_i - p_i)$ ,  $p \in [0, 1]$ .
    Output any root  $p$  of  $S_n(p) = 0$  as  $p_n$ ;
        if there are no roots,  $p_n := (1 + \text{sign } S_n)/2$ .
    Read  $y_n \in \{0, 1\}$ .
END FOR.
    
```

The intuition behind this algorithm is that  $p_n$  is chosen so that  $p_i$  are unbiased forecasts for  $y_i$  on the rounds  $i = 1, \dots, n - 1$  for which  $(p_i, x_i)$  is similar to  $(p_n, x_n)$ .

The K29\* algorithm is defined in the same way except that:

- its parameter  $\mathbf{k}$  is required to be  $\ast$ -admissible with the function  $\mathbf{k}((p, x), (p, x))$  continuous in  $p$  for any fixed  $x \in \mathbf{X}$  (the fact that this requirement is stronger than admissibility was established by Lehto in 1950; for a proof, see, e.g., [8], pp. 46–47);
- the function  $S_n(p)$  in the description of K29 is now defined as

$$S_n(p) := \sum_{i=1}^{n-1} \mathbf{k}((p, x_n), (p_i, x_i))(y_i - p_i) + \frac{1}{2} \mathbf{k}((p, x_n), (p, x_n))(1 - 2p)$$

(intuitively, the second addend, which can be rewritten as

$$\mathbf{k}((p, x_n), (p, x_n))(0.5 - p),$$

adds an element of regularization, i.e., bias towards the “neutral” value  $p_n = 0.5$ , to K29).

According to Mercer’s theorem (a very simple proof of a suitable version can be found in [5], Theorem II.3.1), there exists a function  $\Phi : [0, 1] \times \mathbf{X} \rightarrow H$  (a Hilbert space taking values in an inner product space  $H$  called the Mercer kernel) such that

$$\mathbf{k}(a, b) = \Phi(a) \cdot \Phi(b), \quad \forall a, b \in [0, 1] \times \mathbf{X} \tag{1}$$

( $\cdot$  standing for the inner product in  $H$ ). It is known that, for any  $\mathbf{k}$  and  $\Phi$  connected by (1),  $\mathbf{k}$  is  $\ast$ -admissible if and only if  $\Phi$  is a continuous function of  $p$  for each fixed  $x \in \mathbf{X}$  (the idea of a proof is described in, e.g., [13], Lemma 3).

Now we can state the basic result about K29 and K29\*.

**Theorem 1.** Let  $\mathbf{k}$  be  $\ast$ -admissible and  $\Phi : [0, 1] \times \mathbf{X} \rightarrow H$  be a Mercer kernel such that  $\mathbf{k}(a, b) = \Phi(a) \cdot \Phi(b)$ .

$$\left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right\|^2 \leq \sum_{n=1}^N \|\Phi(p_n, x_n)\|^2, \quad \forall N \in \{1, 2, \dots\}. \tag{2}$$

Let  $\mathbf{k}$  be  $\ast$ -admissible and  $\Phi : [0, 1] \times \mathbf{X} \rightarrow H$  be a Mercer kernel such that  $\mathbf{k}(a, b) = \Phi(a) \cdot \Phi(b)$ .

$$\left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right\|^2 \leq \sum_{n=1}^N p_n(1 - p_n) \|\Phi(p_n, x_n)\|^2, \quad \forall N \in \{1, 2, \dots\}. \tag{3}$$

Let us assume, for simplicity, that

$$C := \sup_{p, x} \|\Phi(p, x)\| < \infty \tag{4}$$

(it is often a good idea to use kernels with  $\|\Phi(p, x)\| \equiv 1$  and, therefore,  $C = 1$ ). Equation (2) then implies

$$\left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right\| \leq C\sqrt{N}, \quad \forall N \in \{1, 2, \dots\}. \tag{5}$$

When  $\Phi$  is absent (in the sense  $\Phi \equiv 1$ ), this shows that the forecasts  $p_n$  are “unbiased” predictions of the true labels  $y_n$ ; the presence of  $\Phi$  implies, for a suitable kernel, “local unbiasedness”. This is further discussed in §5.

### 3 Fermi–Sobolev Spaces

In the following section we will consider an especially suitable kernel for K29 and K29\* and we will state a calibration and resolution result for it. This result will involve a functional norm, and the goal of this preparatory section is to define this norm.

Let  $f : [0, 1]^k \rightarrow \mathbb{R}$  be a smooth function. The  $L_2$  norm

$$\sqrt{\int_0^1 \dots \int_0^1 \left( \frac{\partial^k f(t_1, \dots, t_k)}{\partial t_1 \dots \partial t_k} \right)^2 dt_1 \dots dt_k}$$

of its full cross-derivative will be denoted  $S(f)$ ; it is easy to see that  $S$  is a seminorm.

The norm  $\|f\|_{\text{FS}}$  of a smooth function  $f : [0, 1]^k \rightarrow \mathbb{R}$  is defined by

$$\|f\|_{\text{FS}}^2 := \sum_{\{i_1, \dots, i_m\} \subseteq \{1, \dots, k\}} S^2 \left( \int_0^1 \dots \int_0^1 f(t_1, \dots, t_k) dt_{i_1} \dots dt_{i_m} \right),$$

where  $\int_0^1 \dots \int_0^1 f(t_1, \dots, t_k) dt_{i_1} \dots dt_{i_m}$  is considered as a function of the free variables (the elements of the set  $\{t_1, \dots, t_k\} \setminus \{t_{i_1}, \dots, t_{i_m}\}$ ) and  $\sum_{\{i_1, \dots, i_m\} \subseteq \{1, \dots, k\}}$  stands (in this and similar places) for summation over all subsets, of all sizes  $m = 0, \dots, k$ , of  $\{1, \dots, k\}$  (and so we could have written  $\sum_{m=0}^k \sum_{\{i_1, \dots, i_m\} \subseteq \{1, \dots, k\}}$  instead).

The norm  $\|\cdot\|_{\text{FS}}$  on  $[0, 1]^k$  is the completion of the set of smooth  $f : [0, 1]^k \rightarrow \mathbb{R}$  satisfying  $\|f\|_{\text{FS}} < \infty$  with respect to the norm  $\|\cdot\|_{\text{FS}}$ . It is easy to see that it is in fact a Hilbert space. Elements of this Hilbert space will be called

In the rest of this paper we will usually assume  $\mathbf{X} = [0, 1]^K$  for some  $K \in \{0, 1, \dots\}$  and will be interested in the Fermi–Sobolev space on  $[0, 1]^{K+1}$ .

### 4 The FS and FS\* Algorithms

Suppose  $\mathbf{X} = [0, 1]^K$ . The kernel

$$\mathbf{k}((t_0, (t_1, \dots, t_K)), (t'_0, (t'_1, \dots, t'_K))) := \prod_{k=0}^K (3 \min^2(t_k, t'_k) + 3 \min^2(1 - t_k, 1 - t'_k) + 5) \quad (6)$$

on  $[0, 1] \times \mathbf{X}$  will be called the  $\mathbf{k}$ -kernel, and the K29 (resp. K29\*) algorithm applied to this kernel will be called the  $\mathbf{k}$ -K29 (resp.  $\mathbf{k}$ -K29\*) algorithm. (Intuitively,  $t_0$  and  $t'_0$  are the forecasts and  $(t_1, \dots, t_K)$  and  $(t'_1, \dots, t'_K)$  are the objects.) It is obvious that (6) is  $\mathbf{k}$ -admissible.

One can deduce the following corollary from Theorem 1.

**Theorem 2.** Let  $\mathbf{X} = [0, 1]^K$ ,  $K \in \{0, 1, \dots\}$ . Let  $(p_n)_{n=1}^N$  be a sequence of probabilities such that

$$\left| \sum_{n=1}^N (y_n - p_n) f(p_n, x_n) \right| \leq \left( \frac{2}{\sqrt{3}} \right)^{K+1} \|f\|_{\text{FS}} \sqrt{N} \quad (7)$$

where  $N \geq 1$ ,  $(y_n)_{n=1}^N \in \mathbb{R}^N$ ,  $(p_n)_{n=1}^N \in \mathbb{R}^N$ ,  $f : [0, 1]^{K+1} \rightarrow \mathbb{R}$  is a  $\mathbf{k}$ -admissible kernel.

$$\left| \sum_{n=1}^N (y_n - p_n) f(p_n, x_n) \right| \leq \left( \frac{2}{\sqrt{3}} \right)^{K+1} \|f\|_{\text{FS}} \sqrt{\sum_{n=1}^N p_n(1 - p_n)} \quad (8)$$

where  $N \geq 1$ ,  $(y_n)_{n=1}^N \in \mathbb{R}^N$ ,  $(p_n)_{n=1}^N \in \mathbb{R}^N$ ,  $f : [0, 1]^{K+1} \rightarrow \mathbb{R}$  is a  $\mathbf{k}$ -admissible kernel.

### 5 Informal Discussion of Theorems 1 and 2

In this section we will assume that (4) and, therefore, (5) hold true. We will show that the latter can be interpreted as saying that K29 is well calibrated (this is also true about K29\*, but in this section we will only discuss, for simplicity, the unstarred versions of the K29 and FS algorithms).

First, we briefly discuss the intuitive notion of calibration (for further details, see [4] and [6]). The forecasts  $p_n$ ,  $n = 1, \dots, N$ , are said to be “well calibrated” (or “unbiased in the small”, or “reliable”, or “valid”) if, for any  $p^* \in [0, 1]$ ,

$$\frac{\sum_{n=1, \dots, N: p_n \approx p^*} y_n}{\sum_{n=1, \dots, N: p_n \approx p^*} 1} \approx p^* \quad (9)$$

provided  $\sum_{n=1, \dots, N: p_n \approx p^*} 1$  is not too small. To make sense of the  $\approx$  in the numerator and denominator of (9), we replace each “crisp” point  $p^*$  by a “fuzzy point”  $I_{p^*} : [0, 1] \rightarrow [0, 1]$ ;  $I_{p^*}$  is required to be continuous; we might also want to have  $I_{p^*}(p^*) = 1$  and  $I_{p^*}(p) = 0$  for all  $p$  outside a small neighborhood of

$p^*$ . (The alternative of choosing  $I_{p^*} := \mathbb{I}_{[p_-, p_+]}$ , where  $[p_-, p_+]$  is a short interval containing  $p^*$  and  $\mathbb{I}_{[p_-, p_+]}$  is its indicator function, does not work because of Oakes’s and Dawid’s examples [9, 3];  $I_{p^*}$  can, however, be arbitrarily close to  $\mathbb{I}_{[p_-, p_+]}$ .) With this interpretation, (9) can be rewritten as

$$\frac{\sum_{n=1}^N (y_n - p^*) I_{p^*}(p_n)}{\sum_{n=1}^N I_{p^*}(p_n)} \approx 0. \tag{10}$$

It will be convenient to replace (10) with

$$\frac{\sum_{n=1}^N (y_n - p_n) I_{p^*}(p_n)}{\sum_{n=1}^N I_{p^*}(p_n)} \approx 0; \tag{11}$$

the difference between the left-hand sides of (10) and (11),

$$\frac{\sum_{n=1}^N (p_n - p^*) I_{p^*}(p_n)}{\sum_{n=1}^N I_{p^*}(p_n)}$$

is typically small (viz., is small assuming that  $I_{p^*}(p)$  is small when  $p$  is far from  $p^*$  and that  $I_{p^*}(p_n)$  is not small for many  $n$ ).

Let  $(p^*, x^*)$  be a point in  $[0, 1] \times \mathbf{X}$ . Fix an admissible kernel  $\mathbf{k} : ([0, 1] \times \mathbf{X})^2 \rightarrow \mathbb{R}$  and consider the “soft neighborhood”

$$I_{(p^*, x^*)}(p, x) := \mathbf{k}((p^*, x^*), (p, x)) \tag{12}$$

of the point  $(p^*, x^*)$ . The following is an easy corollary of Theorem 1.

**Corollary 1.** *Let  $\mathbf{k} : ([0, 1] \times \mathbf{X})^2 \rightarrow \mathbb{R}$  be an admissible kernel.*

$$C^2 := \sup_{(p, x) \in [0, 1] \times \mathbf{X}} \mathbf{k}((p, x), (p, x)) < \infty$$

$$\left| \sum_{n=1}^N (y_n - p_n) I_{(p^*, x^*)}(p_n, x_n) \right| \leq C^2 \sqrt{N} \tag{13}$$

for all  $(p^*, x^*) \in [0, 1] \times \mathbf{X}$  and all  $I_{(p^*, x^*)}$ .

Let  $\Phi : [0, 1] \times \mathbf{X} \rightarrow H$  be a function taking values in an inner product space  $H$  and satisfying (1); the constant  $C$  can be equivalently defined by (4). The Cauchy–Schwarz inequality, (4), and (5) imply

$$\begin{aligned} & \left| \sum_{n=1}^N (y_n - p_n) I_{(p^*, x^*)}(p_n, x_n) \right| \\ &= \left| \left( \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right) \cdot \Phi(p^*, x^*) \right| \\ &\leq \left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right\| \|\Phi(p^*, x^*)\| \leq C^2 \sqrt{N}. \quad \blacksquare \end{aligned}$$

Taking as  $\mathbf{k}$  a non-negative admissible kernel that ignores the objects,

$$\mathbf{k}((p, x), (p', x')) = \mathbf{k}(p, p') ,$$

we can rewrite (13) as

$$\left| \frac{\sum_{n=1}^N (y_n - p_n) I_{p^*}(p_n)}{\sum_{n=1}^N I_{p^*}(p_n)} \right| \leq \frac{C^2 \sqrt{N}}{\sum_{n=1}^N I_{p^*}(p_n)} ;$$

comparing this with (11), we can see that K29 is well calibrated provided

$$\sum_{n=1}^N I_{p^*}(p_n) \gg \sqrt{N} .$$

The presence of objects in (13) reflects the fact that, under a suitable choice of the parameter  $\mathbf{k}$ , K29 is not only well calibrated but can also have high “resolution”; we will now briefly discuss the latter property, again informally.

The fact that good calibration is only a necessary condition for good forecasting performance can be seen from the following standard example [4, 6]: if

$$(y_1, y_2, y_3, y_4, \dots) = (1, 0, 1, 0, \dots) ,$$

the forecasts  $p_n = 1/2, n = 1, 2, \dots$ , are well calibrated but rather poor; it would be better to predict with

$$(p_1, p_2, p_3, p_4, \dots) = (1, 0, 1, 0, \dots) .$$

Assuming that each object  $x_n$  contains the information about the parity of  $n$  (which can always be added to  $x_n$ ), we can see that the problem with the forecasting system  $p_n \equiv 1/2$  is its lack of resolution: it does not distinguish between the objects with odd and even  $n$ . In general, we would like each forecast  $p_n$  to be as specific as possible to the current object  $x_n$ ; the resolution of a prediction algorithm is the degree to which it achieves this goal. Equation (13) implies that we can expect unbiasedness of the forecasts  $p_n, n = 1, \dots, N$ , in the “soft neighborhood” of  $(p^*, x^*)$  when

$$\sum_{n=1}^N I_{(p^*, x^*)}(p_n, x_n) \gg \sqrt{N} ;$$

if  $I_{(p^*, x^*)}$  is concentrated around  $(p^*, x^*)$ , this means not only good calibration but also high resolution.

**Remark.** In principle, it can be seen directly (without using Corollary 1) that (5) implies good calibration and high resolution for a suitable  $\Phi$  and large  $N$ : indeed, (5) shows that the forecasts  $p_n$  are unbiased in the neighborhood of each

$(p^*, x^*)$  for functions  $\Phi$  that map distant  $(p, x)$  and  $(p', x')$  to almost orthogonal elements of the feature space (such as  $\Phi$  corresponding to the Gaussian kernel

$$\exp\left(\frac{(p - p')^2 + \|x - x'\|^2}{2\sigma^2}\right)$$

for a small “kernel width”  $\sigma > 0$ ).

We can see that Theorem 1 has implications for the calibration and resolution of the K29 algorithm. It should be admitted, however, that these implications are not immediate and require certain properties of the kernel parameter that we did not specify precisely. The interpretation of Theorem 2 as a statement about the calibration and resolution of the FS algorithm is more straightforward.

Let us discuss, e.g., calibration. Notice that in the case of a function of one variable  $f : [0, 1] \rightarrow \mathbb{R}$  the Fermi–Sobolev squared norm reduces to

$$\|f\|_{\text{FS}}^2 = \left(\int_0^1 f(t) dt\right)^2 + \int_0^1 (f'(t))^2 dt.$$

To interpret (11), consider the following approximation to the indicator function of a short interval  $[p_-, p_+]$  containing  $p^*$ :

$$f(p) := \begin{cases} 1 & \text{if } p_- + \epsilon \leq p \leq p_+ - \epsilon \\ 0 & \text{if } p \leq p_- - \epsilon \text{ or } p \geq p_+ + \epsilon \\ \frac{1}{2} + \frac{1}{2\epsilon}(p - p_-) & \text{if } p_- - \epsilon \leq p \leq p_- + \epsilon \\ \frac{1}{2} + \frac{1}{2\epsilon}(p_+ - p) & \text{if } p_+ - \epsilon \leq p \leq p_+ + \epsilon; \end{cases} \tag{14}$$

we assume that  $\epsilon > 0$  satisfies

$$0 < p_- - \epsilon < p_- + \epsilon < p_+ - \epsilon < p_+ + \epsilon < 1.$$

It is clear that this approximation is a Fermi–Sobolev function. An easy computation shows that (7) implies

$$\left| \sum_{n=1}^N (y_n - p_n) f(p_n) \right| \leq \frac{2}{\sqrt{3}} \sqrt{\left(\frac{1}{\epsilon} + (p_+ - p_-)^2\right)} N \tag{15}$$

for all  $N$ .

It is clear that inequalities analogous to (15) can also be proved for “soft neighborhoods” of points  $(p^*, x^*)$  in  $[0, 1] \times \mathbf{X}$ , and so Theorem 2 also implies high resolution for large  $N$ . Convenient neighborhoods in  $[0, 1]^{K+1}$  can be constructed as tensor products of neighborhoods (14); it can be shown that the Fermi–Sobolev norm of a tensor product is the product of the Fermi–Sobolev norms of the factors (see Appendix C).

An important advantage of (15) over the calibration properties of K29, as discussed above, is that the former implies that good calibration will be eventually attained at an arbitrarily fine scale (and not just at the scale determined by the “width” of the kernel used); a similar remark can be made about resolution as well.



## Acknowledgments

I am grateful to Ingo Steinwart for information about Lehto's example and to the anonymous reviewers for their comments. This work was partially supported by MRC (grant S505/65) and Royal Society.

## References

- [1] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44:427–485, 1997.
- [2] A. Philip Dawid. Calibration-based empirical probability (with discussion). *Annals of Statistics*, 13:1251–1285, 1985.
- [3] A. Philip Dawid. Self-calibrating priors do not exist: Comment. *Journal of the American Statistical Association*, 80:340–341, 1985. This is a contribution to the discussion in [9].
- [4] A. Philip Dawid. Probability forecasting. In Samuel Kotz, Norman L. Johnson, and Campbell B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 7, pages 210–218. Wiley, New York, 1986.
- [5] Joseph L. Doob. *Stochastic Processes*. Wiley, New York, 1953.
- [6] Dean P. Foster and Rakesh V. Vohra. Asymptotic calibration. *Biometrika*, 85:379–390, 1998.
- [7] Sham M. Kakade and Dean P. Foster. Deterministic calibration and Nash equilibrium. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the Seventeenth Annual Conference on Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 33–48, Heidelberg, 2004. Springer.
- [8] Herbert Meschkowski. *Hilbertsche Räume mit Kernfunktion*. Springer, Berlin, 1962.
- [9] David Oakes. Self-calibrating priors do not exist (with discussion). *Journal of the American Statistical Association*, 80:339–342, 1985.
- [10] Mark J. Schervish. Contribution to the discussion in [2]. *Annals of Statistics*, 13:1274–1282, 1985.
- [11] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- [12] Glenn Shafer and Vladimir Vovk. *Probability and Finance: It's Only a Game!* Wiley, New York, 2001.
- [13] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.
- [14] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
- [15] Vladimir Vovk. Defensive prediction with expert advice. These Proceedings.
- [16] Vladimir Vovk. Non-asymptotic calibration and resolution. Technical Report [arXiv:cs.LG/0506004](https://arxiv.org/abs/cs.LG/0506004) (version 2), [arXiv.org](https://arxiv.org/) e-Print archive, July 2005. This is the full version of this paper, containing all proofs.
- [17] Vladimir Vovk, Akimichi Takemura, and Glenn Shafer. Defensive forecasting. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, January 6–8, 2005, Savannah Hotel, Barbados*, pages 365–372. Society for Artificial Intelligence and Statistics, 2005. Available electronically at <http://www.gatsby.ucl.ac.uk/aistats/>.

## Appendix A: Proof of Theorem 1

The proof of Theorem 1 (as well as its statement and the K29 and K29\* algorithms themselves) is based on the game-theoretic approach to the foundations of probability proposed in [12]. A new player, called Skeptic, is added to the learning protocol of §2; the idea is that Skeptic is allowed to bet at the odds defined by Forecaster’s probabilities.

### BINARY FORECASTING GAME I

**Players:** Reality, Forecaster, Skeptic

**Protocol:**

$\mathcal{K}_0 := C$ .

FOR  $n = 1, 2, \dots$ :

Reality announces  $x_n \in \mathbf{X}$ .

Forecaster announces  $p_n \in [0, 1]$ .

Skeptic announces  $s_n \in \mathbb{R}$ .

Reality announces  $y_n \in \{0, 1\}$ .

$\mathcal{K}_n := \mathcal{K}_{n-1} + s_n(y_n - p_n)$ .

END FOR.

The protocol describes not only the players’ moves but also the changes in Skeptic’s capital  $\mathcal{K}_n$ ; its initial value is an arbitrary constant  $C$ .

The crucial (albeit very simple) observation [17] is that for any continuous strategy for Skeptic there exists a strategy for Forecaster that does not allow Skeptic’s capital to grow, regardless of what Reality is doing (a similar observation was made in [7]). To state this observation in its strongest form, we will make Skeptic announce his strategy for each round before Forecaster’s move on that round rather than announce his full strategy at the beginning of the game. Therefore, we consider the following perfect-information game:

### BINARY FORECASTING GAME II

**Players:** Reality, Forecaster, Skeptic

**Protocol:**

$\mathcal{K}_0 := C$ .

FOR  $n = 1, 2, \dots$ :

Reality announces  $x_n \in \mathbf{X}$ .

Skeptic announces continuous  $S_n : [0, 1] \rightarrow \mathbb{R}$ .

Forecaster announces  $p_n \in [0, 1]$ .

Reality announces  $y_n \in \{0, 1\}$ .

$\mathcal{K}_n := \mathcal{K}_{n-1} + S_n(p_n)(y_n - p_n)$ .

END FOR.

**Lemma 1.**

$\mathcal{K}_0 \geq \mathcal{K}_1 \geq \mathcal{K}_2 \geq \dots$

Forecaster can use the following strategy to ensure  $\mathcal{K}_0 \geq \mathcal{K}_1 \geq \dots$ :

- if the function  $S_n(p)$  takes value 0, choose  $p_n$  so that  $S_n(p_n) = 0$ ;
- if  $S_n$  is always positive, take  $p_n := 1$ ;
- if  $S_n$  is always negative, take  $p_n := 0$ . ■

**Proof of the Theorem**

Following the K29 algorithm Forecaster ensures that Skeptic will never increase his capital with the strategy

$$s_n := \sum_{i=1}^{n-1} \mathbf{k}((p_n, x_n), (p_i, x_i)) (y_i - p_i) . \tag{16}$$

The increase in Skeptic’s capital when he follows (16) is

$$\begin{aligned} \mathcal{K}_N - \mathcal{K}_0 &= \sum_{n=1}^N s_n (y_n - p_n) \\ &= \sum_{n=1}^N \sum_{i=1}^{n-1} \mathbf{k}((p_n, x_n), (p_i, x_i)) (y_n - p_n)(y_i - p_i) \\ &= \frac{1}{2} \sum_{n=1}^N \sum_{i=1}^N \mathbf{k}((p_n, x_n), (p_i, x_i)) (y_n - p_n)(y_i - p_i) \\ &\quad - \frac{1}{2} \sum_{n=1}^N \mathbf{k}((p_n, x_n), (p_n, x_n)) (y_n - p_n)^2 . \tag{17} \end{aligned}$$

We can rewrite (17) as

$$\mathcal{K}_N - \mathcal{K}_0 = \frac{1}{2} \left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right\|^2 - \frac{1}{2} \sum_{n=1}^N \|(y_n - p_n) \Phi(p_n, x_n)\|^2 , \tag{18}$$

which immediately implies (2).

To prove (3), notice that

$$(y_n - p_n)^2 = p_n(1 - p_n) + (1 - 2p_n)(y_n - p_n)$$

both for  $y_n = 0$  and for  $y_n = 1$ . Therefore, replacing strategy (16) with

$$s_n := \sum_{i=1}^{n-1} \mathbf{k}((p_n, x_n), (p_i, x_i)) (y_i - p_i) + \frac{1}{2} \mathbf{k}((p_n, x_n), (p_n, x_n)) (1 - 2p_n)$$

(which is the strategy for Skeptic prevented by K29\* from making a profit), we will obtain, instead of (17) and (18),

$$\begin{aligned}
 \mathcal{K}_N - \mathcal{K}_0 &= \sum_{n=1}^N s_n (y_n - p_n) \\
 &= \frac{1}{2} \sum_{n=1}^N \sum_{i=1}^N \mathbf{k}((p_n, x_n), (p_i, x_i)) (y_n - p_n)(y_i - p_i) \\
 &\quad - \frac{1}{2} \sum_{n=1}^N \mathbf{k}((p_n, x_n), (p_n, x_n)) (y_n - p_n)^2 \\
 &\quad + \frac{1}{2} \sum_{n=1}^N \mathbf{k}((p_n, x_n), (p_n, x_n)) (1 - 2p_n)(y_n - p_n) \\
 &= \frac{1}{2} \sum_{n=1}^N \sum_{i=1}^N \mathbf{k}((p_n, x_n), (p_i, x_i)) (y_n - p_n)(y_i - p_i) \\
 &\quad - \frac{1}{2} \sum_{n=1}^N \mathbf{k}((p_n, x_n), (p_n, x_n)) p_n(1 - p_n) \\
 &= \frac{1}{2} \left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right\|^2 - \frac{1}{2} \sum_{n=1}^N p_n(1 - p_n) \|\Phi(p_n, x_n)\|^2 ;
 \end{aligned}$$

this proves (3).

## Appendix B: Proof Sketch of Theorem 2

Let  $M$  be a countable set and  $a = (a_m \mid m \in M)$  be a fixed array of positive numbers summing to 1:

$$\sum_{m \in M} a_m = 1 . \tag{19}$$

The feature space used in this proof will be the set  $H$  of all arrays  $u = (u_m \mid m \in M)$  of real numbers  $u_m$  such that

$$\sum_{m \in M} a_m u_m^2 < \infty .$$

The inner product in  $H$  is defined by

$$u \cdot v := \sum_{m \in M} a_m u_m v_m ;$$

we will use the notation  $\|u\|_a$  for the norm  $\sqrt{u \cdot u}$  in this space.

Let  $(f_m \mid m \in M)$  be an orthogonal array of continuous functions

$$f_m : [0, 1]^{K+1} \rightarrow [-1, 1] \tag{20}$$

(eventually we will be interested in tensor products of cosine functions) considered as elements of  $L_2([0, 1]^{K+1})$ . We are interested in the feature mapping

$$\Phi : t \in [0, 1]^{K+1} \mapsto (f_m(t) \mid m \in M)$$

(as in (6), we use the letter  $t$  to stand, intuitively, for the vector of attributes  $x$  extended by adding the forecast  $p$  on the left). The corresponding kernel is

$$\mathbf{k}(t, t') := \Phi(t) \cdot \Phi(t') = \sum_{m \in M} a_m f_m(t) f_m(t'). \tag{21}$$

This kernel is continuous in  $t$  since, by (19) and (20), the series (21) converges absolutely and uniformly in  $t$ . Therefore,  $\mathbf{k}$  is a valid parameter for K29. Similarly, taking  $t = t'$  in (21), we can see that  $\mathbf{k}$  is a valid parameter for K29\*.

For each function  $f \in L_2([0, 1]^{K+1})$  its  $c_m$  is defined as

$$c_m = \int f f_m \, d\Lambda,$$

where

$$c_m := \frac{1}{\|f_m\|_2^2} \int f f_m \, d\Lambda$$

and  $\Lambda$  is the Lebesgue measure on  $[0, 1]^{K+1}$ .

**Lemma 2.**  $\left| \sum_{n=1}^N (y_n - p_n) f(p_n, x_n) \right| \leq \sqrt{\left( \sum_{m \in M} \frac{c_m^2}{a_m} \right) N}$

$$\left| \sum_{n=1}^N (y_n - p_n) f(p_n, x_n) \right| \leq \sqrt{\left( \sum_{m \in M} \frac{c_m^2}{a_m} \right) N}, \tag{22}$$

$$\sqrt{\left( \sum_{m \in M} \frac{c_m^2}{a_m} \right) \sum_{n=1}^N p_n (1 - p_n)}.$$

First notice that

$$\sup_t \|\Phi(t)\|_a^2 = \sup_t \sum_{m \in M} a_m f_m^2(t) \leq \sum_{m \in M} a_m = 1.$$

Now we can deduce from Theorem 1:

$$\begin{aligned} \left| \sum_{n=1}^N (y_n - p_n) f(p_n, x_n) \right| &= \left| \sum_{m \in M} c_m \sum_{n=1}^N (y_n - p_n) f_m(p_n, x_n) \right| \\ &= \left\| \left( \frac{c_m}{a_m} \right)_{m \in M} \cdot \left( \sum_{n=1}^N (y_n - p_n) f_m(p_n, x_n) \right)_{m \in M} \right\| \\ &\leq \left\| \left( \frac{c_m}{a_m} \right)_{m \in M} \right\|_a \left\| \left( \sum_{n=1}^N (y_n - p_n) f_m(p_n, x_n) \right)_{m \in M} \right\|_a \\ &= \left\| \left( \frac{c_m}{a_m} \right)_{m \in M} \right\|_a \left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, x_n) \right\|_a \leq \sqrt{\left( \sum_{m \in M} \frac{c_m^2}{a_m} \right) N}. \end{aligned}$$

The proof for K29\* is analogous. ■

**Idea of the Proof of the Theorem**

We take the orthogonal array  $(f_m | m \in M)$  to be the standard “half-range” Fourier basis in  $L_2([0, 1]^{K+1})$ :  $M := \{0, 1, \dots\}^{K+1}$  and

$$f_{m_0, \dots, m_K}(t_0, \dots, t_K) := \cos \pi m_0 t_0 \cdots \cos \pi m_K t_K ;$$

the  $L_2$ -norm of such a function is  $2^{-k/2}$ , where  $k$  is the number of non-zero indices among  $m_0, \dots, m_K$ . The corresponding Fourier coefficients are

$$c_{0, \dots, 0, m_{i_1}, 0, \dots, 0, m_{i_k}, 0, \dots, 0} = 2^k \int_0^1 \cdots \int_0^1 f(t_0, \dots, t_K) \cos \pi m_{i_1} t_{i_1} \cdots \cos \pi m_{i_k} t_{i_k} dt_0 \cdots dt_K ,$$

where  $m_{i_1}, \dots, m_{i_k}$  are the non-zero indices among  $m_0, \dots, m_K$ . It turns out that if we take

$$a_{0, \dots, 0, m_{i_1}, 0, \dots, 0, m_{i_k}, 0, \dots, 0} := \left(\frac{3}{4}\right)^{K+1} \frac{2^k}{\pi^{2k}} \frac{1}{m_{i_1}^2 \cdots m_{i_k}^2} , \tag{23}$$

we will obtain

$$\sum_{m \in M} \frac{c_m^2}{a_m} = \left(\frac{4}{3}\right)^{K+1} \|f\|_{\text{FS}}^2 .$$

For details, see [16].

**Derivation of the FS Kernel**

The details of the derivation are again given in [16], but it is based on the formula

$$\sum_{m=1}^{\infty} \frac{\cos \pi m q}{m^2} = \frac{\pi^2}{12} (3q^2 - 6q + 2) ,$$

which is valid for  $q \in [0, 1]$ . This formula can be obtained by combining the standard Fourier expansions

$$x = \frac{\pi}{2} - \frac{4}{\pi} \left( \cos x + \frac{\cos 3x}{3^2} + \frac{\cos 5x}{5^2} + \cdots \right)$$

(valid for  $x \in [0, \pi]$ ) and

$$x^2 = \frac{\pi^2}{3} - 4 \left( \cos x - \frac{\cos 2x}{2^2} + \frac{\cos 3x}{3^2} - \cdots \right)$$

(valid for  $x \in [-\pi, \pi]$ ), and substituting  $\pi q$  for  $x$ .

### Appendix C: A Useful Property of the Fermi–Sobolev Spaces

The following property is useful for understanding of calibration and resolution in the multidimensional case (and we used it in §5).

**Lemma 3.** *If  $f = gh$  then*

$$\|f\|_{\text{FS}} = \|g\|_{\text{FS}} \|h\|_{\text{FS}}.$$

We will only consider the case of smooth  $g$  and  $h$ . Let  $g$  and  $h$  be functions of  $K_1$  and  $K_2$  variables, respectively; therefore,  $f$  is a function of  $K_1 + K_2$  variables and

$$f(t_1^1, \dots, t_{K_1}^1, t_1^2, \dots, t_{K_2}^2) = g(t_1^1, \dots, t_{K_1}^1)h(t_1^2, \dots, t_{K_2}^2).$$

We will let  $\partial t_{\{j_1, \dots, j_k\}}$  and  $dt_{\{j_1, \dots, j_k\}}$  stand for  $\partial t_{j_1} \cdots \partial t_{j_k}$  and  $dt_{j_1} \cdots dt_{j_k}$ , respectively. We find:

$$\begin{aligned} \|f\|_{\text{FS}}^2 &= \sum_{\substack{\{i_1^1, \dots, i_{k_1}^1\} \subseteq \{1, \dots, K_1\} \\ \{i_1^2, \dots, i_{k_2}^2\} \subseteq \{1, \dots, K_2\}}} \int_0^1 \cdots \int_0^1 \left( \frac{\partial^{k_1}}{\partial t_{\{i_1^1, \dots, i_{k_1}^1\}}} \frac{\partial^{k_2}}{\partial t_{\{i_1^2, \dots, i_{k_2}^2\}}} \right. \\ &\quad \left. \int_0^1 \cdots \int_0^1 g(t_1^1, \dots, t_{K_1}^1)h(t_1^2, \dots, t_{K_2}^2) \right. \\ &\quad \left. dt_{\{1, \dots, K_2\} \setminus \{i_1^2, \dots, i_{k_2}^2\}} dt_{\{1, \dots, K_1\} \setminus \{i_1^1, \dots, i_{k_1}^1\}} \right)^2 dt_{\{i_1^2, \dots, i_{k_2}^2\}} dt_{\{i_1^1, \dots, i_{k_1}^1\}} \\ &= \left( \sum_{\{i_1^1, \dots, i_{k_1}^1\} \subseteq \{1, \dots, K_1\}} \int_0^1 \cdots \int_0^1 \left( \frac{\partial^{k_1}}{\partial t_{\{i_1^1, \dots, i_{k_1}^1\}}} \int_0^1 \cdots \int_0^1 g(t_1^1, \dots, t_{K_1}^1) \right. \right. \\ &\quad \left. \left. dt_{\{1, \dots, K_1\} \setminus \{i_1^1, \dots, i_{k_1}^1\}} \right)^2 dt_{\{i_1^1, \dots, i_{k_1}^1\}} \right) \\ &\quad \left( \sum_{\{i_1^2, \dots, i_{k_2}^2\} \subseteq \{1, \dots, K_2\}} \int_0^1 \cdots \int_0^1 \left( \frac{\partial^{k_2}}{\partial t_{\{i_1^2, \dots, i_{k_2}^2\}}} \int_0^1 \cdots \int_0^1 h(t_1^2, \dots, t_{K_2}^2) \right. \right. \\ &\quad \left. \left. dt_{\{1, \dots, K_2\} \setminus \{i_1^2, \dots, i_{k_2}^2\}} \right)^2 dt_{\{i_1^2, \dots, i_{k_2}^2\}} \right) = \|g\|_{\text{FS}}^2 \|h\|_{\text{FS}}^2. \quad \blacksquare \end{aligned}$$

# Defensive Prediction with Expert Advice

Vladimir Vovk

Computer Learning Research Centre,  
Department of Computer Science,  
Royal Holloway, University of London,  
Egham, Surrey TW20 0EX, England  
vovk@cs.rhul.ac.uk

**Abstract.** The theory of prediction with expert advice usually deals with countable or finite-dimensional pools of experts. In this paper we give similar results for pools of decision rules belonging to an infinite-dimensional functional space which we call the Fermi–Sobolev space. For example, it is shown that for a wide class of loss functions (including the standard square, absolute, and log loss functions) the average loss of the master algorithm, over the first  $N$  steps, does not exceed the average loss of the best decision rule with a bounded Fermi–Sobolev norm plus  $O(N^{-1/2})$ . Our proof techniques are very different from the standard ones and are based on recent results about defensive forecasting. Given the probabilities produced by a defensive forecasting algorithm, which are known to be well calibrated and to have high resolution in the long run, we use the Expected Loss Minimization principle to find a suitable decision.

## 1 Introduction

This paper further develops a new approach to probability forecasting which we call “defensive forecasting”. This approach has several appealing features. First, for every constructive law of probability expressed in game-theoretic terms there exists a procedure of defensive forecasting that satisfies this law perfectly [15]. In particular, one can construct a prediction algorithm with high degrees of calibration and resolution in the long term, which is mathematically expressed by explicit inequalities without making any assumptions about the data-generating mechanism [10]. This paper describes a third feature: defensive forecasts lead to efficient decisions; in particular, defensive forecasting allows one to obtain new results in prediction with expert advice.

First papers on prediction with expert advice with general loss functions (e.g., [1, 11]) dealt with countable (often finite) pools of experts. The next step was to consider finite-dimensional pools of experts (e.g., [3, 7, 12]). This paper continues with infinite-dimensional pools of experts. To get an idea of its central results, the reader is advised to start from Corollaries 1–3.



## 2 The FS Algorithm

In this paper we will use a defensive forecasting algorithm introduced in [10] and called the FS algorithm. We will not give an explicit description of the algorithm (which can be found in [10]) and only describe one of its properties.

Our prediction protocol is:

FOR  $n = 1, 2, \dots$ :  
 Reality announces  $x_n \in [0, 1]^K$ .  
 Forecaster announces  $p_n \in [0, 1]$ .  
 Reality announces  $y_n \in \{0, 1\}$ .  
 END FOR.

At each step  $n$  Forecaster observes the  $K$  attributes of an object  $x_n$  and is asked to predict its label  $y_n$ . The FS algorithm is Forecaster’s strategy in this protocol (or, more precisely, the strategy of a defensive forecaster). In Theorem 1 below we will state its property, but before that we will need to give several auxiliary definitions. We will be following [10].

If  $f : [0, 1]^k \rightarrow \mathbb{R}$  is a smooth function, the  $L_2$  norm

$$\sqrt{\int_0^1 \dots \int_0^1 \left( \frac{\partial^k f(t_1, \dots, t_k)}{\partial t_1 \dots \partial t_k} \right)^2 dt_1 \dots dt_k}$$

of its full cross-derivative is denoted  $S(f)$ . We define the Fermi–Sobolev norm of a smooth function  $f : [0, 1]^k \rightarrow \mathbb{R}$  by

$$\|f\|_{\text{FS}}^2 := \sum_{m=0}^k \sum_{\{i_1, \dots, i_m\} \subseteq \{1, \dots, k\}} S^2 \left( \int_0^1 \dots \int_0^1 f(t_1, \dots, t_k) dt_{i_1} \dots dt_{i_m} \right),$$

where  $\int_0^1 \dots \int_0^1 f(t_1, \dots, t_k) dt_{i_1} \dots dt_{i_m}$  is considered as a function of the free variables. The Fermi–Sobolev space on  $[0, 1]^k$  ( $k$  will always be clear from the context) is the completion of the set of such  $f$  satisfying  $\|f\|_{\text{FS}} < \infty$  with respect to the norm  $\|\cdot\|_{\text{FS}}$ . Elements of the Fermi–Sobolev space will be called Fermi–Sobolev functions (or, for brevity, FS functions).

The reader might find it helpful to concentrate at first on the one-dimensional case,  $k = 1$ ; the Fermi–Sobolev squared norm then reduces to

$$\|f\|_{\text{FS}}^2 := \left( \int_0^1 f(t) dt \right)^2 + \int_0^1 (f'(t))^2 dt,$$

the first addend reflecting how centered  $f$  is and the second measuring the volatility of  $f$ .

**Theorem 1 ([10]).** Let  $\{x_n\}_{n=1}^N$  be a sequence of points in  $[0, 1]^K$  and  $\{p_n\}_{n=1}^N$  a sequence of real numbers in  $[0, 1]$ .

$$\left| \sum_{n=1}^N (y_n - p_n) f(p_n, x_n) \right| \leq \left( \frac{2}{\sqrt{3}} \right)^{K+1} \|f\|_{\text{FS}} \sqrt{N} \tag{1}$$

where  $\{y_n\}_{n=1}^N$  is a sequence of real numbers in  $\{0, 1\}$  and  $f : [0, 1]^{K+1} \rightarrow \mathbb{R}$  is a Fermi–Sobolev function.

The right-hand side of (1) is  $O(\sqrt{N})$ ; this term will be as ubiquitous in this paper as it is in the traditional theory (see, e.g., [1] and [4]). In all our results the ratio  $2/\sqrt{3}$  can be replaced by its upper bound 1.155.

Theorem 1 will be useful in our preliminary discussion in the next section, but much stronger results can be derived from its simple modifications, which will be stated in the appendix.

### 3 Preliminary Discussion

In this section we will introduce our decision making protocol, give several definitions needed for the following sections, and prove a simple result (Proposition 1) showing how defensive forecasting is relevant to decision making. A much more useful result will be proved in the next section.

The decision making protocol of this paper is:

FOR  $n = 1, 2, \dots$ :

    Reality announces  $x_n \in [0, 1]^K$ .

    Decision Maker announces  $\gamma_n \in \Gamma$ .

    Reality announces  $y_n \in \{0, 1\}$ .

END FOR.

In applications,  $x_n$  will represent information deemed useful in predicting  $y_n$ . Decision Maker chooses his actions  $\gamma_n$  from a nonempty set  $\Gamma$  and his performance is measured with a loss function  $\lambda : \{0, 1\} \times \Gamma \rightarrow \mathbb{R}$ ; his average loss over the first  $N$  steps is

$$\frac{1}{N} \sum_{n=1}^N \lambda(y_n, \gamma_n).$$

In tradition of prediction with expert advice, Decision Maker will compete against a pool of decision rules, where a decision rule is defined to be a function  $D : [0, 1]^K \rightarrow \Gamma$ ; the average loss suffered by such a decision rule is

$$\frac{1}{N} \sum_{n=1}^N \lambda(y_n, D(x_n)).$$

It is important that no assumptions are made about Reality (cf. [6, 8], where results somewhat similar to ours are obtained under the assumption that  $y_n = D(x_n)$  for all  $n$  and for a fixed decision rule  $D$  belonging to a known functional class). The expected loss  $\text{Exp}_D : [0, 1]^K \rightarrow \mathbb{R}$  of the decision rule  $D$  at a point  $x \in [0, 1]^K$  is defined by

$$\text{Exp}_D(x) := \lambda(1, D(x)) - \lambda(0, D(x)).$$

A probability distribution  $(1-p, p)$  is any function mapping  $[0, 1]$  to  $\Gamma$ ; intuitively, such a function maps a probability distribution  $(1-p, p)$  on  $\{0, 1\}$  to a decision  $\gamma = \gamma(p)$

that is optimal or nearly optimal under this probability distribution. The defect of a choice function  $G$  is

$$\Delta_G := \sup_{p \in [0,1]} \left( \lambda(p, G(p)) - \inf_{\gamma \in \Gamma} \lambda(p, \gamma) \right),$$

where  $\lambda(p, \gamma)$  is the expected loss caused by taking the decision  $\gamma$  when the probability of 1 is  $p$ :

$$\lambda(p, \gamma) := p\lambda(1, \gamma) + (1 - p)\lambda(0, \gamma). \tag{2}$$

If  $\Gamma = [0, 1]$  and the choice function  $G(p) := p$  has zero defect, the loss function is called a *proper scoring rule* (e.g., the square loss and log loss functions, discussed in Section 5, are known to be proper scoring rules). The *Expected Loss*  $\text{Exp}_G : [0, 1] \rightarrow \mathbb{R}$  of the choice function  $G$  is defined by

$$\text{Exp}_G(p) := \lambda(1, G(p)) - \lambda(0, G(p)).$$

We will be interested in decision rules and, in this section, choice functions whose exposure is an FS function.

We will construct decision strategies from FS’s probabilities by minimizing the expected loss. In some cases instead of the literal minimization we might want to use slightly suboptimal (but, e.g., continuous) choice functions. Therefore, we will use the following “soft” version of the Expected Loss Minimization principle: the decision strategy (i.e., Decision Maker’s strategy)  $\gamma_n$  for a choice function  $G$  and a prediction algorithm is the one that outputs  $\gamma_n := G(p_n)$  at the  $n$ th step, where  $p_n$  is the prediction output by the prediction algorithm.

**Proposition 1.** *Let  $D$  be a FS function and  $G$  a choice function with defect  $\Delta_G$ . Then for any prediction algorithm  $\gamma_n$  and any sequence  $(x_n)_{n=1}^N$  of predictions, it holds that*

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^N \lambda(y_n, \gamma_n) &\leq \frac{1}{N} \sum_{n=1}^N \lambda(y_n, D(x_n)) \\ &\quad + \left( \frac{2}{\sqrt{3}} \right)^{K+1} \frac{\|\text{Exp}_D\|_{\text{FS}} + \|\text{Exp}_G\|_{\text{FS}}}{\sqrt{N}} + \Delta_G \end{aligned}$$

where  $\|\cdot\|_{\text{FS}}$  is the FS norm and  $\text{Exp}_D$  is the expected loss of  $D$ .

Notice that

$$\lambda(y, \gamma) - \lambda(p, \gamma) = (y - p)(\lambda(1, \gamma) - \lambda(0, \gamma))$$

always holds (this can be checked by subtracting (2) from  $\lambda(y, \gamma) := y\lambda(1, \gamma) + (1 - y)\lambda(0, \gamma)$ ). In conjunction with Theorem 1 this implies

$$\sum_{n=1}^N \lambda(y_n, \gamma_n) = \sum_{n=1}^N \lambda(y_n, G(p_n))$$

$$\begin{aligned}
 &= \sum_{n=1}^N \lambda(p_n, G(p_n)) + \sum_{n=1}^N (\lambda(y_n, G(p_n)) - \lambda(p_n, G(p_n))) \\
 &= \sum_{n=1}^N \lambda(p_n, G(p_n)) + \sum_{n=1}^N (y_n - p_n)(\lambda(1, G(p_n)) - \lambda(0, G(p_n))) \\
 &\leq \sum_{n=1}^N \lambda(p_n, G(p_n)) + (2/\sqrt{3})^{K+1} \|\text{Exp}_G\|_{\text{FS}} \sqrt{N} \\
 &\leq \sum_{n=1}^N \lambda(p_n, D(x_n)) + N\Delta_G + (2/\sqrt{3})^{K+1} \|\text{Exp}_G\|_{\text{FS}} \sqrt{N} \\
 &= \sum_{n=1}^N \lambda(y_n, D(x_n)) - \sum_{n=1}^N (\lambda(y_n, D(x_n)) - \lambda(p_n, D(x_n))) \\
 &\quad + N\Delta_G + (2/\sqrt{3})^{K+1} \|\text{Exp}_G\|_{\text{FS}} \sqrt{N} \\
 &= \sum_{n=1}^N \lambda(y_n, D(x_n)) - \sum_{n=1}^N (y_n - p_n)(\lambda(1, D(x_n)) - \lambda(0, D(x_n))) \\
 &\quad + N\Delta_G + (2/\sqrt{3})^{K+1} \|\text{Exp}_G\|_{\text{FS}} \sqrt{N} \\
 &\leq \sum_{n=1}^N \lambda(y_n, D(x_n)) + (2/\sqrt{3})^{K+1} \|\text{Exp}_D\|_{\text{FS}} \sqrt{N} \\
 &\quad + N\Delta_G + (2/\sqrt{3})^{K+1} \|\text{Exp}_G\|_{\text{FS}} \sqrt{N}, \quad (3)
 \end{aligned}$$

which completes the proof. ■

The chain (3) demonstrates our proof technique very well; to show that the actual loss of our decision strategy does not exceed the actual loss of a decision rule  $D$  by much, we notice that:

- the actual loss  $\sum_{n=1}^N \lambda(y_n, G(p_n))$  of our decision strategy is approximately equal, by Theorem 1, to the (one-step-ahead conditional) expected loss  $\sum_{n=1}^N \lambda(p_n, G(p_n))$  of our strategy;
- since we used the Expected Loss Minimization principle, the expected loss of our strategy does not exceed the expected loss of  $D$  (assuming  $\Delta_G = 0$ );
- the expected loss of  $D$  is approximately equal to its actual loss (again by Theorem 1).

It will be convenient to introduce the following terminology (very imprecise, to be used in informal discussions only). We say that a prediction algorithm has good  $\lambda$ -regret if the left-hand side of (1) is much less than  $N$  for a wide class of functions  $f : [0, 1]^{K+1} \rightarrow \mathbb{R}$  and large  $N$ . We say that the algorithm has good  $\lambda$ -regret if

$$\left| \sum_{n=1}^N (y_n - p_n) f(p_n) \right| \ll N$$

for a wide class of functions  $f : [0, 1] \rightarrow \mathbb{R}$  and large  $N$ . Finally, we say that the algorithm has good  $\epsilon$ -regret if

$$\left| \sum_{n=1}^N (y_n - p_n) f(x_n) \right| \ll N \epsilon$$

for a wide class of  $f : [0, 1]^K \rightarrow \mathbb{R}$  and for large  $N$ .

### 4 General Result

The main purpose of the previous section was to motivate the somewhat less natural definitions of this section. Our approach so far suffered from two drawbacks:

- The inequality in Proposition 1 can be tightened if we notice that in applying (1) to establishing the first and third inequalities in (3) we need not arbitrary Fermi–Sobolev  $f = f(p, x)$  but only  $f = f(p)$  (known in advance) and  $f = f(x)$ . (In particular, we only need calibration and resolution separately, not calibration-cum-resolution.)
- The requirement that the exposure of the choice function should be continuous (even Fermi–Sobolev) is very restrictive: for example, in the case of the absolute loss function (described in Subsection 5.2), there is no perfect (i.e., with  $\Delta_G = 0$ ) choice function  $G$  with continuous exposure.

These drawbacks will be dealt with in the appendix. To deal with the second one, we will have to modify the FS algorithm so that it outputs extended predictions  $(p_n, q_n) \in [0, 1]^2$ , where  $p_n$  is the prediction of  $y_n$ , as before, and the extra component  $q_n$  will make it possible to design perfect choice functions with continuous exposure.

The  $\mathbb{R}^2$ -square  $[0, 1]^2$  is defined to be the set  $[0, 1]^2$  equipped with the following linear order: if  $(x_1, y_1)$  and  $(x_2, y_2)$  are two points in  $[0, 1]^2$ ,  $(x_1, y_1) < (x_2, y_2)$  means that either  $x_1 < x_2$  or  $x_1 = x_2, y_1 < y_2$ . (Cf. [2], Problem 3.12.3(d).) The topology on the lexicographic square is, as usual, generated by the open intervals

$$(a, b) := \{u \in [0, 1]^2 \mid a < u < b\},$$

$a$  and  $b$  ranging over  $[0, 1]^2$ . As a topological space, the lexicographic square is normal ([2], Problem 1.7.4(d)), compact ([2], Problem 3.12.3(a), [5], Problem 5.C), and connected ([2], Problem 6.3.2(a), [5], Problem 1.I(d)).

Now we redefine a  $\mathbb{R}^2$ -square  $\mathcal{L}$  to be any function  $G$  mapping the lexicographic square  $[0, 1]^2$  to  $\Gamma$ ; in all our examples, it will map a probability distribution  $(1-p, p)$  on  $\{0, 1\}$  and the extra “tie-breaking” component  $q$  to an optimal, under this probability distribution, decision  $G(p, q)$ . The  $\Delta_G$  of  $G$  is

$$\Delta_G := \sup_{(p,q) \in \mathcal{L}} \left( \lambda(p, G(p, q)) - \inf_{\gamma \in \Gamma} \lambda(p, \gamma) \right),$$

where  $\lambda(p, \gamma)$  is defined by (2). In the next section, we will be interested in zero-defect choice functions. The  $\text{Exp}_G : \mathcal{D} \rightarrow \mathbb{R}$  of  $G$  is now defined as

$$\text{Exp}_G(p, q) := \lambda(1, G(p, q)) - \lambda(0, G(p, q)).$$

The following theorem describes a property of the Expected Loss Minimization principle applied to a modification of the FS algorithm which outputs, at each step, a pair  $(p_n, q_n) \in \mathcal{D}$ .

**Theorem 2.** Let  $(x_n)_{n=1}^N$  be a sequence of decisions in  $\mathcal{D}$ . Let  $(p_n, q_n)_{n=1}^N$  be a sequence of pairs in  $\mathcal{D}$  such that  $(p_n, q_n) \in \text{Exp}_G(x_n)$  for all  $n$ . Then

$$\sum_{n=1}^N \lambda(y_n, \gamma_n) \leq \sum_{n=1}^N \lambda(y_n, D(x_n)) + \left( \left( \frac{2}{\sqrt{3}} \right)^K \| \text{Exp}_D \|_{\text{FS}} + 1 \right) \sqrt{ \sum_{n=1}^N p_n(1-p_n) (\text{Exp}_G^2(p_n, q_n) + 1) + N \Delta_G } \tag{4}$$

where  $\Delta_G := \sum_{(p,q) \in \mathcal{D}} p(1-p) (\text{Exp}_G^2(p, q) + 1)$ .

The proof of this theorem will be sketched in the appendix.

### 5 Examples

By a *game* we will mean a pair  $(\Gamma, \lambda)$ , where  $\Gamma$  is the decision space and  $\lambda : \{0, 1\} \times \Gamma \rightarrow \mathbb{R}$  is the loss function. For all games considered in this section decisions  $\gamma \in \Gamma$  may be interpreted as predictions, but we will still refer to them as decisions to avoid confusion with, e.g., FS's forecasts.

#### 5.1 The Square Loss Game

If  $\Gamma = [0, 1]$  and  $\lambda(y, \gamma) = (y - \gamma)^2$ , we have

$$\lambda(1, D(x)) - \lambda(0, D(x)) = (1 - D(x))^2 - (D(x))^2 = 1 - 2D(x) \tag{5}$$

and, for  $G(p, q) := p$ ,

$$\lambda(1, G(p, q)) - \lambda(0, G(p, q)) = (1 - p)^2 - p^2 = 1 - 2p.$$

Therefore,  $|\text{Exp}_G| \leq 1$ , and so Theorem 2 implies

**Corollary 1.** Let  $(x_n)_{n=1}^N$  be a sequence of decisions in  $[0, 1]$ . Let  $(p_n)_{n=1}^N$  be a sequence of pairs in  $[0, 1]$  such that  $p_n \in \text{Exp}_G(x_n)$  for all  $n$ . Then

$$\sum_{n=1}^N \lambda(y_n, p_n) \leq \sum_{n=1}^N \lambda(y_n, D(x_n)) + \frac{1}{\sqrt{2}} \left( \left( \frac{2}{\sqrt{3}} \right)^K \| D^\dagger \|_{\text{FS}} + 1 \right) \sqrt{N}, \tag{6}$$

where  $D^\dagger := 2D - 1$  is the signed version of  $D$ .

### 5.2 The Absolute Loss Game

For this game,  $\lambda(y, \gamma) = |y - \gamma|$  with  $\Gamma = [0, 1]$ . We find

$$\lambda(p, \gamma) = p(1 - \gamma) + (1 - p)\gamma = p + (1 - 2p)\gamma,$$

and so the ideal decision would be

$$\gamma := \begin{cases} 0 & \text{if } p < 1/2 \\ 1 & \text{if } p > 1/2, \end{cases}$$

with an arbitrary choice in the case of a tie,  $p = 1/2$ . This is a discontinuous function of  $p$ , but we can instead use the following continuous choice function on the lexicographic square:

$$G(p, q) := \begin{cases} 0 & \text{if } p < 1/2 \\ 1 & \text{if } p > 1/2 \\ q & \text{if } p = 1/2. \end{cases} \tag{7}$$

Since

$$\lambda(1, \gamma) - \lambda(0, \gamma) = 1 - 2\gamma,$$

$\text{Exp}_D$  is the same as in the square loss case, (5), and

$$\text{Exp}_G(p, q) = 1 - 2G(p, q) = \begin{cases} 1 & \text{if } p < 1/2 \\ -1 & \text{if } p > 1/2 \\ 1 - 2q & \text{if } p = 1/2. \end{cases}$$

Therefore,  $|\text{Exp}_G| \leq 1$ , and we have the following corollary of Theorem 2.

**Corollary 2.**

Let  $\lambda$  be a loss function on  $\Gamma$  with  $\text{Exp}_G$  bounded by 1. Let  $\{x_n\}_{n=1}^N$  be a sequence of observations in  $D$  and  $\{\gamma_n\}_{n=1}^N$  be a sequence of decisions in  $\Gamma$ . Then

### 5.3 The Simple Loss Game

In this subsection we mainly limit ourselves to informal discussions.

The loss function for the simple loss game is the same as for the absolute loss game,  $\lambda(y, \gamma) = |y - \gamma|$ , but  $\Gamma = \{0, 1\}$ . Now the approach we have used so far does not work: since  $\Gamma$  consists of two elements, there is no non-trivial continuous choice function  $G : \Gamma \rightarrow \Gamma$  (every continuous image of  $\Gamma$  is connected: [2], Theorem 6.1.4).

A natural idea ([1]) is to allow Decision Maker to use randomization. The expected loss of an algorithm making decision 1 with probability  $\gamma$  and 0 with probability  $1 - \gamma$  is  $|y - \gamma|$ , where  $y$  is the observed label; therefore, for the simple loss game a randomized decision strategy can guarantee the following analogue of (6):

$$\sum_{n=1}^N \mathbb{E} \lambda(y_n, \gamma_n) \leq \sum_{n=1}^N \lambda(y_n, D(x_n)) + \frac{1}{\sqrt{2}} \left( \left( \frac{2}{\sqrt{3}} \right)^K \|D^\dagger\|_{\text{FS}} + 1 \right) \sqrt{N}, \tag{8}$$

where  $\mathbb{E}$  refers to the strategy’s internal randomization (the decision rules  $D$  can be allowed to take values in  $[0, 1]$ ).

The disadvantage of (8) is that typically we are interested in the strategy’s actual rather than expected loss. Our derivation of (8) shows the role of randomization: with the choice function (7) no randomization is required unless  $p = 1/2$ . Typically, we rarely find ourselves in a situation of complete uncertainty,  $p_n = 1/2$ ; therefore, only a little bit of randomization is needed, essentially for tie breaking. The actual loss will be very close to the expected loss. (It is instructive to compare this with the discussion of Dawid’s example in [15], Subsection 4.4.)

**5.4 The Log Loss Game**

For the log loss game,  $\Gamma = [0, 1]$  and

$$\lambda(y, \gamma) = \begin{cases} -\ln \gamma & \text{if } y = 1 \\ -\ln(1 - \gamma) & \text{if } y = 0 \end{cases}$$

(we temporarily, in this subsection only, allow  $\lambda$  to take the value  $\infty$ ; cf. the discussion at the end of this subsection). As in the square loss case, we use the “perfect” choice function  $G(p, q) := p$ . Since

$$\lambda(1, \gamma) - \lambda(0, \gamma) = -\ln \gamma + \ln(1 - \gamma) = \ln \frac{1 - \gamma}{\gamma}$$

and

$$\sum_{n=1}^N p_n(1 - p_n) \left( \left( \ln \frac{1 - p_n}{p_n} \right)^2 + 1 \right) \leq N \max_{p \in [0,1]} p(1 - p) \left( \left( \ln \frac{1 - p}{p} \right)^2 + 1 \right) \approx 0.526N \leq 0.53N,$$

we obtain from Theorem 2:

**Corollary 3.**  $\sum_{n=1}^N \lambda(y_n, \gamma_n) \leq \sum_{n=1}^N \lambda(y_n, D(x_n)) + 0.73 \left( \left( \frac{2}{\sqrt{3}} \right)^K \left\| \ln \frac{D}{1 - D} \right\|_{\text{FS}} + 1 \right) \sqrt{N}.$

The only problem with this derivation is that the log loss function takes values in  $(-\infty, \infty]$  (since  $\lambda(0, 1) = \lambda(1, 0) = \infty$ ), whereas Theorem 2 assumes that  $\lambda$  takes values in  $\mathbb{R}$ . The details of the corrected derivation are given in [13] (Appendix B).

**5.5 Convex Games**

One can generalize Corollaries 1–3 making them less precise (viz., involve an unspecified constant). In this subsection we consider games  $(\Gamma, \lambda)$  such that  $\Gamma$  is nonempty and



$$C_0 := \inf_{\gamma \in \Gamma} \lambda(0, \gamma), \quad C_1 := \inf_{\gamma \in \Gamma} \lambda(1, \gamma) \tag{9}$$

are finite. As usual,  $\lambda$  is assumed to take values in  $\mathbb{R}$ ; therefore, to cover the log loss game we redefine its decision space to  $\Gamma := (0, 1)$ . It is convenient (see, e.g., [4]) to summarize a game by its

$$\Sigma := \{(x, y) \in \mathbb{R}^2 \mid \exists \gamma \in \Gamma : x \geq \lambda(0, \gamma) \text{ and } y \geq \lambda(1, \gamma)\} ;$$

elements of this set will be called  $(x, y)$ . Superdecisions of the form  $(\lambda(0, \gamma), \lambda(1, \gamma))$  will sometimes be called  $(\gamma, \gamma)$ . We will assume, additionally, that the set  $\Sigma \subseteq \mathbb{R}^2$  is convex and closed. The  $(\lambda, \Sigma)$  of the game is the function

$$f : [C_0, \infty) \rightarrow \mathbb{R} \cup \{\infty\} \\ x \mapsto \inf\{y \mid (x, y) \in \Sigma\} - C_1$$

and its  $(\lambda, \Sigma)$  is

$$g : [C_1, \infty) \rightarrow \mathbb{R} \cup \{\infty\} \\ y \mapsto \inf\{x \mid (x, y) \in \Sigma\} - C_0 ,$$

where, as usual,  $\inf \emptyset := \infty$ ; it is clear that  $f$  and  $g$  are nonnegative everywhere and finite on  $(C_0, \infty)$  and  $(C_1, \infty)$ , respectively.

**Corollary 4.**  $(\Gamma, \lambda)$   $\lambda$   $\Gamma \neq \emptyset$   $\Sigma$   $f$   $g$

$$f'_+(t) = O(t^{-2}), \quad g'_+(t) = O(t^{-2}) \tag{10}$$

$t \rightarrow \infty$   $f'_+$   $g'_+$   $C$   $\gamma_n$

$$\sum_{n=1}^N \lambda(y_n, \gamma_n) \leq \sum_{n=1}^N \lambda(y_n, D(x_n)) + C \left( \left( \frac{2}{\sqrt{3}} \right)^K \|\text{Exp}_D\|_{\text{FS}} + 1 \right) \sqrt{N}$$

$N$   $D$   $\text{Exp}_D$

This corollary (proved in [13], Appendix C) is applicable to the square, absolute, and log loss games; for the log loss game, this follows from the fact that its tails satisfy

$$f'(t) = g'(t) = -\frac{1}{e^t - 1} \sim -e^{-t} = O(t^{-2}) .$$

**Remark.** If the loss function  $\lambda$  is bounded, (10) holds trivially. The right derivatives in (10) can be replaced by left derivatives, since  $|f'_+| \leq |f'_-|$  and  $|g'_+| \leq |g'_-|$  (see, e.g., [9], Theorem 24.1). Condition (10) can be interpreted as saying that the tails should shrink fast enough. The case  $f(t) = g(t) = t^{-1}$  can be considered borderline; Corollary 4 is still applicable in this case, but it ceases to be applicable for tails that shrink less fast.

## 6 Further Research

Two of the most natural directions in which this paper's results can be developed are:

- to extend Theorem 2 and its corollaries to the multi-class case (defensive forecasting for this case is discussed in [14]);
- to formally analyze games with non-convex loss functions along the lines of Subsection 5.3.

## Acknowledgments

I am grateful to the anonymous referees for useful comments. This work was partially supported by MRC (grant S505/65) and Royal Society.

## References

- [1] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44:427–485, 1997.
- [2] Ryszard Engelking. *General Topology*, volume 6 of *Sigma Series in Pure Mathematics*. Heldermann, Berlin, second edition, 1989.
- [3] Yoav Freund. Predicting a binary sequence almost as well as the optimal biased coin. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 89–98, New York, 1996. Association for Computing Machinery.
- [4] Yuri Kalnishkan and Michael V. Vyugin. The Weak Aggregating Algorithm and weak mixability. In *Proceedings of the Eighteenth Annual Conference on Learning Theory*, 2005.
- [5] John L. Kelley. *General Topology*. Van Nostrand, Princeton, NJ, 1957.
- [6] Don Kimber and Philip M. Long. On-line learning of smooth functions of a single variable. *Theoretical Computer Science*, 148:141–156, 1995.
- [7] Jyrki Kivinen and Manfred K. Warmuth. Exponential Gradient versus Gradient Descent for linear predictors. *Information and Computation*, 132:1–63, 1997.
- [8] Philip M. Long. Improved bounds about on-line learning of smooth functions of a single variable. *Theoretical Computer Science*, 241:25–35, 2000.
- [9] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.
- [10] Vladimir Vovk. Non-asymptotic calibration and resolution. These Proceedings.
- [11] Vladimir Vovk. Aggregating strategies. In Mark Fulk and John Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383, San Mateo, CA, 1990. Morgan Kaufmann.
- [12] Vladimir Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.
- [13] Vladimir Vovk. Defensive prediction with expert advice. Technical Report [arXiv:cs.LG/0506041](https://arxiv.org/abs/cs.LG/0506041) (version 2), [arXiv.org](https://arxiv.org/) e-Print archive, July 2005. This is the full version of this paper.
- [14] Vladimir Vovk, Ilia Nourtdinov, Akimichi Takemura, and Glenn Shafer. Defensive forecasting for linear protocols. These Proceedings.

[15] Vladimir Vovk, Akimichi Takemura, and Glenn Shafer. Defensive forecasting. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, January 6–8, 2005, Savannah Hotel, Barbados*, pages 365–372. Society for Artificial Intelligence and Statistics, 2005. Available electronically at <http://www.gatsby.ucl.ac.uk/aistats/>.

## Appendix: Proof Sketch of Theorem 2

This appendix assumes that the reader has read [10] and makes constant references to that paper. All results of [10] were based on the Intermediate Value Theorem. Since we now allow predictions to be elements of the lexicographic square  $\mathcal{X}$ , the Intermediate Value Theorem needs to be replaced with the following lemma.

**Lemma 1.** *Let  $f : \mathcal{X} \rightarrow \mathbb{R}$  be a continuous function. Then the set  $\{x \in \mathcal{X} \mid f(x) = 0\}$  is closed and connected.*

A continuous image of a connected compact set is connected ([2], Theorem 6.1.4) and compact ([2], Theorem 3.1.10). Therefore,  $f(\cdot)$  is a closed interval. ■

The definition of the K29\* algorithm can be easily adapted to the case of predictions in  $\mathcal{X}$ ; this version of the K29\* algorithm will be called the  $\mathcal{X}$ -K29\* algorithm (see [13] for details). We can then prove the following version of Theorem 1 in [10].

**Theorem 3.** *Let  $\mathbf{k} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  be a kernel function.*

$$\mathbf{k}(a, b) = \Phi(a) \cdot \Phi(b), \quad \forall a, b \in \mathcal{X} \times \mathcal{X},$$

where  $\Phi : \mathcal{X} \times \mathcal{X} \rightarrow H$  is a forecast-continuous function and  $\Phi(p, q, x) = \langle \Phi(p, q), x \rangle$  for  $(p, q) \in \mathcal{X} \times \mathcal{X}$  and  $x \in \mathcal{X}$ . Then the  $\mathcal{X}$ -K29\* algorithm achieves a regret of at most  $\mathbf{k}(p_n, q_n)$  for any sequence of points  $(p_n, q_n) \in \mathcal{X} \times \mathcal{X}$ .

$$\left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, q_n, x_n) \right\|^2 \leq \sum_{n=1}^N p_n (1 - p_n) \|\Phi(p_n, q_n, x_n)\|^2 \quad (11)$$

for  $N = 1, 2, \dots$

For a detailed proof, see [13].

In the construction of the FS kernel on  $[0, 1] \times [0, 1]^K = [0, 1]^{K+1}$  given in [10], the fact that the first coordinate of the points in  $[0, 1]^{K+1}$  was special (corresponding to the forecast rather than an attribute) was never used, and we could carry out the same construction taking only the attributes into account, with  $K + 1$  replaced by  $K$ . The resulting kernel will be called the  $\mathcal{X}$ -FS kernel.

Theorem 3 allows us to prove the required results about separate calibration and resolution, decoupling Theorem 1’s calibration-cum-resolution. The case of

calibration is simple: we will just take the feature mapping  $(p, q, x) \mapsto \text{Exp}_G(p, q)$ , where  $G$  is the choice function. Resolution is covered by the following version of Theorem 1 (i.e., Theorem 2 in [10]):

**Theorem 4.** Let  $p_1, \dots, p_N \in [0, 1]$  and  $f : [0, 1]^K \rightarrow \mathbb{R}$  be a function.

$$\left| \sum_{n=1}^N (y_n - p_n) f(x_n) \right| \leq \left( \frac{2}{\sqrt{3}} \right)^K \|f\|_{\text{FS}} \sqrt{\sum_{n=1}^N p_n(1 - p_n)} \tag{12}$$

where  $N \in \mathbb{N}$  and  $f : [0, 1]^K \rightarrow \mathbb{R}$ .

The proof of this result is completely analogous to the proof of Theorem 2 in [10], and we do not reproduce it here. We will complement each prediction  $p_n \in [0, 1]$  to  $(p_n, q_n) \in [0, 1] \times [0, 1]$  choosing  $q_n \in [0, 1]$  arbitrarily.

To achieve both calibration and resolution, we will have to mix the feature mapping  $\Phi_0(p, q, x) := \text{Exp}_G(p, q)$  and the feature mapping  $\Phi_1$  leading to the object FS kernel. The following corollary of Theorem 3 gives a simple way to mix two feature mappings.

**Corollary 5.** Let  $\Phi_j : [0, 1] \times [0, 1] \times \mathbf{X} \rightarrow H_j$ ,  $j = 0, 1$  be feature mappings into Hilbert spaces  $H_j$ .

$$\begin{aligned} & \left\| \sum_{n=1}^N (y_n - p_n) \Phi_j(p_n, q_n, x_n) \right\|^2 \\ & \leq \sum_{n=1}^N p_n(1 - p_n) \left( \|\Phi_0(p_n, q_n, x_n)\|^2 + \|\Phi_1(p_n, q_n, x_n)\|^2 \right) \end{aligned}$$

where  $N \in \mathbb{N}$  and  $j = 0, 1$ .

Define the direct sum  $H$  of  $H_0$  and  $H_1$  as the Cartesian product  $H_0 \times H_1$  equipped with the inner product

$$h \cdot h' = (h_0, h_1) \cdot (h'_0, h'_1) := \sum_{j=0}^1 h_j \cdot h'_j.$$

Now we can define  $\Phi : [0, 1] \times [0, 1] \times \mathbf{X} \rightarrow H$  by

$$\Phi(p, q, x) := (\Phi_0(p, q, x), \Phi_1(p, q, x)) ;$$

the corresponding kernel is

$$\begin{aligned} \mathbf{k}((p, q, x), (p', q', x')) & := \Phi(p, q, x) \cdot \Phi(p', q', x') \\ & = \sum_{j=0}^1 \Phi_j(p, q, x) \cdot \Phi_j(p', q', x') = \sum_{j=0}^1 \mathbf{k}_j((p, q, x), (p', q', x')) , \end{aligned}$$

where  $\mathbf{k}_0$  and  $\mathbf{k}_1$  are the kernels corresponding to  $\Phi_0$  and  $\Phi_1$ , respectively. Applying lexicographic K29\* to this kernel and using (11), we obtain

$$\begin{aligned}
 & \left\| \sum_{n=1}^N (y_n - p_n) \Phi_j(p_n, q_n, x_n) \right\|^2 \\
 & \leq \left\| \left( \sum_{n=1}^N (y_n - p_n) \Phi_0(p_n, q_n, x_n), \sum_{n=1}^N (y_n - p_n) \Phi_1(p_n, q_n, x_n) \right) \right\|^2 \\
 & = \left\| \sum_{n=1}^N (y_n - p_n) \Phi(p_n, q_n, x_n) \right\|^2 \leq \sum_{n=1}^N p_n (1 - p_n) \|\Phi(p_n, q_n, x_n)\|^2 \\
 & = \sum_{n=1}^N p_n (1 - p_n) \sum_{j=0}^1 \|\Phi_j(p_n, q_n, x_n)\|^2 . \quad \blacksquare
 \end{aligned}$$

### Proof Proper

Let us take  $\Phi_0(p, q, x) := \text{Exp}_G(p, q)$  and  $\Phi_1(p, q, x) := (f_m(x))_{m \in M}$ , the latter as in the proof of Theorem 4 (see [10], Appendix B). Remember that  $\|\Phi_1(p, q, x)\|_a \leq 1$ ,  $\forall p, q, x$ . Merging  $\Phi_0$  and  $\Phi_1$  by Corollary 5, we obtain from the proof of Theorem 4 (namely, following the proof of Lemma 2 in [10];  $c_m$  are the Fourier coefficients of  $f$  w.r. to the system  $(f_m)$ ):

$$\begin{aligned}
 \left| \sum_{n=1}^N (y_n - p_n) f(x_n) \right| &= \left| \sum_{m \in M} c_m \sum_{n=1}^N (y_n - p_n) f_m(x_n) \right| \\
 &= \left| \left( \frac{c_m}{a_m} \right)_{m \in M} \cdot \left( \sum_{n=1}^N (y_n - p_n) f_m(x_n) \right)_{m \in M} \right| \\
 &\leq \left\| \left( \frac{c_m}{a_m} \right)_{m \in M} \right\|_a \left\| \left( \sum_{n=1}^N (y_n - p_n) f_m(x_n) \right)_{m \in M} \right\|_a \\
 &= \sqrt{\sum_{m \in M} \frac{c_m^2}{a_m}} \left\| \sum_{n=1}^N (y_n - p_n) \Phi_1(p_n, q_n, x_n) \right\|_a \\
 &\leq \left( \frac{2}{\sqrt{3}} \right)^K \|f\|_{\text{FS}} \sqrt{\sum_{n=1}^N p_n (1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)} . \quad (13)
 \end{aligned}$$

Finally, analogously to (3) and using (13), we obtain for the lexicographic K29\* algorithm with the merged kernel as parameter:

$$\sum_{n=1}^N \lambda(y_n, \gamma_n) = \sum_{n=1}^N \lambda(y_n, G(p_n, q_n))$$

$$\begin{aligned}
 &= \sum_{n=1}^N \lambda(p_n, G(p_n, q_n)) + \sum_{n=1}^N (\lambda(y_n, G(p_n, q_n)) - \lambda(p_n, G(p_n, q_n))) \\
 &= \sum_{n=1}^N \lambda(p_n, G(p_n, q_n)) + \sum_{n=1}^N (y_n - p_n) (\lambda(1, G(p_n, q_n)) - \lambda(0, G(p_n, q_n))) \\
 &\leq \sum_{n=1}^N \lambda(p_n, G(p_n, q_n)) + \sqrt{\sum_{n=1}^N p_n(1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)} \\
 &\leq \sum_{n=1}^N \lambda(p_n, D(x_n)) + N\Delta_G + \sqrt{\sum_{n=1}^N p_n(1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)} \\
 &= \sum_{n=1}^N \lambda(y_n, D(x_n)) - \sum_{n=1}^N (\lambda(y_n, D(x_n)) - \lambda(p_n, D(x_n))) \\
 &\quad + N\Delta_G + \sqrt{\sum_{n=1}^N p_n(1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)} \\
 &= \sum_{n=1}^N \lambda(y_n, D(x_n)) - \sum_{n=1}^N (y_n - p_n) (\lambda(1, D(x_n)) - \lambda(0, D(x_n))) \\
 &\quad + N\Delta_G + \sqrt{\sum_{n=1}^N p_n(1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)} \\
 &\leq \sum_{n=1}^N \lambda(y_n, D(x_n)) + \left(\frac{2}{\sqrt{3}}\right)^K \|\text{Exp}_D\|_{\text{FS}} \sqrt{\sum_{n=1}^N p_n(1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)} \\
 &\quad + N\Delta_G + \sqrt{\sum_{n=1}^N p_n(1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)} \\
 &= \sum_{n=1}^N \lambda(y_n, D(x_n)) + N\Delta_G \\
 &\quad + \left(\left(\frac{2}{\sqrt{3}}\right)^K \|\text{Exp}_D\|_{\text{FS}} + 1\right) \sqrt{\sum_{n=1}^N p_n(1 - p_n) (\text{Exp}_G^2(p_n, q_n) + 1)}. \blacksquare
 \end{aligned}$$

# Defensive Forecasting for Linear Protocols

Vladimir Vovk<sup>1</sup>, Ilia Nouretdinov<sup>1</sup>, Akimichi Takemura<sup>2</sup>, and Glenn Shafer<sup>1,3</sup>

<sup>1</sup> Department of Computer Science, Royal Holloway, University of London,  
Egham, Surrey TW20 0EX, England  
{vovk, ilia, glenn}@cs.rhul.ac.uk

<sup>2</sup> Department of Mathematical Informatics,  
Graduate School of Information Science and Technology, University of Tokyo,  
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan  
takemura@stat.t.u-tokyo.ac.jp

<sup>3</sup> Rutgers Business School – Newark and New Brunswick,  
180 University Avenue, Newark, New Jersey 07102 USA  
gshafer@andromeda.rutgers.edu

**Abstract.** We consider a general class of forecasting protocols, called “linear protocols”, and discuss several important special cases, including multi-class forecasting. Forecasting is formalized as a game between three players: Reality, whose role is to generate objects and their labels; Forecaster, whose goal is to predict the labels; and Skeptic, who tries to make money on any lack of agreement between Forecaster’s predictions and the actual labels. Our main mathematical result is that for any continuous strategy for Skeptic in a linear protocol there exists a strategy for Forecaster that does not allow Skeptic’s capital to grow. This result is a meta-theorem that allows one to transform any constructive law of probability in a linear protocol into a forecasting strategy whose predictions are guaranteed to satisfy this law. We apply this meta-theorem to a weak law of large numbers in inner product spaces to obtain a version of the K29 prediction algorithm for linear protocols and show that this version also satisfies the attractive properties of proper calibration and resolution under a suitable choice of its kernel parameter, with no assumptions about the way the data is generated.

## 1 Introduction

In a recent paper, [14], we suggested a new methodology for designing forecasting strategies. Considering only the simplest case of binary forecasting, we showed that any constructive, in the sense explained below, law of probability can be translated into a forecasting strategy that satisfies this law. In this paper this result is extended to a general class of protocols including multi-class forecasting. In proposing this approach to forecasting we were inspired by [3] and papers further developing [3], although our methods and formal results appear to be completely different.

Whereas the meta-theorem stated in [14] is mathematically trivial, we have to overcome some technical difficulties in the generalization considered in this pa-

per. Our general meta-theorem is stated in §4. The general forecasting protocols covered by this result are introduced and discussed in §§2–3.

In [14] we demonstrated the value of the meta-theorem by applying it to the strong law of large numbers, obtaining from it a kernel forecasting strategy which we called K29. The derivation, however, was informal, involving heuristic transitions to a limit, and this made it impossible to state formally any properties of K29. In this paper we deduce K29 in a much more direct way from the weak law of large numbers and state its properties. (For binary forecasting, this was also done in [13], and the reader might prefer to read that paper first.) The weak law of large numbers is stated and proved in §5, and K29 is derived and studied in §6.

We call the approach to forecasting using our meta-theorem “defensive forecasting”: Forecaster is trying to defend himself when playing against Skeptic. The justification of this approach given in this paper and in [13] is K29’s properties of proper calibration and resolution. Another justification, in a sense the ultimate justification of any forecasts, is given in [12]: defensive forecasts lead to good decisions; this result, however, is obtained for rather simple decision problems requiring only binary forecasts, and we expect that its extensions will require this paper’s results.

The exposition of probability theory needed for this paper is given in [9]. The standard exposition is based on Kolmogorov’s measure-theoretic axioms of probability, whereas [9] states several key laws of probability in terms of a game between the forecaster, the reality, and a third player, the skeptic. The game-theoretic laws of probability in [9] are constructive in that we explicitly construct computable winning strategies for the forecaster in various games of forecasting.

## 2 Forecasting as a Game

Following [9] and [14] we consider the following general forecasting protocol:

### FORECASTING GAME 1

**Players:** Reality, Forecaster, Skeptic

**Parameters:**  $\mathbf{X}$  (sequence of real numbers),  $\mathbf{Y}$  (sequence of real numbers),  $\mathbf{F}$  (sequence of real numbers),  $\mathbf{S}$  (sequence of real numbers),  $\lambda : \mathbf{S} \times \mathbf{F} \times \mathbf{Y} \rightarrow \mathbb{R}$  (function) and  $\mathcal{K}_0 := 1$ .

**Protocol:**

- $\mathcal{K}_0 := 1$ .
- FOR  $n = 1, 2, \dots$ :
  - Reality announces  $x_n \in \mathbf{X}$ .
  - Forecaster announces  $f_n \in \mathbf{F}$ .
  - Skeptic announces  $s_n \in \mathbf{S}$ .
  - Reality announces  $y_n \in \mathbf{Y}$ .
  - $\mathcal{K}_n := \mathcal{K}_{n-1} + \lambda(s_n, f_n, y_n)$ .

END FOR

**Restriction on Skeptic:** Skeptic must choose the  $s_n$  so that his capital is always nonnegative ( $\mathcal{K}_n \geq 0$  for all  $n$ ) no matter how the other players move.



This is a perfect-information protocol: the players move in the order indicated, and each player sees the other player’s moves as they are made. It specifies both an initial value for Skeptic’s capital ( $\mathcal{K}_0 = 1$ ) and a lower bound on its subsequent values ( $\mathcal{K}_n \geq 0$ ). We will say that  $x_n$  are the Forecaster’s moves,  $y_n$  are the Skeptic’s labels,  $(x_n, y_n)$  are the moves, and  $f_n$  are the forecasts.

Book [9] contains several results (game-theoretic versions of limit theorems of probability theory) of the following form: Skeptic has a strategy that guarantees that either a property of agreement between the forecasts  $f_n$  and labels  $y_n$  is satisfied or Skeptic becomes very rich (without risking bankruptcy, according to the protocol). All specific strategies considered in [9] have computable versions. According to Brouwer’s principle (see, e.g., §1 of [11] for a recent review of the relevant literature) they must be automatically continuous; in any case, their continuity can be checked directly. In [14] we showed that, under a special choice of the players’ move spaces and Skeptic’s gain function  $\lambda$ , for any continuous strategy for Skeptic Forecaster has a strategy that guarantees that Skeptic’s capital never increases when he plays that strategy. Therefore, Forecaster has strategies that ensure various properties of agreement between the forecasts and the labels.

The purpose of this paper is to extend the result of [14] to a wide class of Skeptic’s gain functions  $\lambda$ . But first we consider several important special cases of Forecasting Game 1.

### Binary Forecasting

The simplest non-trivial case, considered in [14], is where  $\mathbf{Y} = \{0, 1\}$ ,  $\mathbf{F} = [0, 1]$ ,  $\mathbf{S} = \mathbb{R}$ , and

$$\lambda(s_n, f_n, y_n) = s_n(y_n - f_n). \tag{1}$$

Intuitively, Forecaster gives probability forecasts for  $y_n$ :  $f_n$  is his subjective probability that  $y_n = 1$ . The operational interpretation of  $f_n$  is that it is the price that Forecaster charges for a ticket that will pay  $y_n$  at the end of the  $n$ th round of the game;  $s_n$  is the number (positive, zero, or negative) of such tickets that Skeptic chooses to buy.

### Bounded Regression

This is the most straightforward extension of binary forecasting, considered in [9], §3.2. The move spaces are  $\mathbf{Y} = \mathbf{F} = [A, B]$ , where  $A$  and  $B$  are two constants, and  $\mathbf{S} = \mathbb{R}$ ; the gain function is, as before, (1). This protocol allows one to prove a strong law of large numbers ([9], Proposition 3.3) and a simple one-sided law of the iterated logarithm ([9], Corollary 5.1).

### Multi-class Forecasting

Another extension of binary forecasting is the protocol where  $\mathbf{Y}$  is a finite set,  $\mathbf{F}$  is the set of all probability distributions on  $\mathbf{Y}$ ,  $\mathbf{S}$  is the set of all real-valued functions on  $\mathbf{Y}$ , and

$$\lambda(s_n, f_n, y_n) = s_n(y_n) - \int s_n \, df_n.$$

The intuition behind Skeptic’s move  $s_n$  is that Skeptic buys the ticket which pays  $s_n(y_n)$  after  $y_n$  is announced; he is charged  $\int s_n \, d f_n$  for this ticket.

The binary forecasting protocol is “isomorphic” to the special case of this protocol where  $\mathbf{Y} = \{0, 1\}$ : Forecaster’s move  $f_n$  in the binary forecasting protocol is represented by the probability distribution  $f'_n$  on  $\{0, 1\}$  assigning weight  $f_n$  to  $\{1\}$  and Skeptic’s move  $s_n$  in the binary forecasting protocol is represented by any function  $s'_n$  on  $\{0, 1\}$  such that  $s'_n(1) - s'_n(0) = s_n$ . The isomorphism between these two protocols follows from

$$\begin{aligned} s'_n(y_n) - \int s'_n \, d f'_n &= s'_n(y_n) - s'_n(1)f_n - s'_n(0)(1 - f_n) \\ &= s'_n(y_n) - s'_n(0) - s_n f_n = s_n(y_n - f_n) \end{aligned}$$

(remember that  $y_n \in \{0, 1\}$ ).

**Bounded Mean-Variance Forecasting**

In this protocol,  $\mathbf{Y} = [A, B]$ , where  $A$  and  $B$  are again two constants,  $\mathbf{F} = \mathbf{S} = \mathbb{R}^2$ , and

$$\lambda(s_n, f_n, y_n) = \lambda((M_n, V_n), (m_n, v_n), y_n) = M_n(y_n - m_n) + V_n((y_n - m_n)^2 - v_n) .$$

Intuitively, Forecaster is asked to forecast  $y_n$  with a number  $m_n$  and also forecast the accuracy  $(y_n - m_n)^2$  of his first forecast with a number  $v_n$ . This protocol, although usually without the restriction  $y_n \in [A, B]$ , is used extensively in [9] (e.g., in Chaps. 4 and 5).

An equivalent representation of this protocol is  $\mathbf{Y} = \{(t, t^2) \mid t \in [A, B]\}$ ,  $\mathbf{F} = \mathbf{S} = \mathbb{R}^2$  and

$$\lambda(s_n, f_n, y_n) = \lambda((s'_n, s''_n), (f'_n, f''_n), (t_n, t_n^2)) = s'_n(t_n - f'_n) + s''_n(t_n^2 - f''_n) .$$

The equivalence of the two representations can be seen as follows: Reality’s move  $(x_n, t_n)$  in the first representation corresponds to  $(x_n, y_n) = (x_n, (t_n, t_n^2))$  in the second representation, Forecaster’s move  $(m_n, v_n)$  in the first representation corresponds to  $(f'_n, f''_n) = (m_n, v_n + m_n^2)$  in the second representation, and Skeptic’s move  $(s'_n, s''_n)$  in the second representation corresponds to  $(M_n, V_n) = (s'_n + 2m_n s''_n, s''_n)$  in the first representation. This establishes a bijection between Reality’s move spaces, a bijection between Forecaster’s move spaces, and a bijection between Skeptic’s move spaces in the two representations; Skeptic’s gains are also the same in the two representations:

$$\begin{aligned} s'_n(t_n - f'_n) + s''_n(t_n^2 - f''_n) \\ &= s'_n(t_n - m_n) + s''_n\left(\left((t_n - m_n)^2 + 2(t_n - m_n)m_n + m_n^2\right) - (v_n + m_n^2)\right) \\ &= (s'_n + 2m_n s''_n)(t_n - m_n) + s''_n((t_n - m_n)^2 - v_n) . \end{aligned}$$

### 3 Linear Protocol

Forecasting Game 1 is too general to derive results of the kind we are interested in. In this subsection we will introduce a narrower protocol which will still be wide enough to cover all special cases considered so far.

All move spaces are now subsets of a Euclidean space  $\mathbf{L}$  (i.e.,  $\mathbf{L} = \mathbb{R}^m$  for some positive integer  $m$ ), equipped with the usual dot product “ $\cdot$ ”. The label space is a non-empty bounded subset  $\mathbf{Y} \subset \mathbf{L}$ , Forecaster’s move space  $\mathbf{F}$  is the whole of  $\mathbf{L}$ , and Skeptic’s move space  $\mathbf{S}$  is also the whole of  $\mathbf{L}$ . Skeptic’s gain function is

$$\lambda(s_n, f_n, y_n) = s_n \cdot (y_n - f_n) .$$

Therefore, we consider the following perfect-information game:

FORECASTING GAME 2

**Players:** Reality, Forecaster, Skeptic

**Parameters:**  $\mathbf{X}, \mathbf{L}$  (Euclidean space),  $\mathbf{Y}$  (a non-empty bounded subset of  $\mathbf{L}$ )

**Protocol:**

$$\mathcal{K}_0 := 1 .$$

FOR  $n = 1, 2, \dots$ :

Reality announces  $x_n \in \mathbf{X}$ .

Forecaster announces  $f_n \in \mathbf{L}$ .

Skeptic announces  $s_n \in \mathbf{L}$ .

Reality announces  $y_n \in \mathbf{Y}$ .

$$\mathcal{K}_n := \mathcal{K}_{n-1} + s_n \cdot (y_n - f_n) . \tag{2}$$

END FOR

**Restriction on Skeptic:** Skeptic must choose the  $s_n$  so that his capital is always nonnegative no matter how the other players move.

Let us check that the specific protocols considered in the previous section are covered by this. At first sight, even the binary forecasting protocol is not covered, as Forecaster’s move space is  $[0, 1]$  rather than  $\mathbb{R}$ . It is easy to see, however, that Forecaster’s move  $f_n \notin \overline{\text{co}} \mathbf{Y}$  outside the convex closure  $\overline{\text{co}} \mathbf{Y}$  of the label space (the convex closure  $\overline{\text{co}} A$  of a set  $A$  is defined to be the intersection of all convex closed sets containing  $A$ ) is always inadmissible, in the sense that there exists Skeptic’s reply  $s_n$  making him arbitrarily rich regardless of Reality’s move, and so we can as well choose  $\mathbf{F} := \overline{\text{co}} \mathbf{Y}$ . Indeed, suppose that  $f_n \notin \overline{\text{co}} \mathbf{Y}$  in the linear protocol. Then  $\overline{\text{co}} \mathbf{Y} - f_n$  is a compact convex set not containing the origin. By the hyperplane separation theorem, there exists a vector  $s_n \in \mathbf{L}$  such that

$$s_n \cdot (y_n - f_n) > 0, \quad \forall y_n \in \overline{\text{co}} \mathbf{Y} .$$

By the compactness of  $\overline{\text{co}} \mathbf{Y}$ ,

$$\inf_{y_n \in \mathbf{Y}} s_n \cdot (y_n - f_n) \geq \min_{y_n \in \overline{\text{co}} \mathbf{Y}} s_n \cdot (y_n - f_n) > 0 .$$

Skeptic’s move  $Cs_n$  can make him as rich as he wishes as  $C$  can be arbitrarily large. In what follows, we will usually assume that Forecaster’s move space is  $\overline{\text{co}} \mathbf{Y}$  and use  $\mathbf{F}$  as a shorthand for  $\overline{\text{co}} \mathbf{Y}$ .

Now it is obvious that the binary forecasting, bounded regression, and bounded mean-variance forecasting (in its second representation) protocols are special cases of the linear protocol (perhaps with  $\mathbf{F} = \overline{\text{co}} \mathbf{Y}$ ). For the multi-class forecasting protocol, we should represent  $\mathbf{Y}$  as the vertices

$$y^1 := (1, 0, 0, \dots, 0), \quad y^2 := (0, 1, 0, \dots, 0), \dots, \quad y^m := (0, 0, 0, \dots, 1)$$

of the standard simplex in  $\mathbb{R}^m$ , where  $m$  is the size of  $\mathbf{Y}$ , represent the probability distributions  $f$  on  $\mathbf{Y}$  as vectors  $(f\{y^1\}, \dots, f\{y^m\})$  in  $\mathbb{R}^m$ , and represent the real-valued functions  $s$  on  $\mathbf{Y}$  as vectors  $(s(y^1), \dots, s(y^m))$  in  $\mathbb{R}^m$ .

### 4 Meta-theorem

In this section we prove the main mathematical result of this paper: for any continuous strategy for Skeptic there exists a strategy for Forecaster that does not allow Skeptic’s capital to grow, regardless of what Reality is doing. As in [14], we make Skeptic announce his strategy at the outset of each round rather than at the beginning of the game, and we drop all restrictions on Skeptic. Forecaster’s move space is restricted to  $\mathbf{F} = \overline{\text{co}} \mathbf{Y}$ . The resulting perfect-information game is:

FORECASTING GAME 3

**Players:** Reality, Forecaster, Skeptic

**Parameters:**  $\mathbf{X}, \mathbf{L}$  (Euclidean space),  $\mathbf{Y} \subset \mathbf{L}$  (non-empty and bounded)

**Protocol:**

$\mathcal{K}_0$  is set to a real number.

FOR  $n = 1, 2, \dots$ :

    Reality announces  $x_n \in \mathbf{X}$ .

    Skeptic announces continuous  $S_n : \overline{\text{co}} \mathbf{Y} \rightarrow \mathbf{L}$ .

    Forecaster announces  $f_n \in \overline{\text{co}} \mathbf{Y}$ .

    Reality announces  $y_n \in \mathbf{Y}$ .

$\mathcal{K}_n := \mathcal{K}_{n-1} + S_n(f_n) \cdot (y_n - f_n)$ .

END FOR

**Theorem 1.**  $\mathcal{K}_1 \geq \mathcal{K}_2 \geq \dots$

Fix a round  $n$  and Skeptic’s move  $S_n : \mathbf{F} \rightarrow \mathbf{L}$  (we will refer to  $S_n$  as a vector field in  $\mathbf{F}$ ). Our task is to prove the existence of a point  $f_n \in \mathbf{F}$  such that, for all  $y \in \mathbf{Y}$ ,  $S_n(f_n) \cdot (y - f_n) \leq 0$ .

If for some  $f \in \partial \mathbf{F}$  (we use  $\partial A$  to denote the boundary of  $A \subseteq \mathbf{L}$ ) the vector  $S_n(f)$  is normal and directed exteriorly to  $\mathbf{F}$  (in the sense that  $S_n(f) \cdot (y - f) \leq 0$  for all  $y \in \mathbf{F}$ ), we can take such  $f$  as  $f_n$ . Therefore, we assume, without loss of generality, that  $S_n$  is never normal and directed exteriorly on  $\partial \mathbf{F}$ . Then by

Lemma 1 in Appendix A there exists  $f$  such that  $S_n(f) = 0$ , and we can take such  $f$  as  $f_n$ . ■

**Remark.** Notice that Theorem 1 will not become weaker if the first move by Reality (choosing  $x_n$ ) is removed from each round of the protocol.

## 5 A Weak Law of Large Numbers in the Feature Space

Unfortunately, the usual law of large numbers is not useful for the purpose of designing forecasting strategies (see the discussion in [14]). Therefore, we state a generalized law of large numbers; at the end of this section we will explain connections with the usual law of large numbers. In this section we consider Forecasting Game 2 without the requirement  $\mathcal{K}_0 > 0$  and with the restriction on Skeptic dropped. If we fix a strategy for Skeptic and Skeptic’s initial capital  $\mathcal{K}_0$  (not necessarily a positive number),  $\mathcal{K}_n$  defined by (2) becomes a function of Reality’s and Forecaster’s moves. Such functions will be called

Let  $\Phi : \mathbf{F} \times \mathbf{X} \rightarrow \mathbf{H}$  (as usual,  $\mathbf{F} = \overline{\text{co}} \mathbf{Y}$ ) be a  $\dots$  into an inner product (typically Hilbert) space  $\mathbf{H}$ ;  $\mathbf{H}$  is called the  $\dots$ . The next theorem uses the notion of tensor product; the relevant definitions and facts can be found in Appendix B.

**Theorem 2.**  $\dots$

$$\mathcal{K}_n := \left\| \sum_{i=1}^n (y_i - f_i) \otimes \Phi(f_i, x_i) \right\|^2 - \sum_{i=1}^n \|y_i - f_i\|^2 \|\Phi(f_i, x_i)\|^2 \tag{3}$$

$\dots$  We start by noticing that

$$\begin{aligned} \mathcal{K}_n - \mathcal{K}_{n-1} &= \left\| \sum_{i=1}^{n-1} (y_i - f_i) \otimes \Phi(f_i, x_i) + (y_n - f_n) \otimes \Phi(f_n, x_n) \right\|^2 \\ &\quad - \left\| \sum_{i=1}^{n-1} (y_i - f_i) \otimes \Phi(f_i, x_i) \right\|^2 - \|y_n - f_n\|^2 \|\Phi(f_n, x_n)\|^2 \\ &= 2 \left( \sum_{i=1}^{n-1} (y_i - f_i) \otimes \Phi(f_i, x_i) \right) \cdot ((y_n - f_n) \otimes \Phi(f_n, x_n)) \\ &= 2 \sum_{i=1}^{n-1} ((y_i - f_i) \cdot (y_n - f_n)) (\Phi(f_i, x_i) \cdot \Phi(f_n, x_n)) \end{aligned}$$

(in the last two equalities we used Lemma 2 stated in Appendix B):. Introducing the notation

$$\mathbf{k}((f, x), (f', x')) := \Phi(f, x) \cdot \Phi(f', x'), \tag{4}$$

where  $(f, x), (f', x') \in \mathbf{F} \times \mathbf{X}$ , we can rewrite the expression for  $\mathcal{K}_n - \mathcal{K}_{n-1}$  as

$$\left( 2 \sum_{i=1}^{n-1} \mathbf{k}((f_i, x_i), (f_n, x_n))(y_i - f_i) \right) \cdot (y_n - f_n) .$$

Therefore,  $\mathcal{K}_n$  is the capital process corresponding to Skeptic’s strategy

$$2 \sum_{i=1}^{n-1} \mathbf{k}((f_i, x_i), (f_n, x_n))(y_i - f_i) ; \tag{5}$$

this completes the proof. ■

**More Standard Statements of the Weak Law**

In the rest of this section we explain connections of Theorem 2 with more standard statements of the weak law of large numbers; in this part of the paper we will use some notions introduced in [9]. The rest of the paper does not depend on this material, and the reader may wish to skip the rest of this section.

Let us assume that

$$C := \sup_{(f,x) \in \mathbf{F} \times \mathbf{X}} \|\Phi(f, x)\| < \infty .$$

We will use the notation  $\text{diam}(\mathbf{Y}) := \sup_{y,y' \in \mathbf{Y}} \text{dist}(y, y')$ , where  $\text{dist}(y, y') = \|y - y'\|$  stands for the Euclidean distance in  $\mathbf{L}$ .

For any initial capital  $\mathcal{K}_0$ ,

$$\mathcal{K}_n := \mathcal{K}_0 + \left\| \sum_{i=1}^n (y_i - f_i) \otimes \Phi(f_i, x_i) \right\|^2 - \sum_{i=1}^n \|y_i - f_i\|^2 \|\Phi(f_i, x_i)\|^2$$

is the capital process of some strategy for Skeptic. Suppose a positive integer  $N$  (the duration of the game, or the  $\gamma, \bullet, \mathbf{A}_{i,i}$ ) is given in advance and  $\mathcal{K}_0 := \text{diam}^2(\mathbf{Y})C^2N$ . Then, in the game lasting  $N$  rounds,  $\mathcal{K}_n$  is never negative and

$$\mathcal{K}_N \geq \left\| \sum_{i=1}^N (y_i - f_i) \otimes \Phi(f_i, x_i) \right\|^2 .$$

If we do not believe that Skeptic can increase his capital  $1/\delta$ -fold for a small  $\delta > 0$  without risking bankruptcy, we should believe that

$$\left\| \sum_{i=1}^N (y_i - f_i) \otimes \Phi(f_i, x_i) \right\|^2 \leq \text{diam}^2(\mathbf{Y})C^2N/\delta ,$$

which can be rewritten as

$$\left\| \frac{1}{N} \sum_{i=1}^N (y_i - f_i) \otimes \Phi(f_i, x_i) \right\| \leq \text{diam}(\mathbf{Y})C(N\delta)^{-1/2} . \tag{6}$$

In the terminology of [9], the game-theoretic lower probability of the event (6) is at least  $1 - \delta$ .

The game-theoretic version of Bernoulli’s law of large numbers is a special case of (6) corresponding to  $\Phi(f, x) = 1$ , for all  $f$  and  $x$ ,  $\mathbf{Y} = \{0, 1\}$ , and  $|\mathbf{X}| = 1$  (the last two conditions mean that we are considering the binary forecasting protocol without the objects); as usual, we assume that  $f_i$  are chosen from  $[0, 1]$ . As explained in [9], in combination with the measurability of Skeptic’s strategy guaranteeing (6), this implies that the measure-theoretic probability of the event (6) is at least  $1 - \delta$ , assuming that the  $y_i$  are generated by a probability distribution and that each  $f_i$  is the conditional probability that  $y_i = 1$  given  $y_1, \dots, y_{i-1}$ . This measure-theoretic result was proved by Kolmogorov in 1929 (see [5]) and is the origin of the name “K29 strategy”.

We will see in the next section that the feature-space version (6) of the weak law of large numbers is much more useful than the standard version for the purpose of forecasting, and it will turn out that K29 guarantees (6) with  $\delta = 1$ .

## 6 The K29 Strategy and Its Properties

According to Theorem 1, under the continuity assumption there is a strategy for Forecaster that does not allow  $\mathcal{K}_n$  to grow, where  $\mathcal{K}_n$  is defined by (3). Fortunately (but not unusually), this strategy depends on the feature mapping  $\Phi$  only via the corresponding  $\mathbf{k}$  defined by (4). The continuity assumption needed is that  $\mathbf{k}((f, x), (f', x'))$  should be continuous in  $f$ ; such kernels will be called *continuous kernels*. According to (5), the corresponding forecasting strategy, which we will call the *K29 strategy* with parameter  $\mathbf{k}$ , is to output, on the  $n$ th round, a forecast  $f_n$  satisfying

$$S(f_n) := \sum_{i=1}^{n-1} \mathbf{k}((f_i, x_i), (f_n, x_n))(y_i - f_i) = 0$$

(or, if such  $f_n$  does not exist, the forecast is chosen to be a point  $f_n \in \partial\mathbf{F}$  where  $S(f_n)$  is normal and directed exteriorly to  $\mathbf{F}$ ).

The protocol of this section is essentially that of Forecasting Game 3; as Skeptic ceases to be an active player, it simplifies to:

FOR  $n = 1, 2, \dots$ :  
 Reality announces  $x_n \in \mathbf{X}$ .  
 Forecaster announces  $f_n \in \overline{\text{co}} \mathbf{Y}$ .  
 Reality announces  $y_n \in \mathbf{Y}$ .  
 END FOR

**Theorem 3.** *Let  $\mathbf{k}$  be a continuous kernel. Then the K29 strategy with parameter  $\mathbf{k}$  guarantees (6) with  $\delta = 1$ .*

$$\left\| \sum_{i=1}^n (y_i - f_i) \otimes \Phi(f_i, x_i) \right\| \leq \text{diam}(\mathbf{Y}) C \sqrt{n}, \tag{7}$$

where  $C := \sup_{(f,x) \in \mathbf{F} \times \mathbf{X}} \|\Phi(f, x)\|$ .

The K29 strategy ensures that (3) never increases; therefore,

$$\left\| \sum_{i=1}^n (y_i - f_i) \otimes \Phi(f_i, x_i) \right\|^2 \leq \sum_{i=1}^n \|y_i - f_i\|^2 \|\Phi(f_i, x_i)\|^2 \leq \text{diam}^2(\mathbf{Y}) C^2 n. \quad \blacksquare$$

**Remark.** The property (7) is a special case of (6) corresponding to  $\delta = 1$ ; we gave an independent derivation to make our exposition self-contained and to avoid the extra assumptions used in the derivation of (6), such as the horizon being finite and known in advance.

**Calibration and Resolution**

Two important properties of a forecasting strategy are its calibration and resolution, which we introduce informally. Our discussion in this section extends the discussion in [13], §5, to the case of linear protocols (in particular, to the case of multi-class forecasting). Forecaster’s move space is assumed to be  $\mathbf{F} = \overline{\text{co}} \mathbf{Y}$ .

We say that the forecasts  $f_n$  are *calibrated* if, for any  $f^* \in \mathbf{F}$ ,

$$\frac{\sum_{i=1, \dots, n: f_i \approx f^*} y_i}{\sum_{i=1, \dots, n: f_i \approx f^*} 1} \approx f^*$$

provided  $\sum_{i=1, \dots, n: f_i \approx f^*} 1$  is not too small. (We shorten  $(1/c)v$  to  $v/c$ , where  $v$  is a vector and  $c \neq 0$  is a number.) Proper calibration is only a necessary but far from sufficient condition for good forecasts: for example, a forecaster who ignores the objects  $x_n$  can be perfectly calibrated, no matter how much useful information  $x_n$  contain. (Cf. the discussion in [2].)

We say that the forecasts  $f_n$  are *resolved* if, for any  $(f^*, x^*) \in \mathbf{F} \times \mathbf{X}$ ,

$$\frac{\sum_{i=1, \dots, n: (f_i, x_i) \approx (f^*, x^*)} y_i}{\sum_{i=1, \dots, n: (f_i, x_i) \approx (f^*, x^*)} 1} \approx f^* \tag{8}$$

provided  $\sum_{i=1, \dots, n: (f_i, x_i) \approx (f^*, x^*)} 1$  is not too small.

Instead of “crisp” points  $(f^*, x^*) \in \mathbf{F} \times \mathbf{X}$  we will consider “fuzzy points”  $I : \mathbf{L} \rightarrow [0, 1]$  such that  $I(f^*, x^*) = 1$  and  $I(f, x) = 0$  for all  $(f, x)$  outside a small neighborhood of  $(f^*, x^*)$ . A standard choice would be something like  $I := \mathbb{I}_E$ , where  $E \subset \mathbf{L}$  is a small neighborhood of  $(f^*, x^*)$  and  $\mathbb{I}_E$  is its indicator function, but we will want  $I$  to be continuous (it can, however, be arbitrarily close to  $\mathbb{I}_E$ ).

Let  $(f^*, x^*)$  be a point in  $\mathbf{F} \times \mathbf{X}$ ; we would like the average of  $y_i, i = 1, \dots, n$ , such that  $(f_i, x_i)$  is close to  $(f^*, x^*)$  to be close to  $f^*$ . (Cf. (8).) Fix an admissible Mercer kernel  $\mathbf{k} : (\mathbf{F} \times \mathbf{X})^2 \rightarrow \mathbb{R}$  and consider the “soft neighborhood”

$$I_{(f^*, x^*)}(f, x) := \mathbf{k}((f^*, x^*), (f, x)) \tag{9}$$

of the point  $(f^*, x^*)$ . The following is an easy corollary of Theorem 3.

**Corollary 1.** *If  $\mathbf{k} \geq 0$  then*

$$\left\| \sum_{i=1}^n (y_i - f_i) I_{(f^*, x^*)}(f_i, x_i) \right\| \leq \text{diam}(\mathbf{Y}) C^2 \sqrt{n} \tag{10}$$



Let  $(f^*, x^*) \in \mathbf{F} \times \mathbf{X}$ . Let  $I_{(f^*, x^*)}$  be the inner product on  $\mathbf{H}$  defined by  $I_{(f^*, x^*)}(f, g) = \langle \Phi(f, x), \Phi(g, x^*) \rangle$ .

Let  $\Phi : \mathbf{F} \times \mathbf{X} \rightarrow \mathbf{H}$  be a function taking values in an inner product space  $\mathbf{H}$  and satisfying (4). Theorem 3 then implies

$$\begin{aligned} \left\| \sum_{i=1}^n (y_i - f_i) I_{(f^*, x^*)}(f_i, x_i) \right\| &= \left\| \sum_{i=1}^n (y_i - f_i) (\Phi(f_i, x_i) \cdot \Phi(f^*, x^*)) \right\| \\ &= \left\| \sum_{i=1}^n ((y_i - f_i) \otimes \Phi(f_i, x_i)) \Phi(f^*, x^*) \right\| \\ &\leq \left\| \sum_{i=1}^n (y_i - f_i) \otimes \Phi(f_i, x_i) \right\| \|\Phi(f^*, x^*)\| \leq \text{diam}(\mathbf{Y}) C^2 \sqrt{n} \end{aligned}$$

(the second equality follows from Lemma 4 and the first inequality from Lemma 3 in Appendix B). ■

We can rewrite (10) as

$$\left\| \frac{\sum_{i=1}^n (y_i - f_i) I_{(f^*, x^*)}(f_i, x_i)}{\sum_{i=1}^n I_{(f^*, x^*)}(f_i, x_i)} \right\| \leq \frac{\text{diam}(\mathbf{Y}) C^2 \sqrt{n}}{\sum_{i=1}^n I_{(f^*, x^*)}(f_i, x_i)} \tag{11}$$

(assuming the denominator  $\sum_{i=1}^n I_{(f^*, x^*)}(f_i, x_i)$  is positive); therefore, we can expect proper calibration and resolution in the soft neighborhood of  $(f^*, x^*)$  when

$$\sum_{i=1}^n I_{(f^*, x^*)}(f_i, x_i) \gg \sqrt{n}. \tag{12}$$

In conclusion, we will illustrate (11) on a simple example. Choose the scale  $\sigma > 0$  at which calibration and resolution are sought, suppose  $\mathbf{X}$  is a subset of a Euclidean space, and consider the Gaussian kernel (obviously admissible)

$$\mathbf{k}((f, x), (f', x')) := \exp\left(-\frac{\|(f, x) - (f', x')\|^2}{2\sigma^2}\right); \tag{13}$$

the corresponding soft neighborhoods  $I_{(f^*, x^*)}$  will be Gaussian bells of “size”  $\sigma$ . Fix  $(f^*, x^*) \in \mathbf{F} \times \mathbf{X}$ . If  $n$  is large enough, we expect (12) to hold (indeed, the left-hand side of (12) typically grows as  $\Theta(n)$  as  $n \rightarrow \infty$ ), and so we expect proper calibration and resolution at  $(f^*, x^*)$ .

## 7 Further Research

The main result of this paper is an existence theorem: we did not show how to compute Forecaster’s strategy ensuring  $\mathcal{K}_0 \geq \mathcal{K}_1 \geq \dots$ . (The latter was easy in the case of binary forecasting considered in [14].) It is important to develop computationally efficient ways to find zeros of vector fields. There are several

popular methods for finding zeros, such as the Newton–Raphson method (see, e.g., [7], Chap. 9), but it would be ideal to have efficient methods that are guaranteed to find a zero (or a near zero) in a prespecified time.

In this paper we considered only the case where  $\mathbf{Y}$  is a subset of a finite-dimensional space  $\mathbf{L}$ . There are important protocols (such as the one in [9], p. 360) in which  $\mathbf{Y}$ ,  $\mathbf{F}$ , and  $\mathbf{S}$  are subsets of, e.g., a Banach space. The proof techniques used in this paper, however, depend on the assumption that  $\mathbf{L}$  is finite-dimensional in an essential way.

Finally, it is interesting to study performance guarantees for K29 when used in conjunction with universal kernels [10]. The disadvantage of kernel (13), for example, is that it is not clear how to choose  $\sigma$ : too large  $\sigma$  are useless ((12) holds but calibration and resolution are not useful at a crude scale) and too small  $\sigma$  are not achievable ((12) does not hold). In the binary case, this work is started in [13].

## Acknowledgments

This work was partially supported by MRC (grant S505/65), Royal Society, and the Superrobust Computation Project (Graduate School of Information Science and Technology, University of Tokyo). We are grateful to the anonymous reviewers for their comments.

## References

- [1] Ravi P. Agarwal, Maria Meehan, and Donal O’Regan. *Fixed Point Theory and Applications*. Cambridge University Press, Cambridge, 2001.
- [2] A. Philip Dawid. Probability forecasting. In Samuel Kotz, Norman L. Johnson, and Campbell B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 7, pages 210–218. Wiley, New York, 1986.
- [3] Dean P. Foster and Rakesh V. Vohra. Asymptotic calibration. *Biometrika*, 85:379–390, 1998.
- [4] Alfred Gray. *Tubes*. Birkhäuser, Basel, second edition, 2004.
- [5] Andrei N. Kolmogorov. Sur la loi des grands nombres. *Atti della Reale Accademia Nazionale dei Lincei. Classe di scienze fisiche, matematiche, e naturali. Rendiconti Serie VI*, 185:917–919, 1929.
- [6] Marston Morse. Singular points of vector fields under general boundary conditions. *American Journal of Mathematics*, 51:165–178, 1929.
- [7] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, second edition, 1992.
- [8] Rolf Schneider. *Convex Bodies: The Brunn–Minkowski Theory*. Cambridge University Press, Cambridge, 1993.
- [9] Glenn Shafer and Vladimir Vovk. *Probability and Finance: It’s Only a Game!* Wiley, New York, 2001.
- [10] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[11] Viggo Stoltenberg-Hansen and John V. Tucker. Computable and continuous partial homomorphisms on metric partial algebras. *Bulletin of Symbolic Logic*, 9:299–334, 2003.

[12] Vladimir Vovk. Defensive prediction with expert advice. *These Proceedings*.

[13] Vladimir Vovk. Non-asymptotic calibration and resolution. *These Proceedings*.

[14] Vladimir Vovk, Akimichi Takemura, and Glenn Shafer. Defensive forecasting. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 365–372. Society for Artificial Intelligence and Statistics, 2005. Available electronically at <http://www.gatsby.ucl.ac.uk/aistats/>.

## Appendix A: Zeros of Vector Fields

The following lemma is the main component of the proof of Theorem 1.

**Lemma 1.** Let  $\mathbf{F}$  be a convex set in  $\mathbf{L}$  and  $S : \mathbf{F} \rightarrow \mathbf{L}$  a vector field on  $\mathbf{F}$  such that  $S|_{\mathbf{F}} = 0$ .

If the boundary  $\partial\mathbf{F}$  were assumed to be smooth, the lemma would follow from [6], Theorem  $A_0$  on p. 170; without smoothness, we will have to give an independent proof, starting with a modification of a simple trick from [6].

For each  $\epsilon > 0$  we define

$$\mathbf{F}_\epsilon := \{z \mid \text{dist}(z, \mathbf{F}) \leq \epsilon\},$$

where, as usual,  $\text{dist}(z, \mathbf{F}) := \inf_{f \in \mathbf{F}} \text{dist}(z, f)$ ;  $\mathbf{F}_\epsilon$  is called the  $\epsilon$ -neighborhood of  $\mathbf{F}$  ([4]) or the local parallel set of radius  $\epsilon$  around  $\mathbf{F}$  ([8], §4). Note that  $\mathbf{F}_\epsilon$  can be written as the Minkowski sum  $\mathbf{F}_\epsilon = \mathbf{F} + \epsilon U$  of  $\mathbf{F}$  and the ball  $\epsilon U$  of radius  $\epsilon$  centered at the origin. Therefore, the convexity of  $\mathbf{F}$  implies that  $\mathbf{F}_\epsilon$  is also convex.

Following [6], we extend the vector field  $S$  from  $\mathbf{F}$  to  $\mathbf{F}_1$  ( $\mathbf{F}_\epsilon$  with  $\epsilon = 1$ ) as follows: for any point  $f \in \partial\mathbf{F}_1$  and any  $t \in [0, 1)$ , set

$$S(ty + (1 - t)f) := tS(y) + (1 - t)S(f), \tag{14}$$

where  $y \in \mathbf{F}$  is the unique closest point to  $f$  and  $S(f)$  is defined as  $y - f$ .

Let us first prove that this extension is well defined, i.e., that each point  $p \in \mathbf{F}_1 \setminus \mathbf{F}$  has a unique representation in the form  $ty + (1 - t)f$ , as above. Such a representation exists since we can take  $y$  to be the closest point of  $\mathbf{F}$  to  $p$  and  $f$  to be the point lying on the straight line connecting  $y$  and  $p$  at a distance of 1 from  $y$  in the direction of  $p$ ; it is clear that  $y$  is the closest point to  $f$  in  $\mathbf{F}$  and that  $f \in \partial\mathbf{F}_1$ . Such a representation is unique since  $y$  is uniquely determined as the closest to  $p$  point of  $\mathbf{F}$ ,  $f$  is uniquely determined as the point lying on the straight line connecting  $y$  and  $p$  at a distance of 1 from  $y$  in the direction of  $p$ , and  $t$  is uniquely determined as the distance between  $f$  and  $p$ .

Let us now prove that the extension of  $S$  to  $\mathbf{F}_1$  is continuous. Let

$$p \in (\mathbf{F}_1 \setminus \mathbf{F}) \cup \partial\mathbf{F}, \quad p_k \in \mathbf{F}_1 \setminus \mathbf{F}, \quad k = 1, 2, \dots,$$

be such that  $p_k \rightarrow p$  as  $k \rightarrow \infty$ ; we are required to prove that  $S(p_k) \rightarrow S(p)$ . Represent each  $p_k$  and  $p$  in the form  $p_k = t_k y_k + (1 - t_k) f_k$  and  $p = ty + (1 - t)f$ , as above (i.e.,  $f$  and  $f_k$  are in  $\partial\mathbf{F}_1$  and  $y$  and  $y_k$  are the corresponding closest points in  $\mathbf{F}$ ). It is easy to check that  $y_k \rightarrow y$  (as  $y_k$  and  $y$  are the closest points in  $\mathbf{F}$  to  $p_k$  and  $p$ , respectively), then to check that  $f_k \rightarrow f$  (provided  $f \notin \partial\mathbf{F}$ ), and finally to check that  $t_k \rightarrow t$  (this is true even if  $f \in \partial\mathbf{F}$ , in which case  $t = 1$  and  $t_k \rightarrow 1$ ). This immediately implies  $S(p_k) \rightarrow S(p)$ .

Since  $S$  is never normal and directed exteriorly on  $\partial\mathbf{F}$ ,  $S$  will have no zeros inside  $\mathbf{F}_1 \setminus \mathbf{F}$ . Since the vector field  $S$  is interiorly directed on  $\partial\mathbf{F}_1$  (we will never need a formal definition of “interiorly directed” in this paper), our task would be accomplished if we assumed that the boundary of  $\mathbf{F}_1$  is smooth: we would apply the Poincaré–Hopf theorem to deduce that  $S$  has at least one zero in  $\mathbf{F}_1$  and, therefore, at least one zero in  $\mathbf{F}$ . We will, however, give an argument that does not depend on any smoothness assumptions.

The proof will be complete if we show that the continuous vector field  $S$  in the closed convex set  $\mathbf{F}_1$ , which is normal and interiorly directed on  $\partial\mathbf{F}_1$ , always has at least one zero. We consider the tube  $\mathbf{F}_2$  of radius 2 around  $\mathbf{F}$  and extend the vector field  $S$  to  $\mathbf{F}_2 \setminus \mathbf{F}_1$  by

$$S(ty + (1 - t)f) := S(f)$$

in the notation of (14) but with  $t \in [-1, 0)$  (as before,  $f$  ranges over  $\partial\mathbf{F}_1$  and  $y \in \mathbf{F}$  is the closest point to  $f$ ). Again, it is easy to check that this extension is well defined and continuous.

By the compactness of  $\mathbf{F}_2$ ,

$$C := \max \left( \sup_{f \in \mathbf{F}_2} \|S(f)\|, 1 \right) < \infty.$$

Notice that  $f + tS(f) \in \mathbf{F}_2$  for all  $f \in \mathbf{F}_2$  and all  $t \in [0, 1/C)$  (for  $f \in \mathbf{F}_2 \setminus \mathbf{F}_1$  this follows from the fact that  $f + tS(f)$  lies between  $f$  and the closest point to  $f$  in  $\mathbf{F}$ , and for  $f \in \mathbf{F}_1$  this follows from the fact that the distance between  $f$  and the complement of  $\mathbf{F}_2$  is at least 1). The function

$$\begin{aligned} G : \mathbf{F}_2 &\rightarrow \mathbf{F}_2 \\ f &\mapsto f + \frac{1}{2C} S(f) \end{aligned}$$

is continuous and so, by Schauder’s fixed point theorem (see, e.g., [1], Chap. 4), has a fixed point; it is clear that such a fixed point will be a zero of  $S$ . ■

## Appendix B: Tensor Product

In this appendix we list several definitions and simple facts about tensor products, in the form used in this paper.

The vector space  $\mathbf{L} \otimes \mathbf{H}$  of  $\mathbf{L} = \mathbb{R}^m$  and  $\mathbf{H}$  (an inner product space, perhaps infinite-dimensional) is the vector space  $\mathbf{H}^m$  with the addition and scalar multiplication defined component-wise,

$$(a_1, \dots, a_m) + (b_1, \dots, b_m) := (a_1 + b_1, \dots, a_m + b_m),$$

$$c(a_1, \dots, a_m) := (ca_1, \dots, ca_m),$$

and the inner product

$$(a_1, \dots, a_m) \cdot (b_1, \dots, b_m) := a_1 \cdot b_1 + \dots + a_m \cdot b_m.$$

The tensor product of  $(t_1, \dots, t_m) \in \mathbf{L}$  and  $h \in \mathbf{H}$  is defined to be

$$(t_1, \dots, t_m) \otimes h := (t_1 h, \dots, t_m h).$$

**Lemma 2.**  $t_1, t_2 \in \mathbf{L} \quad h_1, h_2 \in \mathbf{H}$

$$(t_1 \otimes h_1) \cdot (t_2 \otimes h_2) = (t_1 \cdot t_2)(h_1 \cdot h_2).$$

$$\|t \otimes h\|^2 = \|t\|^2 \|h\|^2, \quad t \in \mathbf{L} \quad h \in \mathbf{H}$$

Immediate from the definition. ■

If  $v \in \mathbf{L} \otimes \mathbf{H}$  and  $h \in \mathbf{H}$ , we define the  $vh \in \mathbf{L}$  by the equality

$$(v_1, \dots, v_m)h := (v_1 \cdot h, \dots, v_m \cdot h),$$

where  $(v_1, \dots, v_m) := v$ . The following lemma generalizes (and is an easy implication of) the Cauchy–Schwarz inequality.

**Lemma 3.**  $v \in \mathbf{L} \otimes \mathbf{H} \quad h \in \mathbf{H}$

$$\|vh\| \leq \|v\| \|h\|.$$

Our goal is to prove

$$\|(v_1 \cdot h, \dots, v_m \cdot h)\| \leq \|(v_1, \dots, v_m)\| \|h\|,$$

which is equivalent to

$$(v_1 \cdot h)^2 + \dots + (v_m \cdot h)^2 \leq \|v_1\|^2 \|h\|^2 + \dots + \|v_m\|^2 \|h\|^2;$$

the last inequality follows from  $(v_i \cdot h)^2 \leq \|v_i\|^2 \|h\|^2$  (a special case of the Cauchy–Schwarz inequality). ■

**Lemma 4.**  $t \in \mathbf{L} \quad a, b \in \mathbf{H}$

$$(a \cdot b)t = (t \otimes a)b. \tag{15}$$

If  $t = (t_1, \dots, t_m)$ , both sides of (15) equal  $(t_1(a \cdot b), \dots, t_m(a \cdot b))$ . ■

# Teaching Learners with Restricted Mind Changes

Frank J. Balbach<sup>1</sup> and Thomas Zeugmann<sup>2</sup>

<sup>1</sup> Institut für Theoretische Informatik, Universität zu Lübeck,  
Ratzeburger Allee 160, 23538 Lübeck, Germany  
balbach@tcs.uni-luebeck.de

<sup>2</sup> Division of Computer Science,  
Hokkaido University, Sapporo 060-0814, Japan  
thomas@ist.hokudai.ac.jp

**Abstract.** Within learning theory teaching has been studied in various ways. In a common variant the teacher has to teach all learners that are restricted to output only consistent hypotheses. The complexity of teaching is then measured by the maximum number of mistakes a consistent learner can make until successful learning. This is equivalent to the so-called teaching dimension. However, many interesting concept classes have an exponential teaching dimension and it is only meaningful to consider the teachability of finite concept classes.

A refined approach of teaching is proposed by introducing a neighborhood relation over all possible hypotheses. The learners are then restricted to choose a new hypothesis from the neighborhood of their current one. Teachers are either required to teach finitely or in the limit. Moreover, the variant that the teacher receives the current hypothesis of the learner as *feedback* is considered.

The new models are compared to existing ones and to one another in dependence of the neighborhood relations given. In particular, it is shown that feedback can be very helpful. Moreover, within the new model one can also study the teachability of infinite concept classes with potentially infinite concepts such as languages. Finally, it is shown that in our model teachability and learnability can be rather different.

## 1 Introduction

Teaching has been modeled and investigated in various ways within algorithmic learning theory. Already in Angluin's query model [1, 2] the oracles have some characteristics of teachers. However, they remain completely passive. In order to study teachers in a more active role, several models have been developed, each of which follows one of two basically different approaches.

In the first approach, the goal is to find a teacher  $T$  and a learner such that a given learning task can be carried out by them. For the inductive inference framework, Freivalds *et al.* [8] and Jain *et al.* [14] developed a model in which a rather implicit teacher provides the learning strategy with good examples. Jackson and Tomkins [13] as well as Goldman and Mathias [10, 15] defined models

of teacher/learner pairs where teachers and learners are constructed explicitly. In all these models, some kind of adversary disturbing the teaching process is necessary to avoid collusion between the teacher and the learner. Angluin and Kriškis' [3, 4] model prevents collusion by giving incompatible hypothesis spaces to teacher and learner. This makes simple encoding of the target impossible.

In the second approach, a teacher has to be found that teaches  $\epsilon$ -learners. This prevents collusion, since teaching happens the same way for all learners and cannot be tailored to a specific one. Goldman [11] and Goldman and Kearns [9] substitute the adversarial teacher in the online learning model by a helpful one selecting good examples. They investigate how many mistakes a consistent learner can make in the worst case. In Shinohara and Miyano's [17] model the teacher produces a set of examples for the target concept such that it is the only consistent one in the concept class. The size of this set is the same as the worst case number of mistakes in the online model. This number is termed the *teaching dimension* of the target. Because of this similarity we will from now on refer to both models as the *teaching dimension model*.

One difficulty of teaching in the TD-model results from the teacher not knowing anything about the learners besides them being consistent. In reality a teacher can benefit a lot from knowing the learners' behavior or their current hypotheses. It is therefore natural to ask how teaching can be improved if the teacher may observe the learners' hypotheses after each example.

After translating this question into the TD-model, one sees that there is no gain in sample size at all. The current hypothesis of a consistent learner reveals nothing about its following hypothesis. Even if the teacher knew the hypothesis and provided a special example in response, he can only be sure that the learner's next hypothesis will be consistent. But this was already known to the teacher.

In this paper we extend the TD-model by a neighborhood relation over all hypotheses and by the requirement that all learners may only switch to a hypothesis in the neighborhood of their current one. We then compare basically two variants: In the first, the teacher receives the learner's hypothesis after every example taught. In the second, the teacher has no feedback available. It turns out that in the extended model the existence of feedback can really make a difference. Some concept classes can be taught much faster with feedback than without and some cannot be taught unless feedback is available to the teacher.

As a side effect the model can be used to study the teachability of infinite classes with potentially infinite concepts, e.g., languages. In the class containing all finite languages, for example, all concepts have an infinite teaching dimension and are thus unteachable in the TD-model. With appropriate neighborhood relations this class can be taught, as we shall show in Section 3.

## 2 Preliminaries

A concept  $c$  is a subset of an instance space  $X$  and a concept class is a set of concepts over  $X$ . We consider two instance spaces:  $\{0, 1\}^n$  for Boolean functions and  $\Sigma^*$  for languages over a finite and non-empty alphabet  $\Sigma$ . By  $\mathcal{X} = X \times \{0, 1\}$

we denote the set of  $\langle x, b \rangle$  over  $X$ . An example  $(x, b)$  is either  $\langle x, 1 \rangle$ , if  $b = 1$ , or  $\langle x, 0 \rangle$ , if  $b = 0$ . A concept  $c$  is  $\langle x, b \rangle$  with  $(x, b)$  iff  $x \in c \Leftrightarrow b = 1$ .

Let  $R$  be a set of strings.  $R \subseteq \mathcal{C}$  iff there is a function  $\gamma: R \times X \rightarrow \{0, 1\}$  with  $\mathcal{C} = \{C_r \mid r \in R\}$ , where  $C_r = \{x \mid \gamma(r, x) = 1\}$ . The length of  $r$  is denoted by  $|r|$  and  $\text{size}(c) := \min\{|r| \mid C_r = c\}$  for every  $c \in \mathcal{C}$ . For any set  $S$ , we denote by  $\text{card}(S)$  its cardinality and by  $S^*$  the set of all finite tuples over  $S$ . We use the symbols  $\circ$  for concatenation of tuples and  $\Delta$  for the symmetric difference of two sets. Let  $c$  be a concept and let  $\mathbf{x} \in \mathcal{X}^*$  be a list of examples, then  $\text{in}(\mathbf{x}, c)$  is the set of all examples in  $\mathbf{x}$  that are inconsistent with  $c$ .

A  $\nu$ -neighborhood for a concept  $c$  with respect to  $\mathcal{C}$  is a set  $S$  of examples such that  $c$  is the only concept in  $\mathcal{C}$  consistent with  $S$ . The  $\nu$ -teaching dimension of  $\mathcal{C}$  is the size of the smallest teaching set for  $c$ , the teaching dimension of  $\mathcal{C}$  is  $\text{TD}(\mathcal{C}) = \max\{\text{td}_\nu(c) \mid c \in \mathcal{C}\}$ .

For studying feedback, the learners in our model have to evolve over time. We adopt the online learning model and divide the teaching process into rounds. In each round the teacher provides an example to the learner who then computes a hypothesis from  $R$ . At the end of the round the teacher observes this hypothesis.

Thus, we describe a teacher by a function  $T: R \times R^* \rightarrow \mathcal{X}$  receiving a concept's representation and a sequence of previously observed hypotheses as input and outputting an example.

A learner can be described by a function  $L: \mathcal{X}^* \rightarrow R$  receiving a sequence of examples as input and outputting a hypothesis. Let  $\nu \subseteq R \times R$  be a relation over  $R$ . Then  $L$  is called  $\nu$ -admissible to  $\nu$  iff  $\forall \mathbf{x} \in \mathcal{X}^* \forall z \in \mathcal{X} [(L(\mathbf{x}), L(\mathbf{x} \circ z)) \in \nu]$ , that is  $\nu$  defines the admissible mind changes of  $L$ . Now,  $(R, \nu)$  is a directed graph and we define the neighborhood of  $r \in R$  as  $\text{nb}(r) := \{s \in R \mid (r, s) \in \nu\} \cup \{r\}$  and denote by  $\text{dist}(r, s)$  the length of a shortest path from  $r$  to  $s$ .

In the TD-model, the learner is required to always output a consistent hypothesis. Since in the restricted model all admissible hypotheses might be inconsistent, we have to modify this demand. We require that  $L$  chooses only among the admissible hypotheses with least error with respect to the known examples. Moreover, we require a form of  $\nu$ -admissibility:  $L$  may only change its hypothesis if the new one has a smaller error. This ensures that  $L$  will not change its mind after reaching a correct hypothesis. On the other hand, we also require  $L$  to search for a better hypothesis if it receives an inconsistent example. Otherwise,  $L$  could stay at the initial hypothesis forever and teaching were impossible.

**Definition 1.** Let  $R \subseteq \mathcal{C}$  and  $\nu \subseteq R \times R$ . A  $\nu$ -learner  $L: \mathcal{X}^* \rightarrow R$  is  $\nu$ -admissible to  $\nu$  if  $L(\emptyset) = h_0$  and  $\forall \mathbf{x} \in \mathcal{X}^* \forall z \in \mathcal{X}$

- (1)  $(L(\mathbf{x}), L(\mathbf{x} \circ z)) \in \nu$
- (2)  $L(\mathbf{x}) \neq L(\mathbf{x} \circ z) \Rightarrow \text{card}(\text{in}(\mathbf{x} \circ z, C_{L(\mathbf{x})})) > \text{card}(\text{in}(\mathbf{x} \circ z, C_{L(\mathbf{x} \circ z)}))$
- (3)  $\text{card}(\text{in}(\mathbf{x} \circ z, C_{L(\mathbf{x})})) > \text{card}(\text{in}(\mathbf{x} \circ z, C_s)) \Rightarrow L(\mathbf{x} \circ z) \in \arg \min_{s \in \text{nb}(L(\mathbf{x}))} \text{card}(\text{in}(\mathbf{x} \circ z, C_s))$

We briefly remark that one can think of many plausible variants of the above definition. For instance, the learner could be allowed to change its mind on a



consistent example if its hypothesis is inconsistent with an example received earlier. In this paper, however, all learners follow Definition 1.

The teaching process for a concept  $c \in \mathcal{C}_r$  is fully described by a teacher  $T$  and a learner  $L$  together with an initial hypothesis  $h_0$ . Such a process will result in a series  $(h_i)_{i \in \mathbb{N}}$  of hypotheses and a series  $(z_i)_{i \in \mathbb{N}}$  of examples:  $h_{i+1} = L(z_0, \dots, z_i)$  and  $z_i = T(r, (h_0, \dots, h_i))$ .

**Definition 2.**  $\mathcal{C} \subseteq \mathcal{C}_r$  is teachable to  $\nu$ -learners in the limit with feedback if there exists a teacher  $T: R \times \mathbb{N} \rightarrow \mathcal{X}$  and a learner  $L: \mathcal{X}^{\mathbb{N}} \rightarrow \mathcal{H}$  such that for every  $c \in \mathcal{C}$  there exists a round  $r \in R$  and a hypothesis  $h_0 \in \mathcal{H}$  such that  $(h_i)_{i \in \mathbb{N}}$  is a teaching time of  $T$  on  $r$  and  $L(h_i)_{i \in \mathbb{N}} = c$ .

Note that an infinite teaching time does not imply unteachability of a concept.

For studying the influence of feedback, we also have to define teaching without feedback. In this situation the teacher is modeled as a function  $T: R \times \mathbb{N} \rightarrow \mathcal{X}$ , where the second argument specifies the round. The series of hypotheses is then given by  $h_{i+1} = L(T(r, 0), \dots, T(r, i))$ . With this notation the definition of teaching without feedback is literally the same as Definition 2.

In the situation with feedback the teacher can stop teaching as soon as the learner has reached the goal. If there is no feedback, the teacher may or may not know when to stop. A teacher stopping after finitely many examples and still ensuring the learning success is said to teach finitely. More formally we consider  $T: R \times \mathbb{N} \rightarrow \mathcal{X} \cup \{\perp\}$  where  $\perp$  means ‘‘teaching has stopped.’’

With feedback we do not need to distinguish teaching finitely from teaching in the limit and we shall call this kind of teaching simply teachable.

**Definition 3.**  $\mathcal{C} \subseteq \mathcal{C}_r$  is finitely teachable to  $\nu$ -learners without feedback if there exists a teacher  $T: R \times \mathbb{N} \rightarrow \mathcal{X} \cup \{\perp\}$  and a learner  $L: \mathcal{X}^{\mathbb{N}} \rightarrow \mathcal{H}$  such that for every  $c \in \mathcal{C}$  there exists a round  $r \in R$  and a hypothesis  $h_0 \in \mathcal{H}$  such that  $j = \min\{i \mid T(r, i) = \perp\}$  and  $L(h_i)_{i \in \mathbb{N}} = c$ .

Setting  $\nu = R \times R$  in Definition 3 gives the teacher-directed learning model [11] having no restriction on hypothesis changes. Theorem 4 justifies the use of  $\nu$ 's for studying the impact of feedback on the teaching process.

**Theorem 4.** Let  $\mathcal{C} \subseteq \mathcal{C}_r$ . The following are equivalent: (1)  $\mathcal{C}$  is teachable to  $\nu$ -learners with feedback. (2)  $\mathcal{C}$  is finitely teachable to  $\nu$ -learners without feedback. (3)  $\mathcal{C}$  is teachable to  $\nu$ -learners without feedback. (c)  $c \in \mathcal{C}$  and  $(c) < \infty$ .

The implication 1.  $\Rightarrow$  2.  $\Rightarrow$  3. is clear from the definitions.

It remains to show 3.  $\Rightarrow$  1. Let  $\mathcal{C}$  be teachable to  $\nu$ -learners without feedback for  $\nu = R \times R$  and let  $c \in \mathcal{C}$ . We first prove that for all  $c \in \mathcal{C}$ ,  $(c) < \infty$ .

Suppose there is a  $c^* \in \mathcal{C}$  with  $\text{card}(\mathcal{C}_{c^*}) = \infty$ . Then there is a  $\nu$ -learner  $L$  always assuming a consistent hypothesis not representing  $c^*$ . This is possible because there is no finite set of examples specifying  $c^*$  and because every hypothesis can be reached from every other. Obviously  $L$  cannot be taught  $c^*$  in the limit, not even with feedback; a contradiction.

Now, since all teaching dimensions are finite, we can define a teacher  $T$  that outputs for each  $c \in \mathcal{C}$  a teaching sequence and stops.  $T$  does not need any feedback. Clearly,  $T$  teaches  $\mathcal{C}$  to all  $\nu$ -learners finitely and without feedback, because at the end of teaching there is only one consistent hypothesis left which is certainly reachable.

To see that  $T$  has optimal teaching time for  $c \in \mathcal{C}$  with respect to all three teaching models, we consider a  $\nu$ -learner  $L$  that always outputs a consistent hypothesis not representing  $c$ , unless  $c$  is the only consistent concept in which case  $L$  outputs a representation for  $c$ . It is easy to see that  $c$  cannot be taught to  $L$  with less than  $\text{card}(\mathcal{C}_{c^*})$  examples, no matter whether or not feedback is allowed. ■

Note that Theorem 4 relies on the fact that neither the teacher nor the learners nor the function  $\gamma$  are required to be recursive. Adding these requirements leads to new questions which we skip here due to space constraints.

### 3 Comparison of the Teaching Models

In this section we will apply the new framework to the class  $\mathcal{C}_{fin}$  of all finite languages over an alphabet  $\Sigma$ . This class cannot be taught in the TD-model. By using different  $\nu$ -restrictions we demonstrate various effects.

We fix any total ordering on all strings over  $\Sigma$  and use as representation language  $R$  the set of all comma-separated ordered lists of strings over  $\Sigma$ , i.e.,  $r = w_1, \dots, w_m \in R$  represents the language  $\{w_1, \dots, w_m\}$ . To simplify proofs later, we set  $|r| := \sum_{i=1}^m |w_i|$ , i.e., without counting the commas. We define the allowed transitions from  $r$  to  $s$  by  $(r, s) \in \nu$  iff  $\text{card}(\mathcal{C}_r \Delta \mathcal{C}_s) \leq 1$ . The initial hypothesis is the empty string  $\varepsilon$  representing the empty concept. Now we have:

**Fact 5.**  $\mathcal{C}_{fin}, \nu, \nu', \nu'', \nu''', \nu'''' \in \mathcal{C}_{fin}, \nu, \nu', \nu'', \nu''', \nu'''' \in \mathcal{C}_{fin}$

For a finite language with representation  $w_1, \dots, w_m$  a teacher simply presents all positive examples  $(w_1, 1), \dots, (w_m, 1)$ . In every round, the learners may either add or remove a string from their hypothesis. Starting at the empty language, there is only one possibility to stay consistent with the examples, namely by adding them to the hypothesis. Therefore, after  $m$  rounds all  $\nu$ -learners have arrived at the target hypothesis. ■

Feedback can be utilized when the restriction is modified. We define  $(r, s) \in \nu'$  iff  $\mathcal{C}_s = \mathcal{C}_r \cup \{w_1, w_2\}$  for some  $w_1, w_2 \in \Sigma^*$  or  $\mathcal{C}_s = \mathcal{C}_r \setminus \{w_1\}$ . In both cases, we require that the size of the hypotheses may at most double each round:  $|s| \leq 2|r|$ . In the special case  $r = \varepsilon$  we allow every singleton concept as neighbor:  $(\varepsilon, s) \in \nu'$  for all  $s$  with  $\text{card}(\mathcal{C}_s) = 1$ . For  $\nu'$ -learners there is a big difference in teaching time between teaching with and without feedback.

**Fact 6.**  $\mathcal{C}_{fin}$  is  $\nu'$ -teachable iff  $O(\text{card}(c)) \leq O(\text{size}(c))$

All  $\nu'$ -learners may either add two strings to their hypothesis or remove one. As a consequence, whenever a  $\nu'$ -learner receives a positive example, he can add it to the hypothesis and “invent” another string and add it to the hypothesis as well. Due to the size restriction there are always only finitely many strings that can be invented.

Let  $c^* = \{w_1, \dots, w_m\}$  be a target concept. A teacher with feedback first teaches all strings  $w_i$  as positive examples. After  $w_m$  the hypothesis of each learner contains  $c^*$  plus at most  $m$  invented strings  $u_1, \dots, u_\ell$ . From the feedback, the teacher gets to know these strings and can teach them as negative examples. Since at most one string can be removed per round, the learners have to remove the negative example they are taught and thus arrive at the correct hypothesis after  $\ell$  rounds. Altogether teaching takes at most  $2m = 2\text{card}(c^*)$  rounds. ■

**Fact 7.**  $\mathcal{C}_{fin}$  is  $\nu'$ -teachable iff  $\Omega(2^{\text{size}(c)}) \leq \text{card}(c)$

A suitable teacher is defined as follows. Let  $c \in \mathcal{C}_{fin}$ . First of all, the teacher gives all strings of length at most  $2\text{size}(c)$  that are not in  $c$  as negative examples. Afterwards, all strings in  $c$ , starting with a longest one, are taught as positive examples. The initial hypothesis is consistent with all negative examples, hence no hypothesis change happens. During the positive examples, the learners cannot include any strings outside of  $c$  into their hypotheses, since all these strings either have been ruled out by the negative examples or are too long to be included. Also, since a longest string is taught first, the hypothesis growth limitation cannot be violated by positive examples included later. Hence, all  $\nu'$ -learners must reach the target hypothesis after the positive examples are taught.

For the lower bound, let  $T$  be a teacher that teaches  $\mathcal{C}_{fin}$  finitely without feedback to all  $\nu'$ -learners. Let  $\mathbf{a}$  and  $\mathbf{b}$  be symbols from the alphabet and  $c^* = \{\mathbf{a}^m, \mathbf{b}\}$  a concept of size  $m + 1$  for an arbitrary  $m > 2$ . Let  $z_0, \dots, z_M$  be all examples taught by  $T$  on concept  $c^*$ . Let  $L$  be a  $\nu'$ -learner.

Clearly both strings,  $\mathbf{a}^m$  and  $\mathbf{b}$ , must occur as positive examples, otherwise the  $\nu'$ -learner  $L_0$  that never “invents” a string could not be taught. Moreover,  $\mathbf{a}^m$  must occur before  $\mathbf{b}$ , since otherwise  $L_0$  would at some point have  $\mathbf{b}$  as hypothesis. But because of the growth restriction,  $\mathbf{b}$  cannot be changed to  $\mathbf{a}^m$ ,  $\mathbf{b}$  later, thus  $L_0$  cannot learn  $c^*$ . Let  $z_{j_1} = (\mathbf{a}^m, 1)$  be the the first occurrence of  $\mathbf{a}^m$  and let  $z_{j_2} = (\mathbf{b}, 1)$  be the first occurrence of  $\mathbf{b}$ .

It suffices to show that  $z_1, \dots, z_M$  contains all strings of length at most  $m - 1$ . This implies  $M \geq 2^m - 1 = \Omega(2^{\text{size}(c^*)})$ . Assume there were a string  $\hat{w} \notin c^*$  with  $|\hat{w}| \leq m - 1$  which is not taught. We give a  $\nu'$ -learner  $L$  that does not arrive at  $c^*$  during teaching. On  $z_{j_1}$ ,  $L$  switches to hypothesis  $h_{j_1+1} = \mathbf{a}^m$  and does not change it until  $z_{j_2}$  arrives. Then  $L$  chooses the hypothesis  $h_{j_2+1} = \mathbf{a}^m, \mathbf{b}, \hat{w}$  which is incorrect, but consistent with the examples so far. The length restriction is

obeyed, since  $\text{size}(\mathbf{a}^m) = m = |\mathbf{b}| + |\hat{w}|$ . From then on,  $L$  will never change the hypothesis, since the only inconsistent example,  $(\hat{w}, 0)$ , is never taught according to the assumption.

As  $m$  can be chosen arbitrarily large, there is no bound on the number of examples needed that depends on  $\text{card}(c^*)$  only. ■

If we remove the size restriction from  $\nu'$  we obtain  $\nu''$ .

**Fact 8.**  $\mathcal{C}_{fin}$  is not teachable to  $\nu''$ -learners in the limit without feedback.

Suppose there is a teacher that finitely teaches  $\mathcal{C}_{fin}$  to  $\nu''$ -learners without feedback. Let  $c = \{w_1, w_2\} \in \mathcal{C}_{fin}$ . Then a learner that, when the second positive example arrives, “invents” a word not occurring in the examples does not arrive at a correct hypothesis, a contradiction.

Next, we describe a teacher  $T$  which teaches  $\mathcal{C}_{fin}$  finitely with feedback. On  $c \in \mathcal{C}_{fin}$ ,  $T$  first gives all positive examples. This may lead to at most  $\text{card}(c)$  superfluous strings in the hypothesis of a  $\nu''$ -learner.  $T$  observes these strings and gives them as negative examples, thus forcing all learners to remove the excessive strings and to reach the correct hypothesis.

A teacher for teaching  $\mathcal{C}_{fin}$  in the limit without feedback, first teaches all positive examples. Again, a  $\nu''$ -learner’s hypothesis may contain finitely many excessive strings. By teaching all strings outside the target concept, the superfluous strings can be removed in the limit. ■

Finally we define  $\nu'''$ . It differs from  $\nu''$  in that a string may only be removed from the hypothesis if neither its predecessor nor its successor (with respect to the fixed ordering on  $\Sigma^*$ ) is contained in the hypothesis.

**Fact 9.**  $\mathcal{C}_{fin}$  is not teachable to  $\nu'''$ -learners in the limit without feedback.

Suppose there is a teacher  $T$  which teaches  $\mathcal{C}_{fin}$  to  $\nu'''$ -learners in the limit without feedback. Let  $c^* = \{w_1, w_2, w_3\} \in \mathcal{C}_{fin}$  and let  $(z_i)_{i \in \mathbb{N}}$  be the sequence of examples taught by  $T$  on  $c^*$ . All three strings must occur in the example sequence, otherwise the learner that does not “make up” strings could not be taught. Let  $z_{j_i} = (w_i, 1)$  be first occurrence of  $w_i$  for  $i = 1, 2, 3$ . Without loss of generality, we assume  $j_1 < j_2 < j_3$ .

We now construct a  $\nu'''$ -learner  $L$  which fails on the above example sequence. After  $z_{j_1}$ ,  $L$ ’s hypothesis is  $w_1$ . When taught  $z_{j_2}$ ,  $L$  adds  $w_2$  to the hypothesis, as well as a string  $u_1 \notin c^*$  such that (1) neither  $u_1$  nor its successor  $u_2$  occurs in  $z_1, \dots, z_{j_3}$ , and (2)  $u_2 \notin c^*$ . When taught  $z_{j_3}$ ,  $L$  adds  $w_3$  and  $u_2$  to the hypothesis. Adding  $u_2$  is possible, because it has not yet occurred as negative example. At this point  $L$ ’s hypothesis contains the strings  $u_1$  and  $u_2$  neither of which can be deleted any more. Thus,  $L$  cannot end up with a correct hypothesis (because of the definition of  $\nu'''$ ), a contradiction.

Teaching  $\mathcal{C}_{fin}$  to  $\nu'''$ -learners finitely with feedback can be done as follows. Let  $c^* \in \mathcal{C}_{fin}$  be the target concept. The teacher first teaches all negative examples

that are predecessors or successors of a string in  $c^*$ . Then all positive examples are taught and as soon as the teacher discovers that a learner has introduced a wrong string  $u$  into the hypothesis, the negative example  $(u, 0)$  is given. The string  $u$  cannot be predecessor or successor of any other string in the hypothesis and is thus deleted from the hypothesis. After at most  $(2 + 1 + 1) \cdot \text{card}(c) = O(\text{size}(c))$  examples all  $\nu'''$ -learners have reached the target. ■

If we denote by  $\mathcal{C}_{\text{no feedback}}, \mathcal{C}_{\text{feedback}}, \mathcal{C}_{\text{in the limit}}$  the set of all  $(\mathcal{C}, R, \nu, h_0)$  such that  $\mathcal{C}$  is finitely teachable without feedback, with feedback or in the limit, respectively, we have just proved the following theorem.

**Theorem 10.**  $\mathcal{C}_{\text{no feedback}} \subset \mathcal{C}_{\text{feedback}} \subset \mathcal{C}_{\text{in the limit}}$

The teaching times in our model can hardly be compared to the teaching dimension, since the latter depends only on  $\mathcal{C}$ , whereas different choices of  $\nu$  can lead to different teaching times for the same  $\mathcal{C}$ .

### 4 Finding Teachers

The problem of finding an optimal teacher (with or without feedback) for  $\nu$ -learners is NP-hard, since it is a generalization of finding an optimal teaching set, namely if  $\nu = R \times R$  (see [17, 9, 5]).

Concept classes over finite instance spaces can always be taught in the TD-model. Given  $\nu$ -learners, however, the first question is whether teaching is possible at all. We shall show that this is difficult to decide in general.

The next theorem assumes that  $\mathcal{C}$  and  $\nu$  over an instance space  $X$  and representation language  $R$  are represented as a 0-1-valued matrix with  $\text{card}(R)$  rows and  $\text{card}(X) + \text{card}(R)$  columns. Each row describes the represented concept in the first  $\text{card}(X)$  bits, and its neighborhood in the last  $\text{card}(R)$  bits (cf. Fig 1).

	$x_1$	$\bar{x}_1$	$x_2$	$\bar{x}_2$	$x_3$	$\bar{x}_3$	$x_4$	$\bar{x}_4$	$w$	$y_1$	$y'_1$	$y_2$	$y'_2$	$y_3$	$y'_3$	$y_4$	$y'_4$	$r_0$	$r_1$	$r_2$	$r_3$	$s_0$	$s_1$	$s_2$	$s_3$	$s_4$	$s^*$	
$r_0$	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
$r_1$	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
$r_2$	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
$r_3$	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
$s_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0
$s_1$	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0
$s_2$	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0
$s_3$	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
$s_4$	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
$s^*$	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Fig. 1.** Example for the reduction in Theorem 11 for the formula  $F = (v_1 \vee \bar{v}_2 \vee v_3) \wedge (v_2 \vee v_4 \vee v_1) \wedge (\bar{v}_1 \vee v_3 \vee \bar{v}_4)$ . The left part of the matrix defines  $\mathcal{C}$ , the right one  $\nu$ .

**Theorem 11.**

Instance:  $\mathcal{C}, R, \nu, c^*, \text{card}(X)$

Question:  $c^* \in \nu$

The proof is by reduction from 3-SAT. Let  $F = K_1 \wedge \dots \wedge K_m$  be a formula in 3-CNF with clauses  $K_1, \dots, K_m$  and variables  $v_1, \dots, v_n$ . Define  $X_F, \mathcal{C}_F, R_F$  and  $\nu_F$  as follows.  $X_F$  contains instances  $x_1, \bar{x}_1, \dots, x_n, \bar{x}_n$  and  $y_1, y'_1, \dots, y_n, y'_n$  and an instance  $w$ , hence  $\text{card}(X_F) = 4n + 1$ .  $R_F$  contains the representations  $r_0, r_1, \dots, r_m, s_0, s_1, \dots, s_n, s^*$ . The initial hypothesis  $r_0$  represents  $\{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$ , the target concept  $c^* := \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n, w\}$  will be represented by  $s^*$ . For each clause  $K_i$  we use  $r_i$  to represent a concept that consists of all instances except of  $x_j$  for all  $v_j$  in  $K_i$  and except of  $\bar{x}_j$  for all  $\bar{v}_j$  in  $K_i$ . Finally  $s_0$  represents  $X_F$  and  $s_i$  represents the concept  $\{x_{i+1}, \bar{x}_{i+1} \dots x_n, \bar{x}_n, y_{i+1}, y'_{i+1}, \dots, y_n, y'_n, w\}$  for  $i = 1, \dots, n$  (see Fig. 1).

The relation  $\nu_F$  contains  $(r_0, r_i)$  for all  $i = 1, \dots, n$  and  $(r_0, s_0)$  and  $(s_i, s_{i+1})$  for  $i = 0, \dots, n - 1$ , as well as  $(s_n, s^*)$ . The only path from the initial to the target hypothesis is  $r_0, s_0, s_1, \dots, s_n, s^*$ . If one of the  $r_i$ 's is reached, teaching has failed as these representations are dead ends.

$\mathcal{C}_F$  and  $\nu_F$  can easily be computed and encoded as a  $(4n + 1 + m + 1 + n + 2) \cdot (m + 1 + n + 2) = O((n + m)^2)$  size matrix. Therefore the reduction is polynomial.

Let  $F$  be satisfied by an assignment  $\beta: \{v_1, \dots, v_n\} \rightarrow \{0, 1\}$ . We have to show that  $c^*$  is teachable in the environment defined above. A successful example sequence consists of (1) for all  $i = 1, \dots, n$  the positive examples  $x_i$ , if  $\beta(v_i) = 1$ , or  $\bar{x}_i$ , if  $\beta(v_i) = 0$ ; (2) the positive example  $w$ ; (3) the sequence  $y_1, y'_1, \dots, y_n, y'_n$  of negative examples; (4) any positive example not yet presented. All examples before  $w$  are consistent with  $r_0$ , hence no mind change can take place. A mind change is then triggered by teaching  $(w, 1)$ . By their definition all  $r_i$ 's are inconsistent with the examples taught at Step (1), whereas  $s_0$  certainly is consistent. Therefore all  $\nu_F$ -learners will hypothesize  $s_0$  after Step (2). Teaching  $y_1$  and  $y'_1$  causes two inconsistencies with  $s_0$ , but  $s_1$  has only one error (either  $x_1$  or  $\bar{x}_1$ , depending on  $\beta$ ). It follows that all learners are forced to  $s_1$ . Similarly one can see that after teaching  $y_2, y'_2, \dots, y_n, y'_n$  all learners have reached  $s_n$ . Now each missing positive example triggers a mind change to  $s^*$ . This shows that  $c^*$  is teachable.

Let  $F$  be a formula such that  $c^* \in \mathcal{C}_F$  is teachable to all  $\nu_F$ -learners. Let  $z_1, \dots, z_\ell \in \mathcal{X}$  be a sequence of examples such that all  $\nu_F$ -learners starting at  $r_0$  end up in  $s^*$ . We have to show that  $F$  can be satisfied.

The idea of the proof is as follows. First we show that after a certain example all learners must have reached  $s_0$ . At this point, for all  $i = 1, \dots, n$  both  $x_i$  and  $\bar{x}_i$  have been taught. To prove this we show that, if for some  $i$  both  $x_i$  and  $\bar{x}_i$  have been taught, then it is impossible to force all learners to reach  $s^*$ . Finally we define a satisfying assignment  $\beta$  depending, for each  $i$ , on whether  $x_i$  or  $\bar{x}_i$  occurs in the sample.

As long as the teacher teaches examples different from  $w$ ,  $r_0$  is consistent and no mind change happens. Therefore, for some  $k, z_k = (w, 1)$ . At this point a mind change must happen. If there were no mind change, all neighbors of  $r_0$ ,

i.e.,  $r_1, \dots, r_m, s_0$ , had more errors than  $r_0$ . This cannot be repaired, thus all learners would remain in  $r_0$  forever.

Since the example sequence eventually leads to  $s^*$ , the hypothesis after example  $z_k$  must be  $s_0$ . Furthermore none of the  $y$ -examples can have been taught: Otherwise all neighbors of  $r_0$  had at least one error (the  $y$ -example) and  $r_0$  had exactly one error (the example  $w$ ), hence no change from  $r_0$  had occurred.

Since the only way to  $s^*$  is via  $s_1, \dots, s_n$ , the teacher must now provide examples that make all learners switch to  $s_1$ . Now, the point is that, if  $x_1, \bar{x}_1$  occur in the sample, then  $s_1$  has two errors, but if only one of these examples occurs, then  $s_1$  has only one error. If the hypothesis is to be switched to  $s_1$ , the teacher must provide examples such that  $s_0$  has at least two errors (otherwise there were no better hypothesis in the neighborhood). Since  $s_0$  and  $s_1$  are identical with respect to all instances except  $x_1, \bar{x}_1, y_1, y'_1$ , such errors can only be generated by teaching  $y_1$  as well as  $y'_1$ . But even then, the mind change can only be performed if  $s_1$  has less than two errors. Thus, since  $s_1$  is reached via the example sequence, it follows that not both  $x_1$  and  $\bar{x}_1$  have been taught.

In a similar way it can be shown that  $s_{i+1}$  can only be reached from  $s_i$  if not both  $x_{i+1}$  and  $\bar{x}_{i+1}$  appear in the sample. Note that teaching  $x_i$  or  $\bar{x}_i$  if the learners have reached  $s_i$  is possible, but does not influence the following mind changes, because the concepts  $s_{i+1}, \dots, s_n$  are identical with respect to  $x_1, \bar{x}_1, \dots, x_i, \bar{x}_i$ .

Altogether it follows that when all learners changed to  $s_0$  for all  $i$  either  $x_i$  or  $\bar{x}_i$  had not been in the sample taught so far. Therefore the assignment  $\beta$  is well-defined by  $\beta(v_i) = 1$  iff  $x_i$  appears among the examples  $z_1, \dots, z_k$ .

It remains to show that  $\beta$  satisfies  $F$ . This is clear from the definition of the  $r_i$ 's. If  $\beta$  did not satisfy a clause  $K_j$  then  $r_j$  is consistent with whatever  $x$ -examples have been taught before  $w$ . Thus,  $r_j$  is an equally good neighbor as  $s_0$  and there will be a  $\nu_F$ -learner choosing  $r_j$  instead of  $s_0$ . But this is a contradiction to the assumption that all such learners reach  $s^*$ . ■

For infinite instance spaces or classes (and infinite  $\nu$ ) the next theorem applies.

**Theorem 12.**

Input  $\langle \mathcal{C}, \nu \rangle$   
 Output  $\langle \mathcal{C}, \nu \rangle$

We use  $\mathbb{N}$  as instance space and as representation language. Let  $\mathcal{C} = \{\{0, \dots, i\} \mid i \geq 1\} \cup \{\mathbb{N}\}$  a concept class. A concept  $\{0, \dots, i\}$  is represented by  $i$ , and 0 represents the concept  $\mathbb{N}$ . Let  $(\varphi_i)_{i \in \mathbb{N}}$  be an effective enumeration of all partial recursive functions. For all  $j \in \mathbb{N}$  we define an effective enumeration  $(\nu_j)_{j \in \mathbb{N}}$  by

$$\nu_j(r, s) = \begin{cases} 1, & \text{if } r + 1 = s \text{ or } (s = 0 \text{ and } \varphi_j(j) \text{ is defined after } \leq r \text{ steps}), \\ 0, & \text{otherwise.} \end{cases}$$

It suffices to show that  $\mathcal{C}$  is teachable to  $\nu_j$ -learners iff  $\varphi_j(j)$  is defined. Let  $\mathcal{C}$  be teachable to  $\nu_j$ -learners. Then  $\mathbb{N}$  can be taught, hence the representation 0

must be reachable in the graph  $(\mathbb{N}, \nu_j)$ . From the definition of  $\nu_j$  follows that  $\varphi_j(j)$  is defined.

For the converse let  $\varphi_j(j)$  be defined after  $r$  steps. Then  $\mathbb{N}$  can be taught by the example sequence  $(2, 1), \dots, (r, 1), (r + 2, 1)$ , where the last example ensures that the only consistent neighbor of  $r$  is 0. Concepts  $\{0, \dots, i\}$  can be taught by  $(i + 1, 0), (2, 1), \dots, (i, 1)$ , where the first example prohibits a transition to hypothesis 0. ■

## 5 Teaching Without Feedback

A teacher  $T$  without feedback knows all learners' initial hypotheses  $h_0$ , but can quickly lose track of them during teaching. On the other hand,  $T$  can rule out neighbors  $r$  of  $h_0$  by giving examples consistent with  $h_0$ , but inconsistent with  $r$ . If in such a way  $T$  can eliminate all but one neighbor  $r'$ , he effectively forces all learners to switch to  $r'$ . By continuing in this manner,  $T$  always knows all learners' hypotheses even without feedback. If the enforced hypotheses approach the target,  $T$  will be successful. Figure 2 describes this strategy more formally.

- 1  $r := h_0$ ;
- 2 **while**  $C_r \neq c^*$  **do**:
  - 2.1 Find  $s \in Nb(r)$ ,  $S \subseteq \mathcal{X}$ , and  $z \in \mathcal{X}$  such that (1)  $C_r$  is consistent with  $S$ , but not with  $z$ , (2)  $s$  is the only neighbor of  $r$  consistent with  $S \cup \{z\}$ , and (3)  $dist(s, r^*) < dist(r, r^*)$ ;
  - 2.2 Teach  $S$  in arbitrary order and then  $z$ ;
  - 2.3  $r := s$ ;

**Fig. 2.** A simple general strategy for teaching without feedback by forcing all learners to make the same mind changes. The initial hypothesis is  $h_0$ ,  $r^*$  represents the target.

The feasibility of this strategy depends on Step 2.1. If teaching does not need to be finite, the condition in Step 2 does not need to be checked. Albeit simple, the strategy works surprisingly often for natural concept classes and  $\nu$ -restrictions. In the following we give some examples. Some proofs are omitted due to lack of space; the reader is referred to [6].

First, we consider the class of all monomials over  $n$  variables. Let  $R = \{0, 1, *\}^n$  and define  $(r, s) \in \nu$  iff  $r$  and  $s$  differ only in one “bit.” As initial hypothesis  $h_0 = *^n$  is used.

**Fact 13.**

Let  $c^*$  be a concept represented by  $r^*$ . We use the “standard” minimum teaching set for monomials that can be constructed in time  $O(n^2)$  (see [9, 17]). Let  $k_1, \dots, k_\ell$  be the positions of all constants in  $r^*$ . The teaching set consists of two positive examples  $x_0^+, x_1^+$  which result from substituting all  $*$ 's with zeroes and ones, respectively. Furthermore it contains one negative example  $x_i^-$  for each



$k_i$  where the  $k_i$ -th bit is inverted and all  $*$ 's are replaced by zeroes. Let  $T$  teach the sequence  $\langle x_0^+, x_1^+, x_1^-, \dots, x_\ell^- \rangle$ .

$T$  follows the strategy of Fig. 2: After the first inconsistent example,  $x_1^-$ , all  $\nu$ -learners are forced to a consistent hypothesis in the neighborhood of  $*^n$ . The only such hypothesis is obtained from  $*^n$  by setting the  $k_1$ -th “bit” to the correct value. This reduces the distance from the target by one. Each of the remaining examples forces all learners to set one  $*$ -bit of their hypothesis to a constant. After  $x_\ell^-$  all constants are set correctly and the target is reached. ■

At first glance, the new and the TD-model show little difference with regard to monomials, since we can use teaching sets also for  $\nu$ -learners. However, not every teaching set could be used for teaching. Even the same teaching set might fail if the examples are given in the wrong order. For example, consider  $r^* = 11**$  which has a teaching set with  $x_0^+ = 1100$ ,  $x_1^+ = 1111$ ,  $x_1^- = 0100$ ,  $x_2^- = 1000$ . Teaching those examples in reverse order can lead to the following hypothesis sequence:  $0***$ ,  $00**$ ,  $00**$ ,  $00**$ . The last hypothesis is not only incorrect, it is even impossible to reach  $r^*$  from it (given the examples taught so far).

As another natural concept class, together with a representation, we consider the class of all Boolean functions of  $n$  variables represented by decision trees. A decision tree is a binary tree whose internal nodes are labeled with a variable and whose leaves are labeled either as positive or as negative. An instance  $x \in \{0, 1\}^n$  traverses the tree beginning at the root and at each internal node choosing the left child if that node’s variable is satisfied and the right child otherwise, until a leaf is reached. Thus each tree represents a concept  $c \subseteq \{0, 1\}^n$  containing all positively classified instances.

Each learner starts at the tree consisting of only one negative leaf. In each round one leaf may be substituted by an internal node that has two differently labeled leaves as children. This specifies a relation  $\nu_{DT}$  over all decision trees.

**Fact 14.** *The teaching dimension of the class of all Boolean functions of  $n$  variables with respect to  $\nu_{DT}$  is  $2^n$ .*

The teaching dimension with respect to all Boolean functions is  $2^n$  for all concepts. As we have seen, for  $\nu$ -learners based on decision trees, teaching can often be successful with much fewer examples.

One can think of three situations where the above strategy either fails or is inefficient due to lack of feedback: (1) it is impossible to enforce a certain mind change by ruling out all but one neighbor; (2) correcting a wrong hypothesis afterwards is cheaper than preventing all possible errors beforehand; (3) there are several equivalent, but syntactically different hypotheses in the neighborhood.

We have already seen examples of situations (1) in Fact 9, and of situation (2) in Facts 6 and 7. In the following we construct an example for (3).

We consider the class  $\langle (y_1, b_1), \dots, (y_m, b_m), (*, 0) \rangle$  of variables  $y_1, \dots, y_m$  and bits  $b_i \in \{0, 1\}$ . An instance  $x \in \{0, 1\}^n$  runs through the list starting at the node  $(y_1, b_1)$  until it satisfies a variable, say  $y_j$ , in which case

it is classified as  $b_j$ . The default node  $(*, 0)$  classifies all instances as negative that do not satisfy any of the variables  $y_1, \dots, y_m$ .

We use two kinds of learners obeying different neighborhood relations. Both start at a decision list with only a positive default node  $(*, 1)$  whose only neighbor is the list  $\langle (*, 0) \rangle$  with a negative default node. All learners may insert nodes of the form  $(y, 0)$  in any position of the list. However, restrictions apply with regard to nodes of the form  $(y, 1)$ . Learners of the first kind are allowed to substitute the  $(*, 1)$  node of the hypothesis by an arbitrary positive node or to insert such a node at the  $i$ -th position of the list. Learners of the second kind may only substitute the  $(*, 1)$  node or insert at the  $i$ -th position of the list. In both cases, the default node must not be substituted.

To distinguish the hypotheses of both kinds of learners we label the decision lists with either B or E specifying whether modifications are allowed at the beginning or at the end of the list, respectively. We have therefore two relations,  $\nu_B$  and  $\nu_E$ , with exactly one common representation, the initial hypothesis  $\langle (*, 1) \rangle$ . If we join both relations at this point, we get a relation  $\nu_{DL}$ . Thus, a  $\nu_{DL}$ -learner will, after receiving the first negative example, switch to either  $\langle (*, 0) \rangle_B$  or  $\langle (*, 0) \rangle_E$  and then act like a  $\nu_B$ - or a  $\nu_E$ -learner.

Intuitively, examples suitable for  $\nu_B$ -learners can lead  $\nu_E$ -learners into a dead end hypothesis and vice versa. Hence, it is important for the teacher to know what type of learner he teaches. This can be recognized by the B/E-extension of the hypotheses, which requires feedback.

**Fact 15.** Let  $\mathcal{C}$  be a concept class in  $\nu_{DL}$  with  $m$  variables. Then, for any  $\nu_{DL}$ -learner  $L$ , there exists a teacher  $T$  such that  $T$  can teach  $\mathcal{C}$  to  $L$  with at most  $m + 1$  examples.

## 6 Comparison with Learning

Such comparisons have been done in the mistake bound model between teacher-directed learning and self-directed learning. In many natural concept classes, the best learner can always learn with fewer mistakes than the best teacher needs to teach all consistent learners [12, 9, 11]. Rivest und Yin [16] use cryptographic assumptions to construct a concept class where a teacher needs less examples than the best learner, if both are restricted to polynomial time algorithms. Ben-David and Eiron [7] construct such classes without relying on cryptographic assumptions.

Teaching and learning can also be compared according to the sample complexity instead of the mistake bound. This amounts to a comparison of the teaching dimension  $TD(\mathcal{C})$  with the number  $Q(\mathcal{C})$  of membership queries necessary. Goldman and Kearns [9] observed that for all  $\mathcal{C}$ ,  $TD(\mathcal{C}) \geq Q(\mathcal{C})$ , i.e., being taught is generally simpler than learning by oneself. This contrasts with the mistake bound model.

We will have a brief look at how the introduction of the  $\nu$ -relation influences the relationship between teaching and learning. To do so, we give the  $\nu$ -learners access to a membership oracle. Note that still all conditions of Definition 1 apply.

For example, a  $\nu$ -learner must try to change his mind when the oracle's answer is inconsistent with the current hypothesis.

The next two facts demonstrate that in our model teachability and learnability can be rather different.

**Fact 16.** Let  $X = \{x_1, x_2, x_3\}$ ,  $C = \{c_0, c_1, c_2, c_3, c_4\}$ ,  $R = \{r_0, r_1, r_2, r_3, r_4\}$ ,  $\nu \subseteq R \times R$  and  $\mathcal{C} = \{c_0, c_1, c_2, c_3, c_4\}$ . Then  $\mathcal{C}$  is teachable but not learnable by  $\nu$ -learners.

Let  $X = \{x_1, x_2, x_3\}$ ,  $c_0 = \emptyset$ ,  $c_1 = \{x_1\}$ ,  $c_2 = \{x_1, x_3\}$ ,  $c_3 = \{x_2\}$ ,  $c_4 = \{x_2, x_3\}$  and  $R = \{r_0, r_1, r_2, r_3, r_4\}$  with  $r_i$  representing  $c_i$ . Finally,  $\nu$  contains  $(r_0, r_i)$  for  $i = 1, 2, 3, 4$ .

The concept  $c_1$  can be taught using the instances  $x_3, x_1$ ;  $c_2$  by  $x_2, x_3$ ;  $c_3$  by  $x_3, x_2$ ; and  $c_4$  by  $x_1, x_3$ . Thus  $\mathcal{C}$  can be taught without feedback to  $\nu$ -learners.

Assume there is a  $\nu$ -learner  $L$  with access to a membership oracle.

1.  $L$  queries  $x_1$ . On answer "1",  $L$  must change its hypothesis to either  $r_1$  or  $r_2$ . If  $L$  chooses  $r_1$  than it cannot learn  $c_2$  since there is no way back to  $r_0$ . Similar, if  $r_2$  is chosen,  $L$  cannot learn  $c_1$  any more.
2.  $L$  queries  $x_2$ . Analogous to Case 1 with concepts  $c_3$  and  $c_4$ .
3.  $L$  queries  $x_3$ . Analogous to Case 1 with concepts  $c_2$  and  $c_4$ . ■

**Fact 17.** Let  $X = \{x_1, x_2\}$ ,  $C = \{c_0, c_1, c_2\}$ ,  $R = \{r_0, r_1, r_2, r'_1\}$ ,  $\nu \subseteq R \times R$  and  $\mathcal{C} = \{c_0, c_1, c_2\}$ . Then  $\mathcal{C}$  is teachable but not learnable by  $\nu$ -learners.

Let  $X = \{x_1, x_2\}$ ,  $c_0 = \emptyset$ ,  $c_1 = \{x_1\}$ ,  $c_2 = \{x_1, x_2\}$ , and let  $R = \{r_0, r_1, r'_1, r_2\}$  with  $r_i$  representing  $c_i$  and additionally  $r'_1$  representing  $c_1$ . Let  $\nu = \{(r_0, r_1), (r_0, r'_1), (r_1, r_2)\}$ .

A  $\nu$ -learner works as follows. First query  $x_1$ . If the answer is "0", then the target must be  $c_0$  and  $L$  stops. If the answer is "1", change to hypothesis  $r_1$  and query  $x_2$ . If the answer is "0", the target is  $c_1$  and  $L$  stops, otherwise  $L$  switches to  $r_2$  and stops. Hence, this  $\nu$ -learner learns  $\mathcal{C}$ .

Let  $T$  be a teacher. We show that  $T$  cannot teach  $c_2$ . Let  $z$  be the first example taught. If  $z = (x_1, 1)$  there is a  $\nu$ -learner going to  $r'_1$  from where  $r_2$  cannot be reached. Consequently,  $T$  has to begin with  $z = (x_2, 1)$  which causes no hypothesis change. As soon as  $T$  teaches  $(x_2, 1)$  there is a learner switching to  $r'_1$ . This learner will never reach  $r_2$ . Thus,  $\mathcal{C}$  cannot be taught. ■

## 7 Conclusion and Further Research

In our model several effects regarding feedback can be observed. Feedback can be useless, helpful, or even indispensable for teaching. In addition, natural infinite concept classes can be taught in this model and the relationship between teachability and learnability is more diverse than in the TD-model.

The variety of possible results stems mostly from the ability to define  $\nu$  arbitrarily. We have also used rather artificial  $\nu$ 's in some places. It would therefore be interesting to put some natural restrictions on  $\nu$ , e.g., some relation between syntax (distance in the  $(R, \nu)$ -graph) and semantics (number of errors).

The strategy of Section 5, which often makes teaching without feedback possible, relies on the (somewhat unrealistic) feature of our models that all learners remember all examples (especially the consistent ones). It seems natural to study feedback for learners with some sort of memory limitation.

Further directions of research include adding computability restrictions to the teachers and/or learners, teaching with only positive examples, and other types of feedback, e.g., answering teacher's questions.

**Acknowledgments.** The authors heartily thank the anonymous referees for many valuable comments. The second author has been supported by the

## References

- [1] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
- [2] D. Angluin. Queries revisited. In *Algorithmic Learning Theory, 12th International Conference, ALT 2001, Proc.*, vol. 2225 of *Lecture Notes in Artificial Intelligence*, pages 12–31. Springer, 2001.
- [3] D. Angluin and M. Krikis. Teachers, learners and black boxes. In *Proc. 10th Annual Conference on Computational Learning Theory*, pages 285–297, ACM Press, New York, NY, 1997.
- [4] D. Angluin and M. Krikis. Learning from different teachers. *Machine Learning*, 51(2):137–163, 2003.
- [5] M. Anthony, G. Brightwell, D. Cohen, and J. Shawe-Taylor. On exact specification by examples. In *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, pages 311–318. ACM Press, New York, NY, 1992.
- [6] F.J. Balbach and T. Zeugmann. Teaching Learners that can only Perform Restricted Mind Changes, TCS Technical Report, Series A, TCS-TR-A-05-5, Division of Computer Science, Hokkaido University, July 18, 2005.
- [7] S. Ben-David and N. Eiron. Self-directed learning and its relation to the VC-dimension and to teacher-directed learning. *Machine Learning*, 33(1):87–104, 1998.
- [8] R. Freivalds, E. B. Kinber, and R. Wiehagen. Learning from good examples. In *Algorithmic Learning for Knowledge-Based Systems*, vol. 961 of *Lecture Notes in Artificial Intelligence*, pages 49–62. Springer, 1995.
- [9] S. A. Goldman and M. J. Kearns. On the complexity of teaching. *J. of Comput. Syst. Sci.*, 50(1):20–31, 1995.
- [10] S. A. Goldman and H. D. Mathias. Teaching a smarter learner. *J. of Comput. Syst. Sci.*, 52(2):255–267, 1996.
- [11] S. A. Goldman, R. L. Rivest, and R. E. Schapire. Learning binary relations and total orders. *SIAM J. Comput.*, 22(5):1006–1034, Oct. 1993.
- [12] S. A. Goldman and R. H. Sloan. The power of self-directed learning. *Machine Learning*, 14(3):271–294, 1994.
- [13] J. Jackson and A. Tomkins. A computational model of teaching. In *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, pages 319–326. ACM Press, New York, NY, 1992.
- [14] S. Jain, S. Lange, and J. Nessel. Learning of r.e. languages from good examples. In *Algorithmic Learning Theory, 8th International Workshop, ALT '97, Proc.*, vol. 1316 of *Lecture Notes in Artificial Intelligence*, pages 32–47. Springer, 1997.

- [15] H. D. Mathias. A model of interactive teaching. *J. of Comput. Syst. Sci.*, 54(3):487–501, 1997.
- [16] R. L. Rivest and Y. L. Yin. Being taught can be faster than asking questions. In *Proc. 8th Annual Conference on Computational Learning Theory*, pages 144–151. ACM Press, New York, NY, 1995.
- [17] A. Shinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8(4):337–348, 1991.

# Author Index

- Balbach, Frank J. 474  
Bao, Jie 13  
Bennet, Rotem 183  
Bousquet, Olivier 63  
Bradshaw, Gary 10  
Bshouty, Nader H. 183
- Caragea, Doina 13  
Carlucci, Lorenzo 241  
Case, John 241  
Chapelle, Olivier 78  
Chen, Pai-Hsuen 45  
Chernov, Alexey 414  
Clark, Alexander 283
- Elomaa, Tapio 371  
Eyraud, Rémi 283
- Fakcharoenphol, Jittat 135  
Fan, Rong-En 45  
Fernau, Henning 297
- Geng, Zhi 92  
Goldberg, Paul W. 157  
Gretton, Arthur 63  
Guttman, Omri 171
- Harada, Shigeaki 343  
Henderson, Matthew 386  
He, Yang-Bo 92  
Honavar, Vasant 13  
Hutter, Marcus 356, 414
- Jain, Sanjay 1, 226, 241, 256, 327
- Kato, Hirotaka 211  
Kijirikul, Boonserm 135  
Kinber, Efim 256  
King, Ross D. 12  
Kowalczyk, Adam 78  
Kujala, Jussi 371
- Lange, Steffen 226  
Liang, Xun 92
- Lin, Chih-Jen 45  
Lindner, Wolfgang 198
- Martin, Eric 327  
Maruoka, Akira 343  
Matsumoto, Satoshi 211  
Mesterharm, Chris 399  
Miyahara, Tetsuhiro 211
- Ng, Yen Kaow 269  
Nouretdinov, Ilia 459
- Palmer, Nick 157  
Pathak, Jyotishman 13  
Poland, Jan 356  
Pudi, Vikram 122
- Rao, M.R.K. Krishna 312  
Ryabko, Daniil 148
- Schölkopf, Bernhard 63  
Shafer, Glenn 459  
Shawe-Taylor, John 386  
Shinohara, Takeshi 269  
Smalheiser, Neil R. 11  
Smola, Alex 63  
Stephan, Frank 241, 327
- Takemura, Akimichi 459  
Takimoto, Eiji 343  
Thonangi, Risi 122  
Tomita, Etsuji 1
- Ulrich Simon, Hans 1
- Vishwanathan, S.V.N. 171  
Vovk, Vladimir 429, 444, 459
- Watanabe, Kazuho 107  
Watanabe, Sumio 107  
Williamson, Robert C. 171
- Žerovnik, Janez 386  
Zeugmann, Thomas 474  
Zhang, Jun 13  
Zilles, Sandra 226