

Ebrahim Bagheri
Jackie C. K. Cheung (Eds.)

LNAI 10832

Advances in Artificial Intelligence

31st Canadian Conference on Artificial Intelligence, Canadian AI 2018
Toronto, ON, Canada, May 8–11, 2018
Proceedings

 Springer

Lecture Notes in Artificial Intelligence

10832

Subseries of Lecture Notes in Computer Science

LNAI Series Editors

Randy Goebel

University of Alberta, Edmonton, Canada

Yuzuru Tanaka

Hokkaido University, Sapporo, Japan

Wolfgang Wahlster

DFKI and Saarland University, Saarbrücken, Germany

LNAI Founding Series Editor

Joerg Siekmann

DFKI and Saarland University, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/1244>

Ebrahim Bagheri · Jackie C. K. Cheung (Eds.)

Advances in Artificial Intelligence

31st Canadian Conference on Artificial Intelligence, Canadian AI 2018
Toronto, ON, Canada, May 8–11, 2018
Proceedings

Editors

Ebrahim Bagheri
Ryerson University
Toronto, ON
Canada

Jackie C. K. Cheung
McGill University
Montréal, QC
Canada

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-319-89655-7 ISBN 978-3-319-89656-4 (eBook)
<https://doi.org/10.1007/978-3-319-89656-4>

Library of Congress Control Number: 2018939448

LNCS Sublibrary: SL7 – Artificial Intelligence

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at CAI 2018: the 31st Canadian Conference on Artificial Intelligence held May 8–11, 2018, at York University in Toronto, Ontario. CAI is an event organized by the Canadian Artificial Intelligence Association. The event is collocated with the Canadian Graphics Interface and the Computer and Robot Vision conferences. These events (AIGICRV 2018) bring together hundreds of leaders in research, industry, and government, as well as Canada’s most accomplished students. We are pleased to once again showcase Canada’s ingenuity, innovation, and leadership in artificial intelligence research and advanced information and communications technology.

There were 72 submissions to the main conference track, of which 67 were valid and sent for peer review. We ran a double-blind review process, where each submission was reviewed by at least two Program Committee members. Based on the recommendations of the committee members, 16 submissions were accepted as long papers (24%) and 18 submissions as short papers (27%). In addition, the program included seven papers presented at the Graduate Student Symposium, chaired by Luiza Antonie and Kate Larson, which ran an independent review process, and four papers in the Industry Track, chaired by Don Turnbull. We would like to thank Luiza, Kate, and Don for organizing these sessions and chairing the review processes.

We would also like to thank the Program Committee members and the external reviewers of the main conference, the Graduate Student Symposium, and the Industry Track reviewers, for their time and effort in providing valuable reviews in a timely manner. We thank all the authors for submitting their contributions and the authors of accepted papers for preparing the final version of their papers and presenting their work at the conference.

The conference was enriched by three keynote speakers, who are leaders in the field from academia and industry: Jian Pei (Simon Fraser University), Amanda Stent (Bloomberg), and Peter van Beek (Waterloo). We are grateful to them for their time and participation in this event.

The Canadian Conference on Artificial Intelligence is sponsored by the Canadian Artificial Intelligence Association (CAIAC). We would like to thank and acknowledge the work of the executive committee of CAIAC, Ziad Kobti, Leila Kosseim, Xin Wang, Marina Sokolova, and Cory Butz. They kept the conference organization on track and running smoothly. We would also like to thank Fabrizio Gotti, who designed and maintained the conference website with the utmost efficiency. We would like to express our sincere gratitude to Michael Jenkin, the General Chair of AI/GI/CRV 2018, for his help with the organization of the conference.

Finally, we thank our Financial sponsors, including Springer, Borealis AI, Element AI, and the National Research Council of Canada.

March 2018

Ebrahim Bagheri
Jackie C. K. Cheung

Organization

Program Committee

Esma Aimeur	University of Montreal, Canada
Xiangdong An	University of Tennessee Martin, USA
Eric Beaudry	Université du Québec À Montréal (UQAM), Canada
Colin Bellinger	University of Alberta, Canada
Virendra Bhavsar	University of New Brunswick, Canada
Nizar Bouguila	Concordia University, Canada
Scott Buffett	National Research Council Canada, IIT - e-Business, Canada
Cory Butz	University of Regina, Canada
Laurence Capus	Laval University, Canada
Eric Charton	Yellow Pages
Colin Cherry	Google
David Chiu	University of Guelph, Canada
Paul Cook	University of New Brunswick, Canada
Lyne Da Sylva	University of Montreal, Canada
Abdoulaye Baniré Diallo	Université du Québec à Montréal, Canada
Chris Drummond	NRC Institute for Information Technology
Audrey Durand	McGill University, Canada
Faezeh Ensan	Ferdowsi University of Mashhad, Iran
Ahmed Esmin	Federal University of Lavras, Brazil
Jocelyne Faddoul	Saint Mary's University, Canada
Atefeh Farzindar	University of Southern California, USA
Paola Flocchini	University of Ottawa, Canada
Vincent Francois	McGill University, Canada
Michel Gagnon	Polytechnique Montreal, Canada
Yong Gao	The University of British Columbia, Canada
Michael Guerzhoy	St. Michael's Hospital and University of Toronto, Canada
Howard Hamilton	University of Regina, Canada
Jesse Hoey	University of Waterloo, Canada
Jimmy Huang	University of York
Diana Inkpen	University of Ottawa, Canada
Ilya Ioshikhes	University of Ottawa, Canada
Aminul Islam	University of Louisiana at Lafayette, USA
Vlado Keselj	Dalhousie University, Canada
Fazel Keshkar	St. John's University, Canada
Kamyar Khodamoradi	University of Alberta, Canada
Richard Khoury	Laval University, Canada
Ziad Kobti	University of Windsor, Canada

Leila Kosseim	Concordia University, Canada
Adam Krzyzak	Concordia University, Canada
Matt Kyan	University of York
Sébastien Lallé	The University of British Columbia, Canada
Luc Lamontagne	Laval University, Canada
Philippe Langlais	University of Montreal, Canada
Edith Law	University of Waterloo, Canada
Pawan Lingras	Saint Mary's University, Canada
Rongxing Lu	University of New Brunswick, Canada
Simone Ludwig	North Dakota State University, USA
Masoud Makrehchi	University of Ontario Institute of Technology, Canada
Brad Malin	Vanderbilt University, USA
Stan Matwin	Dalhousie University, Canada
Gordon McCalla	University of Saskatchewan, Canada
Robert Mercer	University of Western Ontario, Canada
Marie-Jean Meurs	Université du Québec à Montréal (UQAM), Canada
Evangelos Milios	Dalhousie University, Canada
Malek Mouhoub	University of Regina, Canada
Jian-Yun Nie	University of Montreal, Canada
Roger Nkambou	Université du Québec À Montréal (UQAM), Canada
Zeinab Noorian	Ryerson University, Canada
Fred Popowich	Simon Fraser University, Canada
Guillaume Rabusseau	McGill University, Canada
Sheela Ramanna	University of Winnipeg, Canada
Siamak Ravanbakhsh	The University of British Columbia, Canada
Robert Reynolds	Wayne State University, USA
Samira Sadaoui	University of Regina, Canada
Fatiha Sadat	Université du Québec À Montréal (UQAM), Canada
Javad Sadri	Concordia University, Canada
Eugene Santos	Dartmouth College, USA
Weiming Shen	National Research Council Canada
Marina Sokolova	University of Ottawa and Institute for Big Data Analytics, Canada
Bruce Spencer	University of New Brunswick, Canada
Stan Szpakowicz	University of Ottawa, Canada
Jian Tang	HEC Montreal and MILA, Canada
Thomas Tran	University of Ottawa, Canada
Don Turnbull	The University of Texas at Austin, USA
Richard Valenzano	University of Toronto, Canada
Petko Valtchev	Université du Québec À Montréal, Canada
Herke Van Hoof	McGill University, Canada
Chun Wang	Concordia University, Canada
Xin Wang	University of Calgary, Canada
René Witte	Concordia University, Canada
Dan Wu	University of Windsor, Canada
Yang Xiang	University of Guelph, Canada

Jingtao Yao	University of Regina, Canada
Harry Zhang	University of New Brunswick, Canada
Xiaodan Zhu	Queen's University, Canada
Nur Zincir-Heywood	Dalhousie University, Canada

Contents

Long Papers

Compressing Bayesian Networks: Swarm-Based Descent, Efficiency, and Posterior Accuracy	3
<i>Yang Xiang and Benjamin Baird</i>	
De-Causalizing NAT-Modeled Bayesian Networks for Inference Efficiency	17
<i>Yang Xiang and Dylan Loker</i>	
A Novel Evaluation Methodology for Assessing Off-Policy Learning Methods in Contextual Bandits	31
<i>Negar Hassanpour and Russell Greiner</i>	
Synthesizing Controllers: On the Correspondence Between LTL Synthesis and Non-deterministic Planning	45
<i>Alberto Camacho, Jorge A. Baier, Christian Muise, and Sheila A. McIlraith</i>	
Logic-Based Benders Decomposition for Two-Stage Flexible Flow Shop Scheduling with Unrelated Parallel Machines	60
<i>Yingcong Tan and Daria Terekhov</i>	
Advice-Based Exploration in Model-Based Reinforcement Learning	72
<i>Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith</i>	
Deep Super Learner: A Deep Ensemble for Classification Problems	84
<i>Steven Young, Tamer Abdou, and Ayse Bener</i>	
One Single Deep Bidirectional LSTM Network for Word Sense Disambiguation of Text Data	96
<i>Ahmad Pesaranhader, Ali Pesaranhader, Stan Matwin, and Marina Sokolova</i>	
MedFact: Towards Improving Veracity of Medical Information in Social Media Using Applied Machine Learning.	108
<i>Hamman Samuel and Osmar Zaïane</i>	
Reranking Candidate Lists for Improved Lexical Induction.	121
<i>Laurent Jakubina and Philippe Langlais</i>	

Analysis of Social Media Posts for Early Detection of Mental Health Conditions 133
Antoine Briand, Hayda Almeida, and Marie-Jean Meurs

Motor Bearing Fault Diagnosis Using Deep Convolutional Neural Networks with 2D Analysis of Vibration Signal 144
M. M. Manjurul Islam and Jong-Myon Kim

Mobile App for Detection of Counterfeit Banknotes 156
Tamarafinide V. Dittimi and Ching Y. Suen

A Multiagent Framework for Understanding Addiction 169
Wasif Khan and Robin Cohen

Infusing Domain Knowledge to Improve the Detection of Alzheimer’s Disease from Everyday Motion Behaviour 181
Chao Bian, Shehroz S. Khan, and Alex Mihailidis

An Incremental Machine Learning Algorithm for Nuclear Forensics 194
Chris Drummond

Short Papers

MML-Based Approach for Determining the Number of Topics in EDCM Mixture Models 211
Nuha Zamzami and Nizar Bouguila

Constrained Bayesian Optimization for Problems with Piece-wise Smooth Constraints 218
Aliakbar Gorji Daronkolaei, Amir Hajian, and Tonya Custis

Dimensionality Reduction and Visualization by Doubly Kernelized Unit Ball Embedding 224
Behrouz Haji Soleimani and Stan Matwin

Accelerated Gradient and Block-Wise Gradient Methods for Big Data Factorization 231
M. Reza Peyghami, Kevin Yang, Shengyuan Chen, Zijiang Yang, and Masoud Ataei

Learning Belief Revision Operators 239
Aaron Hunter

Solving Constraint Satisfaction Problems Using Firefly Algorithms 246
Mahdi Bidar, Malek Mouhoub, Samira Sadaoui, and Mohsen Bidar

An AI Planning-Based Approach to the Multi-Agent Plan Recognition Problem 253
Maayan Shvo, Shirin Sohrabi, and Sheila A. McIlraith

Predicting Transportation Modes of GPS Trajectories Using Feature Engineering and Noise Removal 259
Mohammad Etemad, Amílcar Soares Júnior, and Stan Matwin

Prediction of Container Damage Insurance Claims for Optimized Maritime Port Operations 265
Ashwin Panchapakesan, Rami Abielmona, Rafael Falcon, and Emil Petriu

Drug-Target Interaction Network Predictions for Drug Repurposing Using LASSO-Based Regularized Linear Classification Model 272
Jiaying You, Md. Mohaiminul Islam, Liam Grenier, Qin Kuang, Robert D. McLeod, and Pingzhao Hu

Optimal Scheduling for Smart Charging of Electric Vehicles Using Dynamic Programming 279
Karol Lina López and Christian Gagné

Combining MCTS and A3C for Prediction of Spatially Spreading Processes in Forest Wildfire Settings 285
Sriram Ganapathi Subramanian and Mark Crowley

Text-Based Detection of Unauthorized Users of Social Media Accounts. 292
Milton King, Dima Alhadidi, and Paul Cook

N-Gram Based Approach for Automatic Prediction of Essay Rubric Marks 298
Magdalena Jankowska, Colin Conrad, Jabez Harris, and Vlado Kešelj

Matching Résumés to Job Descriptions with Stacked Models 304
Peng Xu and Denilson Barbosa

Towards a Comprehensive Evaluation of Recommenders: A Cognition-Based Approach 310
Alaa Alslaity and Thomas Tran

A Sentence-Level Sparse Gamma Topic Model for Sentiment Analysis 316
Tao Chen and Jeffrey Parsons

Topic Detection and Document Similarity on Financial News 322
Saeede Sadat Asadi Kakhki, Can Kavaklioglu, and Ayse Bener

Graduate Student Symposium Papers

Software Defect Prediction from Code Quality Measurements
via Machine Learning 331
Ross MacDonald

Automated Scheduling: Reinforcement Learning Approach
to Algorithm Policy Learning 335
Yingcong Tan

Estimating Vineyard Grape Yield from Images 339
Tanya Monga

Real-Time Deep Learning Pedestrians Classification
on a Micro-Controller 344
Zhaoyang Huang

A Unified Evaluation Framework for Recommenders 351
Alaa Alslaity

Early Detection of Alzheimer’s Disease Using Deep Learning 355
Laura McCrackin

Learning with Prior Domain Knowledge and Insufficient
Annotated Data. 360
Matthew Dirks

Industry Track

Predicting Crime Using Spatial Features 367
Fateha Khanam Bappee, Amílcar Soares Júnior, and Stan Matwin

A Tool for Defining and Simulating Storage Strategies
on the Smart Grid 374
Dan Russell and Aaron Hunter

Decision Assist for Self-driving Cars 381
*Sriram Ganapathi Subramanian, Jaspreet Singh Sambee,
Benyamin Ghojogh, and Mark Crowley*

Rule Mining and Prediction Using the Flek Machine – A New Machine
Learning Engine 388
Abbas Taher

Author Index 395

Long Papers



Compressing Bayesian Networks: Swarm-Based Descent, Efficiency, and Posterior Accuracy

Yang Xiang^(✉) and Benjamin Baird

University of Guelph, Guelph, Canada
yxiang@uoguelph.ca

Abstract. Local models in Bayesian networks (BNs) reduce space complexity, facilitate acquisition, and can improve inference efficiency. This work focuses on Non-Impeding Noisy-AND Tree (NIN-AND Tree or NAT) models whose merits include linear complexity, being based on simple causal interactions, expressiveness, and generality. We present a swarm-based constrained gradient descent for more efficient compression of BN CPTs (conditional probability tables) into NAT models. We show empirically that multiplicatively factoring NAT-modeled BNs allows significant speed up in inference for a reasonable range of sparse BN structures. We also show that such gain in efficiency only causes reasonable approximation errors in posterior marginals in NAT-modeled real world BNs.

Keywords: Uncertainty · Bayesian Networks
Causal independence models

1 Introduction

Local models (modeling the CPT over an effect and its causes) in BNs, such as noisy-OR [9], noisy-MAX [2, 3], context-specific independence (CSI) [1], recursive noisy-OR [6], NIN-AND Tree or NAT [12], DeMorgan [7], cancellation model [11], and tensor-decomposition [10], reduce the space complexity of BNs, facilitate knowledge acquisition, and can improve efficiency for inference. This work focuses on NAT models [12], whose merits include linear complexity, being based on simple causal interactions (reinforcement and undermining), expressiveness (recursive mixture, multi-valued, ordinal and nominal [13]), and generality (generalizing noisy-OR, noisy-MAX [14], and DeMorgan [12]). In addition, NAT models support much more efficient inference, where two orders of magnitude speedup in lazy propagation is achieved in very sparse NAT-modeled BNs [14]. Since causal independence encoded in a NAT model is orthogonal to CSI, NAT models provide an alternative mechanism to CSI for efficient inference in BNs.

This work advances the state of the art of NAT modeling around three issues. Given a general BN, it can be compressed into a NAT-modeled BN for improved

space and inference efficiency. A key step of compression is to parameterize alternative NAT models through constrained gradient descent. We investigate a swarm-based constrained gradient descent for more efficient compression.

NAT-modeling has been shown to improve inference efficiency in very sparse BNs. In this work, we apply NAT-modeling to BNs with a wider range of structural densities, and assess the impact on inference efficiency in that range.

Compression of a general BN into a NAT-modeled BN introduces approximation errors. The compression errors have been evaluated through Kullback–Leibler divergence and Euclidean distance between the target CPT (from the general BN) and the NAT CPT [13]. In this work, we investigate the impact of compression errors on the posterior marginals from inference, which provide a more direct evaluation of approximation errors of NAT-modeling.

The remainder of the paper is organized as follows. Section 2 reviews the background on NAT modeling. The swarm-based parameterization is presented in Sect. 3. Evaluations on efficiency gain in inference due to NAT modeling and on posterior accuracy are reported in Sects. 4 and 5, respectively.

2 Background on NAT Models

NAT models deal with *uncertain* causes, that can render their effects but do not always do so. The effect and causes in NAT models are *causal variables*.

Definition 1 [13]. *A variable x that can be either inactive or be active in multiple ways, and is involved in a causal relation, is a **causal variable** if when all causes of the effect are inactive, the effect is inactive with certainty.*

A causal variable can be ordinal or nominal, and hence is more general than the *graded* variable, commonly assumed in noisy-OR or noisy-MAX, e.g., in [2]. The inactive value of a causal variable e is indexed as e^0 , and its active values are indexed arbitrarily. In practice, some orders of indexing on active values are preferred over others. However, the semantics of NAT models does not dictate choice on such orders.

In general, we denote an effect by e and the set of all causes of e by $C = \{c_1, \dots, c_n\}$. The domain of e is $D_e = \{e^0, \dots, e^\eta\}$ ($\eta > 0$) and the domain of c_i ($i = 1, \dots, n$) is $D_i = \{c_i^0, \dots, c_i^{m_i}\}$ ($m_i > 0$). An active value may be written as e^+ or c_i^+ .

A causal event is a *success* or *failure* depending on whether e is rendered active at a certain range of values, is *single-causal* or *multi-causal* depending on the number of active causes, and is *simple* or *congregate* depending on the range of effect values.

A *simple single-causal success* is an event that cause c_i of value c_i^j ($j > 0$) caused effect e to occur at value e^k ($k > 0$), when every other cause is inactive. Denote the event probability by $P(e^k \leftarrow c_i^j) = P(e^k | c_i^j, c_z^0 : \forall z \neq i)$ ($j > 0$). A multi-causal success involves a set $X = \{c_1, \dots, c_q\}$ ($q > 1$) of active causes, where each $c_i \in X$ has the value c_i^j ($j > 0$), when every other cause $c_m \in C \setminus X$ is inactive. A *congregate multi-causal success* is an event such that causes in X

collectively caused the effect to occur at value e^k ($k > 0$) or values of higher indexes, when every other cause is inactive. We denote the probability of the event as

$$P(e \geq e^k \leftarrow c_1^{j_1}, \dots, c_q^{j_q}) = P(e \geq e^k | c_1^{j_1}, \dots, c_q^{j_q}, c_z^0 : c_z \in C \setminus X) \quad (j > 0),$$

where $X = \{c_1, \dots, c_q\}$ ($q > 1$). It can also be denoted $P(e \geq e^k \leftarrow \underline{x}^+)$.

A *congregate single-causal failure* refers to an event where $e < e^k$ ($k > 0$) when cause c_i has value $c_i^{j_i}$ ($j_i > 0$) and every other cause is inactive. We denote the probability of the event as $P(e < e^k \leftarrow c_i^{j_i}) = P(e < e^k | c_i^{j_i}, c_z^0 : \forall z \neq i) \quad (j_i > 0)$.

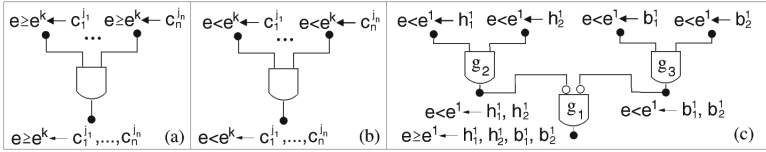


Fig. 1. A direct NIN-AND gate (a), a dual NIN-AND gate (b), and a NAT (c)

A NAT consists of two types of NIN-AND gates, each over disjoint sets of causes W_1, \dots, W_q . An input event of a *direct* gate (Fig. 1 (a)) is $e \geq e^k \leftarrow \underline{w}_i^+$ and the output event is $e \geq e^k \leftarrow \underline{w}_1^+, \dots, \underline{w}_q^+$. An input of a *dual* gate (Fig. 1 (b)) is $e < e^k \leftarrow \underline{w}_i^+$ and the output event is $e < e^k \leftarrow \underline{w}_1^+, \dots, \underline{w}_q^+$. The probability of the output event of a gate is the product of probabilities of its input events.

Interactions among causes may be reinforcing or undermining, either between causes or groups of causes, as specified in Definition 2.

Definition 2 [12]. Let e^k be an active effect value, $R = \{W_1, \dots\}$ be a partition of a set $X \subseteq C$ of causes, $R' \subset R$, and $Y = \cup_{W_i \in R'} W_i$. Sets of causes in R **reinforce** each other relative to e^k , iff $\forall R' P(e \geq e^k \leftarrow \underline{y}^+) \leq P(e \geq e^k \leftarrow \underline{x}^+)$. They **undermine** each other relative to e^k , iff $\forall R' P(e \geq e^k \leftarrow \underline{y}^+) > P(e \geq e^k \leftarrow \underline{x}^+)$.

Direct gates model undermining and dual gates model reinforcing. A NAT organizes multiple gates into a tree and expresses mixtures of reinforcing and undermining recursively, as illustrated in Fig. 1(c). A NAT specifies the interaction between each pair of c_i and c_j , denoted by the *PCI bit* $pci(c_i, c_j) \in \{u, r\}$, where u stands for undermining and r for reinforcing. The collection of PCI bits is the *PCI pattern* of the NAT. The PCI pattern for the NAT in Fig. 1(c) is $\{pci(h_1, h_2) = r, pci(h_1, b_1) = u, pci(h_1, b_2) = u, pci(h_2, b_1) = u, pci(h_2, b_2) = u, pci(b_1, b_2) = r\}$. A NAT can be uniquely identified by its PCI pattern [16].

From the NAT in Fig. 1(c) and probabilities of its input events, in the general form $P(e^k \leftarrow c_i^{j_i})$ ($j, k > 0$), called *single-causals*, $P(e \geq e^1 \leftarrow h_1^1, h_2^1, b_1^1, b_2^1)$ can be obtained. From the single-causals and all derivable NATs, the NAT CPT

$P(e|h_1, h_2, b_1, b_2)$ is uniquely defined [12]. A NAT model for $|C| = n$ is specified by a NAT topology and a set of single-causals of space linear on n .

A general CPT over C and e has space exponential on n . The complexity is reduced to being linear by compressing the CPT into a NAT model, with the following main steps. (1) Extract one or more PCI patterns from the target CPT [13]. (2) Retrieve NAT structures that are compatible with the PCI patterns [15]. (3) Search for numerical parameters for each NAT structure and return the NAT structure and its parameters that best approximate the target CPT [13].

3 Parameterizing NAT Models by Swarm Descent

Parameterization (the step (3) above) is a time consuming compression step (see experiment at end of section). This section presents a novel method to improve its efficiency. To clarify key issues, we present the existing method [13] algorithmically so that the enhancement can be elaborated.

The input includes a target CPT $P_T(e|C)$ and a set Ψ of candidate NATs over C and e . The output is a NAT $Y \in \Psi$ and a set SC of single-causals associated with Y . The criterion is to select the NAT model $M = (Y, SC)$ such that its CPT P_M best approximates P_T . The difference of P_M from P_T to be minimized is average Kullback–Leibler divergence,

$$KL(P_T, P_M) = \frac{1}{Z} \sum_{i=0}^{Z-1} \sum_j P_T(i, j) \log \frac{P_T(i, j)}{P_M(i, j)},$$

where i indexes conditional probability distributions (CPDs) in P_T , j indexes probabilities in each CPD, and Z counts the CPDs. This is achieved by constrained gradient descent described below. In the experimental study, average Euclidean distance

$$ED(P_T, P_M) = \sqrt{\frac{1}{K} \sum_{i=0}^{Z-1} \sum_j (P_T(i, j) - P_M(i, j))^2}$$

is also obtained, where K counts probabilities in P_T .

The gradient g_M is the set of partial derivatives of $KL(P_T, P_M)$ relative to each single-causal $x \in SC$, where $g_M(x) = \partial KL(P_T, P_M) / \partial x$. In a *single-causal descent step*, a single-causal x is revised into $x - \alpha * g_M(x)$, where α is the descent scale (e.g., 0.01). A *descent step* consists of $|SC|$ single-causal descent steps, one per $x \in SC$. During a descent step, each x in the general form $P(e^k \leftarrow c_i^j)$ ($j, k > 1$) is constrained by $x \in (0, 1)$. In addition, for every c_i^j , elements of SC are collectively constrained by $\sum_{k=1}^{\eta} P(e^k \leftarrow c_i^j) < 1$. One *descent round* consists of up to *MaxStep* (e.g., 200) descent steps, as specified in Algorithm 1.

Algorithm 1. $Descent(P_T, Y, SC)$

```

1 for step = 1 to MaxStep,
2   compute gradient  $g_M$  from  $P_T$  and  $M = (Y, SC)$ ;
3   if  $g_M$  signifies convergence, return  $SC$ ;
4   for each  $x \in SC$ ,  $x = x - \alpha * g_M(x)$ , subject to relevant constraints;
5 return  $SC$ ;

```

Given a NAT Y , the returning SC and the KL distance of NAT model $M = (Y, SC)$ is often dependent on the initial single-causals SC . To get the best KL distance from Y , $MaxRound$ (e.g., 10) of descent rounds are performed as Algorithm 2.

Algorithm 2. $MultiDescent(P_T, Y)$

```

1 BestSC = null, BestKL =  $\infty$ ;
2 for round = 1 to MaxRound,
3   randomly initialize a set  $SC$  of single-causals;
4    $SC' = Descent(Y, SC)$ ;
5   compute  $KL(P_T, P_M)$ , where  $M = (Y, SC')$ ;
6   if  $KL(P_T, P_M) < BestKL$ , then  $BestSC = SC'$ ,  $BestKL = KL(P_T, P_M)$ ;
7 return (BestSC, BestKL);

```

Parameterization step selects the best NAT from candidates as Algorithm 3.

Algorithm 3. $ParameterizeNAT(P_T, \Psi)$

```

1 BestNAT = null, BestSC = null, BestKL =  $\infty$ ;
2 for each NAT  $Y \in \Psi$ ,
3    $(SC, KL) = MultiDescent(P_T, Y)$ ;
4   if  $KL < BestKL$ , then  $BestNAT = Y$ ,  $BestSC = SC$ ,  $BestKL = KL$ ;
5 return (BestNAT, BestSC);

```

The above method is time consuming (see experiment at end of section). We observe that it conducts descent rounds independently, one for each candidate NAT and each initial SC set. In the following, we apply particle swarm optimization [5] to develop a novel swarm-based constrained gradient descent, where dependency between multiple descent rounds is introduced to improve efficiency.

A *particle* captures a descent round relative to a NAT Y and an initial set SC of single-causals. The descent round is divided into multiple *descent intervals*, where each interval involves *IntervalLength* (e.g., 10) descent steps. The operation of a particle during a descent interval is specified as Algorithm 4. It is similar to Algorithm 1 but over a smaller number of steps. Instead of returning SC , it saves SC , a flag, gradient, and KL distance as attributes of the particle, to be used for evaluating the particle performance.

Algorithm 4. *ParticleDescent*(P_T, Y, SC)

```

1  init flag converged = false;
2  for step = 1 to IntervalLength,
3    compute gradient  $g_M$  from  $P_T$  and  $M = (Y, SC)$ ;
4    if  $g_M$  signifies convergence,  $converged = true$  and break;
5    for each  $x \in SC$ ,  $x = x - \alpha * g_M(x)$ , subject to relevant constraints;
6    compute  $kl = KL(P_T, P_M)$ , where  $M = (Y, SC)$ ;
7    save  $SC$ ,  $converged$ ,  $g_M$ , and  $kl$ ;

```

A *particle group* is a set of particles that share the same NAT Y . Performances of particles in a group G are evaluated after each *intra-group check period* consisting of a constant of *InGroupCheckPeriod* (e.g., 2) descent intervals. The evaluation is specified in Algorithm 5, where each particle $R \in G$ is associated with attributes such as its α value, *converged* flag, gradient, and KL distance at the end of the last interval of the period. These attributes can be accessed by, e.g., $R.converged$ and $R.kl$. For each $R.g_M$, we denote the average of its element by $R.\overline{g_M}$.

In Algorithm 5 below, each particle $R \in G$ not yet converged is evaluated as *promising*, *average*, or *poor*. A particle is promising, if its average KL distance is better than the group average and it is faster in descending. A particle is poor performing, if its average KL distance is worse than the group average and it is slower in descending. Each promising particle is rendered to descend even faster, and each poor performing particle is halted, where γ and ρ are threshold scales (e.g., ≥ 1).

Algorithm 5. *InGroupCheck*(G)

```

1   $avgKL = (\sum_{R \in G} R.kl) / |G|$ ;
2   $avgGradient = (\sum_{R \in G} R.\overline{g_M}) / |G|$ ;
3  for each particle  $R \in G$ , where  $R.converged = false$ ,
4    if  $R.kl < \gamma * avgKL$  and  $R.\overline{g_M} > avgGradient$ , increase  $R.\alpha$ ;
5    else if  $R.kl > \rho * avgKL$ ,  $R.\overline{g_M} < avgGradient$ , and  $|G| > 1$ ,
6      remove  $R$  from  $G$ ;
7  return  $G$ ;

```

Each candidate NAT is processed by exactly one particle group. Performances of all particle groups are evaluated after each *inter-group check period* consisting of a constant of *InterGroupCheckPeriod* (e.g., 4) descent intervals. As the outcome of evaluation, poorly performing groups are terminated. The evaluation is specified in Algorithm 6, where Φ denotes the set of all particle groups, and β is a threshold scale (e.g., 1.25). Lines 1 to 4 determine average group performance, and the remainder uses it to identify poorly performing groups.

Algorithm 6. *InterGroupCheck*(Φ)

```

1   $avgKL = 0$ ;
2  for each particle group  $G \in \Phi$ ,
3     $G.bestKL = \min_{R \in G} R.kl$ ,  $avgKL = avgKL + G.bestKL$ ;
4   $avgKL = avgKL / |\Phi|$ ;
5  for each particle group  $G \in \Phi$ ,
6    if  $G.bestKL > \beta * avgKL$  and  $|\Phi| > 1$ , remove  $G$  from  $\Psi$ ;
7  return  $\Phi$ ;

```

Algorithm 7 is the top level algorithm for swarm-based descent. *GroupSize* defines the initial number of particles per particle group. It serves the role equivalent to *MaxRound* in Algorithm 2. *MaxInterval* (e.g., 10) controls the total number of descent intervals. Lines 1 to 5 initialize all particle groups. Particles descend in lines 6 to 11, subject to intra-group and inter-group evaluations and eliminations. The remainder selects the best NAT model emerging from the descent.

Algorithm 7. *ParameterizeBySwarm*(P_T, Ψ)

```

1  create a set  $\Phi$  of  $|\Psi|$  particle groups;
2  for each group  $G \in \Phi$ ,
3    assign a distinct NAT  $Y \in \Psi$  to  $G$  as  $G.Y$ ;
4    create GroupSize particles in  $G$ ;
5    for each particle  $R \in G$ , init  $R.\alpha$  and a set of single-causals as  $R.SC$ ;
6  for interval = 1 to MaxInterval,
7    for each group  $G \in \Phi$ ,
8      for each particle  $R \in G$ ,  $R$  runs ParticleDescent( $P_T, G.Y, R.SC$ );
9    if (interval mod InGroupCheckPeriod) = 0,
10     for each group  $G \in \Phi$ ,  $G = \text{InGroupCheck}(G)$ ;
11   if (interval mod InterGroupCheckPeriod) = 0,  $\Phi = \text{InterGroupCheck}(\Phi)$ ;
12  $BestNAT = null$ ,  $BestSC = null$ ,  $BestKL = \infty$ ;
13 for each group  $G \in \Phi$ ,
14   for each particle  $R \in G$ ,
15     if  $R.kl < BestKL$ , then  $BestNAT = G.Y$ ,  $BestSC = R.SC$ ,  $BestKL = R.kl$ ;
16 return ( $BestNAT, BestSC$ );

```

We observe that although *ParticleDescent* by multiple particles are executed sequentially, they are independent until intra-group evaluations. To further improve efficiency, we also investigated a parallel version of swarm-based descent, where *ParticleDescent* by each particle is run on a separate thread. The primary difference from Algorithm 7 is to execute line 8 by threads. All particle threads must also be completed before line 9 is executed.

The three alternative methods for parameterization, referred to as CGD (constrained gradient descent), SSD (sequential swarm-based descent), and PSD (parallel swarm-based descent), are evaluated experimentally. Five batches of target CPTs are randomly generated with the number of causes $n = 5, 6, 7, 8, 9$, respectively. Each batch consists of 2 groups with the variable domain size bounded at $s = 3, 4$, respectively. Each group consists of 50 CPTs. Hence, the

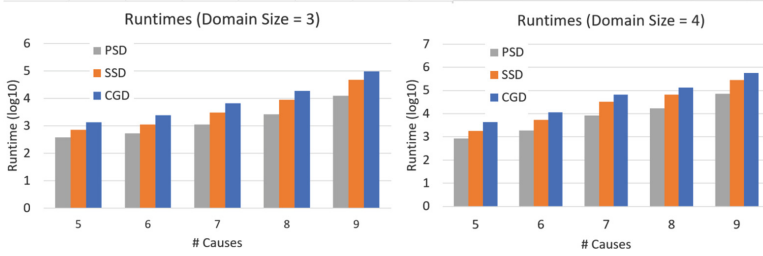


Fig. 2. Runtimes (msec) of PSD, SSD, and CGD in Log10

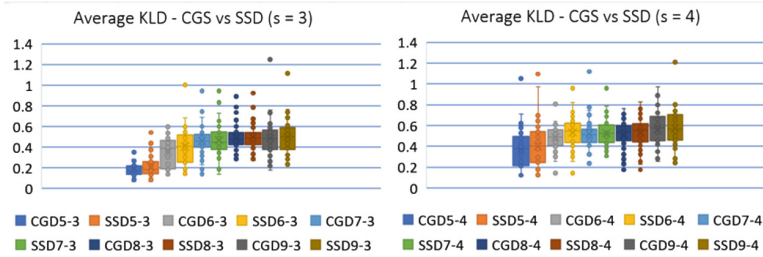


Fig. 3. Average KL distance between target and NAT CPTs from CGD and SSD

experiment consists of a total of $5 * 2 * 50 = 500$ target CPTs. Each CPT is run by each of the 3 methods, using a 6-core desktop at 3.7 GHz clock speed.

Figure 2 summarizes runtime. For each method, as n grows from 5 to 9, computation cost grows by about 100 times. Change of domain sizes from 3 to 4 only increases the computational cost, but does not affect the relative efficiency of the three methods. PSD is consistently more efficient than SSD, and SSD is consistently more efficient than CGD. Furthermore, as n grows from 5 to 9, the advantage of PSD over SSD and that of SSD over CGD becomes more pronounced. At $n = 9$, PSD is about 3.1 times faster than SSD and SSD is about 3.1 times faster than CGD. This results in PSD being about one order of magnitude faster than CGD.

Figure 3 compares the average KL distance of compressed CPTs from target CPTs by CGD and SSD (PSD has the same KL distance as SSD). Figure 4 compares the average ED distance. As n grows from 5 to 9, the approximation errors tend to increase, but the average ED distances are between 0.23 and 0.32.

Errors by SSD are similar to those of CGD, and are more often (but not always) slightly larger. The slightly large errors can be attributed to early termination of some particles or particle groups. Although they do not perform well in the early descent rounds (and hence are terminated by SSD), they could lead to more accurate approximations later on if allowed to continue. Hence, the swarm-based descent should be viewed as a good heuristic and trade-off of slight accuracy for efficiency, rather than as a dominate strategy.

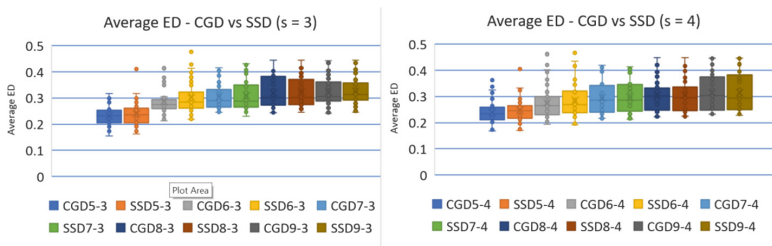


Fig. 4. Average ED distance between target and NAT CPTs from CGD and SSD

4 Impact of MF of NAT Models on Inference Efficiency

In this section, we investigate the impact of NAT modeling on efficiency of inference with BNs. We consider BNs where the CPT of every non-root variable of 2 or more parents is a NAT model, which is referred to as a *NAT-modeled* BN. For the inference method, we focus on lazy propagation (LP) [8]. In order to perform LP with NAT-modeled BNs, we compile them through multiplicative factorization (MF) [14]. In particular, the NAT model of each CPT is structured into a hybrid graph, where link potentials for undirected links and family potentials for directed families are assigned based on parameters of the NAT model. As a result, the NAT-modeled BN is converted into a Markov network and then compiled into a lazy junction tree (JT) for LP. We refer to BNs compiled as so as MF of NAT-modeled BNs (MF-BN).

For comparison, we created a *peer* BN (PR-BN) for each NAT-modeled BN, where each NAT model is expanded into a tabular CPT. The peer BN is then compiled into a lazy JT for LP. In an earlier work, it has been shown that for very sparse BNs (5% more links than singly connected), MF-BNs support up to 2 orders of magnitude speedup in LP than peer BNs [14]. The question of our interest is whether the speedup in LP extends to less sparse NAT-modeled BNs. We provide positive empirical evidence below.

We simulated NAT-modeled BNs with 100 variables per BN. The maximum number of parents per variable in each BN is bounded at $n = 8, 10, 12$, respectively. The uniform domain size of all variables is controlled at $m = 3, 4$, respectively. The structural density of BNs is controlled by adding $w = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\%$ of links to a singly connected network, respectively. Hence, there are a total of $3 \times 2 \times 10 = 60$ distinct (n, m, w) combinations. For each combination, we simulated 10 BNs. This amounts to a total of 600 NAT-modeled BNs.

For each NAT-modeled BN, we perform LP with its MF-BN and PR-BN, and compare runtimes from a laptop of 2.8 GHz clock speed. Figure 5 shows MF-BN runtimes, that grow significantly as BN structures become denser (we omit analysis based on tree-width growth due to space). On the other hand, little growth is observed as n grows from 8 to 12. We attribute this to MF of NAT models, which breaks large BN CPTs into smaller MF factors.

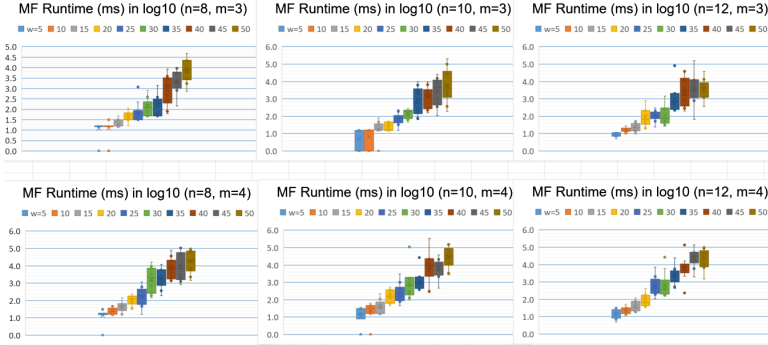


Fig. 5. Runtimes for MF-BNs

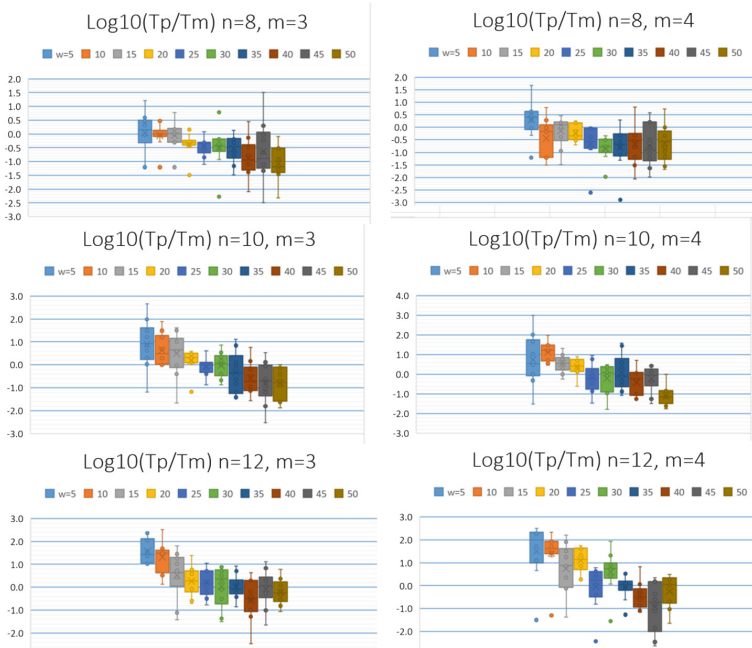


Fig. 6. Ratios of PR-BN runtimes (T_P) versus MF-BN runtimes (T_M)

Figure 6 shows ratios of PR-BN runtimes versus MF-BN runtimes. MF-BNs have superior efficiency for very sparse structures ($w = 5\%$), and the superiority grows as n and m grow. This is consistent with findings in [14]. For $n = 10, 12$, MF-BNs run faster than PR-BNs by up to 2 orders of magnitude (984 times faster in one case). The superiority decreases as w grows (denser structures), but persists up to $w = 20\%$ for $n = 10$, and up to $w = 30\%$ for $n = 12$. Hence, the advantage of MF of NAT-modeling extends to a wider range of sparse structures than reported

[14]. Between $w = 25$ and 35% for $n = 10$ and $w = 35$ and 45% for $n = 12$, either MF-BNs or PR-BNs may run faster depending on particular BNs.

5 Posterior Accuracy After NAT-Modeling

CPT compression errors are assessed in Sect. 3, among others. This section investigates the degree in which they translate into errors of posterior marginals in inference. Eight real world discrete BNs are selected from the well-known *bnlearn* repository. The selection criterion is that the BN must contain a sufficient number of variables whose CPTs are suitable for NAT-modeling. More specifically, the following variables are not suited for NAT-modeling.

1. Root: Its CPT is trivial (not conditional).
2. Single parent: Its tabular CPT is not exponentially spaced.
3. Some CPDs in the CPT are partially deterministic: The causes are not uncertain causes as assumed by NAT models.
4. The variable has a large domain (e.g., of size 20) but a few parents (e.g., 4). As analyzed in [14], MF of NAT models are not more efficient than tabular CPTs for such variables.

Table 1 lists selected BNs. In their NAT-modeled versions, variables unsuited for NAT-modeling keep original CPTs. CPTs of remaining variables in each BN (percentage in 4th column) are NAT-modeled and subject to compression errors. These errors translate into errors in posterior marginals during inference. The objective of the experiment is to assess the magnitude of errors in posterior marginals, in relation to CPT compression errors.

For each BN, we perform LP on the original version (RW-BN) as well as the NAT-modeled version (NT-BN), conditioned on the same observation over 10% of randomly selected variables. For each pair of RW-BN and NT-BN, 10 runs of LP are performed with distinct observations. This amounts to a total of $8 * 2 * 10 = 160$ LP runs.

Table 1. Real world BNs (Andes⁻: Andes with 3 isolated nodes removed)

Network	# Nodes	# NAT CPTs	% NAT CPTs	# links	w
Alarm	37	16	43.2	46	28
Andes ⁻	220	106	48.2	338	54
Barley	48	28	58.3	84	79
Child	20	6	30.0	25	32
Hepar2	70	33	47.1	123	78
Insurance	27	8	29.6	52	100
Win95pts	76	8	10.5	112	49
Munin	1041	11	1.1	1397	34

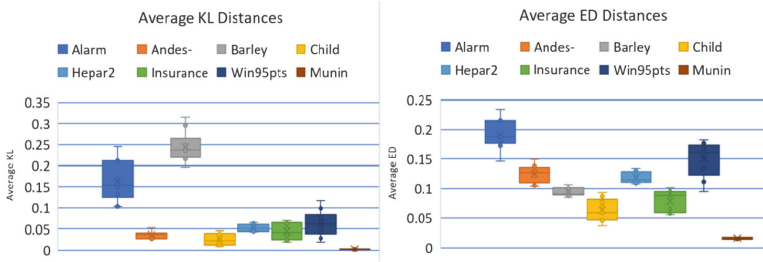


Fig. 7. Average KL and ED distances from LP runs

Average KL and ED distances (Sect. 3) are shown in Fig. 7. We observe that both distances are reasonably small. Hence, the posterior marginals are reasonably accurate, even though a significant percentage of CPTs in each BN (between 30 and 50% for most BNs) are NAT-modeled. The posterior errors are generally smaller than CPT compression errors (Figs. 3 and 4). That is, CPT compression errors are attenuated, rather than amplified, by the inference.

Figure 8 (left) summarizes the NT-BN runtime. Ratios of RW-BN runtimes over NT-BN runtimes are summarized in Fig. 8 (right). NT-BNs for Win95pts and Munin run LP faster than RW-BNs, which can be attributed to their lower density levels (w in Table 1). For other BNs, their density levels are generally higher than the threshold level observed in Sect. 4, and their NT-BNs run LP slower than RW-BNs.

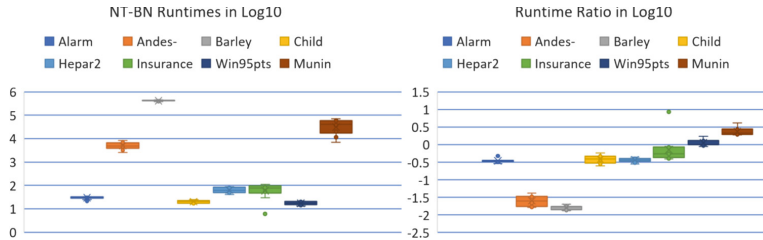


Fig. 8. Left: NT-BN runtimes; Right: Runtime ratios (RW-BN over NT-BN)

In summary, the result demonstrates that NAT-modeling maintains reasonable posterior accuracy in inference (while reducing BN CPT space from being exponential to being linear). As MF of NAT-modeled BNs are more efficient in inference when BN structures are sparse, a promising direction for future work is to learn sparse NAT-modeled BNs directly from data (rather than building a dense BN and then compressing it).

6 Conclusion

Contributions of this work are on local modeling in BNs for more efficient inference, with focus on NAT models. First, we proposed swarm-based constrained gradient descent that allows one order of magnitude faster search and parameterization of NAT-models. Second, our experimental study shows that by multiplicatively factorizing NAT-modeled BNs, LP efficiency can be improved for a range of sparse BN structures. In particular, up to 2 orders of magnitude efficiency gain can be obtained when BN structural densities are up to about 30% more links beyond being singly-connected. Finally, our experimental study of NAT-modeling real world BNs for LP inference demonstrated reasonable accuracy in posterior marginals that tend to be more accurate than CPT compression.

A number of directions for future research can be envisioned. We have NAT-modeled real world BNs by keeping deterministic CPTs. They may be approximated by NAT models or other compact local models, e.g., algebraic decision diagrams [4], and then integrated with NAT models in a same BN. The range of structural densities where MF of NAT modeled BNs show superior LP performance suggests direct learning of NAT modeled BNs constrained to that density range. Finally, MF of NAT models is linear in number of causes but exponential in the effect domain size. Methods other than MF without such limitation may be explored.

Acknowledgement. Financial support from NSERC Discovery Grant to first author is acknowledged.

References

1. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Horvitz, E., Jensen, F. (eds.) Proceedings of 12th Conference on Uncertainty in Artificial Intelligence, pp. 115–123 (1996)
2. Diez, F.J.: Parameter adjustment in Bayes networks: the generalized noisy OR-gate. In: Heckerman, D., Mamdani, F.J. (eds.) Proceedings of 9th Conference on Uncertainty in Artificial Intelligence, pp. 99–105. Morgan Kaufmann (1993)
3. Henrion, M.: Some practical issues in constructing belief networks. In: Kanal, L.N., Levitt, T.S., Lemmer, J.F. (eds.) Uncertainty in Artificial Intelligence, vol. 3, pp. 161–173. Elsevier Science Publishers (1989)
4. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: SPUDD: stochastic planning using decision diagrams. In: Proceedings of 15th Conference on Uncertainty in Artificial Intelligence, pp. 279–288 (1999)
5. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)
6. Lemmer, J.F., Gossink, D.E.: Recursive noisy OR - a rule for estimating complex probabilistic interactions. *IEEE Trans. Syst. Man Cybern. Part B* **34**(6), 2252–2261 (2004)

7. Maaskant, P.P., Druzdzel, M.J.: An independence of causal interactions model for opposing influences. In: Jaeger, M., Nielsen, T.D. (eds.) Proceedings of 4th European Workshop on Probabilistic Graphical Models, Hirtshals, Denmark, pp. 185–192 (2008)
8. Madsen, A.L., Jensen, F.V.: Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artif. Intell.* **113**(1–2), 203–245 (1999)
9. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
10. Vomlel, J., Tichavský, P.: An approximate tensor-based inference method applied to the game of minesweeper. In: van der Gaag, L.C., Feelders, A.J. (eds.) PGM 2014. LNCS (LNAI), vol. 8754, pp. 535–550. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11433-0_35
11. Woudenbergh, S., van der Gaag, L.C., Rademaker, C.: An intercausal cancellation model for Bayesian-network engineering. *Inter. J. Approximate Reasoning* **63**, 3247 (2015)
12. Xiang, Y.: Non-impeding noisy-AND tree causal models over multi-valued variables. *Int. J. Approximate Reasoning* **53**(7), 988–1002 (2012)
13. Xiang, Y., Jiang, Q.: NAT model based compression of Bayesian network CPTs over multi-valued variables. *Computational Intelligence* (2017). <https://doi.org/10.1111/coin.12126>. Paper version in press
14. Xiang, Y., Jin, Y.: Efficient probabilistic inference in Bayesian networks with multi-valued NIN-AND tree local models. *Int. J. Approximate Reasoning* **87**, 67–89 (2017)
15. Xiang, Y., Liu, Q.: Compression of Bayesian networks with NIN-AND tree modeling. In: van der Gaag, L.C., Feelders, A.J. (eds.) PGM 2014. LNCS (LNAI), vol. 8754, pp. 551–566. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11433-0_36
16. Xiang, Y., Truong, M.: Acquisition of causal models for local distributions in Bayesian networks. *IEEE Trans. Cybern.* **44**(9), 1591–1604 (2014)



De-Causalizing NAT-Modeled Bayesian Networks for Inference Efficiency

Yang Xiang^(✉) and Dylan Loker

University of Guelph, Guelph, Canada
yxiang@uoguelph.ca

Abstract. Conditional independence encoded in Bayesian networks (BNs) avoids combinatorial explosion on the number of variables. However, BNs are still subject to exponential growth of space and inference time on the number of causes per effect variable in each conditional probability table (CPT). A number of space-efficient local models exist that allow efficient encoding of dependency between an effect and its causes, and can also be exploited for improved inference efficiency. We focus on the Non-Impeding Noisy-AND Tree (NIN-AND Tree or NAT) models due to its multiple merits. In this work, we develop a novel framework, *de-causalization* of NAT-modeled BNs, by which causal independence in NAT models can be exploited for more efficient inference. We demonstrate its exactness and efficiency impact on inference based on lazy propagation (LP).

Keywords: Bayesian nets · Causal independence models
Probabilistic inference

1 Introduction

Conditional independence encoded in BNs avoids combinatorial explosion on the number of variables. However, BNs are still subject to exponential growth of space and inference time on the number of causes per effect variable in each CPT. A number of space-efficient local models exist that allow efficient encoding of dependency between an effect and its causes. They include noisy-OR [9], noisy-MAX [2,4], context-specific independence (CSI) [1], recursive noisy-OR [5], Non-Impeding Noisy-AND Tree (NIN-AND Tree or NAT) [13], DeMorgan [6], tensor-decomposition [10], and cancellation model [11].

We consider expressing BN CPTs as, or compressing them into, NAT models [13]. The merits of NAT models include being based on simple causal interactions (reinforcement and undermining), expressiveness (recursive mixture, multi-valued), and generality (generalizing noisy-OR, noisy-MAX [15], and DeMorgan [13]). Since causal independence encoded in a NAT model is orthogonal to CSI, NAT models provide an alternative to CSI for efficient local modeling in BNs.

Local models not only reduce space and time to acquire CPT parameters, they can also be exploited to improve inference efficiency. Through multiplicative

factorization (MF) of NAT-modeled BNs, inference based on LP [7] was made up to two orders of magnitude faster in very sparse BNs [15]. In this work, we develop a framework alternative to MF which is referred to as *de-causalization* of NAT-modeled BNs, where causal independence in NAT models can be exploited for more efficient inference. We also evaluate its impact on LP efficiency.

The remainder of the paper is organized as follows. Section 2 reviews the background on NAT modeling. In Sects. 3 and 4, we present how to de-causalize dual and direct NIN-AND gate models. The tree-width of the de-causalized representation is further reduced in Sect. 5. This de-causalization is extended to a general NAT model in Sect. 6 and then to a NAT-modeled BN in Sect. 7. The impact of de-causalization is empirically evaluated in Sect. 8.

2 Background on NAT Models

This section briefly reviews background on NAT models. More details can be found in [13, 14]. A NAT model is defined over an effect e and a set of n causes $C = \{c_1, \dots, c_n\}$ that are multi-valued, where $e \in D_e = \{e^0, \dots, e^\eta\}$ ($\eta \geq 1$) and $c_i \in \{c_i^0, \dots, c_i^{m_i}\}$ ($i = 1, \dots, n, m_i \geq 1$). C and e form one family (a child variable plus its parents) in a BN, whose dependence is quantified by a CPT by default. Values e^0 and c_i^0 are *inactive*. Other values (may be written as e^+ or c_i^+) are *active*. A higher index often means higher intensity (graded or ordinal variables), but that is not necessary (see [14] for generalization to nominal variables).

A causal event is a *success* or *failure* depending on if e is active up to a given value, is *single-* or *multi-causal* depending on the number of active causes, and is *simple* or *congregate* depending on value range of e . For instance, $P(e^k \leftarrow c_i^j) = P(e^k | c_i^j, c_z^0 : \forall z \neq i)$ ($j > 0$) is probability of a *simple single-causal success*, and

$$P(e \geq e^k \leftarrow c_1^{j_1}, \dots, c_q^{j_q}) = P(e \geq e^k | c_1^{j_1}, \dots, c_q^{j_q}, c_z^0 : c_z \in C \setminus X)$$

is probability of a *congregate multi-causal success*, where $j_1, \dots, j_q > 0$, $X = \{c_1, \dots, c_q\}$ ($q > 1$). The latter may be denoted as $P(e \geq e^k \leftarrow \underline{x}^+)$. Interactions among causes may be reinforcing or undermining as defined below.

Definition 1. Let e^k be an active effect value, $R = \{W_1, \dots, W_m\}$ ($m \geq 2$) be a partition of a set $X \subseteq C$ of causes, $S \subset R$, and $Y = \cup_{W_i \in S} W_i$. Sets of causes in R reinforce each other relative to e^k , iff $\forall S P(e \geq e^k \leftarrow \underline{y}^+) \leq P(e \geq e^k \leftarrow \underline{x}^+)$. They undermine each other iff $\forall S P(e \geq e^k \leftarrow \underline{y}^+) > P(e \geq e^k \leftarrow \underline{x}^+)$.

A NAT has multiple NIN-AND gates. A *direct* gate involves disjoint sets of causes W_1, \dots, W_m . Each input event is a success $e \geq e^k \leftarrow \underline{w}_i^+$ ($i = 1, \dots, m$), e.g., Fig. 1(a) where each W_i is a singleton. The output event is $e \geq e^k \leftarrow \underline{w}_1^+, \dots, \underline{w}_m^+$. The probability of output event of a direct NIN-AND gate is

$$P(e \geq e^k \leftarrow \underline{w}_1^+, \dots, \underline{w}_m^+) = \prod_{i=1}^m P(e \geq e^k \leftarrow \underline{w}_i^+). \quad (1)$$

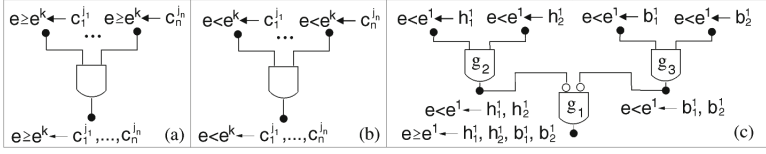


Fig. 1. A direct NIN-AND gate (a), a dual NIN-AND gate (b), and a NAT (c).

Direct gates encode undermining causal interactions. Each input event of a *dual* gate is a failure $e < e^k \leftarrow \underline{w}_i^+$, e.g., Fig. 1(b). The output event is $e < e^k \leftarrow \underline{w}_1^+, \dots, \underline{w}_m^+$. The probability of output event of a dual NIN-AND gate is

$$P(e < e^k \leftarrow \underline{w}_1^+, \dots, \underline{w}_m^+) = \prod_{i=1}^m P(e < e^k \leftarrow \underline{w}_i^+). \quad (2)$$

Dual gates encode reinforcement causal interactions.

Figure 1(c) shows a NAT, where causes h_1 and h_2 reinforce each other, and so do b_1 and b_2 . However, the two groups undermine each other. From the NAT and probabilities of its input events, in the general form $P(e^k \leftarrow c_i^j)$ ($j, k > 0$), called *single-causals*, $P(e \geq e^1 \leftarrow h_1^1, h_2^1, b_1^1, b_2^1)$ can be obtained. From the single-causals and all derivable NATs [12], CPT $P(e|h_1, h_2, b_1, b_2)$ is uniquely specified [13]. A NAT model is specified by the topology and a set of single-causals with a space linear on n .

A BN where the CPT of every family of size 3 or larger is a NAT model is a *NAT-modeled BN*. A discrete BN where every CPT is tabular has a space complexity of $O(N \kappa^n)$, where N is the number of variables, κ is the size of largest variable domains, and $n+1$ is the largest family size. On the other hand, a NAT-modeled BN has a linear space complexity of $O(N \kappa n)$. The efficiency of NAT-modeled BNs can extend to inference. A MF framework has been developed [15], where each NAT model in the BN is converted into a hybrid network segment by exploiting causal independence. The multiplicatively factorized NAT-model BN allows up to two orders of magnitude speedup in LP for very sparse BNs.

3 De-Causalizing Dual NIN-AND Gate Models

First, we de-causalize a dual NIN-AND gate model, a building block of NAT models. It has been shown that a dual NIN-AND gate over multi-valued variables is equivalent to noisy-MAX [15]. A BN segment can be used [3] to structure noisy-MAX models. Given the equivalence between a dual NIN-AND gate and a noisy-MAX model, any structuring of noisy-MAX is also applicable to dual gates. Nevertheless, we give below an alternative justification of the mapping of dual gates to the BN segment, that is direct and hence more intuitive.

Figure 2 shows the BN segment structure, where root variables are the n causes and the leaf is the effect e . For each cause c_i , a probabilistic auxiliary

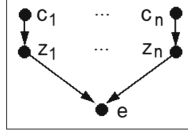


Fig. 2. The DAG structure of BN segment of an NIN-AND gate model.

child variable z_i is introduced, whose domain is D_e . It represents the impact of cause c_i to effect e . The CPT at z_i , referred as *single-causal* (SC) CPT, is

$$P(z_i = e^j | c_i = c_i^k) = \begin{cases} 1, & \text{if } e^j = e^0 \text{ and } c_i^k = c_i^0, \\ P(e^j \leftarrow c_i^k), & \text{if } e^j > e^0 \text{ and } c_i^k > c_i^0. \end{cases} \quad (3)$$

The 1st formula says that when c_i is inactive, it cannot render e active. The 2nd formula expresses the impact to e when c_i is active. The CPT at e , referred to as a *MAX* CPT, encodes a *MAX* function as follows, where the domain of every variable is D_e and the number of α_i variables is finite.

$$P(\tau | \alpha_1, \alpha_2, \dots) = \begin{cases} 1, & \text{if } \tau = \text{MAX}(\alpha_1, \alpha_2, \dots), \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

For the MAX CPT at e , τ is substituted by e and $\alpha_1, \alpha_2, \dots$ by z_1, \dots, z_n .

Definition 2. Let G be the DAG in Fig. 2 over $C = \{c_1, \dots, c_n\}$ and e , and CP be the set of CPTs specified by Eqs. (3) and (4). Then $\Phi = (C, e, G, CP)$ is the **BN segment for a dual NIN-AND gate model**.

We show below that the BN segment Φ is equivalent to the dual gate model illustrated in Fig. 1(b).

Proposition 1. Let $\Phi = (C, e, G, CP)$ be a BN segment for a dual NIN-AND gate model. Then the CPT $P(e | c_1, \dots, c_n)$ from Φ defined by the marginalized product

$$\sum_{z_1, \dots, z_n} (P(e | z_1, \dots, z_n) \prod_{i=1}^n P(z_i | c_i))$$

is the same as that defined by the dual NIN-AND gate model.

Proof: The dual NIN-AND gate model is characterized by Eq. (2). When each set of causes is a singleton, Eq. (2) becomes the following where, without losing generality, c_1, \dots, c_m are active and c_{m+1}, \dots, c_n are inactive:

$$P(e < e^k \leftarrow c_1^+, \dots, c_m^+) = \prod_{i=1}^m P(e < e^k \leftarrow c_i^+), \quad (k = 1, \dots, \eta).$$

It is equivalent to

$$P(e \leq e^k \leftarrow c_1^+, \dots, c_m^+) = \prod_{i=1}^m P(e \leq e^k \leftarrow c_i^+), \quad (k = 0, \dots, \eta - 1),$$

which is a cumulative causal distribution. If Φ has the same cumulative distribution (which we show below), then $P(e|c_1, \dots, c_n)$ from Φ is also the same as that of the dual gate model.

Assume that c_1, \dots, c_m are active and c_{m+1}, \dots, c_n are inactive. In Φ , since the CPT by Eq. (4) encodes the *MAX* function, we have

$$P(e \leq e^k \leftarrow c_1^+, \dots, c_m^+) = \sum_{MAX(z_1, \dots, z_n) \leq e^k} P(z_1, \dots, z_n | c_1^+, \dots, c_m^+, c_{m+1}^0, \dots, c_n^0).$$

Since $MAX(z_1, \dots, z_n) \leq e^k$ iff $z_i \leq e^k$ for $i = 1, \dots, n$, the above is equal to

$$\begin{aligned} & \sum_{z_1 \leq e^k, \dots, z_n \leq e^k} P(z_1, \dots, z_n | c_1^+, \dots, c_m^+, c_{m+1}^0, \dots, c_n^0) \\ &= \sum_{z_1 \leq e^k} \dots \sum_{z_n \leq e^k} P(z_1, \dots, z_n | c_1^+, \dots, c_m^+, c_{m+1}^0, \dots, c_n^0). \end{aligned}$$

By the DAG structure of Φ , z_i is independent of z_j for $i \neq j$ given c_i . Hence, the above equals

$$\begin{aligned} & \sum_{z_1 \leq e^k} \dots \sum_{z_n \leq e^k} (P(z_1 | c_1^+) \dots P(z_m | c_m^+) P(z_{m+1} | c_{m+1}^0) \dots P(z_n | c_n^0)) \\ &= \sum_{z_1 \leq e^k} P(z_1 | c_1^+) \dots \sum_{z_m \leq e^k} P(z_m | c_m^+) \dots \sum_{z_{m+1} \leq e^k} P(z_{m+1} | c_{m+1}^0) \dots \sum_{z_n \leq e^k} P(z_n | c_n^0). \end{aligned}$$

Since $\sum_{z_i \leq e^k} P(z_i | c_i^0) = 1$ for $i = m+1, \dots, n$, the above equals

$$\sum_{z_1 \leq e^k} P(z_1 | c_1^+) \dots \sum_{z_m \leq e^k} P(z_m | c_m^+) = \prod_{i=1}^m P(z_i \leq e^k \leftarrow c_i^+).$$

From Eq. (3), the above equals $\prod_{i=1}^m P(e \leq e^k \leftarrow c_i^+)$. Hence, we have

$$P(e \leq e^k \leftarrow c_1^+, \dots, c_m^+) = \prod_{i=1}^m P(e \leq e^k \leftarrow c_i^+). \quad \square$$

4 De-Causalizing Direct NIN-AND Gate Models

Next, we de-causalize a direct NIN-AND gate model, another building block of NAT models. The structure of BN segment is the same as Fig. 2. However, the domain of each auxiliary variable z_i is $D_a = \{e^0, \dots, e^\eta, aaci\}$, where an extra value *aaci* (all above causes inactive) is added to D_e . Its semantics are elaborated below. When values of z_i are compared, the relation $e^0 < \dots < e^\eta < aaci$ is assumed. Note that events in Fig. 1 are causal events, while events in Fig. 2 are not. Hence the name *de-causalization*.

The CPT at z_i , referred to as a *single-causal-plus* (SC^+) CPT, is the following, where $^+$ signifies the enlarged domain of z_i beyond D_e :

$$\begin{cases} P(z_i = aaci | c_i = c_i^0) = 1, \\ P(z_i = e^j | c_i = c_i^k) = P(e^j \leftarrow c_i^k), \quad (e^j > e^0, c_i^k > c_i^0). \end{cases} \quad (5)$$

The 1st formula explicitly signifies that c_i is inactive (the above cause is inactive). The 2nd formula covers all cases where c_i is active. The CPT at e , referred to as *PMIN* CPT, encodes a pseudo-MIN (PMIN) function below over a finite set of arguments, where each argument has domain D_a and function range is D_e :

$$PMIN(\alpha_1, \alpha_2, \dots) = \begin{cases} e^0, & \text{if } \forall_i \alpha_i = aaci, \\ MIN(\alpha'_1, \dots, \alpha'_m), & \text{if } \alpha'_1, \dots, \alpha'_m \neq aaci \ (m > 0). \end{cases}$$

The PMIN CPT at e is the following:

$$P(\tau | \alpha_1, \alpha_2, \dots) = \begin{cases} 1, & \text{if } \forall_i \alpha_i = aaci \ \wedge \ \tau = e^0, \\ 1, & \text{if } \alpha'_1, \dots, \alpha'_m \neq aaci \ (m > 0) \ \wedge \ \tau = MIN(\alpha'_1, \dots, \alpha'_m). \end{cases} \quad (6)$$

We define the BN segment below and establish its soundness.

Definition 3. Let G be the DAG in Fig. 2 over $C = \{c_1, \dots, c_n\}$ and e , and CP be the set of CPTs specified by Eqs. (5) and (6). Then $\Phi = (C, e, G, CP)$ is the **BN segment for a direct NIN-AND gate model**.

We show below that the BN segment Φ is equivalent to the dual gate model illustrated in Fig. 1(a). The proof is omitted due to space limit.

Proposition 2. Let $\Phi = (C, e, G, CP)$ be a BN segment for a direct NIN-AND gate model. Then the CPT $P(e | c_1, \dots, c_n)$ from Φ defined by the marginalized product

$$\sum_{z_1, \dots, z_n} (P(e | z_1, \dots, z_n) \prod_{i=1}^n P(z_i | c_i))$$

is the same as that defined by the direct NIN-AND gate model.

5 Reducing Tree-Width of BN Segment

The complexity of probabilistic reasoning with a BN is critically dependent on its tree-width. By reducing the tree-width of BN segments, the tree-width of a BN may also be reduced. The BN segments presented in previous sections have a tree-width of n . Below, we take advantage of deterministic CPTs in Eqs. (4) and (6), and apply parent divorcing [8] to reduce the tree-width of these BN segments from n to 2.

Figure 3 shows the enhanced DAG structure. A total of $n - 2$ deterministic auxiliary variables y_i are introduced. Since the DAG is a directed tree where each node has no more than two parents, its tree-width is 2.

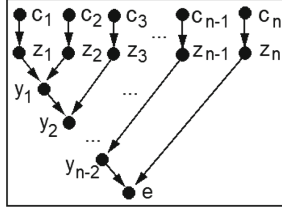


Fig. 3. The DAG structure of BN segment by applying parent divorcing.

For the enhanced BN segment of a dual gate model, the domain of each y_i is D_e . The CPT at each y_i ($i = 1, \dots, n - 2$) and e is a MAX CPT defined by Eq. (4). It can be shown that the collection of CPTs at y_i and e is equivalent to the single MAX CPT $P(e|z_1, \dots, z_n)$ described in Sect. 3. We omit the proof due to space considerations.

Assume that all cause variables have the same domain size $\eta + 1$ as e . The total size of the CPT collection is $(n - 1)(\eta + 1)^3$, while the single CPT has a size of $(\eta + 1)^{n+1}$. For $n = \eta = 4$, the two sizes are 375 and 3125.

For the enhanced BN segment of a direct gate model, the domain of each y_i is D_a . The CPT at e is a PMIN CPT defined by Eq. (6), where condition variables are y_{n-2} and z_n . When one of y_{n-2} and z_n is not *aaci*, the MIN function is trivialized.

The CPT at each y_i ($i = 1, \dots, n - 2$), referred to as a $PMIN^+$ CPT, encodes the following pseudo-MIN-plus ($PMIN^+$) function:

$$PMIN^+(\alpha_1, \alpha_2) = \begin{cases} aaci, & \text{if } \alpha_i = aaci \ (i = 1, 2), \\ MIN(\alpha'_1, \alpha'_m), & \text{if } \alpha'_1, \alpha'_m \neq aaci \ (m > 0). \end{cases}$$

When $m = 1$, the MIN function is trivial. The $PMIN^+$ CPT at each y_i is the following, where τ is substituted by y_i , and α_i are substituted by parents of y_i :

$$P(\tau|\alpha_1, \alpha_2) = \begin{cases} 1, & \text{if } \alpha_i = aaci \ (i = 1, 2) \ \wedge \ \tau = aaci, \\ 1, & \text{if } \alpha'_1, \alpha'_m \neq aaci \ (m > 0) \ \wedge \ \tau = MIN(\alpha'_1, \alpha'_m). \end{cases} \quad (7)$$

The 1st formula signifies that all causes above y_i are inactive, so that the non-impeding behavior of a direct gate model is enabled. It can be shown that the collection of CPTs at y_i and e are equivalent to the single PMIN CPT described in Sect. 4. The size of the CPT collection is $(n - 2)(\eta + 2)^3 + (\eta + 1)(\eta + 2)^2$, while the single PMIN CPT has a size of $(\eta + 1)(\eta + 2)^n$. For $n = \eta = 4$, the two sizes are 612 and 6480.

6 De-Causalizing NAT Models

A NAT model generally consists of multiple NIN-AND gates organized into a tree. To de-causalize a general NAT model, we apply the BN segment for each

gate and interface the gate segments so that the BN segment of the NAT model encodes the exact CPT of the NAT model. If an NIN-AND gate feeds into another in the NAT model, its effect variable is replaced with a *quasi-effect* variable.

Consider the NAT in Fig. 4(a), where labels of causal events have been simplified (e.g., input events to gates) or omitted (e.g., output events). Suppose that the leaf gate g_2 is dual. Then g_1 is direct. The (enhanced) BN segment of the NAT is shown in (b). The BN segment of g_1 consists of causes variables c_i ($i = 1, 2, 3$), probabilistic auxiliary variables z_i ($i = 1, 2, 3$), deterministic auxiliary variable y_1 , and quasi-effect variable q . This segment can be implemented as in Sects. 4 and 5, except that the variable e there is renamed as q .

The BN segment of g_2 consists of causes variables c_i ($i = 4, 5$), quasi-effect variable q as an input from g_1 , probabilistic auxiliary variables z_j ($j = 4, 5$), deterministic auxiliary variable y_2 , and effect variable e . This segment can be implemented as in Sects. 3 and 5, except that the quasi-effect variable q should be treated in the same way as probabilistic auxiliary variables z_j ($j = 4, 5$).

Next, suppose that the leaf gate g_2 is direct and g_1 is dual. The BN segment of the NAT is the same as in Fig. 4(b). However, the BN segment of dual gate g_1 must be modified relative to that of Sects. 3 and 5. In Sects. 3 and 5, auxiliary variables z_i and y_i , as well as the effect e , have the domain D_e . This is no longer valid. Since g_1 is not the leaf gate, it now feeds into the direct gate g_2 . To support non-impeding behavior of the direct gate, domains of z_i , y_i , and quasi-effect q have to be enlarged into D_a .

Due to this enlargement, SC CPTs cannot be applied to z_i ($i = 1, 2, 3$), and MAX CPTs cannot be applied to y_1 and q . Instead, auxiliary variables z_i ($i = 1, 2, 3$) adopt SC^+ CPTs (Eq. (5)). A new form of CPT is needed for y_1 and q . It is referred to as $PMAX^+$ CPTs, and encodes the following pseudo-MAX-plus ($PMAX^+$) function, where domain of each argument and function range are D_a :

$$PMAX^+(\alpha_1, \alpha_2) = \begin{cases} aaci, & \text{if } \alpha_i = aaci \ (i = 1, 2), \\ MAX(\alpha'_1, \alpha'_m), & \text{if } \alpha'_1, \alpha'_m \neq aaci \ (m > 0). \end{cases}$$

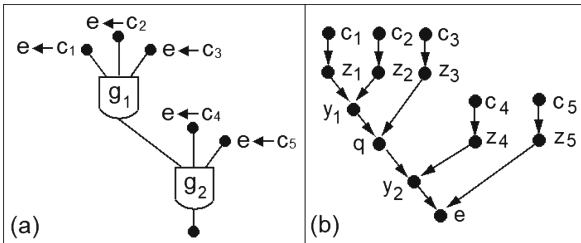


Fig. 4. (a) A NAT. (b) The enhanced BN segment.

The PMAX⁺ CPTs at y_1 and q are the following:

$$P(\tau|\alpha_1, \alpha_2) = \begin{cases} 1, & \text{if } \alpha_i = aaci \ (i = 1, 2) \wedge \tau = aaci, \\ 1, & \text{if } \alpha'_1, \alpha'_m \neq aaci \ (m > 0) \wedge \tau = MAX(\alpha'_1, \alpha'_m). \end{cases} \quad (8)$$

The BN segment of the direct leaf gate g_2 can be encoded as Sects. 4 and 5, except that the quasi-effect q should be treated in the same way as auxiliary variables z_4 and z_5 .

In general, domains of auxiliary variables (both probabilistic and deterministic) and quasi-effect variables should be set as summarized in Table 1, where level 0 is the leaf level. The primary criteria are to keep the domain as small as possible, while ensuring non-impeding behavior of direct gates.

CPTs for auxiliary, quasi-effect, and effect variables should be set as summarized in Table 2, where the last column refers to effect (level 0) or quasi-effect (level 1+). The primary criteria are to maintain exact CPT as corresponding NIN-AND gate model, while ensuring non-impeding behavior of direct gates downstream.

It can be shown formally that the collection of CPTs in the BN segment of the NAT model ensures the exact $P(e|c_1, \dots, c_n)$ of the NAT model. We omit the formal analysis due to space restriction. Instead, we demonstrate the exactness empirically in Sect. 8.

Table 1. Summary of variable domains

		Auxiliary variable	Quasi-effect
Level 0	Dual gate	D_e	NA
	Direct gate	D_a	NA
Level 1	Dual gate	D_a	D_a
	Direct gate	D_a	D_e
Level 2+	Dual gate	D_a	D_a
	Direct gate	D_a	D_a

Table 2. Summary of variable CPTs

Level	Gate	Probabilistic aux	Deterministic aux	(Quasi)-effect
0	Dual	SC CPT	MAX CPT	MAX CPT
	Direct	SC ⁺ CPT	PMIN ⁺ CPT	PMIN CPT
1	Dual	SC ⁺ CPT	PMAX ⁺ CPT	PMAX ⁺ CPT
	Direct	SC ⁺ CPT	PMIN ⁺ CPT	PMIN CPT
2+	Dual	SC ⁺ CPT	PMAX ⁺ CPT	PMAX ⁺ CPT
	Direct	SC ⁺ CPT	PMIN ⁺ CPT	PMIN ⁺ CPT

7 De-Causalizing NAT-Modeled Bayesian Networks

To de-causalize a NAT-modeled BN, for each NAT model family (child e plus parents c_1, \dots, c_n), delete the directed link from each parent to the child (as well as the CPT of the child), reconnect the family by the de-causalizing BN segment, and assign a CPT to each variable (except c_1, \dots, c_n) as presented above.

Consider the example NAT-modeled BN in Fig. 5, where the NAT model over family of v_8 is shown with simplified labeling, and all variables are ternary. The gate g_3 is direct and remaining gates are dual.

The de-causalized BN is shown in Fig. 6. For causes v_i ($i = 1, \dots, 7$) in that order, the probabilistic auxiliary variables are $x_{10}, x_{16}, x_{11}, x_{20}, x_{17}, x_{21}, x_{18}$, respectively. For gate g_2 , the quasi-effect is q_{13} . For gate g_1 , the deterministic auxiliary variable is y_{19} and the quasi-effect is q_{12} . For gate g_3 , the deterministic auxiliary variables are y_{14} and y_{15} . A BN with the DAG in Fig. 5(left), where all variables are ternary and all CPTs are tabular, has 6642 numerical parameters (values in all CPTs). The de-causalized BN has 489 parameters.

Let a NAT-modeled BN be over the set V of variables and its de-causalized BN be over the set $V \cup W$ of variables, where W is the set of all auxiliary and

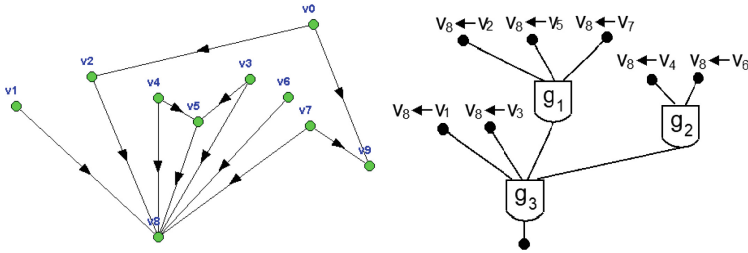


Fig. 5. Left: DAG of a NAT-model BN. Right: NAT-model over family of v_8 .

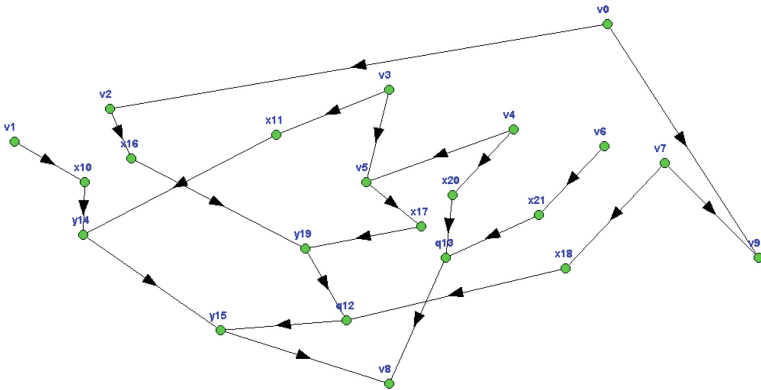


Fig. 6. The NAT-modeled BN in Fig. 5 after de-causalization.

quasi-effect variables. Let $P(V)$ be the joint probability distribution (JPD) of the NAT-modeled BN, and $P(V, W)$ be the JPD of the de-causalized BN. Since for each replaced BN family, the CPT $P(e|c_1, \dots, c_n)$ specified by the de-causalizing segment is equal to the original NAT CPT of the family, we have

$$\sum_{w \in W} P(V, W) = P(V).$$

The de-causalized BN can be used for probabilistic reasoning using any standard inference algorithm. Only observations over variables in V can be entered, as variables in W are not observable. In Sect. 8, we demonstrate the posterior marginals thus computed from de-causalized BNs are exact as computed from original NAT-modeled BNs.

8 Experiments

The 1st experiment evaluates the space of a NAT model-CPT as a tabular CPT (TAB), of a de-causalized CPT without parent divorcing (DEC), and of a de-causalized CPT with parent divorcing (DPD). The numbers of causes per CPT are $n = 5, 7, 9, 11$. The uniform domain sizes of variables in each CPT are $d = 3, 5, 7$. For each combination of (n, d) , 30 random NAT topologies are generated. Hence, a total of $4 * 3 * 30 = 360$ distinctly structured NAT models are evaluated.

Figure 7 show spaces of CPTs in \log_{10} by TAB, DEC, and DPD when $d = 7$. Due to space consideration, we omit presentation of result for $d = 3, 5$. Space of TAB CPTs are completely determined by n and d , and are constant. Spaces of both DEC and DPD CPTs are sensitive to the NAT topology, but DPD CPTs are only slightly so. DEC CPTs are often more space-efficient than TAB CPTs. But for some NAT topologies, they are less efficient. For instance, the 11th DEC CPT for $n = 11$ and $d = 7$ is less efficient than the TAB CPT, whose NAT has two gates and one of them has 10 inputs. DPD CPTs are always the most efficient, and are 5 orders of magnitude more efficient than TAB CPTs for $n = 11$ and $d = 7$.

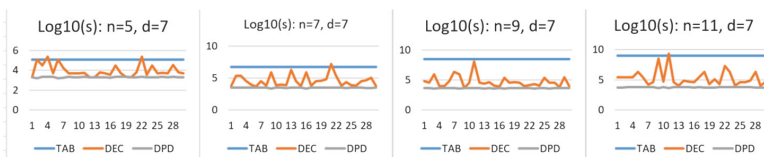


Fig. 7. Spaces (s) of CPTs as TAB, DEC, and DPD, where $d = 7$.

The 2nd experiment evaluates the impact of de-causalization on inference efficiency, where the inference method is LP. We simulated NAT-modeled BNs with 100 variables per BN. The maximum number of parents per variable in

each BN is bounded at $m = 6, 8, 10, 12$, respectively. The uniform domain size of all variables is controlled at $s = 2, 3$, respectively. The structural density of BNs is controlled by adding $w = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55\%$ of links to a singly connected network, respectively. Hence, there are a total of $4 \times 2 \times 11 = 88$ distinct (m, s, w) combinations. For each combination, we simulated 10 BNs. This amounts to a total of 880 NAT-modeled BNs.

For each NAT-modeled BN, we created a *normalized* BN (NM-BN) where each NAT model is expanded into a tabular CPT, and a de-causalized BN (DC-BN). Both NM-BN and DC-BN are compiled for LP, conditioned on the same observation over 10% of randomly selected variables. For each pair of NM-BN and DC-BN, LP resulted in the same posterior marginals, which empirically demonstrates exactness of de-causalization. LP runtimes for $m = 10, 12$, using a desktop of 3.4 GHz clock speed, are summarized in Fig. 8. Runtimes for $m = 6, 8$ are omitted due to space restriction.

The inference becomes harder as m, s and w grow. For sparse BN structures, as inference becomes harder, DC-BNs become more advantageous than NM-BNs. For instance, with $w = 5$, as m and s grow, the runtime by DC-BNs become significantly less than NM-BNs. At $(m = 12, s = 3, w = 5)$, LPs with DC-BNs are two orders of magnitude faster than NM-BNs.

Furthermore, as m and s grow, the range of structural densities where DC-BNs are more efficient than NM-BNs grows as well. For instance, for $(m = 6, s = 3)$, DC-BNs and NM-BNs tie in runtime around $w = 20$. As m grows to 8, 10, 12, the corresponding structural density grows to $w = 30, 50, 55$, respectively.

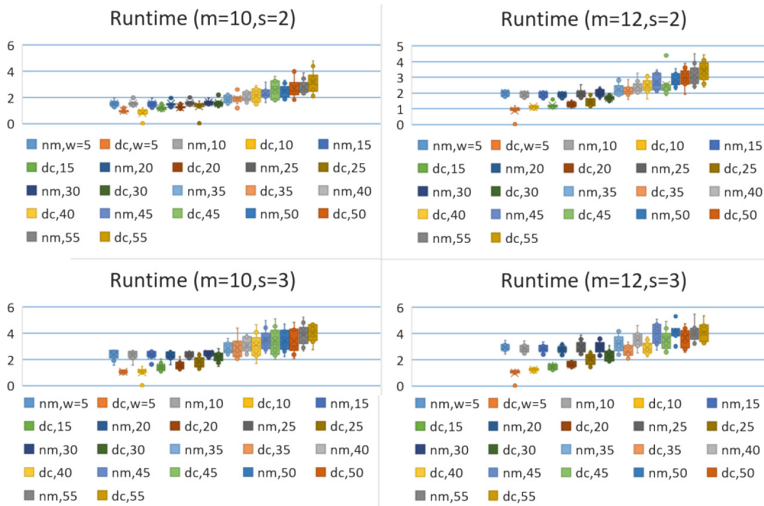


Fig. 8. LP runtimes (msec in log10) for NM-BNs and DC-BNs where $m = 10, 12$.

9 Conclusion

The main contribution of this work is the novel de-causalization framework, by which a NAT-modeled BN is converted into a de-causalized BN for inference computation. An existing alternative is the MF framework. In comparison, the MF frame has a limitation where one potential for each NIN-AND gate is exponential on η (domain size of effect e). In the de-causalization framework, no component is exponential. We demonstrated that the de-causalized BNs support exact inference, and can speed up LP inference by up to two orders of magnitude for a wide range of sparse BN structures. We are currently investigating the efficiency impact of de-causalization, when inference is performed through sum-product networks (SPNs).

Acknowledgement. Financial support from NSERC Discovery Grant to first author is acknowledged.

References

1. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Horvitz, E., Jensen, F. (eds.) *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pp. 115–123 (1996)
2. Diez, F.J.: Parameter adjustment in Bayes networks: the generalized noisy OR-gate. In: Heckerman, D., Mamdani, A. (eds.) *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pp. 99–105. Morgan Kaufmann (1993)
3. Diez, F.J., Druzdzel, M.J.: Canonical probabilistic models for knowledge engineering. Technical report cisiad-06-01, UNED (2007)
4. Henrion, M.: Some practical issues in constructing belief networks. In: Kanal, L.N., Levitt, T.S., Lemmer, J.F. (eds.) *Uncertainty in Artificial Intelligence 3*, pp. 161–173. Elsevier Science Publishers (1989)
5. Lemmer, J.F., Gossink, D.E.: Recursive noisy OR - a rule for estimating complex probabilistic interactions. *IEEE Trans. Syst. Man Cybern. Part B* **34**(6), 2252–2261 (2004)
6. Maaskant, P.P., Druzdzel, M.J.: An independence of causal interactions model for opposing influences. In: Jaeger, M., Nielsen, T.D. (eds.) *Proceedings of the 4th European Workshop on Probabilistic Graphical Models*, Hirtshals, Denmark, pp. 185–192 (2008)
7. Madsen, A.L., Jensen, F.V.: Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artif. Intell.* **113**(1–2), 203–245 (1999)
8. Olesen, K.G., Kjrulff, U., Jensen, F., Jensen, F.V., Falck, B., Andreassen, S., Andersen, S.K.: A munin network for the median nerve—a case study on loops. *Appl. Artif. Intell.* **3**(2–3), 385–403 (1989)
9. Pearl, J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco (1988)
10. Vomlel, J., Tichavský, P.: An approximate tensor-based inference method applied to the game of minesweeper. In: van der Gaag, L.C., Feelders, A.J. (eds.) *PGM 2014. LNCS (LNAI)*, vol. 8754, pp. 535–550. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11433-0_35

11. Woudenberg, S., van der Gaag, L.C., Rademaker, C.: An intercausal cancellation model for Bayesian-network engineering. *Int. J. Approximate Reason.* **63**, 32–47 (2015)
12. Xiang, Y.: Acquisition and computation issues with NIN-AND tree models. In: Myllymaki, P., Roos, T., Jaakkola, T. (eds.) *Proceedings of the 5th European Workshop on Probabilistic Graphical Models*, Finland, pp. 281–289 (2010)
13. Xiang, Y.: Non-impeding noisy-AND tree causal models over multi-valued variables. *Int. J. Approximate Reason.* **53**(7), 988–1002 (2012)
14. Xiang, Y., Jiang, Q.: NAT model based compression of Bayesian network CPTs over multi-valued variables. *Comput. Intell.* (2017). <https://doi.org/10.1111/coin.12126>. (Paper version in press)
15. Xiang, Y., Jin, Y.: Efficient probabilistic inference in Bayesian networks with multi-valued NIN-AND tree local models. *Int. J. Approximate Reason.* **87**, 67–89 (2017)



A Novel Evaluation Methodology for Assessing Off-Policy Learning Methods in Contextual Bandits

Negar Hassanpour^(✉) and Russell Greiner

University of Alberta, Edmonton, Canada
{hassanpo,rgreiner}@ualberta.ca

Abstract. We propose a novel evaluation methodology for assessing off-policy learning methods in contextual bandits. In particular, we provide a way to use data from any given Randomized Control Trial (RCT) to generate a range of observational studies with synthesized “outcome functions” that can match the user’s specified degrees of sample selection bias, which can then be used to comprehensively assess a given learning method. This is especially important in evaluating methods developed for precision medicine, where deploying a bad policy can have devastating effects. As the outcome function specifies the real-valued quality of *any* treatment for any instance, we can accurately compute the quality of any proposed treatment policy. This paper uses this evaluation methodology to establish a common ground for comparing the robustness and performance of the available off-policy learning methods in the literature.

Keywords: Contextual bandits · Off-policy learning
Evaluation method

1 Introduction

Precision medicine is a rapidly emerging field that tries to determine which specific treatment (*e.g.*, surgery, drug therapy, etc.) leads to the best outcome for each patient. This not only improves the patients’ quality of life, but also typically reduces the health-care costs, especially in situations where the first course of treatment does not provide desirable outcomes, which means the patient has to receive a second (hopefully better) treatment. Many attempt to learn models for precision medicine from observed data; here, it is often necessary to answer counterfactual questions such as: “*Would this patient have lived longer, had she received an alternative treatment?*”. To answer such questions, it is required to know the underlying causal relationships between the patient’s attributes and the outcomes associated with each potential treatment. Such causal relationships can only be learned from experimental studies that involve making interventions

R. Greiner—The authors were supported by NSERC and Amii.

and collecting data on-line [1]. As such studies are often not available, we have to approximate the causal effects from off-line datasets [2].

In a Randomized Control Trial (RCT), the treatment assignment T to a patient is independent of the patient attributes X (see Fig. 1a). This makes it straightforward to infer the causal effects on a *population* level [2]. However, it is not possible to run an RCT for every causal query; since, they are at best expensive, or in most cases infeasible. By contrast, in observational studies, the health-care provider suggests a treatment based on the patient’s attributes, according to her training and/or the established clinical pathway. This is *sample selection bias*, as the assignment of treatment T depends on the patient’s health history/attributes X (see Fig. 1b). However, while RCTs are rare, observational studies are abundant and are therefore our main source of data for developing algorithms that produce the personalized models needed for precision medicine.



Fig. 1. Types of data collection

The whole setup can be viewed in a *contextual bandit* setting [3] where, given a vector of attributes $x_i \in X$ describing patient i and her received treatment $t_i \in T$, we observe an outcome value $y(x_i, t_i) \in \mathbb{R}$. Treatment is selected according to an established clinical pathway represented by a conditional probability distribution $\pi_0(t|x)$. Following the literature, we refer to this as the *logging policy* (a.k.a. “behaviour policy” in reinforcement learning [4]). Also note that, for each patient, we only get to observe the outcome $y(x_i, t_i)$ associated with the received treatment t_i and not the outcomes associated with the alternative treatment(s) ($t \neq t_i$). Since such *counterfactual* outcome(s) are inherently “unobservable” (and not just “unobserved”), there is no way to truly determine the best personalized treatment for each patient. The case of $y(x_i, t_i) \in \mathbb{R}^{\geq 0}$ might represent life expectancy after treatment, while $y(x_i, t_i) \in \{0, 1\}$ might indicate whether a patient would die or survive. In general, such contextual bandit datasets have the following information: $\mathcal{D} = \{ [x_i, t_i, y(x_i, t_i), \pi_0(t_i|x_i)] \}_{i=1..n}$. The goal here is to find the best policy, $\pi^*(t|x)$, whose most likely selected treatment $t^* = \arg \max_t \pi^*(t|x)$ for patient x , indeed matches the best one. In other words, had we known all outcomes (observed *and* counterfactuals(s)), we want:

$$t^* = \arg \max_t y(x, t) \quad (1)$$

The problem of learning an optimal policy from observational studies (a.k.a. “off-policy learning” [4]) is not limited to medical applications. It also

appears in A/B testing, ad-placement systems [5, 6], recommender systems [7], intelligent tutoring systems [8], etc. It is challenging to evaluate a proposed policy given an observational dataset, due to (i) the inherent unobservability of the counterfactual outcomes, and (ii) the unknown degree of existing selection bias – *i.e.*, the strength of the link between X and T . To overcome these two issues, we have to somehow create synthetic datasets with full information on outcome (*i.e.*, both factual and counterfactual). However, to the best of our knowledge, all the existing approaches only evaluate their proposed method on an observational study generated under a fixed level of selection bias – see Sect. 3.1 for more details. This paper addresses this gap by proposing an evaluation methodology to establish a common ground for comparing the efficiency and effectiveness of the proposed methods for off-policy learning from observational datasets. Our contribution is an algorithm that, given an RCT dataset, can synthesize a spectrum of observational datasets that exhibit various degrees of selection bias.

2 Background

All the existing methods for off-line policy learning must address the sample selection bias to effectively learn an optimal policy from bandit feedback data. Many, including [9, 10], use rejection sampling to extract a RCT-like subset of data. That is, they find and discard a subset of instances, such that the probability of assigning any treatment becomes uniform for all the remaining samples. This approach is extremely data inefficient, which might be acceptable for applications that have access to enormous amounts of data, such as ad-placement in search engines. However, this inefficiency means it cannot be used for small medical datasets. Besides the methods mentioned above, there are two other main approaches for handling the sample selection bias: (i) outcome prediction and (ii) utility maximization. Below, we briefly summarize such methods.

2.1 Outcome Prediction

Outcome prediction (OP) methods (a.k.a. “direct methods”) try to learn a regression fit that predicts the outcome for patient i given feature vector x_i for any treatment $t \in T$ (*i.e.*, $\hat{y}(x_i, t)$). Assuming the learned regression fit is accurate, we can rank different treatments in terms of their estimated outcome and choose the treatment that has the best outcome. This can be done following Eq. 1 by replacing $y(x, t)$ with the predicted (counter)factual outcomes $\hat{y}(x, t)$.

Although this method seems straightforward, obtaining an accurate regression fit is often not easy as this requires addressing (i) selection bias and (ii) modeling bias. While there are ways to deal with selection bias, such as importance sampling (*cf.* [5]) modeling bias is more challenging to address, because it is not easy to identify the right model and/or features to produce an accurate estimator. Moreover, the outcome prediction method tries to answer the harder question of accurately predicting the counterfactual outcomes, while all we need to find an optimal policy is to *rank* the potential treatments. Therefore, other methods have been proposed that try to maximize a utility function instead.

2.2 Inverse Propensity Scoring

Inverse Propensity Scoring (IPS), an *importance sampling* technique that adjusts the weights of different instances, is one of the main ways to address the selection bias problem. Here, we use a variant of the early works of Horvitz and Thompson [11] and Rosenbaum and Rubin [12], since we are interested in evaluating a stochastic policy $\pi(t|x)$ as opposed to a deterministic one. This variant has been used in many closely-related applications, such as off-policy reinforcement learning [4], off-policy learning for contextual bandits [9,10], and counterfactual learning with causal graphs [5]. First, we formulate a utility function with importance sampling weights $\frac{\pi(\cdot)}{\pi_0(\cdot)}$ as:

$$\widehat{U}_{IPS}(\pi) = \frac{1}{n} \sum_{i=1}^n \frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)} y(x_i, t_i) \quad (2)$$

where n is the number of instances in the given contextual bandit dataset, $\pi_0(t_i|x_i)$ is the policy used to sample treatments in the training set (*i.e.*, the “logging policy”), $\pi(t_i|x_i)$ is the probability of selecting t_i given x_i by the proposed policy $\pi(\cdot)$, and $y(x_i, t_i)$ is the observed outcome. Note that IPS is an unbiased estimator of the true [unknown] utility $U(\pi)$, meaning:

$$\mathbb{E}_{\mathcal{D}}[\widehat{U}_{IPS}(\pi)] = U(\pi) \quad (3)$$

for any $\pi(\cdot)$ provided that $\pi_0(\cdot)$ has a non-zero value everywhere in its support [12]. Ultimately, the optimal policy π^* is obtained by:

$$\pi^* = \arg \max_{\pi \in \Pi} \widehat{U}(\pi) = \arg \min_{\pi \in \Pi} \widehat{R}(\pi) \quad (4)$$

where Π is the hypothesis space containing all possible policies and $\widehat{R}(\pi) = -\widehat{U}(\pi)$ is the empirical risk. Although unbiased, the IPS estimator has a high variance due to the $\pi_0(t_i|x_i)$ term in its denominator. That is, some importance sampling weights (*i.e.*, $\frac{\pi(\cdot)}{\pi_0(\cdot)}$) will be very large for any instance with small π_0 – *i.e.*, treatments that had a small chance of being selected, but were selected anyway. The most common approach is to clip the weights [5]:

$$w_i = \min \left\{ M, \frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)} \right\} \quad (5)$$

where the M hyperparameter is an upper bound for the importance sampling weights. The risk calculated with the clipped weights is denoted as $\widehat{R}^M(\pi)$.

2.3 Doubly Robust Estimator

As discussed earlier, methods based on OP enjoy a low variance estimate at the cost of a high bias. On the other hand, although IPS is an unbiased estimator, it suffers from a high variance. This motivated Dudík *et al.* [13] to propose the

Doubly Robust (DR) estimator, which leverages the strengths and mitigates the weaknesses of the above mentioned methods. Prior to [13], Robins *et al.* [14] had proposed a variant of this method for evaluating deterministic policies that have binary choice of treatment. DR’s utility function is formulated as:

$$\widehat{U}_{DR}(\pi) = \frac{1}{n} \sum_{i=1}^n \left[\frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)} \left(y(x_i, t_i) - \widehat{y}(x_i, t_i) \right) + \mathbb{E}_{t \sim \pi|x_i} [\widehat{y}(x_i, t)] \right] \quad (6)$$

where $\widehat{y}(x, t)$ is the regression fit (obtained from an OP method) that predicts the (counter)factual outcome for any given patient x and treatment $t \in T$.

2.4 Self-Normalized Estimator

In order to alleviate the high variance problem, the doubly robust estimator employs a regression fit to predict counterfactual outcomes to be used as *additive* terms in the utility function (see Eq. 6). Alternatively, Swaminathan and Joachims [15] take a *multiplicative* approach by proposing the Self-Normalized (SN) estimator, which is a stochastic variant of the method proposed by Hirano *et al.* [16] for evaluating deterministic policies with a binary choice of treatment. Self-Normalized (SN) estimator uses the fact that $\mathbb{E} \left[\sum_{i=1}^n \frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)} \right] = n$, which motivates replacing n with $\sum_{i=1}^n \frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)}$ in Eq. 2, leading to:

$$\widehat{U}_{SN}(\pi) = \sum_{i=1}^n \frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)} y(x_i, t_i) \bigg/ \sum_{i=1}^n \frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)} \quad (7)$$

The intuition is: since the main source of variance is the importance sampling weights appearing in the numerator of the utility function, having a similar factor in the denominator may cancel out some of the variability.

2.5 Counterfactual Risk Minimization

Swaminathan and Joachims [17] studied the variance of the IPS estimator with clipped weights under the Empirical Risk Minimization (ERM) principle (see Eq. 4) to prove the following generalization bound:

$$P \left[\exists \pi \in \Pi : R(\pi) > \widehat{R}^M(\pi) + \lambda \sqrt{\frac{\widehat{Var}(u(\cdot))}{n}} + C \right] \leq \gamma \quad (8)$$

where $R(\pi)$ is the true risk, $u(\cdot) = y(x, t) \min \left\{ M, \frac{\pi(t|x)}{\pi_0(t|x)} \right\}$, $\widehat{Var}(\cdot)$ is the estimated variance of $u(\cdot)$, $0 < \gamma < 1$ bounds the probability that this generalization bound fails, and λ and C are constants. This suggests adding the square-root term to the ERM objective function:

$$\pi^* = \arg \min_{\pi \in \Pi} \left\{ \widehat{R}^M(\pi) + \lambda \sqrt{\frac{\widehat{Var}(u(\pi))}{n}} \right\} \quad (9)$$

This addition to the objective function yields the Counterfactual Risk Minimization (CRM) principle [17], which is designed to penalize high empirical variance in weighted observed outcomes. The CRM principle can be used along with any utility function described in Sects. 2.2 to 2.4.

3 Evaluation Methodology

As described earlier, observational studies inherently contain partial information, as we only observe the outcome for the one treatment that was administered to each patient. When evaluating a new learned policy, however, due to the variance of the utility estimators, it is challenging to know if the new proposed policy is indeed better than the in-place policy. The best way to determine the effectiveness of a new policy is to actually deploy it on-site, record the factual outcomes for a reasonable period of time, and then analyze the results. However, it is neither ethical, nor allowed by the health-care community, to deploy a policy that has a chance of producing results that may reduce the patients’ quality of life. Therefore, we need to synthesize bandit datasets in such a way that their counterfactual outcomes are also known, merely for the purpose of evaluation. In the following two sections, we first review the existing evaluation methodology [18] and point out its shortcomings, and then explain our proposed evaluation methodology, which allows for a more comprehensive assessment of a proposed algorithm in terms of robustness to various degrees of selection bias.

3.1 The Existing Approach

Beygelzimer and Langford [18] proposed an approach that converts the training partition of a full-information binary multi-label supervised dataset¹ $\mathcal{D}^* = \{(x_i, t_i^*)\}_{i=1..n}$ with $t_i^* \in \{0, 1\}^k$ into a partial-information bandit dataset for training off-policy learning methods.² They view each label t_i^* as the best possible treatment for patient i – that is, the outcome value-function $y(x_i, t_i)$ is defined such that $y(x_i, t_i^*) > y(x_i, \neg t_i)$, where $\neg t_i$ is any of the treatments other than t_i^* , as this ensures that Eq. 1 holds, and so the optimal policy $\pi^*(t|x_i)$ will prefer t_i^* . One can convert this supervised dataset into a bandit dataset by sampling a set of new labels $t_i \sim h_0(t|x_i)$ for each x_i , where $h_0(\cdot)$ is the underlying mechanism that creates this observational study. This allows a single subject to appear many ($r < 2^k$) times in the dataset, each time associated with a different treatment.

In many applications, such as ad-placement, the underlying treatment assignment mechanism (*i.e.*, the deployed algorithm) is known [5]. To mimic the same situation, Swaminathan and Joachims [19] set $h_0(t|x)$ to be a logistic regression function, whose parameters are learned from a small portion (*e.g.*, 5%) of the supervised training set. This $h_0(\cdot)$ is then used to guide the sampling process of

¹ This is not one-hot encoding as there may be instances with multiple associated labels – *e.g.*, a news article concerning political initiatives on climate change.

² Note that the test set remains intact for evaluating the learned policy.

new labels t_i for each x_i , and log the propensities $\pi_0(t_i|x_i) = h_0(t_i|x_i)$. Finally, the outcome $y(x_i, t_i)$ is calculated as the Jaccard index between the supervised label t_i^* and the bandit label(s) t_i (s). This completes the procedure of generating a bandit dataset $\mathcal{D} = \{[x_i, t_i, y(x_i, t_i), \pi_0(t_i|x_i)]\}_{i=1..n}$.

There are several reasons why this evaluation framework is not appropriate for assessing off-policy learning methods for medical observational studies:

1. It is not clear how to map the concept of binary multi-label $\in \{0, 1\}^k$ to treatment, *e.g.*, letting each bit in a label vector to be 1 (resp. 0) refer to taking (resp. not taking) a certain medication, a multi-label target would mean a combination of several drugs. However, due to drug interactions, such combinations might neutralize the effect of the treatment or worse, be detrimental to the patient’s health. Therefore, unless there is a principled way to consider such interactions, a single class label seems more appropriate.
2. Using the Jaccard index to define outcomes implies assigning equal importance to various treatment options. However, this assumption does not hold in medicine since receiving the wrong treatment might be catastrophic for some cases while minor for others. Here, continuous measures such as *survival time* ($y \in \mathbb{R}^{\geq 0}$) that directly correspond to the consequences of the assigned treatment on the patient’s health status seem more appropriate.
3. Unlike applications such as ad-placement, where the underlying mechanism of action selection is known, it may not be fully understood in medical observational studies (*e.g.*, clinical pathways). In reality, we never have access to [even a small] subset of data with ground truth labels. Hence, the propensities $\pi_0(\cdot)$ have to be calculated directly from the bandit dataset, as opposed to readily deriving them from $h_0(\cdot)$ (*i.e.*, estimated from 5% of supervised data).

3.2 The Proposed Approach

This section discusses our proposed evaluation methodology and its advantages over the existing approach. In addition to overcoming the shortcomings of the existing approach, we want to address the following requirements: (i) design a bandit dataset that is as realistic as possible in terms of similarity to an actual medical observational study (Sect. 3.2); and (ii) include a procedure to generate many different observational studies from a single RCT dataset to allow for comprehensive evaluation of learning methods for contextual bandits (Sect. 3.2).

Designing a Bandit Dataset: We require that the designed bandit dataset be as similar as possible to a real medical observational dataset. Therefore, instead of converting a supervised dataset to a bandit dataset, we directly work with a real-world RCT dataset as the source and from it synthesize various observational studies with different degrees of sample selection bias.³ This makes sense because there is no selection bias in RCT datasets and therefore, one can often estimate

³ This means the X values are realistic. By contrast, we do not know whether the X values from a supervised dataset look like realistic [medical] observational studies.

the counterfactual outcomes reliably. In addition to the primary constraints (*e.g.*, $t \in \{0, 1\}$ and $y \in \mathbb{R}^{\geq 0}$), we want to preserve the statistical characteristics of the original (source) RCT dataset; characteristics such as:

- (i) Average Treatment Effect: $ATE = \frac{1}{N_1} \sum y(x_i, 1) - \frac{1}{N_0} \sum y(x_i, 0)$, where N_1 (resp., N_0) is the number of subjects assigned to $t = 1$ (resp. $t = 0$); and
- (ii) Coefficient of Determination: $R_t^2 = 1 - \frac{\sum [y(x_i, t_i) - f_t(x_i)]^2}{\sum [y(x_i, t_i) - \bar{y}]^2}$ for $t_i \in \{t_0, t_1, \dots\}$, where \bar{y} is the mean of y . Coefficient of Determination is calculated on each treatment arm separately and measures the amount of variance in the response variable that can be explained by the observed explanatory variables.⁴

Given a RCT dataset with two treatment arms (*i.e.*, $t \in \{0, 1\}$), we first fit two Gaussian Process (GP) [20] models $f_t(\cdot)$ to calculate an initial estimate of the counterfactual outcomes; one for each treatment arm. More concretely, $f_t(x)$ provides a mean $\mu_t(x)$ along with a standard deviation $\sigma_t(x)$ that indicates the confidence of estimation at any point x in the function domain. We can now calculate the counterfactual outcome for each subject. For example, the counterfactual outcome for a subject whose received treatment was $t = 1$ (with observed outcome $y(x_i, 1)$) is defined as: $\hat{y}(x_i, 0) = \mu_0(x_i) + k_0 \times \sigma_0(x_i)$, where k_0 is determined such that the average *personalized* treatment effect calculated on the N_1 subjects whose received treatment was $t = 1$ (*i.e.*, $\widehat{ATE}_1 = \frac{1}{N_1} \sum_{i \text{ s.t. } t_i=1} (y(x_i, 1) - \hat{y}(x_i, 0))$) matches the ATE calculated on the original RCT dataset. Solving for $\widehat{ATE}_1 = ATE$ and $\widehat{ATE}_0 = ATE$ yields:

$$k_t = \left(ATE - (2t - 1) \frac{1}{N_{-t}} \sum (\mu_t(x_i) - y(x_i, -t_i)) \right) / \frac{1}{N_t} \sum \sigma_t(x_i) \quad (10)$$

This procedure ensures that any synthetic RCT data generated by random re-assignment of treatments will have an \widehat{ATE} close to the original ATE .

We also want our synthetic datasets to match the R_t^2 measure on every treatment arm t . To do so, we first calculate \widehat{R}_t^2 for each treatment arm t , on all subjects, using either observed, or counterfactual outcomes as derived in the previous step. Then, if the \widehat{R}_t^2 was higher than the original R_t^2 value, we modify the counterfactual outcomes by adding noise to them as follows:

$$\hat{y}(x_i, t) += e_t \times \epsilon_i, \quad t \neq t_i \quad (11)$$

where e_t is the amplitude of the noise (tuned such that \widehat{R}_t^2 matches R_t^2) and $\epsilon \sim U(-0.5, 0.5)$. As $\mathbb{E}[\epsilon] = 0$, $\hat{y}(x_i, t)$'s expected increase is 0, and therefore we expect that \widehat{ATE} would not change.

With this complete set of outcomes (observed as well as counterfactual), we can determine the best treatment for each patient (*i.e.*, ground truth labels),

⁴ A low R^2 measure suggests that there must exist [some] unobserved confounder(s) that [significantly] contribute to the outcome.

following Eq. 1. It is also possible to synthesize any observational study (including RCT) by simply designing an appropriate $h_0(\cdot)$ function. Figures 2a and b respectively show the scatter plots of an original RCT dataset and a sample synthetic RCT generated from it, following the procedure described above. The next section explains how our proposed evaluation methodology can synthesize various observational studies, covering a wide range of selection bias.

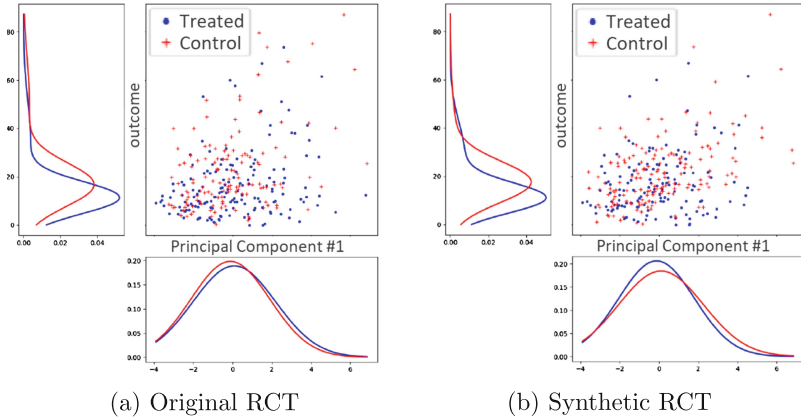


Fig. 2. The proposed method generates a synthetic RCT dataset (right) that is very similar to a real RCT dataset (left)

Various Generating Policies $h_0(\cdot)$: Unlike [15,17,19], our proposed evaluation methodology decouples the generating policy $h_0(\cdot)$ from the supervised dataset. This means we can easily design different $h_0(\cdot)$ policies with various degrees of selection bias and/or conservatism, which in turn enables us to study the behaviours/robustness of different learning algorithms under such various circumstances. In order to create a bandit dataset, our basis function for sampling labels (*i.e.*, treatments) is a sigmoid function:

$$\sigma(z) = \sigma_{\alpha,\beta}(z) = \frac{1}{1 + e^{-\alpha(z-\beta)}} \quad (12)$$

where $|z|$ is the amount of improvement in outcome for a patient in case she receives the best treatment; z is positive for $t^* = 1$ class and negative for $t^* = 0$ class. For instances in $t^* = 1$ class, the treatments t are then drawn according to $\sigma(z)$ via rejection sampling and for $t^* = 0$ class according to $1 - \sigma(z)$.

Parameter α in Eq. 12 controls the degree of selection bias. With $\alpha = 0$, $\sigma(z)$ is a uniform distribution, which results in synthesizing a RCT dataset. Increased α creates a more biased dataset such that at the limit of $\alpha = \infty$, $\sigma(z)$ becomes a step function at β . At $\beta = 0$, a larger α increases the chance that a sampled treatment t is equal to its respective ground truth label t^* . We can simulate the tendency towards prescribing a certain treatment more than the alternative(s)

(*i.e.*, conservatism) by modifying β . As such, a $\beta > 0$ would assign treatment 0 to more patients and treatment 1 to fewer ones, and vice versa for $\beta < 0$.

4 Experiments, Results, and Discussions

We experimented the proposed framework with the following two RCT datasets:

Acupuncture. RCT [21, 22] was designed to study the potential benefit of acupuncture (in addition to the standard care) for treatment of chronic headache disorders. It has 18 features, all measured prior to applying any treatment. There were two main outcomes: “severity score” and “headache frequency”, each measured at two points in time: “immediately after the treatment is completed” and “at one year follow-up”. Out of 401 participants, we use the 295 subjects with no missing values. Due to space limitation, we only report the performance results on one of the main outcomes (*i.e.*, severity score at one year follow-up) in the main text (the rest are closely similar). For this outcome, $ATE = -6.15$, $R_0^2 = 0.68$, and $R_1^2 = 0.33$, while our synthesized RCTs has $\widehat{ATE} = -6.17(0.76)$, $\widehat{R}_0^2 = 0.60(0.05)$, and $\widehat{R}_1^2 = 0.36(0.07)$.

Hypericum. RCT [23] was designed to assess the acute efficacy of a standardized extract of the herb St. John’s Wort in treatment of patients with major depression disorder. This study has three arms (placebo, hypericum, and an SSRI medication). The primary outcome measure is Hamilton Depression scale at the end of week 8. We compiled 278 features from assessment forms. In our experiments, we use the “hypericum” and “SSRI” as the binary treatment options (82 and 79 patients in each arm respectively). The original RCT has $ATE = -2.25$, $R_0^2 = 0.24$, and $R_1^2 = 0.00$. Our synthesized RCTs has $\widehat{ATE} = -2.65(0.97)$, $\widehat{R}_0^2 = 0.15(0.10)$, and $\widehat{R}_1^2 = 0.04(0.09)$.

The following methods are compared in terms of classification accuracy on the ground truth labels derived following the procedure described in Sect. 3.2. **Baseline:** predict the majority class. **Logger:** use the logging policy $\pi_0(\cdot)$ as the classifier. **Outcome Prediction:** find a regression fit with a simple linear least squares method with L2 regularization (OP), then use Eq. 1 to predict the best label (*i.e.*, treatment). **Inverse Propensity Scoring:** use the ERM objective function (IPS-ERM), as well as the CRM objective function (IPS-CRM) to learn a new policy $\pi(t|x)$ that acts as our classifier.⁵ **Doubly Robust:** use the same regression function as OP for the regression component with either ERM (DR-ERM) or CRM (DR-CRM) objective functions to obtain $\pi(t|x)$. **Self-Normalized:** use either ERM or CRM objective functions (SN-ERM and SN-CRM respectively) to obtain $\pi(t|x)$.

Figure 3 summarizes the performance results; showing the effect of changing α at $\beta = 0$ in Figs. 3a and c, and changing β at $\alpha = 0.05$ in Figs. 3b and d. Each

⁵ Our implementation of IPS (and SN below) is obtained from Policy Optimizer for Exponential Models (POEM [19]). We extended POEM substantially to include a way to deal with the missing components (*i.e.*, OP and DR), as well as implementation of the proposed evaluation methodology.

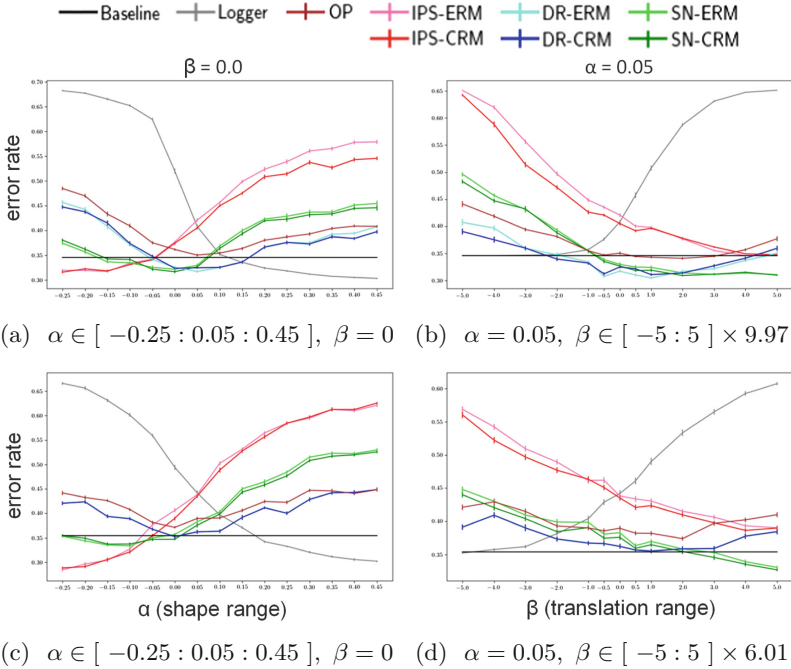


Fig. 3. Mean and $\frac{1}{10}$ standard deviation of the classification error rates on the “Acupuncture” (top) and “Hypericum” (bottom) datasets; best viewed in color (Color figure online)

point on the plots represents the mean classification error rate across 25 runs and its respective error bar indicates a fraction (10%) of the standard deviation of the error rates (in order to maintain the plots’ clarity). Also note that the Baseline accuracy for neither of the datasets is 1.0, meaning that not all patients benefit from receiving “acupuncture” or “SSRI”; indeed, for some, “no acupuncture” or “St. John’s Wort” achieves a better outcome, *i.e.*, personalized medicine.

Effects of changing α . As α increases, it is trivial that the Logger’s accuracy would improve since a higher α produces a bandit dataset with a higher tendency towards sampling the ground truth treatments more often. Moreover, OP’s prediction of (counter)factual outcomes is not accurate as α moves away from 0, resulting in a bad performance for DR as well. IPS’s performance is also correlated with α and it tends to do worse as α increases, as opposed to that of Logger’s. SN tends to perform worse as $|\alpha|$ increases since this imposes π_0 to be small in parts of its domain which, due to weight clipping, results in $\mathbb{E} \left[\sum_{i=1}^n \frac{\pi(t_i|x_i)}{\pi_0(t_i|x_i)} \right] \neq n$. This breaks the fundamental idea of SN (see Sect. 2.4). This suggests that SN is only useful for datasets that are close to RCT.

Effects of changing β . We know that the effect of changing β varies relative to the degree of class imbalance in a dataset. In ours, since the majority class is labeled as “1”, a $\beta > 0$ would result in assigning fewer samples with label $t = 1$. This, in turn, would generate a bandit dataset that is more exploratory (which SN seems to prefer); but, on the other hand, is far from the true underlying label distribution (which is not desirable for OP and DR).

ERM versus CRM principle. We found that the CRM principle often improves SN’s performance (not statistically significant though). However, it does not do so for IPS and DR. In general, our results indicate that if a reliable OP method is available, then DR is the most effective and robust method for various α and β values. However, we should bear in mind that most OP (and as a result DR) methods require more processing power than IPS and SN. Therefore, we face a trade-off between a quick response versus a more accurate one.

5 Future Directions and Conclusions

The proposed evaluation methodology can be extended in three directions:

1. Increase the pool size of possible treatments from $|T| = 2$ (*i.e.*, $t \in \{0, 1\}$). This could cover combining several medications or other medical interventions.
2. Explore ways to extend this evaluation methodology for sequential observational studies. This is not trivial since the space of all possible decisions grows exponentially as we progress through the course of treatment.
3. We can create observational datasets that contain censored samples, *i.e.*, only a *lower bound* of the survival time of some patients is available. Such datasets can then be used to develop and evaluate survival analysis/prediction based methods that can handle sample selection bias.

In this paper, we proposed a novel evaluation methodology for assessing off-policy learning methods in contextual bandits. Unlike the existing methodology (*cf.*, [18]), our approach allows for a comprehensive assessment of the learning methods in terms of performance and robustness with respect to various degrees of sample selection bias. Moreover, it does not require the underlying mechanism for data generation to be known, and it better matches medical applications as it allows the outcomes to be more realistic ($y \in \mathbb{R}^{\geq 0}$). Using the proposed evaluation methodology, we assessed several prominent off-policy learning methods in contextual bandits – namely, outcome prediction, Inverse Propensity Scoring, Doubly Robust [13], Self-Normalized [15], and Counterfactual Risk Minimization principle [17] – on observational datasets synthesized using two RCT datasets. Our analyses identify the conditions under which a certain off-policy learning method performs best (*e.g.*, SN is preferable for a close-to-RCT dataset). Such analysis was not possible with [18]’s evaluation methodology as it has no means to generate such diverse observational datasets in terms of selection bias. Thus, we believe the proposed evaluation methodology should become a standard way

for comprehensive assessment of new off-policy learning methods in contextual bandits, especially in costly applications such as precision medicine where deploying a bad policy can have devastating effects.

References

1. Pearl, J.: Causality. Cambridge University Press, New York (2009)
2. Imbens, G.W., Rubin, D.B.: Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction. Cambridge University Press, New York (2015)
3. Wang, C.C., Kulkarni, S.R., Poor, H.V.: Bandit problems with side observations. *IEEE Trans. Autom. Control* **50**(3), 338–355 (2005)
4. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, vol. 1. MIT Press, Cambridge (1998)
5. Bottou, L., Peters, J., Candela, J.Q., Charles, D.X., Chikering, M., Portugaly, E., Ray, D., Simard, P.Y., Snelson, E.: Counterfactual reasoning and learning systems: the example of computational advertising. *JMLR* **14**(1), 3207–3260 (2013)
6. Li, L., Chen, S., Kleban, J., Gupta, A.: Counterfactual estimation and optimization of click metrics in search engines: a case study. In: Proceedings of the 24th International Conference on World Wide Web. ACM (2015)
7. Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., Joachims, T.: Recommendations as treatments: debiasing learning and evaluation. In: Proceedings of the 33rd International Conference on Machine Learning, vol. 48 (2016)
8. Liu, Y.E., Mandel, T., Brunskill, E., Popovic, Z.: Trading off scientific knowledge and user learning with multi-armed bandits. In: Educational Data Mining (2014)
9. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web. ACM (2010)
10. Li, L., Chu, W., Langford, J., Wang, X.: Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In: Proceedings of the 4th International Conference on Web Search and Data Mining, Hong Kong (2011)
11. Horvitz, D.G., Thompson, D.J.: A generalization of sampling without replacement from a finite universe. *J. Am. Stat. Assoc.* **47**(260), 663–685 (1952)
12. Rosenbaum, P.R., Rubin, D.B.: The central role of the propensity score in observational studies for causal effects. *Biometrika* **70**, 41–55 (1983)
13. Dudík, M., Langford, J., Li, L.: Doubly robust policy evaluation and learning. In: International Conference on Machine Learning (2011)
14. Robins, J.M., Rotnitzky, A., Zhao, L.P.: Estimation of regression coefficients when some regressors are not always observed. *J. Am. Stat. Assoc.* **89**(427), 846–866 (1994)
15. Swaminathan, A., Joachims, T.: The self-normalized estimator for counterfactual learning. In: Advances in Neural Information Processing Systems (2015)
16. Hirano, K., Imbens, G.W., Ridder, G.: Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica* **71**(4), 1161–1189 (2003)
17. Swaminathan, A., Joachims, T.: Counterfactual risk minimization: learning from logged bandit feedback. In: International Conference on Machine Learning (2015)
18. Beygelzimer, A., Langford, J.: The offset tree for learning with partial labels. In: Proceedings of the 15th ACM SIGKDD. ACM (2009)
19. Swaminathan, A., Joachims, T.: Batch learning from logged bandit feedback through counterfactual risk minimization. *JMLR* **16**, 1731–1755 (2015)

20. Rasmussen, C.E., Williams, C.K.: Gaussian Processes for Machine Learning, vol. 1. MIT Press, Cambridge (2006)
21. Vickers, A.J., Rees, R.W., Zollman, C.E., McCarney, R., Smith, C.M., Ellis, N., Fisher, P., Van Haselen, R.: Acupuncture for chronic headache in primary care: large, pragmatic, randomised trial. *BMJ* **328**(7442), 744 (2004)
22. Vickers, A.J.: Whose data set is it anyway? Sharing raw data from randomized trials. *Trials* **7**(1), 15 (2006)
23. Hypericum Depression Trial Study Group, et al.: Effect of Hypericum perforatum (St. John's Wort) in major depressive disorder: a randomized controlled trial. *JAMA* **287**(14), 1807–1814 (2002)



Synthesizing Controllers: On the Correspondence Between LTL Synthesis and Non-deterministic Planning

Alberto Camacho¹(✉), Jorge A. Baier², Christian Muise³,
and Sheila A. McIlraith¹

¹ Department of Computer Science, University of Toronto, Toronto, Canada
acamacho@cs.toronto.edu

² Chilean Center for Semantic Web Research,
Pontificia Universidad Católica de Chile, Santiago, Chile

³ IBM Research, Cambridge Research Center, Cambridge, USA

Abstract. Linear Temporal Logic (LTL) synthesis can be understood as the problem of building a controller that defines a winning strategy, for a two-player game against the environment, where the objective is to satisfy a given LTL formula. It is an important problem with applications in software synthesis, including controller synthesis. In this paper we establish the correspondence between LTL synthesis and fully observable non-deterministic (FOND) planning. We study LTL interpreted over both finite and infinite traces. We also provide the first explicit compilation that translates an LTL synthesis problem to a FOND problem. Experiments with state-of-the-art LTL FOND and synthesis solvers show automated planning to be a viable and effective tool for highly structured LTL synthesis problems.

Keywords: Automated planning · Controller synthesis · LTL
Non-deterministic planning

1 Introduction

The problem of synthesizing software, including controllers, from logical specification is a fundamental problem in AI and computer science more generally. Church's synthesis problem was first posed by Church in 1957 in the context of synthesizing digital circuits from a logical specification [1] and is considered one of the most challenging problems in reactive systems [2]. Two common approaches to solving the problem have emerged: reducing the problem to the emptiness problem of tree automata, and characterizing the problem as a two-player game.

In 1989, Pnueli and Rosner examined the problem of reactive synthesis using Linear Temporal Logic (LTL) [3] as the specification language (henceforth "LTL synthesis") viewing the problem as a two-player game, and showing that this

problem was 2EXPTIME-complete [4]. This discouraging result has been mitigated by the identification of several restricted classes of LTL for which synthesis is less complex. For example, if the LTL specification is restricted to the class of so-called *Generalized Reactivity(1)* (GR1) formulae, an N^3 -time algorithm exists [2]. Today, a number of synthesis tools exist with varying effectiveness (e.g., Acacia+ [5], Lily [6]).

Recent work has explored various connections between automated planning and synthesis (e.g., [7–12]) but has not provided a full mapping between the two problems, nor have the practical implications of such a mapping been explored from an automated planning perspective. In this paper we investigate the relationship between (LTL) synthesis and automated planning, and in particular (LTL) Fully Observable Non-Deterministic (FOND) planning. We do so by leveraging a correspondence between FOND and 2-player games. This work is inspired by significant recent advances in the computational efficiency of FOND planning that have produced FOND planners that scale well in many domains (e.g., myND [13] and PRP [14]). Our insights are that just as SAT can be (and has been) used as a black-box solver for a myriad of problems that can be reduced to SAT, so too can FOND be used as a black-box solver for suitable problems. Establishing the connection between FOND and 2-player games not only provides a connection to LTL synthesis – the primary subject of this exploration – it also provides the key to leveraging FOND for other problems.

In Sect. 3 we establish the correspondence between LTL synthesis and strong solutions to FOND planning. In Sect. 4 we provide the first automatic translation of a realizability problem into a planning problem, described in the Planning Domain Definition Language (PDDL), the de facto standard input language for automated planners. Experiments with state-of-the-art LTL synthesis and FOND solvers illustrate that the choice of formalism and solver can have a dramatic impact. Indeed, planning-based approaches excel if the problem is highly structured and the uncertainty largely restricted, as is the case for synthesis problems associated with engineered physical devices.

2 Preliminaries

FOND: A FOND planning problem is a tuple $\langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$, where \mathcal{F} is a set of *fluents*; $\mathcal{I} \subseteq \mathcal{F}$ characterizes what holds initially; $\mathcal{G} \subseteq \mathcal{F}$ characterizes the goal condition; and \mathcal{A} is the set of actions. The set of literals of \mathcal{F} is $Lits(\mathcal{F}) = \mathcal{F} \cup \{\neg f \mid f \in \mathcal{F}\}$. Each action $a \in \mathcal{A}$ is associated with $\langle Pre_a, Eff_a \rangle$, where $Pre_a \subseteq Lits(\mathcal{F})$ is the precondition and Eff_a is a set of outcomes of a . We sometimes write $oneof(Eff_a)$ to emphasize that Eff_a is non-deterministic. Each outcome $e \in Eff_a$ is a set of conditional effects of the form $(C \rightarrow \ell)$, where $C \subseteq Lits(\mathcal{F})$ and $\ell \in Lits(\mathcal{F})$. Given a planning state $s \subseteq \mathcal{F}$ and a fluent $f \in \mathcal{F}$, we say that s satisfies f , denoted $s \models f$, iff $f \in s$. In addition $s \models \neg f$ if $f \notin s$, and $s \models L$ for a set of literals L , if $s \models \ell$ for every $\ell \in L$. Action a is *applicable* in state s if $s \models Pre_a$. We say s' is a *result of applying a in s* iff, for one outcome $e \in Eff_a$, s' is equal to $s \setminus \{f \mid (C \rightarrow \neg f) \in e, s \models C\} \cup \{f \mid (C \rightarrow f) \in e, s \models C\}$.

A *policy* p , is a partial function from states to actions such that if $p(s) = a$, then a is applicable in s . An *execution* π of a policy p in state s is a sequence $s_0, a_0, s_1, a_1, \dots$ (finite or infinite), where $s_0 = s$, and such that every state-action-state substring s, a, s' are such that $p(s) = a$ and s' is a result of applying a in s . Finite executions ending in a state s are such that $p(s)$ is undefined.

A finite execution π *achieves* a set of literals L if its ending state s is such that $s \models L$. An infinite execution π *achieves* a set of literals L if there exists a state s that appears infinitely often in π and that is such that $s \models L$. An infinite execution σ is *fair* iff whenever s, a occurs infinitely often within σ , then so does s, a, s' , for every s' that is a result of applying a in s [15]. Note this implies that finite executions are fair. A policy p is a *strong plan* (resp. *strong-cyclic plan*) for a FOND problem $P = \langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$, iff every execution (resp. fair execution) of p over \mathcal{I} satisfies the goal \mathcal{G} .

Linear Temporal Logic: Linear Temporal Logic (LTL) is a propositional logic extended with temporal modal operators *next* (\circ) and *until* (\cup). The set of LTL formulae over a set of propositions \mathcal{P} is defined inductively as follows. p is a formula if $p \in \mathcal{P}$ or the constant \top . If φ_1 and φ_2 are LTL formulas, then so are $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\circ\varphi_1$ and $\varphi_1 \cup \varphi_2$. Let $\sigma = s_0, s_1, \dots$ be an infinite sequence of subsets of \mathcal{P} , and φ be an LTL formula. Then σ *satisfies* φ , denoted as $\sigma \models \varphi$ iff $\sigma, 0 \models \varphi$, where:

$$\begin{aligned} \sigma, i \models p, \text{ for each } p \in \mathcal{P} \cup \{\top\} &\text{ iff } s_i \models p & \sigma, i \models \neg\varphi &\text{ iff } \sigma, i \models \varphi \text{ does not hold} \\ \sigma, i \models \varphi_1 \wedge \varphi_2 &\text{ iff } \sigma, i \models \varphi_1 \text{ and } \sigma, i \models \varphi_2 & \sigma, i \models \circ\varphi &\text{ iff } \sigma, (i+1) \models \varphi \\ \sigma, i \models \varphi_1 \cup \varphi_2 &\text{ iff there exists a } j \geq i \text{ such that} & & \\ & \sigma, j \models \varphi_2, \text{ and } \sigma, k \models \varphi_1, \text{ for each } k \in \{i, i+1, \dots, j-1\} & & \end{aligned}$$

Intuitively, the *next* operator tells what needs to hold in the next time step, and the *until* operator tells what needs to hold until something else holds. The modal operators *eventually* (\diamond) and *always* (\square) are defined by $\diamond\varphi \equiv \top \cup \varphi$, $\square\varphi \equiv \neg\diamond\neg\varphi$. Additional constants and operators are defined by following conventional rules as follows $\perp \equiv \neg\top$, $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$.

LTL over Finite Traces: LTL_f is a variant of LTL interpreted over finite traces [16]. Such finite variants have been applied to problems in verification, model checking and planning. LTL_f and LTL share the same syntax, but their interpretations differ. For example, the LTL_f formula $\diamond\neg\circ\top$ is true in a finite trace, whereas the same formula in LTL evaluates false on infinite traces. Similarly, *weak next* must often replace *next* to avoid unintended interpretations of LTL over finite traces. (See [16] for details.)

Automata: There is a well-established correspondence between LTL and automata. A *Non-deterministic Büchi Automaton* (NBA) is a tuple $M = (Q, \Sigma, \delta, q_0, Q_{Fin})$, where Q is the set of automaton states, Σ is the alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, q_0 is the initial state, and $Q_{Fin} \subseteq Q$ is the set of accepting states. The automaton is *deterministic* (DBA) when for each $q \in Q$, and $s \in \Sigma$, there exists a unique $q' \in Q$ such that $(q, s, q') \in \delta$. A *run* of

M on an infinite word $\sigma = s_0, s_1, \dots$ of elements in Σ is a sequence of automaton states, starting in q_0 , such that $(q_i, s_i, q_{i+1}) \in \delta$ for all $i \geq 0$. A run is accepting if it visits an infinite number of accepting states. Finally, we say that M accepts σ if there is an accepting run of M on σ . *Non-deterministic Finite-state Automata* (NFAs) differ from NBAs in that the acceptance condition is defined on *finite* words: a word $\sigma = s_0, s_1, \dots, s_m$ is accepting if $q_{m+1} \in Q_{Fin}$. Finally, *Deterministic Finite-state Automata* are NFAs where the transition relation is deterministic.

Given an LTL formula φ , it is possible to construct an NBA A_φ that accepts σ iff $\sigma \models \varphi$. The construction is worst-case exponential in the size of φ [17]. It is not always possible to construct a DBA, and the construction is double exponential. Similar results hold for LTL_f: it is always possible to construct an NFA (resp. DFA) that accepts σ iff $\sigma \models \varphi$, and the construction is worst-case exponential (resp. double exponential) [18].

LTL FOND: Recently [11] extended FOND with LTL goals. An LTL FOND problem is a tuple $\langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$, where \mathcal{G} is an LTL or LTL_f formula over \mathcal{F} , and $\mathcal{F}, \mathcal{I}, \mathcal{A}$ are defined as in FOND planning. In short, LTL FOND executions are defined just like in FOND, and a policy is a strong-cyclic (resp. strong) plan for problem P if each fair (resp. unrestricted) execution π results in a sequence of states σ such that $\sigma \models \mathcal{G}$.

LTL Synthesis: Intuitively, the LTL synthesis problem [4] describes a two-player game between a controller and the environment. The game consists of an infinite sequence of turns. In each turn the environment chooses an action, and the controller then chooses another. Each action corresponds to setting the values of some variables. The controller has a winning strategy if, no matter how the environment plays, the sequences of states generated satisfy a given LTL formula φ . Formally, a synthesis problem is a tuple $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$, where $\mathcal{X} = \{x_1, \dots, x_n\}$, the environment variables, and $\mathcal{Y} = \{y_1, \dots, y_m\}$, the controller variables, are disjoint sets. An LTL formula over $\mathcal{X} \cup \mathcal{Y}$ is *realizable* if there exists a function $f : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$ such that for every infinite sequence of subsets of \mathcal{X} , $X_1 X_2 \dots$, it holds that $\pi = (X_1 \cup f(X_1)), (X_2 \cup f(X_1 X_2)) \dots$ is such that $\pi \models \varphi$. Intuitively, no matter what the choice of the environment is, which is given by the sequence $X_1 X_2 \dots$, the controller has a strategy, given by f , that ensures formula φ is satisfied in the resulting game. The *synthesis* problem corresponds to computing function f . Synthesis has also been studied over *finite* sequences of turns using LTL_f specifications (e.g. [10]). In the rest of the paper, we write LTL synthesis to also refer to LTL_f synthesis, and make the distinction explicit only when necessary.

3 Relationship Between FOND and Synthesis

Both LTL synthesis and FOND are related to two-player games: in both problems an agent (or controller) seeks a solution that achieves a condition regardless of the choices made by the environment. There are, however, two important differences.

First, in LTL synthesis the controller reacts to the environment; in other words, the environment “plays first”, while the controller “plays second”.¹ In FOND, the play sequence is inverted since the environment decides the outcome of an action, which is in turn defined by the agent (controller). Second, state-of-the-art FOND solvers find strong-cyclic solutions predicated on an assumption of fairness in the environment, which is not an assumption inherent to LTL synthesis. Thus a correct mapping between FOND and Synthesis must handle fairness correctly.

Previous work has explored the relation between FOND and synthesis. [9] show how to translate FOND as a reactive synthesis problem by expressing fairness constraints as temporal logic formulae. [16] sketches a mapping from FOND to LTL synthesis, in which the effects of actions are specified using LTL. This approach, however, does not dive into the details of the inverted turns. Neither do the works by [7, 19], which show a correspondence between two-player game structures and FOND planning. In the rest of the section we provide an explicit mapping between LTL FOND and LTL synthesis. Efficiency is the focus of the next section.

To establish a correspondence between LTL synthesis and LTL FOND, we address the inverted turns by considering the negation of realizability. Observe that an instance $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ is not realizable iff there exists a sequence $X_1 X_2 X_3 \dots$ of subsets of \mathcal{X} such that for every function $f : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$:

$$X_1 \cup f(X_1), X_2 \cup f(X_1 X_2), X_3 \cup f(X_1 X_2 X_3) \dots \models \neg \varphi$$

Note that what comes after the “iff” maps directly into an instance of LTL FOND: we define the problem $P_\varphi = \langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$ such that fluents are the union of all variables (i.e., $\mathcal{F} = \mathcal{X} \cup \mathcal{Y}$), and the set of actions is the set of subsets of \mathcal{X} (i.e., $\mathcal{A} = \{a_x \mid x \subseteq \mathcal{X}\}$). Intuitively action a_x is always executable (has empty preconditions) and deterministically sets to true the variables in x and to false the variables in $\mathcal{X} \setminus x$. In addition, it non-deterministically sets the values of variables in \mathcal{Y} to every possible combination. Formally, $\text{Eff}_{a_x} = \{e_{x,y} \mid y \subseteq \mathcal{Y}\}$, where each $e_{x,y} = \{f \mid f \in x \cup y\} \cup \{\neg f \mid f \in (\mathcal{X} \cup \mathcal{Y}) \setminus (x \cup y)\}$. Finally, we set $\mathcal{I} = \{\}$ and $\mathcal{G} = \circ \neg \varphi$.

A more involved argument follows for LTL_f synthesis. In this case, an instance $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ is not realizable iff for every finite m there exists a sequence $X_1 X_2 \dots X_m$ of subsets of \mathcal{X} such that, for every function $f : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$:

$$X_1 \cup f(X_1), \dots, (X_1 \dots X_m) \cup f(X_1 \dots X_m) \models \neg \varphi$$

What follows after the “iff” cannot be directly mapped into an instance of LTL FOND, because the formula above has to hold for all m . We can mitigate for this by adding a new variable to \mathcal{P}_φ , y_{ok} , that acts like any other variable in \mathcal{Y} . The goal of \mathcal{P}_φ is the LTL_f formula $\mathcal{G} = \circ(\neg \varphi \wedge \diamond(y_{ok} \wedge \neg \circ \top)) \vee \diamond(\neg y_{ok} \wedge \neg \circ \top)$.

Theorem 1. *LTL synthesis problem $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ is realizable iff P_φ has no strong plan.*

¹ The problem with inverted turns, where the agent “plays first”, has also been studied (e.g. [5]).

In the other direction, let $P = \langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$ be an LTL FOND problem. We now construct a synthesis problem $\langle \mathcal{X}_P, \mathcal{Y}_P, \varphi_P \rangle$ following well-known encodings of planning as SAT [20]; we use LTL to establish a connection between a state and its successors, instead of different variables, and we consider explicitly that actions have a number of outcomes. The specification φ_P is: $\varphi_P := \varphi_{init} \rightarrow (\varphi_{env} \rightarrow (\varphi_{agt} \wedge \varphi_g))$. Intuitively, φ_{init} models the initial state \mathcal{I} , φ_{env} and φ_{agt} model the dynamics in \mathcal{P} , and φ_g is the LTL goal formula \mathcal{G} .

For each action in $a \in \mathcal{A}$, we create a variable $a \in \mathcal{X}_P$. Each fluent $f \in \mathcal{F}$ is also a variable in \mathcal{X}_P . Variables in \mathcal{Y}_P are used to choose one of the non-deterministic outcomes of each action; this way if the action with the largest number of outcomes has n outcomes, we create $\lceil \log n \rceil$ variables, whose objective is to “choose” the outcome for an action. To model the preconditions of the action, we conjoin in φ_{env} , for each action a the formula $\Box(a \rightarrow \bigwedge_{\ell \in Pre_a} \ell)$. We express the fact that only one action can execute at a time by conjoining to φ_{env} the formulae $\Box \bigvee_{a \in \mathcal{A}} a$, and $\Box(a \rightarrow \neg a')$, for each $a' \in \mathcal{A}$ different from a . To model the fact that the environment selects the outcome being performed, for each action outcome e we create a variable a_e in \mathcal{X}_P . For each action $a \in \mathcal{A}$ and outcome $e \in Eff_a$, φ_{agt} has formulae of the form $\Box(a \wedge \chi_{a,e} \rightarrow a_e)$, where $\chi_{a,e}$ is a formula over \mathcal{Y}_P , which intuitively “selects” outcome e for action a . For space, we do not go into the details of how to encode $\chi_{a,e}$. However, these formulae have the following property: for any action a , given an assignment for \mathcal{Y}_P variables there is exactly one $e \in Eff_a$ for which $\chi_{a,e}$ becomes true. This models the fact that the \mathcal{Y}_P variables are used to select the outcomes.

Finally, we now conjoin to φ_{env} formulae to express the dynamics of the domain. Specifically we add successor-state-axiom-like expressions [21] of the form:

$$\Box(\circ f \equiv (\phi_f^+ \vee (f \wedge \neg \phi_f^-))), \quad \text{for each } f \in \mathcal{F}$$

where ϕ_f^+ is a formula that encodes the conditions under which f becomes true after an outcome has occurred, and where ϕ_f^- encodes the conditions under which f becomes false in the next state. Both of these formulae can be computed from Eff_a [21], and have fluents a_e for $e \in Eff_a$. Finally, φ_{init} is the conjunction of the fluents in the initial state \mathcal{I} , and φ_g is the goal formula, \mathcal{G} . When the goal of \mathcal{P} is an LTL_f formula, the construction conjoins $\circ\top$ to the successor state axioms in φ_{env} .

Now, it is not hard to see that there exists a strong solution to the LTL problem P iff there exists a (finite for LTL_f goals, infinite for LTL) sequence of settings of the \mathcal{X}_P variables, such that for every sequence of settings of the \mathcal{Y} variables (i.e., for every function $f : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$), it holds that $(X_1 \cup Y_1), (X_2 \cup Y_2), (X_3 \cup Y_3), \dots \models \varphi_P$.

Theorem 2. *LTL FOND problem $P = \langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$ has a strong plan if and only if $\langle \mathcal{X}_P, \mathcal{Y}_P, \neg \varphi_P \rangle$ is not realizable.*

4 Approach

In Sect. 3 we established the correspondence between existence of solutions to LTL synthesis, and existence of strong solutions to LTL FOND planning. In this section we introduce the first translation from LTL synthesis into FOND planning (and by inclusion, into LTL FOND), and a translation for LTL_f specifications.

Our approach to solve an LTL synthesis problem $\mathcal{P} = \langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ as FOND consists of three stages. First, we pre-process \mathcal{P} . Second, we compile it into a standard FOND problem \mathcal{P}' . Finally, we solve \mathcal{P}' with a strong-cyclic planner. Extracting a strategy for \mathcal{P} from a solution to \mathcal{P}' is straightforward, and we omit the details for lack of space.

Automaton Transformation: In a pre-processing stage, we simplify the specification by removing from \mathcal{X} and \mathcal{Y} those variables that do not appear in φ . Then, we transform φ into an automaton, $A_\varphi = (Q, \Sigma, \delta, q_0, Q_{Fin})$, that can be a DBA when the LTL formula is interpreted over infinite traces, or an NFA (or DFA, by inclusion) when the specification is an LTL_f formula. In addition to DBAs, our algorithm can seamlessly handle NBAs at the cost of losing its completeness guarantee. NBAs are a good alternative to DBAs as they are usually more compact, and only a subset of LTL formulae can be transformed into DBAs. The transition relation δ in A_φ implicitly defines the conditions under which the automaton in state q is allowed to transition to state q' . These conditions are known as *guards*. Formally, $\text{guard}(q, q') = \bigvee_{(q,s,q') \in \delta} s$. In our case, elements of the alphabet Σ are conjunctions of boolean variables, that allow for guard formulae to be described in a compact symbolic form. In what follows, we assume guard formulae $\text{guard}(q, q') = \bigvee_m c_m$ are given in DNF, where each clause c_m is a conjunction of boolean state variables. We denote as δ^* the set of tuples $T_m = (q, c_m, q')$ for each pair (q, q') with $\text{guard}(q, q') \neq \perp$, and for each clause c_m in $\text{guard}(q, q')$. For convenience, we write $\text{guard}(T_m) = c_m$, and refer to elements of δ^* as *transitions*. Wherever convenient, we drop the subindex of transitions and simply write T .

In the second stage, we compile $\mathcal{P} = \langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ with automaton A_φ into a parameterized FOND problem $\mathcal{P}'(\mathcal{X}, \mathcal{Y}, A_\varphi, H) = \langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$ that integrates the dynamics of A_φ with a two-player game between the environment and the agent. Before introducing the technical details of the compilation, we give an overview. The compilation simulates automaton states by means of fluents q , one for each automaton state with the same name. Planning states s have the following property: *an automaton fluent q is true in s iff for some σ , a run σ of A_φ finishes in q* . Notably, the input word σ can be obtained directly from the state-action plan that leads the initial state to s in the search tree. When the input to the algorithm is a non-deterministic automaton (NBA or NFA), planning states can simultaneously capture multiple runs of the automaton in parallel by simultaneously having multiple q fluents set to true.

The acceptance condition behaves differently for Büchi and non-Büchi automata, and this is reflected in our compilation. For Büchi automata, the planning process expands a graph that simulates the environment and agent

moves, and the search for solutions becomes a search for strong cyclic components that hit an accepting state infinitely often. The latter property is checked by means of *tokenized* fluents q^t , one for each q . Intuitively, the truth of $q \wedge q^t$ in state s indicates a commitment to progress runs finishing in q into runs that reach an accepting state. Conversely, $s \models q \wedge \neg q^t$ represents that such a commitment has been accomplished. The role of the parameter H is twofold: it bounds the horizon in the search for cycles, and it allows the use of strong-cyclic solvers to find solutions to a problem whose non-determinism has unrestricted fairness.

The compilation runs in two sequentially alternating modes simulating each two-player turn. The *environment mode* simulates the environment moves, which are non-deterministic and uncontrollable to the agent. In *automaton mode*, the agent moves are simulated and the automaton state fluents are synchronized according to valid transitions in δ^* . Auxiliary fluents q^s and $q^{s,t}$ are used to store the value of automaton state fluents q and q^t prior to synchronization, so that more than one transition can be simulated in the case of non-deterministic automata compilations. When an accepting state q is recognized, the agent can set the token fluents q^t to commit to progress the runs that finish in q into a run that hits another accepting state.

The dynamics of the compilation are similar for non-Büchi automata. The exception is that accepting runs are recognized whenever an accepting automaton state fluent is reached, and there is no need to commit to reaching another accepting state. Consequently, tokenized fluents q^t and $q^{s,t}$ are not needed but have been kept, for generality, in the algorithm below.

The sets of fluents (\mathcal{F}) and actions (\mathcal{A}) of the problem are listed below. In what follows, we describe the technical details of the compilation.

$$\begin{aligned} \mathcal{F} &= \{q, q^s, q^t, q^{s,t} \mid q \in Q\} \cup \{goal\} \cup \{next(h+1, h)\}_{0 \leq h < H} \cup \{turn_k\}_{1 \leq k \leq |\mathcal{X}|} \\ &\cup \{at_horizon(h)\}_{0 \leq h \leq H} \cup \{env_mode, aut_mode, can_switch, can_accept\} \\ &\cup \{v_x, v_{\neg x}\}_{x \in \mathcal{X}} \cup \{v_y, v_{\neg y}\}_{y \in \mathcal{Y}} \\ \mathcal{A} &= \{move_k\}_{1 \leq k \leq |\mathcal{X}|} \cup \{trans_T\}_{T \in \delta^*} \cup \{switch2aut, switch2env, accept\} \end{aligned}$$

Environment Mode: In the environment mode, the dynamics of the problem simulates the move of the environment. As this move is not controllable by the agent, it can be simulated with a non-deterministic action that has $2^{|\mathcal{X}|}$ effects, each simulating an assignment to variables in \mathcal{X} . Fluents v_l simulate the truth value of variables in $\mathcal{X} \cup \mathcal{Y}$. More precisely, v_x (resp. $v_{\neg x}$) indicates that $x \in \mathcal{X}$ is made true (resp. false), and similarly for $y \in \mathcal{Y}$. In order to reduce the explosion in non-deterministic action effects, we simulate the environment's move with a cascade of non-deterministic actions $move_k$, each one setting (v_{xk}) or unsetting ($v_{\neg xk}$) the value of a variable x_k in \mathcal{X} ,

$$\begin{aligned} Pre_{move_k} &= \{env_mode, turn_k\} \\ Eff_{move_k} &= oneof(\{v_{xk}, \neg v_{\neg xk}\}, \{\neg v_{xk}, v_{\neg xk}\}) \cup \Psi_k \end{aligned}$$

where $\Psi_k = \{\text{turn}_{k+1}, \neg\text{turn}_k\}$ if $k < |\mathcal{X}|$, and $\Psi_k = \{\text{can_switch}, \neg\text{turn}_k\}$ if $k = |\mathcal{X}|$. After the environment's move has been simulated, the *switch2aut* action switches the dynamics to automaton mode, and the automaton configuration (represented by fluents of the form q and q^t) is *frozen* into copies q^s and $q^{s,t}$. Special predicates $\text{at_horizon}(h)$ capture the number of turns from the last recognized accepting state in the plan. If $h < H$, the horizon value is incremented by one.

Automaton Mode: The automaton mode simulates the assignment to variables in \mathcal{Y} and the automaton state transitions. Whereas the update in the automaton configuration is usually understood as a *response* to the observation to variables in $\mathcal{X} \cup \mathcal{Y}$, the dynamics of the encoding take a different perspective: the agent can decide which automaton transitions to perform, and then set the variables in \mathcal{Y} so that the transition guards are satisfied. Such transitions are simulated by means of *trans_T* actions, one for each $T = (q_i, \text{guard}(T), q_j) \in \delta^*$:

$$\begin{aligned} \text{Pre}_{\text{trans}_T} &= \{\text{aut_mode}, q_i^s, \neg q_j\} \cup \{\neg v_{-l}\}_{l \in \text{guard}(T)} \\ \text{Eff}_{\text{trans}_T} &= \{q_j\} \cup \{v_l\}_{l \in \text{guard}(T)} \cup \Psi_{\text{trans}_T} \end{aligned}$$

where $\Psi_{\text{trans}_T} = \{q_i^{s,t} \rightarrow q_j^t\}$ if $q_j \notin Q_{\text{Fin}}$ and $\Psi_{\text{trans}_T} = \{\text{can_accept}\}$ otherwise. A transition $T = (q_i, \text{guard}(T), q_j)$ can be simulated when there exists a run of the automaton finishing in q_i (as such, q_i had to be *frozen* into q_i^s by means of *switch2aut*). Preconditions include the set $\{\neg v_{-l} \mid l \in \text{guard}(T)\}$, that checks that the transition guard is not violated by the current assignment to variables. Here, we abuse notation and write $l \in \text{guard}(T)$ if the literal l appears in $\text{guard}(T)$. As usual, we use the equivalence $\neg(\neg l) = l$. The effects $\{v_l \mid l \in \text{guard}(T)\}$ set the variables in \mathcal{Y} so that the guard is satisfied and T can be fired. In parallel, the automaton state fluent q_j is set, as to reflect the transition T . According to the semantics of the tokenized fluents, when $q_i^{s,t}$ holds in the current state the token is progressed into q_j^t to denote a commitment to reach an accepting state. If q_j is indeed an accepting state, then the tokenized fluent is not propagated and instead the fluent *can_accept* is set. Notably, the conditional effects $q_i^{s,t} \rightarrow q_j^t$ do *not* delete the copies q_i^s and $q_i^{s,t}$. This allows the agent to simulate more than one transition when the automaton is a non-deterministic, thereby capturing multiple runs of the automaton in the planning state (although it is not obliged to simulate all transitions). When the automaton is deterministic, the effects of *trans_T* allow for at most one transition can be simulated. Finally, the fluent q_j appears negated in the preconditions of *trans_T* merely for efficiency purposes, as executing *trans_T* when q_j is true has no value to the plan (and *trans_T* can be safely pruned).

The agent has two action mechanisms to switch back to environment mode: *accept* and *switch2env*. At any time during automaton mode, the agent can execute *switch2env* causing all *frozen* copies q^s and $q^{s,t}$ to be deleted. The purpose

of *Regularize*, which is optional and defined as $\{\neg v_z, \neg v_{\neg z} \mid z \in \mathcal{X} \cup \mathcal{Y}\}$, is to improve search performance.

$$Pre_{switch2env} = \{aut_mode\}$$

$$Eff_{switch2env} = \{env_mode, \neg aut_mode, turn_1\} \cup \{\neg q^s, \neg q^{s,t} \mid q \in Q\} \cup Regularize$$

The *accept* action is useful to compilations based on Büchi automata, and recognizes runs that have satisfied a commitment to hit an accepting state. At least one of these runs exist if fluent *can_accept* (which is part of the preconditions) holds true. By executing *accept*, the agent *forgets* those runs that did not satisfy the commitment to hit an accepting state, and commits to progress the rest of the runs into runs that hit another accepting state. The agent can postpone action *accept* as much as necessary in order to progress all relevant runs into runs that hit an accepting state. Action *accept* has a non-deterministic effect *goal*, introduced artificially as a method to find infinite plans that visit accepting states infinitely often. Full details can be found in [11].

$$Pre_{accept}(h) = \{aut_mode, can_accept, at_horizon(h)\}$$

$$Eff_{accept}(h) = oneof(\{goal\},$$

$$\{turn_1, at_horizon(0), \neg at_horizon(h)\} \cup \{env_mode, \neg aut_mode, \neg can_accept\} \cup$$

$$\{q^s \rightarrow \neg q^s, q^{s,t} \rightarrow \neg q^{s,t} \mid q \in Q\} \cup \{q \wedge q^t \rightarrow \{\neg q, \neg q^t\} \mid q \in (Q \setminus Q_{Fin})\} \cup$$

$$\{q \wedge \neg q^t \rightarrow q^t \mid q \in Q\} \cup Regularize)$$

Initial and Goal States: The initial state of is $\mathcal{I} = \{next(h+1, h) \mid h \in 0 \dots H\} \cup \{q_0, env_mode, turn_1, at_horizon(0)\}$. When the input of the algorithm is a Büchi automaton, the goal is $\mathcal{G} = \{goal\}$. For NFAs and DFAs, the goal is $\mathcal{G} = \{can_accept\}$.

These steps comprise our compilation of LTL synthesis into FOND, Syn2FOND.

Definition 1 (Syn2FOND). For LTL synthesis problem $\mathcal{P} = \langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$, (NBA, DBA, NFA, or DFA) automaton A_φ , and parameter H , the Syn2FOND compilation constructs the FOND problem $\mathcal{P}'(\mathcal{X}, \mathcal{Y}, A_\varphi, H) = \langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$ as described above.

Solutions to the compiled problem \mathcal{P}' yield solutions to \mathcal{P} (cf. Theorem 3). The iterated search of solutions to Syn2FOND compilations (with $H = 1, 2 \dots, 2^{|\mathcal{Q}|}$) is guaranteed to succeed, if \mathcal{P} is realizable, when the input automaton is a DBA, NFA, or DFA (cf. Theorem 4). This follows, intuitively, from the fact that if a solution exists, then a strong cyclic policy can be unfolded and simulated in a Syn2FOND compilation search graph. If the agent's strategy cannot always guarantee hitting an accepting state within $H \leq 2^{|\mathcal{Q}|}$ turns, then the environment can force a non-accepting cycle – i.e., the environment has a winning strategy that prevents the agent from satisfying the specification. With deterministic automata, the bound can be lowered to $H \leq |\mathcal{Q}|$. We illustrate below with a counter-example that completeness is not guaranteed for NBAs.

Theorem 3 (soundness). *Strong-cyclic plans to the Syn2FOND compilation \mathcal{P}' correspond to solutions for \mathcal{P} .*

Theorem 4 (completeness). *For a realizable LTL synthesis problem $\mathcal{P} = \langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$, and NFA automaton A_φ , the Syn2FOND compilation \mathcal{P}' is solvable for some $H \leq 2^{|\mathcal{Q}|}$. When A_φ is a DBA or DFA, the bound can be lowered to $H \leq |\mathcal{Q}|$.*

Incompleteness of the NBA-based compilation. The NBA-based compilation is not guaranteed to preserve solutions. Let $\varphi = \circ(\Box x \vee \Diamond \neg x)$, and consider the NBA A_φ with states $Q = \{q_0, q_1, q_2, q_3\}$, $\delta^* = \{(q_0, \top, q_1), (q_0, \top, q_2), (q_1, x, q_1), (q_2, x, q_2), (q_3, \top, q_3)\}$, and $Q_{Fin} = \{q_1, q_3\}$. The environment can consistently play x a finite, but unbounded number of times before playing $\neg x$ – at which point the runs of the automaton that finish in q_2 must not have been forgotten. There is no bounded parameter H that can satisfy such a requirement.

5 Evaluation

Our main objective for the evaluation is to give a sense of when to choose one formalism over another. Although the same problems can be represented as either LTL synthesis or FOND planning, the choice of formalism can have a dramatic impact on the time required to find a solution. We would expect the FOND setting to be better suited for problems with more “structure”, and our results serve to illustrate this hypothesis. In our experiments, we used state-of-the-art synthesis and FOND tools Acacia+ [5] and PRP [14]. Our Syn2FOND algorithm was implemented in a tool we named SynKit. SynKit uses Spot [22] to transform LTL formulae into NBA, and PRP as FOND planner.

We considered some representative problems from both synthesis and FOND perspectives, retrieved from the Syntcomp and IPC competitions, respectively. The first group of problems – **lily** and **loadcomp** (*load*, for short) – come from the synthesis community. The second group of problems – **ttw** and **ctw** – come from the FOND benchmark tireworld. We performed experiments with Acacia⁺ and our synthesis tool Syn2FOND equipped with PRP as strong cyclic planner. Additionally, we tested PRP in some problems directly encoded as FOND. The first thing to note is the drastic performance hit that can occur converting from one formalism to another. Going from LTL synthesis to FOND is workable in some instances, but the opposite direction proved impossible for even the simplest **ttw** problems that can be solved very efficiently in its native FOND formulation. Automata transformations become a bottleneck in the synthesis tools, causing time (TLE, 30 min) and memory (MLE, 512 MB) limit exceptions. This is because the specification requires complex constraints to properly maintain the reachable state-space. It is this “structure” that we conjecture the synthesis tools struggle with, and test separately below with two newly introduced domains.

We encoded directly and compactly in both LTL and FOND the newly introduced domain **build**. The **build** domain addresses the problem of building maintenance, and requires the agent to maintain which rooms have their lights on or off depending on the time of day and whether or not people are in the room. The environment controls the transition between day and night, as well as when people enter or leave a room. The agent must control the lights in response. Problems **build-p#** have # rooms, and problems **build-irr-p#-n** introduce n rooms that may non-deterministically be vacuumed at night (controlled by the environment). Note that this is irrelevant to the task of turning lights on or off (the vacuuming can be done in any lighting condition). For the **build-irr** domain, we could see that the synthesis tools scale far worse as the number of rooms increases (both in generating automata and performing the synthesis). Further, as the number of rooms that need to be vacuumed (which is irrelevant to computing a controller), the relevance reasoning present in the FOND planner is able to cope by largely ignoring the irrelevant aspects of the environment. Conversely, the synthesis component of Acacia+ struggles a great deal. This highlights the strength of the FOND tools for leveraging state relevance to solve problems efficiently.

Finally, we created a synthetic domain that lets us tune the level of “structure” in a problem: more structure leads to fewer possibilities for the environment to act without violating a constraint or assumption. In the **switches** domain, a total of n switches, $s_1 \dots s_n$, initially switched on, need to be all switched off eventually. The environment affects the state of the switches non-deterministically. However, the dynamics of the environment is such that immediately after the agent switches off s_k , the environmental non-determinism can only affect the state of a certain number of switches $s_{k'}$, with $k' > k$. A trivial strategy for the agent is to switch off s_1 to s_n in that order. We encoded a series of **switches** problems natively as LTL specifications in TLSF format and also as FOND. Table 1 shows how Acacia+, Syn2FOND, and PRP fared with these problems. They are in three distinct sets (each of increasing number of switches), and within each set the problems range from most structured to least (by varying k). The problems in the first two groups are solved quite readily by PRP, and so the trend is less clear, but for the larger problems we find that PRP struggles when there is less structure and the environment can flip many switches without violating an assumption. This trend also manifests in the Syn2FOND compilations. On the other side, we find that the most structured problems are the most difficult for Acacia+, and the compilation into automata becomes the bottleneck again. On the other hand, the synthesis becomes easier when there is less structure (i.e., more switches can be flipped).

The structure we tune in the **switches** domain is one property of a problem that may favour FOND technology over the synthesis tools. Another is the presence of state variables that are irrelevant. Other notions, such as causal structure in the problem, may also play an important role in features that separate the effectiveness of the two formalisms. We plan to investigate these possibilities in future work.

Table 1. Performance of LTL synthesis and FOND planning on the **switches** problems.

Problem	Acacia+			Syn2FOND(PRP)				PRP
	Aut	$N \times Q $	Syn	H	Aut	$ Q $	Search	Search
p4-0	212	6×131	0.01	3	0.16	13	3.10	0.01
p4-1	11.1	6×195	0.03	3	0.17	13	2.56	0.32
p4-2	1.55	6×181	0.02	3	0.16	13	1.26	0.66
p4-3	0.58	6×46	0.02	3	0.10	12	1.04	0.22
p5-0	TLE	—	—	3	1.33	15	1.02	0.01
p5-1	TLE	—	—	3	1.59	15	0.88	3.30
p5-2	16991	7×548	0.02	3	1.25	15	0.94	2.14
p5-3	533	7×452	0.08	3	1.34	15	15.8	3.82
p5-4	111	7×236	0.06	3	0.59	14	10.2	4.66
p6-0	TLE	—	—	3	9.11	17	1.90	0.01
p6-1	TLE	—	—	3	11.8	17	1.80	4.90
p6-2	TLE	—	—	3	11.4	17	1.78	5.54
p6-3	TLE	—	—	3	10.7	17	1.52	16.7
p6-4	TLE	—	—	3	10.4	17	3.66	25.4
p6-5	TLE	—	—	3	6.15	16	3.32	26.2

6 Concluding Remarks

LTL synthesis is an important and challenging problem for which broadly effective tools have remained largely elusive. Motivated by recent advances in the efficiency of FOND planning, this work sought to examine the viability of FOND planning as a computational tool for the realization of LTL synthesis. To this end, we established the theoretical correspondence between LTL synthesis and strong solutions to FOND planning. We also provided the first approach to automatically translate a realizability problem, given by a specification in LTL or LTL_f , into a planning problem described in PDDL. Experiments with state-of-the-art LTL synthesis and FOND solvers highlighted properties that challenged or supported each of the solvers. Our experiments show automated planning to be a viable and effective tool for highly structured LTL synthesis problems.

Acknowledgements. The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Fondecyt grant numbers 1150328 and 1161526.

References

1. Church, A.: Applications of recursive arithmetic to the problem of circuit synthesis. In: Summaries of the Summer Institute of Symbolic Logic, Cornell University, vol. 1, pp. 3–50 (1957)
2. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Saar, Y.: Synthesis of reactive (1) designs. *J. Comput. Syst. Sci. JCSS* **78**(3), 911–938 (2012)
3. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57 (1977)
4. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL, pp. 179–190 (1989)
5. Bohy, A., Bruyère, V., Filiot, E., Jin, N., Raskin, J.-F.: Acacia+, a tool for LTL synthesis. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 652–657. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_45
6. Jobstmann, B., Bloem, R.: Optimizations for LTL synthesis. In: FMCAD, pp. 117–124 (2006)
7. De Giacomo, G., Felli, P., Patrizi, F., Sardiña, S.: Two-player game structures for generalized planning and agent composition. In: AAAI (2010)
8. Patrizi, F., Lipovetzky, N., Geffner, H.: Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In: IJCAI (2013)
9. Sardiña, S., D’Ippolito, N.: Towards fully observable non-deterministic planning as assumption-based automatic synthesis. In: IJCAI, pp. 3200–3206 (2015)
10. De Giacomo, G., Vardi, M.Y.: Synthesis for LTL and LDL on finite traces. In: IJCAI, pp. 1558–1564 (2015)
11. Camacho, A., Triantafyllou, E., Muise, C., Baier, J.A., McIlraith, S.A.: Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In: AAAI, pp. 3716–3724 (2017)
12. Camacho, A., Baier, J.A., Muise, C.J., McIlraith, S.A.: Finite LTL synthesis as planning. In: ICAPS (2018, to appear)
13. Mattmüller, R., Ortlieb, M., Helmert, M., Bercher, P.: Pattern database heuristics for fully observable nondeterministic planning. In: ICAPS, pp. 105–112 (2010)
14. Muise, C., McIlraith, S.A., Beck, J.C.: Improved non-deterministic planning by exploiting state relevance. In: ICAPS, pp. 172–180 (2012)
15. Geffner, H., Bonet, B.: A concise introduction to models and methods for automated planning. *Synth. Lectu. Artif. Intell. Mach. Learn.* **7**(2), 1–141 (2013)
16. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI (2013)
17. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. *Inf. Comput.* **115**(1), 1–37 (1994)
18. Baier, J.A., McIlraith, S.A.: Planning with temporally extended goals using heuristic search. In: ICAPS, pp. 342–345 (2006)
19. Kissmann, P., Edelkamp, S.: Solving fully-observable non-deterministic planning problems via translation into a general game. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS (LNAI), vol. 5803, pp. 1–8. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04617-9_1
20. Rintanen, J., Heljanko, K., Niemelä, I.: Planning as satisfiability: parallel plans and algorithms for plan search. *Artif. Intell. AIJ* **170**(12–13), 1031–1080 (2006)

21. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge (2001)
22. Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., Xu, L.: Spot 2.0 - a framework for LTL and ω -automata manipulation. In: ATVA, pp. 122–129 (2016)



Logic-Based Benders Decomposition for Two-Stage Flexible Flow Shop Scheduling with Unrelated Parallel Machines

Yingcong Tan^(✉) and Daria Terekhov^(✉)

Mechanical, Industrial and Aerospace Engineering,
Concordia University, Montréal, Canada

t_yingco@encs.concordia.ca, daria.terekhov@concordia.ca

Abstract. We study a two-stage flexible flow shop scheduling problem (FFSP) with the objective of makespan minimization. There is a single machine in stage 1 and unrelated parallel machines in stage 2. We propose a logic-based Benders decomposition (LBBD) algorithm, which decomposes this problem into a mixed-integer programming (MIP) master problem that sequences jobs on stage 1 and assigns jobs to machines on stage 2, and a set of constraint programming sub-problems that aim to find a feasible schedule on stage 2. Extensive computational results show that LBBD outperforms the best-known MIP model for this problem in terms of both computational time and ability to prove optimality over the majority of test instances. Additional experiments show that the superiority of LBBD over the monolithic MIP model holds regardless of whether algorithm tuning features are applied.

Keywords: Constraint satisfaction · Scheduling · Flexible flow shop
Unrelated parallel machines · Logic-based Benders decomposition
Mixed-integer programming · Constraint programming

1 Introduction

Automated planning and scheduling continues to be an important part of artificial intelligence research and practice [4, 9, 10, 23]. A commonly-occurring [7, 18], but not well-studied, scheduling environment is the flexible flow shop. The flexible flow shop scheduling problem (FFSP) consists of finding an optimal schedule of jobs on a set of stages with one or multiple parallel machine(s) at each stage. All jobs follow the same route through the stages. The parallel machines can be identical (job processing times are the same for all machines), uniform (job processing times are determined by machine-specific speed factors) or unrelated (job processing times are both machine and job-dependent). For instance, the development of an automated scheduling tool for operating theatre scheduling where an operating theatre contains one operating room and one recovery room with multiple beds requires an efficient method of solving FFSP [7]. Surgeries are performed in the operating room (i.e., stage 1), after which patients are transferred

to the recovery room (i.e., multiple parallel machines in stage 2). Depending on the setting of the recovery room (e.g., availability of medical devices and medical experts), the recovery time may vary by patient and room (i.e., unrelated parallel machines).

Early work on FFSP mainly focused on developing heuristic algorithms, especially for the two-stage scenario [8, 19]. Among complete approaches, mixed-integer programming (MIP) models [15, 20] and custom branch and bound (B&B) algorithms [6, 8] have been prevalent [19]. According to a comprehensive review of mathematical models for flexible *job shop* problems, which are generalizations of FFSP [5], a precedence-variable based MIP model proposed by [16] provides the best overall performance in terms of average computational time.

According to a comprehensive survey of 200 papers on FFSPs in 2010 [19], only about 5% of the literature considered *unrelated* parallel machines at the second stage. The literature on the unrelated parallel machine scheduling problem, $RM||C_{max}$, is also scarce [2]. Most recently, the unrelated parallel machine problem with sequence-dependent setup times was addressed by Tran et al. [22] via logic-based Benders decomposition and branch-and-check. Their methods can solve instances of $RM||C_{max}$ of up to 60 jobs and 5 parallel machines within a 3-h time window. Bülbül and Şen [2] develop a Benders decomposition approach for the unrelated parallel machine scheduling problem with the objective of total weighted completion time, showing that it is able to solve instances of up to 1000 jobs and 8 parallel machines.

We focus on the problem of makespan minimization in a two-stage FFSP with a single machine in stage 1 and multiple unrelated parallel machines in stage 2, denoted as $FF2|(1, RM)|C_{max}$ [19], and known to be NP-hard [8]. Our chosen method is logic-based Benders decomposition (LBBD), which is known to be powerful for solving scheduling problems that involve assignment and scheduling decisions [4, 9–13, 22]. However, $FF2|(1, RM)|C_{max}$ involves multiple stages, each requiring sequencing decisions to be made, and thus does not allow for a natural separation of assignment and scheduling tasks. As a result, we decompose the problem in a stage-based manner where job assignment and sequencing tasks exist simultaneously in the master problem. Furthermore, to the best of our knowledge, decomposition-based algorithms have never been implemented for solving FFSP, even for the case with two stages.

This paper develops a logic-based Benders decomposition (LBBD) algorithm for $FF2|(1, RM)|C_{max}$ which is able to solve instances of up to 100 jobs and 10 parallel machines. LBBD significantly outperforms an existing state-of-the-art MIP model with and without algorithm tuning features. To the best of our knowledge, our LBBD approach, implemented in commercially-available software, is the first decomposition-based algorithm applied to FFSPs.

This paper is organized as follows. The FFSP problem of interest is formally defined in Sect. 2. The proposed LBBD algorithm is described in Sect. 3. Section 4 presents computational results and discussion of the performance of our LBBD algorithm. Section 5 concludes the paper.

2 Problem Definition

Let $\mathcal{J} = \{1, \dots, n\}$ be the set of jobs, $\mathcal{K} = \{1, 2\}$ be the set of stages, and $\mathcal{I}^{(k)}$ be the set of parallel machines at stage $k \in \mathcal{K}$. We define $\mathcal{I}^{(1)} = \{1\}$ and $\mathcal{I}^{(2)} = \{1, \dots, m\}$. In $FF2|(1, RM)|C_{max}$, each job $j \in \mathcal{J}$ needs to be processed on the single stage 1 machine first, and then on one of a set of unrelated parallel machines in the second stage. The feature of unrelated parallel machines in the second stage means that a job j can have different processing times on different machines, i.e., we define $p_{kij} \in \mathbb{R}^+$ to be the processing time of job j on machine i at stage k .

Following the recent work showing the superior performance of disjunctive MIP models for both job shop [14] and flexible job shop [5] settings, we develop the following specialized MIP model for $FF2|(1, RM)|C_{max}$.

The decision variables are: S_{kij} and C_{kij} , the start and completion times, respectively, of job j on machine i of stage k ; C_{max} , the makespan; V_{kij} , an assignment variable that is 1 if job j is assigned to machine i in stage k and 0 otherwise; and X_{kijg} is 1 if job j precedes job g on machine i in stage k and 0 otherwise. In addition, M is an arbitrarily large number.

The model is then stated as follows:

$$\text{Minimize } C_{max} \tag{1}$$

$$\text{subject to } \sum_{i \in \mathcal{I}^{(2)}} V_{2ij} = 1 \quad j \in \mathcal{J} \tag{2}$$

$$C_{max} \geq \sum_{i \in \mathcal{I}} C_{2ij} \quad j \in \mathcal{J} \tag{3}$$

$$S_{2ij} + C_{2ij} \leq V_{2ij}M \quad j \in \mathcal{J}, i \in \mathcal{I}^{(2)}, \tag{4}$$

$$C_{2ij} - p_{2ij} \geq S_{2ij} - (1 - V_{2ij})M \quad j \in \mathcal{J}, i \in \mathcal{I}^{(2)} \tag{5}$$

$$S_{kij} \geq C_{kig} - (X_{kijg})M \quad j, g \in \mathcal{J}, i \in \mathcal{I}^{(k)}, k \in \mathcal{K} \tag{6}$$

$$S_{kig} \geq C_{kij} - (1 - X_{kijg})M \quad j, g \in \mathcal{J}, i \in \mathcal{I}^{(k)}, k \in \mathcal{K} \tag{7}$$

$$\sum_{i \in \mathcal{I}^{(2)}} S_{2ij} \geq \sum_{i \in \mathcal{I}^{(1)}} C_{1ij} \quad j \in \mathcal{J} \tag{8}$$

$$S_{kij}, C_{kij} \geq 0; V_{kij}, X_{kijg} \in \{0, 1\} \quad j, g \in \mathcal{J}, i \in \mathcal{I}^{(k)}, k \in \mathcal{K} \tag{9}$$

The objective is makespan minimization. Constraint (2) ensures that every job must be assigned to one and only one machine in stage 2 where parallel machines exist. Constraint (3) states that C_{max} is the largest completion time. Note that each job will have only one non-zero C_{kij} over all machines i in each stage k , which is enforced by subsequent constraints.

Constraint (4) ensures that S_{2ij} and C_{2ij} take positive values if and only if job j is assigned to machine i in stage k (i.e., $V_{2ij} = 1$). Job completion times on each machine are defined in constraint (5) using the same logic. Constraints (6) & (7) are disjunctive constraints which define the precedence of two jobs, j & g , on machine i in stage k . In a FFSP with two stages, a job j can not start

in stage 2 before its operation at stage 1 is completed – this fact is reflected by constraint (8). Constraints (9) ensure non-negativity and integrality of the decision variables.

We add two more constraints to the model:

$$\sum_{j \in \mathcal{J}} V_{1ij} = 0 \quad i = \{2, \dots, n\}, \quad (10)$$

$$C_{max} \geq \min_{j \in \mathcal{J}} p_{11j} + \sum_{j \in \mathcal{J}} p_{2ij} V_{2ij} \quad i \in \mathcal{I}^{(2)}. \quad (11)$$

Since there is only a single machine in stage 1, Constraint (10) disables the feature of parallel machines in stage 1. Consequently, all decision variables with machine index $i > 1$ in stage 1 are dummy variables in the model (i.e., $V_{1ij} = 0$, $X_{1ijg} = 0$, $S_{1ij} = 0$, $C_{1ij} = 0$, $\forall j \in \mathcal{J}, i > 1$). Constraint (11) defines a strengthened lower bound for the objective function: the makespan has to be at least the smallest job duration on stage 1 plus the sum of job durations assigned to a particular machine at stage 2.

3 Logic-Based Benders Decomposition

LBBD is a generalized decomposition method in which an optimality proof scheme and a method of generating Benders cuts must be developed individually for each problem class. To solve $FF2|(1, RM)|C_{max}$, we decompose it into a master problem which is a relaxation of the original MIP model (Sect. 2), and multiple independent single-machine scheduling sub-problems with release dates (i.e., $1|r_i|C_{max}$). A MIP model is formulated for the master problem. A set of independent constraint programming (CP) models are solved in series as the sub-problems. Such a hybrid framework of MIP and CP models has been widely successful [1, 11–13, 21].

3.1 Master Problem

The master problem is obtained by relaxing the disjunctive constraints of job precedence on stage 2. Thus, solving the master problem provides only a partial job sequence on stage 1 and job-machine assignment for stage 2. The MIP formulation of the master problem is:

$$\text{Minimize } C_{max} \quad (12)$$

$$\text{subject to } \text{Const. (2) – (5), (8), (10), (11)}$$

$$S_{1ij} \geq C_{1ig} - (X_{1ijg})M \quad j, g \in \mathcal{J}, i \in \mathcal{I}^{(1)}, \quad (13)$$

$$S_{1ig} \geq C_{1ij} - (1 - X_{1ijg})M \quad j, g \in \mathcal{J}, i \in \mathcal{I}^{(1)} \quad (14)$$

$$\text{Benders cuts} \quad (15)$$

$$S_{kij}, C_{kij} \geq 0; V_{kij}, X_{kijg} \in \{0, 1\} \quad j, g \in \mathcal{J}, i \in \mathcal{I}, k \in \mathcal{K} \quad (16)$$

Constraints (13) & (14) are relaxations of constraints (6) & (7) obtained by setting $k = 1$. Constraints (2) – (5), (8), (10), (11) from the original MIP model (Sect. 2) are also included. Note that the LB strengthening inequality (11) produces a valid lower bound with respect to the original MIP model. Therefore, adding this constraint helps with LB strengthening without compromising the validity of the master problem. In addition, constraint (15) represents the Benders cuts that will be added to the master problem iteratively. Details on how to generate such cuts are provided in Sect. 3.3.

The solutions obtained from the master problem are infeasible with respect to the original MIP model. Thus, the optimal objective value of the master problem provides a lower bound on the optimal makespan. In the LBB algorithm, this lower bound value is recorded iteratively and used for proving optimality.

3.2 Sub-problems

After solving the master problem, job completion times at stage 1 and job-machine assignment at stage 2 are known. Thus, we generate m independent sub-problems, one for each machine $i \in \mathcal{I}^{(2)}$, to compute the complete job schedule in stage 2. Our CP model for solving sub-problem i in iteration h is:

$$\text{Minimize } C_{max}^{ih} \quad (17)$$

$$\text{subject to } C_{max}^{ih} \geq job_j.end \quad j \in \{\mathcal{J} \mid V_{2ij} = 1\} \quad (18)$$

$$job_j.duration = p_{kij} \quad j \in \{\mathcal{J} \mid V_{2ij} = 1\} \quad (19)$$

$$job_j.start \geq C_{11j} \quad j \in \{\mathcal{J} \mid V_{2ij} = 1\} \quad (20)$$

$$NoOverlap(job_j) \quad j \in \{\mathcal{J} \mid V_{2ij} = 1\} \quad (21)$$

The objective (17) and constraint (18) ensure makespan minimization on each machine $i \in \mathcal{I}^{(2)}$. Constraint (19) defines the duration of each job. Constraint (20) states that the earliest start time of job j in stage 2 is greater than or equal to its corresponding completion time in stage 1. Constraint (21) ensures that all jobs are sequenced without overlap, since all machines are of unary capacity. Note that job_j , $j \in \mathcal{J}$, is an interval variable with three associated values: duration ($job_j.duration$), start time ($job_j.start$) and end time ($job_j.end$). The constraints of the sub-problem are active only if $V_{2ij} = 1$, i.e., they are defined only for jobs that have been assigned to machine i . In each iteration h , m independent CP models are solved in series.

By solving m sub-problems at iteration h , the LBB algorithm produces a complete job schedule with makespan $Z_{sp}^h = \max_{i \in \mathcal{I}^{(2)}} C_{max}^{ih}$ which is feasible with respect to the original MIP model (Sect. 2), but not necessarily optimal. Therefore, Z_{sp}^h is an upper bound on the optimal makespan, which is updated iteratively every time a better feasible solution is found.

3.3 Benders Cuts

After each iteration, a set of Benders cuts are added to the master problem. Benders cuts remove non-optimal solutions, then force the master problem to

find new solutions in future iterations. Observe that every solution of the master problem can be fully represented with the combination of V_{2ij} (job-machine assignment in stage 2) and X_{11jg} (job sequencing in stage 1). Based on this observation, two solutions of the master problem are different if they have different job-machine assignment in stage 2, or different job sequence in stage 1 or both. Therefore, we define the following Benders cut at iteration h :

$$C_{max} \geq Z_{sp}^h \left(1 - \sum_{i \in \mathcal{I}^{(2)}, j \in \mathcal{J}: \hat{V}_{2ij}^h = 1} (1 - V_{2ij}) - \sum_{j, g \in \mathcal{J}: \hat{X}_{11jg}^h = 1} (1 - X_{11jg}) \right) \quad (22)$$

where C_{max} is the master problem objective, Z_{sp}^h is the sub-problem makespan in iteration h , and \hat{V}_{2ij}^h and $\hat{X}_{11jg}^h \forall i \in \mathcal{I}^{(2)}, j \in \mathcal{J}$, are the solutions of the master problem in iteration h . The above inequality ensures that the master problem will have a different solution in future iterations $h' > h$ if and only if such a solution improves the objective function value. Given the same assignment of values to all variables, i.e., $V_{2ij} = \hat{V}_{2ij}^h, X_{11jg} = \hat{X}_{11jg}^h \forall i \in \mathcal{I}^{(2)}, j, g \in \mathcal{J}$, both summation terms are equal to 0. Then, the right-hand side of (22) equals to Z_{sp}^h , providing a bound for new solutions. This constraint is inactive if at least one of V_{2ij}, X_{11jg} takes a different value in future iterations. The proof of validity of inequality (22), which constitutes a *no-good* cut [12], is similar to the proof in [17] and is omitted here.

3.4 Optimality Condition

The LBB algorithm iterates between the master problem and the sub-problems. The master problem provides a lower bound value; the sub-problem provides an upper bound value. A solution is proved to be optimal when upper and lower bounds converge. In the LBB framework, the obtained solution is proven to be optimal if one of the following two conditions is met in an iteration h :

1. $Z_{mp}^h = \bar{Z}$, that is, the master problem objective value in iteration h (Z_{mp}^h) coincides with the best upper bound value (\bar{Z});
2. $Z_{sp}^h = Z_{mp}^h$, that is, in iteration h , the objective value of sub-problem (Z_{sp}^h) coincides with the objective value of master problem (Z_{mp}^h).

3.5 Algorithm Strengthening

We exploit some characteristics of $FF2|(1, RM)|C_{max}$ to strengthen LBB.

Strengthening Lower Bound. With a single machine in stage 1, all jobs will be scheduled in sequence without any idle time. Let g denote the job which is completed last in stage 1. Then the makespan must be greater or equal to the completion time of job g in stage 2. Furthermore, the completion time of job g in stage 2 is minimized if it starts immediately after its completion in stage 1 and

if, in addition, it is assigned to the machine at stage 2 on which its processing time is shortest. Formally,

$$C_{max} \geq \sum_{i \in \mathcal{I}^{(2)}} C_{2ig} V_{2ig} = \sum_{i \in \mathcal{I}^{(2)}} \left(\underbrace{ST_{2ig}}_{=\sum_{j \in \mathcal{J}} p_{11j}} + \underbrace{WaitTime}_{\geq 0} + \underbrace{p_{2ig}}_{\geq \min_{j \in \mathcal{J}, i \in \mathcal{I}^{(2)}} p_{2ij}} \right) V_{2ig}$$

Using these observations, we derive the following lower bound: $C_{max} \geq \sum_{j \in \mathcal{J}} p_{11j} + \min_{j \in \mathcal{J}, i \in \mathcal{I}^{(2)}} p_{2ij}$, which is valid for both the MIP model (Sect. 2) and the LBB master problem (Sect. 3.1).

Tightening Big- M . As reported in the literature [3], setting the value of big M properly helps with solving MIP models. In this study, M is used in constraints (4)–(7) in the MIP model, and (13)–(14) in the master problem. In constraint (4), M must be larger than the sum of S_{2ij} and C_{2ij} to ensure its validity, but needs to be larger than either one of S_{2ij} or C_{2ij} to ensure the validity of other constraints. Thus, a value for M that is valid for constraint (4) is also valid for all other constraints. In particular, given constraint (4), $M \geq \max_{i \in \mathcal{I}^{(2)}, j \in \mathcal{J}} S_{2ij} + \max_{i \in \mathcal{I}^{(2)}, j \in \mathcal{J}} C_{2ij}$. Since

$$\begin{aligned} \max_{i \in \mathcal{I}^{(2)}, j \in \mathcal{J}} C_{2ij} &\geq \max_{i \in \mathcal{I}^{(2)}, j \in \mathcal{J}} S_{2ij} + \sum_{j \in \mathcal{J}: V_{2ij}=1} p_{2ij} && i \in \mathcal{I}^{(2)} \\ &\geq \sum_{j \in \mathcal{J}} p_{11j} + \sum_{j \in \mathcal{J}} \max_{i \in \mathcal{I}^{(2)}} p_{2ij}, \end{aligned}$$

we find that $M = 2 \sum_{j \in \mathcal{J}} p_{11j} + \sum_{j \in \mathcal{J}} \max_{i \in \mathcal{I}^{(2)}} p_{2ij}$.

4 Computational Study

In this section, we test our LBB algorithm against the state-of-the-art MIP model (Sect. 2). The goal of our experiments is to computationally demonstrate the performance superiority of our LBB algorithm in two respects: (1) ability to find and prove an optimal solution and (2) the time required to do so.

4.1 Experimental Setup

All experiments are performed on a Intel Core i5 2.53 GHz CPU with 4 GB of main memory with a time limit of 20 min. All MIP and CP models are implemented in *CPLEX Studio 12.6*. Due to the lack of an agreed set of benchmarks for FFSPs [19], job durations in test instances are commonly generated from a uniform distribution with different ranges [8, 22]. In this paper, we use the discrete uniform distribution with two different ranges [1, 5] (narrow) and [1, 100] (wide). We generate instances with 10, 20, 50 and 100 jobs, and 2, 5 and 10 unrelated parallel machines. One hundred instances are generated with each combination, except the combination of 10 jobs and 10 parallel machines whose job-machine ratio in stage 2 (i.e. $n/RM = 1$) is too low. Therefore, there are 2200 test instances in total. Instances with these problem sizes are widely used in the FFSP literature and parallel machine scheduling problems [2, 8, 22].

4.2 Experimental Results and Discussions

A summary of computational results is presented in Tables 1 and 2. Performance of LBB and MIP is evaluated in terms of (a) the number of instances for which optimality could not be proven within 20 min, which we refer to as *unsolved* and denote as *un.* in the tables, (b) the optimality gap, calculated as $gap = 100\%(\bar{Z} - \underline{Z})/\bar{Z}$, where \bar{Z} and \underline{Z} are the best upper and lower bounds recorded within 20 min (if applicable) and calculated only for instances where a feasible solution was found, and (c) the computational time required to find and prove an optimal solution, denoted as *Time* in the table (for the purposes of this

Table 1. Statistical analysis for narrow-range instances ($p_{kij} \sim U[1, 5]$). Best performance for each problem size for each metric is in bold.

		un./ <i>Ave_{gap}</i> (%)/ <i>Std_{gap}</i> (%)		Time (s), Ave./Std./95% C.I.	
n	RM	MIP	LBB	MIP	LBB
10	2	0/NA/NA	0/NA/NA	0.35/0.39/0.08	0.25/0.53/0.10
	5	0/NA/NA	0/NA/NA	0.33/0.18/0.04	0.11/0.07/0.01
20	2	3/1.70/0.14	1/2.47/NA	51.33/228.55/45.34	13.46/120.00/23.81
	5	0/NA/NA	0/NA/NA	6.27/3.85/0.76	0.32/0.20/0.04
	10	0/NA/NA	0/NA/NA	9.49/5.45/1.08	0.35/0.14/0.040
50	2	31/0.86/0.38	0/NA/NA	500.01/520.72/103.32	29.28/121.03/24.02
	5	9/0.66/0.03	0/NA/NA	374.58/340.31/67.53	9.72/15.84/3.14
	10	1/0.65/NA	0/NA/NA	190.60/197.58/39.20	7.98/4.01/0.80
100	2	80/2.88/3.23	1/0.33/NA	1003.35/397.93/78.96	122.40/169.51/33.63
	5	63/1.04/1.02	1/0.35/NA	954.01/382.12/75.82	137.96/168.26/33.39
	10	48/0.64/0.31	1/0.11/NA	961.15/282.20/55.99	152.30/186.12/36.93

Table 2. Statistical analysis for wide-range instances ($p_{kij} \sim U[1, 100]$). Best performance for each problem size for each metric is in bold.

		un.(MP)/ <i>Ave_{gap}</i> (%)/ <i>Std_{gap}</i> (%)		Time (s), Ave./Std./95% C.I.	
n	RM	MIP	LBB	MIP	LBB
10	2	0/NA/NA	2/0.56/NA	11.88/72.83/14.45	24.65/168.67/33.47
	5	0/NA/NA	0/NA/NA	2.97/16.88/3.35	0.36/1.07/0.21
20	2	32/ 0.49/0.37	31(17)/2.77/3.95	402.23/554.68/110.06	375.16/555.15/110.15
	5	29/ 0.21/0.16	10(10) /NA/NA	373.09/539.46/107.04	200.87/442.57/87.81
	10	20/ 0.20/0.17	11(8) /0.85/0.51	298.94/471.28/93.51	178.87/423.11/83.95
50	2	83/ 0.53/0.55	65(45) /5.78/4.98	1039.34/385.38/76.47	801.7/556.1/110.34
	5	86/ 0.25/0.27	35(31) /1.04/1.36	1122.43/237.5/47.13	446.02/559.68/111.05
	10	73/ 0.10/0.07	15(12) /1.27/1.35	984.5/391.71/77.72	231.52/443.66/88.03
100	2	97/ 1.7/2.06	80(62) /2.42/2.42	1178.93/145.86/28.94	1011.93/416.59/82.66
	5	98/0.82/1.28	49(45) / 0.57/0.55	1189.52/75.79/15.04	767.33/474.35/94.12
	10	95/0.44/0.46	34(32) / 0.33/0.49	1177.31/100.94/20.03	600.81/496.19/98.45

calculation, the computational time for instances where optimality is not proven within the time limit is recorded as 20 min). For all performance metrics, we present the sample average and standard deviation. Note that standard deviation is not calculated if the number of unsolved instances is low, reported as ‘NA’ in the table. For LBB, ‘(MP)’ represents the number of instances where the first master problem could not be solved within the time limit.

Run Time. LBB has a smaller average run time over all instance classes except the class of 10-job, 2-machine, $p_{kij} \sim U[1, 100]$ instances. More importantly, there is no overlap in the 95% CIs over the majority of instance classes except ones generated from $p_{kij} \sim U[1, 100]$ with 10 and 20 jobs. The performance superiority of the LBB algorithm over the MIP model continues to improve as the instance complexity level increases (i.e., more jobs and parallel machines). One might notice that the MIP model has a smaller standard deviation for computational time than LBB for some large instance classes (i.e., 50-job, 100-job with $p_{kij} \sim U[1, 100]$ in Table 2). For these classes, MIP could not prove optimality in the majority of instances, resulting in the same run time of 20 min being recorded in our calculations for all these instances – thus, while the average run time is high, the variability (measured via standard deviation) is low.

Run Time of Master Problem and Sub-problem. On average, the proportion of run time spent on solving the master problem and sub-problems is 99.67% and 0.36%, respectively. For some instances with a large number of jobs and/or parallel machines (262 out of 1100 instances with $p_{kij} \sim U[1, 100]$ in Table 2), LBB can not solve the master problem even once. In FFSP, the job sequence in stage 1 is highly correlated with job-machine assignment in stage 2. Thus, even the master problem (i.e., a relaxation of $FF2|(1, RM)|C_{max}$) is hard to solve.

Number of Unsolved Instances. For instances generated from $U[1, 5]$, the MIP model solved all instances with 10 and 20 jobs; however, the number of unsolved instances increases sharply for 50-job and 100-job instances to 41/300 and 191/300, respectively. On the other hand, LBB has only 4/1100 unsolved instances, which is a substantial improvement over the MIP model. Similar, though less pronounced, behavior was observed over $U[1, 100]$ instances: 613/1100 and 332/1100 could not be solved for MIP and LBB, respectively.

Despite the fact that LBB is able to prove optimality in a greater number of instances, there are also many instances (i.e., 262 out of 1100 instances generated from $p_{kij} \sim U[1, 100]$) where LBB can not find a feasible solution. This performance is due to the fact that LBB fails to solve the master problem for even one iteration, revealing a limitation of LBB. In this case, the optimality gap is calculated only over instances where a feasible solution is found within the time limit. On the other hand, the MIP model always finds a feasible solution and provides a tight optimality gap, which is important when optimality cannot be proven. The MIP model demonstrates a better performance in terms of optimality gap over 20-job and 50-job instances generated from $p_{kij} \sim U[1, 100]$.

Another observation is that the number of unsolved instances tends to decrease, unexpectedly, for both MIP and LBB, as the number of parallel machine increases. We conjecture that this effect is due to stronger lower bound values derived from constraints (11) and the lower bound presented in Sect. 3.5, since with the same number of jobs and a larger number of parallel machines, the average number of jobs on each machine decreases.

Processing Time Ranges. Our test instances are generated from the discrete uniform distribution ranges $[1, 5]$ and $[1, 100]$. Both MIP and LBB perform better for instances generated from $U[1, 5]$. There are 235 and 4 (out of 1100) unsolved instances for MIP and LBB, respectively. For instances of the same size with $U[1, 100]$, the total number of unsolved instances for MIP and LBB increase sharply to 613 and 332 (out of 1100) respectively. Similarly, we observed a sharp increase in the average computational times for both methods. Our future work includes classification of problem instances based on features such as the range of processing times.

Algorithm Tuning Features. A common limitation discussed in the literature is that extensive algorithm tuning restricts generalization and causes problems for practical implementation. Although these concerns arise mostly for heuristic and meta-heuristic algorithms, we believe the same concerns are applicable in the development of complete methods. In this study, we used three different ways to ‘tune’ our methods: the strengthened lower bound and tightening of the big M value described in Sect. 3.5, and constraint (11). To test the impact of these features, we conducted a preliminary experiment using the same instances as above generated from $U[1, 5]$ with 10 jobs, 2 and 5 machines with all these features disabled. The M value was set to 10000 (an arbitrary large value). Results are shown in Fig. 1. We see that the run time of both MIP and LBB increases as expected. However, the LBB algorithm still provides substantial

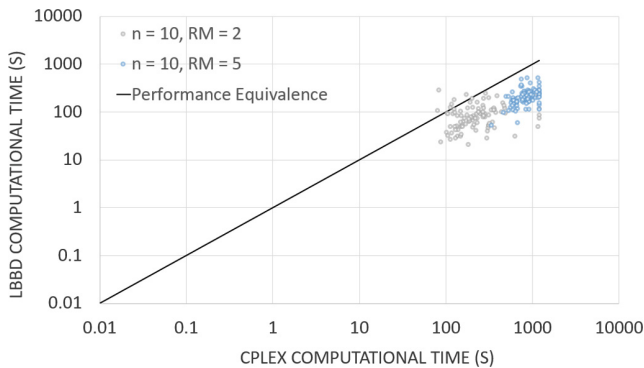


Fig. 1. Run time comparison of MIP and LBB without algorithm tuning features over the same 10-job, 2- or 5-parallel-machine instances with $P_{kij} \sim U[1, 5]$

improvements over the MIP model. This property provides flexibility for practical implementation since in-depth problem specific knowledge is not required to obtain improvements from LBBDD.

Additionally, the tuning methods we used were developed based on the MIP model. We did not exploit any special structure of the master problem or sub-problem to further strengthen LBBDD specifically. Thus, the advantage of LBBDD for FFSP is due primarily to the decomposition itself.

5 Conclusion

We present a logic-based Benders decomposition algorithm for solving two-stage flexible flow shop problems with one machine at stage 1 and two unrelated parallel machines at stage 2. We performed extensive computational experiments over instances with processing times generated from the uniform distribution with different ranges, and also with/without algorithm tuning features. Our LBBDD algorithm is shown to perform better than a specialized MIP model in terms of computational time and number of instances in which optimality is proven within 20 min. These performance gains are apparent even without algorithm-specific tuning for LBBDD. For large-size instances that could not be solved using LBBDD, the main reason was the long run time of the master problem and hence testing branch-and-check [1, 21, 22] on these problem instances is an immediate future work direction.


References

1. Beck, J.C.: Checking-up on branch-and-check. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 84–98. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15396-9_10
2. Bülbül, K., Şen, H.: An exact extended formulation for the unrelated parallel machine total weighted completion time problem. *J. Sched.* **20**(4), 373–389 (2017)
3. Camm, J.D., Raturi, A.S., Tsubakitani, S.: Cutting big M down to size. *Interfaces* **20**(5), 61–66 (1990)
4. Ciré, A.A., Coban, E., Hooker, J.N.: Logic-based Benders decomposition for planning and scheduling: a computational analysis. *Knowl. Eng. Rev.* **31**(5), 440–451 (2016)
5. Demir, Y., İşleyen, S.K.: Evaluation of mathematical models for flexible job-shop scheduling problems. *Appl. Math. Model.* **37**(3), 977–988 (2013)
6. Dessouky, M.M., Dessouky, M.I., Verma, S.K.: Flowshop scheduling with identical jobs and uniform parallel machines. *Eur. J. Oper. Res.* **109**(3), 620–631 (1998)
7. Fei, H., Meskens, N., Chu, C.: A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Comput. Ind. Eng.* **58**(2), 221–230 (2010)
8. Gupta, J.N., Tunc, E.A.: Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *Int. J. Prod. Res.* **29**(7), 1489–1502 (1991)
9. Heching, A., Hooker, J.N.: Scheduling home hospice care with logic-based Benders decomposition. In: Quimper, C.-G. (ed.) CPAIOR 2016. LNCS, vol. 9676, pp. 187–197. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33954-2_14

10. Hooker, J.N.: Job sequencing bounds from decision diagrams. In: Beck, J.C. (ed.) CP 2017. LNCS, vol. 10416, pp. 565–578. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66158-2_36
11. Hooker, J.N.: Logic-based methods for optimization. In: Borning, A. (ed.) PPCP 1994. LNCS, vol. 874, pp. 336–349. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58601-6_111
12. Hooker, J.N.: A hybrid method for the planning and scheduling. *Constraints* **10**(4), 385–401 (2005)
13. Hooker, J.N., Ottosson, G.: Logic-based Benders decomposition. *Math. Program.* **96**(1), 33–60 (2003)
14. Ku, W.Y., Beck, J.C.: Mixed integer programming models for job shop scheduling: a computational analysis. *Comput. Oper. Res.* **73**, 165–173 (2016)
15. Manne, A.S.: On the job-shop scheduling problem. *Oper. Res.* **8**(2), 219–223 (1960)
16. Özgüven, C., Özbakır, L., Yavuz, Y.: Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Appl. Math. Model.* **34**(6), 1539–1548 (2010)
17. Roshanaei, V., Luong, C., Aleman, D.M., Urbach, D.R.: Collaborative operating room planning and scheduling. *INFORMS J. Comput.* **29**(3), 558–580 (2017)
18. Ruiz, R., Maroto, C.: A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *Eur. J. Oper. Res.* **169**(3), 781–800 (2006)
19. Ruiz, R., Vázquez-Rodríguez, J.A.: The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **205**(1), 1–18 (2010)
20. Thomalla, C.S.: Job shop scheduling with alternative process plans. *Int. J. Prod. Econ.* **74**(1), 125–134 (2001)
21. Thorsteinsson, E.S.: Branch-and-check: a hybrid framework integrating mixed integer programming and constraint logic programming. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 16–30. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45578-7_2
22. Tran, T.T., Araujo, A., Beck, J.C.: Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS J. Comput.* **28**(1), 83–95 (2016)
23. Tran, T.T., Vaquero, T., Nejat, G., Beck, J.C.: Robots in retirement homes: applying off-the-shelf planning and scheduling to a team of assistive robots. *J. Artif. Intell. Res.* **58**, 523–590 (2017)



Advice-Based Exploration in Model-Based Reinforcement Learning

Rodrigo Toro Icarte^{1,2}(✉) , Toryn Q. Klassen¹,
Richard Anthony Valenzano^{1,3}, and Sheila A. McIlraith¹

¹ Department of Computer Science, University of Toronto, Toronto, Canada
{[rntoro](mailto:rntoro@cs.toronto.edu), [toryn](mailto:toryn@cs.toronto.edu), [rvalenzano](mailto:rvalenzano@cs.toronto.edu), [sheila](mailto:sheila@cs.toronto.edu)}@cs.toronto.edu

² Vector Institute, Toronto, Canada

³ Element AI, Toronto, Canada

Abstract. Convergence to an optimal policy using model-based reinforcement learning can require significant exploration of the environment. In some settings such exploration is costly or even impossible, such as in cases where simulators are not available, or where there are prohibitively large state spaces. In this paper we examine the use of advice to guide the search for an optimal policy. To this end we propose a rich language for providing advice to a reinforcement learning agent. Unlike constraints which potentially eliminate optimal policies, advice offers guidance for the exploration, while preserving the guarantee of convergence to an optimal policy. Experimental results on deterministic grid worlds demonstrate the potential for good advice to reduce the amount of exploration required to learn a satisficing or optimal policy, while maintaining robustness in the face of incomplete or misleading advice.

Keywords: Markov decision process · Reinforcement learning
Model-based learning · Linear temporal logic · Advice

1 Introduction

Reinforcement Learning (RL) methods can often be used to build intelligent agents that learn how to maximize long-term cumulative reward through interaction with the environment. Doing so generally requires extensive exploration of the environment, which can be infeasible in real-world environments where exploration can be unsafe or require costly resources. Even when there is access to a simulator and exploration is safe, the amount of interaction needed to find a reasonable policy may be prohibitively expensive.

In this paper we investigate the use of *advice* as a means of guiding exploration. Indeed, when humans try to master a new task, they certainly learn through exploration, but they also avail themselves of linguistically expressed advice from other humans. Here we take “advice” to be recommendations regarding behaviour that may describe suboptimal ways of doing things, may not be universally applicable, or may even contain errors. However, even in these cases

people often extract value and we aim to have RL agents do likewise. We use advice to guide the exploration of an RL agent during its learning process so that it more quickly finds useful ways of acting.

We use the language of Linear Temporal Logic (LTL) [1] for providing advice. The advice vocabulary is drawn from state features, together with the linguistically inspired temporal modalities of LTL (e.g., “*Turn out the lights before you leave the office*” or “*Always avoid potholes in the road*”). We propose a variant of the standard model-based RL algorithm *R-MAX* [2] that adds the ability to use given advice to guide exploration. Experimental results on randomly generated (deterministic) grid worlds demonstrate that our approach can effectively use good advice to reduce the number of training steps needed to learn a strong policy, and also recover from misleading advice.

2 Preliminaries

Example environment: Consider a grid world in which the agent starts at some initial location. At various locations there are doors, keys, walls, and nails. The agent can move deterministically in the four cardinal directions, unless there is a wall or locked door in the way. The agent can only enter a location with a door when it has a key, after which point the door and key disappear (i.e., the door remains open). The agent automatically picks up a key whenever it visits a location with a key. The agent receives a reward of -1 for every action, unless it enters a location with nails (reward of -10) or reaches the red door with a key (reward of $+1000$, and the episode ends). Figure 1, which we use as a running example below, depicts an instance of this domain, in which there is a single door and it is red.

In this environment, we may wish to advise the agent to avoid the nails, or to get the key before going to the door. In order to provide advice to an arbitrary agent, we must have a vocabulary from which the advice is constructed. For this purpose, we define a *signature* as a tuple $\Sigma = \langle \Omega, C, \text{arity} \rangle$ where Ω is a finite set of predicate symbols, C is a finite set of constant symbols, and $\text{arity} : \Omega \rightarrow \mathbb{N}$ assigns an arity to each predicate. For example, in the grid-world environment, we use a signature with only a single predicate called **at** (i.e., $\Omega = \{\mathbf{at}\}$ and $\text{arity}(\mathbf{at}) = 1$), where $\mathbf{at}(c)$ states that the agent is at the same location as c . Each object in the domain will be represented with a single constant in C (i.e., $\text{key1}, \text{door1}, \dots$). Intuitively, we use this signature to reference different elements about states in the environment when providing advice.

We define $GA(\Sigma) \stackrel{\text{def}}{=} \{P(c_1, \dots, c_{\text{arity}(P)}) \mid P \in \Omega, c_i \in C\}$. That is, $GA(\Sigma)$ is the set of all *ground atoms* of the first order language with signature Σ . A *ground literal* is either a ground atom or the negation of a ground atom, so $\text{lit}(\Sigma) \stackrel{\text{def}}{=} GA(\Sigma) \cup \{\neg p : p \in GA(\Sigma)\}$ is the set of ground literals. A *truth assignment* can be given by a set $\tau \subseteq \text{lit}(\Sigma)$ such that for every $a \in GA(\Sigma)$, exactly one of a and $\neg a$ is in τ . Let $T(\Sigma)$ be the set of all truth assignments.

A *Markov Decision Process (MDP)* with an initial state and a signature is a tuple $\mathcal{M} = \langle S, s_0, A, p, \gamma, \Sigma, L \rangle$ where S is a finite set of *states*, $s_0 \in S$

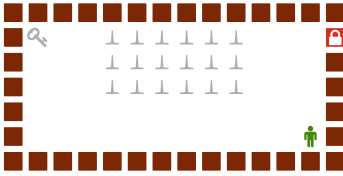


Fig. 1. The agent receives reward for going to the locked red door after having visited the key. It is penalized for stepping on nails. (Color figure online)

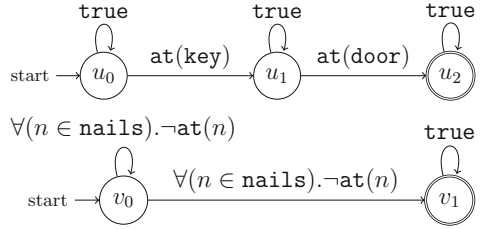


Fig. 2. Advice example, with NFAs corresponding to the LTL formula $\diamond(\text{at}(\text{key}) \wedge \bigcirc \diamond \text{at}(\text{door})) \wedge \square \forall (n \in \text{nails}). \neg \text{at}(n)$

is the initial state, A is a finite set of *actions*, p is a function that specifies *transition probabilities* where $p(s', r|s, a)$ is the probability of transitioning to s' and receiving reward $r \in \mathbb{R}$ if action $a \in A$ is taken in state s , $\gamma \in (0, 1]$ is the *discount factor*, $\Sigma = \langle \Omega, C, \text{arity} \rangle$ is a signature, and $L : S \rightarrow T(\Sigma)$ labels each state with a truth assignment. For example, in our grid-world, the labeling function $L(s)$ makes $\text{at}(c)$ true if and only if the location of the agent is equal to the location of c in state s . Note that as $GA(\Sigma)$ is finite, we could equivalently consider a state label to be a vector of *binary features*, with the i th entry being 1 if the i th ground atom holds in that state, and 0 otherwise. Below, we assume that the agent does not know the transition probability function p (as usual in RL).

3 Providing and Utilizing Advice While Learning

In this section, we describe our LTL advice language and how corresponding automata can be used to monitor satisfaction of such advice. We then describe our method for using the automata to guide exploration in a variant of R-MAX.

3.1 Linear Temporal Logic: A Language for Providing Advice

Providing advice to an agent requires a language for communicating that advice. Given that RL agents are typically engaged in an extended interaction with the environment, this language must allow us to suggest how the agent should behave over time (e.g. “*Get the key and then go to the locked door.*”). To this end, we use Linear Temporal Logic (LTL), a modal temporal logic originally proposed for the verification of reactive systems [1], that has subsequently been used to represent temporally extended goals and preferences in planning [3]. Here, we use it as the basis for expressing advice.

Suppose that we have an MDP with a signature $\mathcal{M} = \langle S, s_0, A, p, \gamma, \Sigma, L \rangle$ for which we wish to provide advice. The language of LTL contains formulae consisting of propositional symbols, which we take to be the ground atoms of

$GA(\Sigma)$ (e.g. $\text{at}(\text{key}2)$), and all formulae that can be constructed from other formulae using the standard set of connectives from propositional logic — namely *and* (\wedge), *or* (\vee), and *negation* (\neg) — and the temporal operators *next* (\bigcirc) and *until* (U). From these temporal operators, we can also derive other useful operators such as *always* (\square) and *eventually* (\diamond). We will also make use of *universal* (\forall) and *existential* (\exists) quantifiers in abbreviating conjunctions and disjunctions. If $T = \{t_1, \dots, t_k\} \subseteq C$ is a set of constant symbols, then $\forall(x \in T).\varphi(x) \stackrel{\text{def}}{=} \varphi(t_1) \wedge \dots \wedge \varphi(t_k)$ and $\exists(x \in T).\varphi(x) \stackrel{\text{def}}{=} \varphi(t_1) \vee \dots \vee \varphi(t_k)$.

To provide some intuition about LTL, we consider some possible example advice formulae for the problem in Fig. 1 which use the unary operators \square , \diamond , and \bigcirc . The formula $\square\neg\text{at}(\text{nail}1)$ literally means “always, it is not the case that the agent is at the location of nail1”; used as advice it can be taken to say that “at all times the agent *should* not be at nail1.” The formula $\diamond(\text{at}(\text{key}) \wedge \bigcirc(\diamond\text{at}(\text{door})))$ can be understood as “the agent should eventually get to a state where it is at the key and then eventually get to a state where it is at the door.” If $\text{nails} \subset C$ is the set of objects that are nails, we can use $\square\forall(n \in \text{nails}).\neg\text{at}(n)$ to advise that at all times, the agent should not be at a location where there is a nail.

The truth value of an LTL formula φ is determined relative to a sequence $\sigma = \langle s_0, \dots, s_n \rangle$ of states from \mathcal{M} (i.e., the states visited in an episode). The truth of a ground atom at time t is determined by the label of s_t , and the truth values of more complicated formulae are built up according to the formal semantics of LTL (see De Giacomo et al. [4] for more details).

3.2 From LTL to Finite State Automata

Any LTL formula φ can be converted into a *Nondeterministic Finite State Automaton (NFA)* such that a finite sequence of states σ will be accepted by the NFA if and only if σ satisfies φ [4,5]. We can represent the NFA as a directed graph with edges labelled by formulae from \mathcal{L}_Σ , the subset of LTL formulae that does not include temporal operators. Each edge represents a set of NFA transitions, one for each truth assignment satisfying the edge label. Because the NFA is non-deterministic, it may be in multiple states at once. Intuitively, the state(s) that an NFA is in after visiting a sequence of MDP states represent the progress that has been made towards satisfying φ . Reaching an accepting NFA state means that the current trace satisfies the advice formula.

To translate from LTL to automata we use the system developed by Baier and McIlraith [5], which, for computational efficiency, constructs a set \mathcal{N} of small NFAs rather than one potentially very large NFA. \mathcal{N} is considered to accept σ if every NFA in \mathcal{N} accepts σ . For example, Fig. 2 shows the two NFAs generated from $\diamond(\text{at}(\text{key}) \wedge \bigcirc\diamond\text{at}(\text{door})) \wedge \square\forall(n \in \text{nails}).\neg\text{at}(n)$. This advice states that the agent should get the key and then go to the door, and also avoid nails. We now demonstrate how we track the NFA state set for the top NFA. At the beginning of an episode, the agent is in the initial state of the MDP and the NFA is in state u_0 . Thus, the NFA state set is initialized to $\{u_0\}$. This NFA state set will remain constant until the agent reaches an MDP state s' for which

$\text{at}(\text{key}) \in L(s')$ (i.e., the agent gets the key). Since the transition to u_1 is now possible in the NFA, but it is also possible to remain in u_0 because true holds in s' , the NFA state set is updated to $\{u_0, u_1\}$. The state set will then remain constant until the agent reaches the door.

We note that the above procedure may lead to an empty NFA state set on some NFAs and state sequences. For example, notice that in the bottom NFA in Fig. 2, there is no transition to follow when the agent enters a state with a nail. This occurs because once the agent is at a nail, the episode can never satisfy the advice formula regardless of how the rest of the episode proceeds. We call such situations *NFA dead-ends*. Since the advice may still be useful even if it has been violated, we handle NFA dead-ends as follows: if an NFA state set becomes empty, we revert it to the previous set. The NFA in Fig. 2 will therefore continue to suggest that the agent avoid nails even if the agent already failed to do so.

3.3 Background Knowledge Functions

Advice like “get the key” would not be very useful if the agent had no notion of what behaviour might lead to the key. To give advice, we must presuppose that the advised agent has some capacity to understand and apply the advice. To this end, we assume that the agent has a *background knowledge function* $h_B : S \times A \times \text{lit}(\Sigma) \rightarrow \mathbb{N}$, where $h_B(s, a, \ell)$ is an estimate of the number of primitive actions needed to reach the first state s' where the literal ℓ is true (i.e., where $\ell \in L(s')$) if we execute a in s . Intuitively, h_B represents the agent’s prior knowledge — which may not be perfectly accurate — about how to make ground atomic formulae either true or false.

Since h_B only provides estimates with respect to individual ground literals, we believe that for many applications it should be relatively easy to manually define (or learn, e.g. [6]) a reasonable h_B . For example, in the grid world environment, we defined the background knowledge function so that

$$h_B(s, a, \text{at}(c)) = |\text{pos}(\text{agent}, s).x - \text{pos}(c, s).x| + \\ |\text{pos}(\text{agent}, s).y - \text{pos}(c, s).y| + \Delta$$

where $\text{pos}(c, s)$ provides the coordinates of c in state s and Δ is equal to -1 if the action a points toward c from the agent’s position and 1 if it does not. Hence, if the agent is three locations to the left of the key, $h_B(s, \text{right}, \text{at}(\text{key}))$ will return 2 , even if there is a wall in the way. Furthermore, we defined $h_B(s, a, \neg\text{at}(c))$ to be equal to 1 if $h_B(s, a, \text{at}(c)) = 0$ and 0 otherwise.

Given any background knowledge function h_B , we can construct a function $h : S \times A \times \mathcal{L}_\Sigma \rightarrow \mathbb{N}$ that extends h_B to provide an estimate of the number of primitive actions needed to satisfy an arbitrary formula $\varphi \in \mathcal{L}_\Sigma$. We recursively define $h(s, a, \varphi)$ as follows:

$$h(s, a, \ell) = h_B(s, a, \ell) \text{ for } \ell \in \text{lit}(\Sigma) \\ h(s, a, \psi \wedge \chi) = \max\{h(s, a, \psi), h(s, a, \chi)\} \\ h(s, a, \psi \vee \chi) = \min\{h(s, a, \psi), h(s, a, \chi)\}$$

In the next sections, we describe how to drive the agent’s exploration using h .

3.4 Advice-Based Action Selection

Suppose the agent is at a state s in the MDP, with NFA state sets $q^{(0)}, \dots, q^{(m)}$. Intuitively, following the advice in s involves having the agent take actions that will move the agent through the edges of each NFA towards its accepting states. To this end, we begin by identifying *useful* NFA edges which may actually lead to progress towards the accepting states. Let us write $(q, \beta, q') \in \delta^{(i)}$ if there is an edge from q to q' that is labelled by the formula β in the i th NFA. We say that an edge $(q, \beta, q') \in \delta^{(i)}$ is useful if q is not an accepting state and there exists a path in the NFA from q' to an accepting state that does not have q along it. We then let $useful(q^{(i)})$ denote the set of all useful edges that are from NFA states in $q^{(i)}$. We now define the *advice guidance* formula $\hat{\varphi}$ as follows:

$$\hat{\varphi} \stackrel{\text{def}}{=} \bigvee_{i=0}^m \left[\bigvee_{(q, \beta, q') \in useful(q^{(i)})} \text{to_DNF}(\beta) \right]$$

where the function $\text{to_DNF} : 2^{\mathcal{L}_\Sigma} \rightarrow \mathcal{L}_\Sigma$ converts the formula β to disjunctive normal form. Notice that the formula $\hat{\varphi}$ will be satisfied by any action that achieves one of the formulas needed to transition over a useful edge. We define $\hat{h}(s, a) = h(s, a, \hat{\varphi})$ which can be used to rank how close each action is to making progress in satisfying the advice guidance formula $\hat{\varphi}$.

In addition to guidance, it is important to, if possible, avoid NFA dead-ends. To do so, we define the *advice warning* formula $\hat{\varphi}_w$ as follows:

$$\hat{\varphi}_w \stackrel{\text{def}}{=} \bigwedge_{i=0}^m \left[\bigvee_{q \in q^{(i)} \text{ and } (q, \beta, q') \in \delta^{(i)}} \text{to_DNF}(\beta) \right]$$

and use it to define the set $W(s) \stackrel{\text{def}}{=} \{a \in A(s) : h(s, a, \hat{\varphi}_w) \neq 0\}$. The idea is that W contains those actions which the evaluation function predicts will lead to NFA dead-ends. In the next section, we describe how we recommend the agent to disfavor actions from W while also being guided by \hat{h} .

3.5 Incorporating Advice in R-MAX

Model-based Reinforcement Learning solves MDPs by learning the transition probabilities and rewards. The R-MAX family of algorithms [2] are model-based methods that explore the environment by assuming that unknown transitions give maximal reward R_{max} . In practice, if R_{max} is big enough, this means that the agent plans towards reaching the closest unknown transition.

We propose a simple variant of an R-MAX algorithm that can take advantage of the advice. Instead of planing towards *the closest* unknown transition, we plan towards the closest unknown transition that is not in W and has minimum \hat{h} -value. If every reachable unknown transition is in W , then we just go to the closest unknown transition with minimum \hat{h} -value. This planning step can be done using LAO* [7].

```

1 Function Rmax_with_advice( $S, A, L, h_B, \varphi_{\text{advice}}, C, N$ )
2    $T_{\text{unknown}} \leftarrow \emptyset; \hat{p} \leftarrow \text{initialize\_empty\_model}();$ 
3   for  $s, a \in S \times A$  do
4      $n(s, a) \leftarrow 0;$ 
5      $T_{\text{unknown}} \leftarrow T_{\text{unknown}} \cup (s, a);$ 
6    $t \leftarrow 0; \pi \leftarrow \emptyset; s \leftarrow \text{get\_initial\_state}();$ 
7   while  $t < N$  do
8      $t \leftarrow t + 1;$ 
9     if  $\text{is\_terminal\_state}(s)$  or  $\text{cannot\_be\_reached}(s, \hat{p}, T_{\text{unknown}})$  then
10       $s \leftarrow \text{get\_initial\_state}();$ 
11     if  $s \notin \pi$  then
12        $\pi \leftarrow \text{policy\_towards\_min\_heuristic}(\hat{p}, \varphi_{\text{advice}}, T_{\text{unknown}}, L, h_B);$ 
13        $a \leftarrow \text{sample\_action}(\pi(s));$ 
14        $s', r \leftarrow \text{execute\_action}(s, a);$ 
15        $\hat{p} \leftarrow \text{update\_model}(\hat{p}, s, a, s', r);$ 
16        $n(s, a) \leftarrow n(s, a) + 1;$ 
17       if  $n(s, a) = C$  then
18          $T_{\text{unknown}} \leftarrow T_{\text{unknown}} \setminus (s, a);$ 
19        $s \leftarrow s';$ 
20   return  $\text{compute\_optimal\_policy}(\hat{p});$ 

```

Algorithm 1. R-Max with Advice algorithm.

Algorithm 1 shows the pseudo-code of our R-MAX variant. Its inputs are the MDP states S , action set A , labelling function L , background knowledge function h_B , the advice formula φ_{advice} , the number $C \geq 1$ of times that a transition has to be tried before being labeled as *known*, and the number of actions N that the agent can execute during training. The algorithm starts by marking every state-action as unknown. It also defines an auxiliary variable $n(s, a)$ that counts how many times the transition (s, a) has been tried and an empty model of the estimated environment's *transition probabilities* \hat{p} . In every iteration of the main loop, the agent selects the next action a to execute using a partial policy π , which encodes the optimal way to reach the closest unknown transition with minimum \hat{h} -value that is not in W (as previously described). Then, the agent executes a and receives a reward r and the next state s' from the environment. The probabilities in the estimated model \hat{p} are updated to reflect this observation of the transition (s, a, r, s') . Whenever a transition (s, a) is tried C times, it is removed from T_{unknown} . If a terminal state is reached or it is not possible to reach unknown transitions from the current state, the environment is restarted. After executing N actions, the algorithm determines the optimal policy with respect to \hat{p} (using, for instance, value iteration) and returns it. For a deterministic MDP, whenever N is sufficiently large, that policy will also be optimal for the MDP.

Theorem 1. *Given an MDP with a deterministic transition function, there exists a number N_0 so that Algorithm 1 converges to an optimal policy if $N \geq N_0$.*

Proof (sketch). Since the MDP is deterministic, after an action is attempted even one time in a state, the estimate of the probabilities (all 0 or 1) over the outcomes of that state-action pair will be exact. So there will be exact estimates for all transitions the algorithm considers *known* (even if $C = 1$). As each episode ends after a finite number of steps, eventually all transitions reachable from the initial state become known, since the agent plans to visit unknown transitions as long as there are any (regardless of the advice). By that point, the MDP can be solved exactly by the agent, and so the optimal policy can be found.

For a non-deterministic MDP, we expect that our approach would converge to a near-optimal policy in the same sense that R-MAX does [2], but further investigation is needed.

4 Evaluation and Discussion

We tested our approach using various pieces of advice on grid world problems of the sort defined in Sect. 2, using the signature and background knowledge functions described in Sects. 2 and 3. In addition to the domain elements of walls, keys, and nails described earlier, some problems also had *cookies* and *holes*. When the agent reaches a location with a cookie, it gets a reward of +10 and the cookie disappears. For reaching a hole, the reward is -1000 and the episode ends. As a baseline, we reported the performance of standard R-MAX, which does not use any form of advice. To measure performance, we evaluated the agent’s policy every 100 training steps. At training time, the agent explored the environment to learn a good model. At test time, we evaluated the best policy that the agent could compute using its current model (ignoring the unknown transitions). We used a discount factor of 1 in all experiments.

Figure 4a shows the median performance (over 20 independent trials) in a 25×50 version of the motivating example in Fig. 1. Our approach allows the agent to quickly find a policy that follows the advice and then slowly converges to an optimal policy. This is the case even with the deliberately misleading advice N, which told the agent to go to every nail. As the quality of the advice decreases,

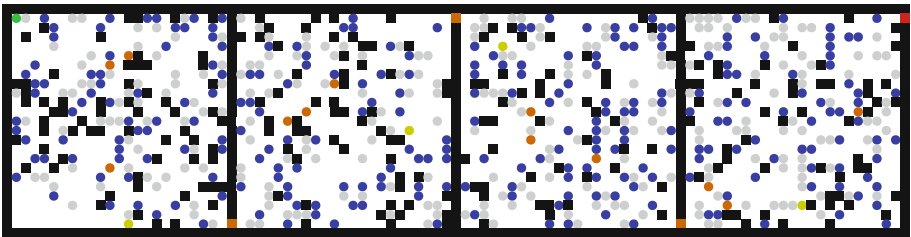


Fig. 3. An example randomly generated grid world map, containing an agent (●), walls (■), nails (■), holes (●), keys (●), cookies (●), doors (■), and a final door (■). (Color figure online)

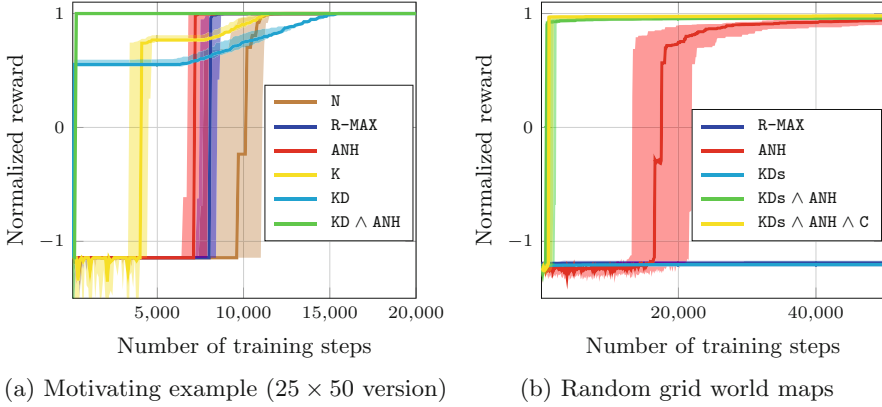


Fig. 4. In (a) and (b), reward is normalized so that 1 represents the best policy found on each map. The shaded areas represent the first and third quartiles. The formulae in the legends are explained by Table 1. Graphs are best viewed in colour. (Color figure online)

Table 1. Abbreviations for legends in Fig. 4

Abbreviation	Advice formula (and informal meaning)
R-MAX	<i>No advice</i>
	Standard R-MAX
C	$\forall(c \in \text{cookies}). \diamond \text{at}(c)$
	Get all the cookies
ANH	$\square(\forall(x \in \text{nails} \cup \text{holes}). \neg \text{at}(x))$
	Avoid nails and holes
K	$\diamond \text{at}(\text{key})$
	Get the key.
KD	$\diamond(\text{at}(\text{key}) \wedge \bigcirc \diamond(\text{at}(\text{door})))$
	Get the key and then go to the door
KDs	$\forall(k \in \text{keys}). \diamond(\text{at}(k) \wedge \bigcirc \diamond(\exists(d \in \text{doors}). \text{at}(d)))$
	For every key in the map, get it and then go to a door
N	$\forall(x \in \text{nails}). \diamond \text{at}(x)$
	Go to every nail

the agent’s initial performance also does. We note that R-MAX, without advice, has poor initial performance, but converges faster to an optimal policy than when using the more detailed advice of K or KD (see Table 1 for what these formulae are). As such, there is a trade-off between quickly learning the model (exploring nearby areas) and moving towards promising states suggested by advice.

We also randomly generated 10 grid world problems (Fig. 3 shows one of them), each consisting of a sequence of four 25×25 rooms with doors between consecutive rooms. The agent always starts in the upper-left corner of the leftmost room and the red (i.e., goal) door is on the rightmost wall. Each space within a room had a $1/9$ probability of being each of a nail, hole, or wall. A key and three cookies were randomly placed in each room among the remaining empty spaces, such that each key, door, and cookie was reachable from the starting location. These grid world maps are challenging because there is sparse reward and it is fairly easy to die.

Figure 4b shows the performance over the 10 maps using four different pieces of advice. We ran 5 trials per piece of advice on each map; the graph reports the median performance across both trials and maps. Without advice, the agent was never able to reach the last door in 50 thousand training steps. Providing advice that the agent should get keys and go to doors (KDs) was also not enough, because the agent always fell in a hole before reaching its target. Stronger results were seen with safety-like advice to avoid holes and nails (ANH), and the best performance was seen when this safety-like advice was combined with advice that guides agent progress (KDs \wedge ANH and KDs \wedge ANH \wedge C).

These results are encouraging. Firstly, they show the potential for advice to play a key role in scaling Reinforcement Learning. In particular, some problems are just too hard to expect RL to find good policies without the guidance provided by advice (e.g., Fig. 4b). Secondly, they show that even well-intended advice could result in sub-optimal behaviour if it were treated as a hard constraint. For instance, the initial performance of KD in Fig. 4a is a policy that satisfies the advice in an optimal number of steps, but gets sub-optimal reward (due to the nails). This further highlights the need for techniques like ours that are based on advice (guidance) instead of constraints (pruning). Still, our experiments are limited to a deterministic setting, and investigating the role of advice in non-deterministic domains is an important future work direction.

5 Related Work

The idea (and recognition of the importance) of constructing agents that can take advice of some sort dates back to John McCarthy’s hypothetical *advice taker* system [8]. For MDPs, several works have proposed agents that accept *hard constraints* in the form of linear temporal logic safety constraints (e.g. [9, 10]) or high level task specifications (e.g. [11, 12]). Such constraints differ from the notion of advice studied in this paper. In particular, such hard constraints eliminate certain traces from consideration, potentially eliminating optimal policies. In contrast, advice only suggests behaviour and as such is more appealing when we want to provide guidance without pruning alternative behaviours.

In RL, the use of “advice” has been mainly focused on human *feedback* and *critique*. Human feedback allows an external observer to guide an agent (while it solves an MDP) by giving it positive and negative feedback (e.g. [13, 14]), whereas critique proposes alternative actions to some states in a trace execution

(e.g. [15,16]). In this work, we study a different modality of advice, which is given offline (prior to the agent’s first interaction with the environment).

Maclin and Shavlik [17] were the first to propose using offline advice in RL. They defined a procedural programming language for encoding the advice. This language allows the user to recommend primitive actions using *If-Then* rules and loops. Since then, several extensions to this work have been done (e.g. [18,19]). Recently, Krening et al. [20] proposed a substantially different approach. They grounded natural language advice into a list of (*object,action*) recommendations (or warnings). The idea is to encourage (or discourage) the agent to perform a primitive action when it interacts with particular classes of objects. In contrast to those works, our advice language supports specification of temporally extended advice in terms of properties of states. For many simple pieces of LTL advice, such as *eventually get the key*, it is unclear how to express them as primitive action recommendations using the previous advice languages.

In another recent work, Andreas et al. [21] proposed *policy sketches*. A policy sketch is a sequence of sub-goals given by the user to solve a task. In a multi-task setting, Andreas et al. show how to use the sketch to compose (previously learned) policies for each sub-task to solve a novel task. However, their advice language is quite constrained in comparison with LTL advice. In particular, advice such as *avoid nails* or *get cookies* cannot be expressed as a sequence of sub-goals (unless some unnecessary order is imposed to sequentialize the advice).

6 Concluding Remarks

This work studied how to exploit linguistically expressed advice in Reinforcement Learning. Advice has the distinctive feature of being a recommendation, but not a task specification. As such, techniques for exploiting advice should provide guidance without pruning alternative, and possibly optimal, behaviours. Three main challenges arise when doing so. First, we need a common vocabulary to communicate the advice to the agent. We handled this by defining a *signature* over the states that is understandable by both the agent and the user. Then, sophisticated pieces of advice were constructed over the signature using LTL. Second, the agent needs some sort of background knowledge to be able to *follow* the advice. In this work, we introduced a background knowledge function for that purpose. Finally, the agent needs a way to reason about how and when to use the advice. Our approach constantly looks to *as soon as possible* reach promising states for advancing towards satisfying the advice. However, getting obsessed with following the advice might result in slower convergence to an optimal policy (as discussed in Sect. 4). We encourage future works to pay special attention to these three dimensions when advising RL agents.

Acknowledgement. This research was supported by NSERC and CONICYT. A preliminary *non-archival* version of this work was presented at RLDM (2017).

References

1. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57 (1977)
2. Brafman, R., Tennenholtz, M.: R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.* **3**, 213–231 (2002)
3. Bacchus, F., Kabanza, F.: Using temporal logics to express search control knowledge for planning. *Artif. Intell.* **116**(1–2), 123–191 (2000)
4. De Giacomo, G., Masellis, R.D., Montali, M.: Reasoning on LTL on finite traces: insensitivity to infiniteness. In: AAI, pp. 1027–1033 (2014)
5. Baier, J., McIlraith, S.: Planning with first-order temporally extended goals using heuristic search. In: AAI, pp. 788–795 (2006)
6. Peng, B., MacGlashan, J., Loftin, R., Littman, M., Roberts, D., Taylor, M.: A need for speed: adapting agent action speed to improve task learning from non-expert humans. In: AAMAS, pp. 957–965 (2016)
7. Hansen, E., Zilberstein, S.: LAO*: a heuristic search algorithm that finds solutions with loops. *Artif. Intell.* **129**(1–2), 35–62 (2001)
8. McCarthy, J.: Programs with common sense. RLE and MIT Computation Center (1960)
9. Lacerda, B., Parker, D., Hawes, N.: Optimal and dynamic planning for markov decision processes with co-safe LTL specifications. In: IROS, pp. 1511–1516 (2014)
10. Wen, M., Topcu, U.: Probably approximately correct learning in stochastic games with temporal logic specifications. In: IJCAI, pp. 3630–3636 (2016)
11. Andre, D., Russell, S.J.: Programmable reinforcement learning agents. In: NIPS, pp. 1019–1025 (2000)
12. Shapiro, D., Langley, P., Shachter, R.: Using background knowledge to speed reinforcement learning in physical agents. In: AA, pp. 254–261 (2001)
13. Isbell, C., Shelton, C.R., Kearns, M., Singh, S., Stone, P.: A social reinforcement learning agent. In: AA, pp. 377–384 (2001)
14. Knox, W.B., Stone, P.: Tamer: training an agent manually via evaluative reinforcement. In: ICDL, pp. 292–297 (2008)
15. Judah, K., Roy, S., Fern, A., Dietterich, T.G.: Reinforcement learning via practice and critique advice. In: AAI, pp. 481–486 (2010)
16. Griffith, S., Subramanian, K., Scholz, J., Isbell, C., Thomaz, A.L.: Policy shaping: integrating human feedback with reinforcement learning. In: NIPS (2013)
17. Maclin, R., Shavlik, J.: Creating advice-taking reinforcement learners. *Mach. Learn.* **22**(1–3), 251–281 (1996)
18. Maclin, R., Shavlik, J., Torrey, L., Walker, T., Wild, E.: Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In: AAI, pp. 819–824 (2005)
19. Kunapuli, G., Odom, P., Shavlik, J.W., Natarajan, S.: Guiding autonomous agents to better behaviors through human advice. In: ICDM, pp. 409–418 (2013)
20. Krening, S., Harrison, B., Feigh, K., Isbell, C., Riedl, M., Thomaz, A.: Learning from explanations using sentiment and advice in RL. *IEEE Trans. Cogn. Dev. Syst.* **9**(1), 44–55 (2016)
21. Andreas, J., Klein, D., Levine, S.: Modular multitask reinforcement learning with policy sketches. In: ICML, pp. 166–175 (2017)



Deep Super Learner: A Deep Ensemble for Classification Problems

Steven Young^{1(✉)}, Tamer Abdou^{1,2(✉)}, and Ayse Bener^{1(✉)}

¹ Data Science Laboratory, Ryerson University, Toronto, ON M5B 2K3, Canada
{[steven.young](mailto:steven.young@ryerson.ca), [tamer.abdou](mailto:tamer.abdou@ryerson.ca), [ayse.bener](mailto:ayse.bener@ryerson.ca)}@ryerson.ca
² Faculty of Science, Arish University, North Sinai 45516, Egypt

Abstract. Deep learning has become very popular for tasks such as predictive modeling and pattern recognition in handling big data. Deep learning is a powerful machine learning method that extracts lower level features and feeds them forward for the next layer to identify higher level features that improve performance. However, deep neural networks have drawbacks, which include many hyper-parameters and infinite architectures, opaqueness into results, and relatively slower convergence on smaller datasets. While traditional machine learning algorithms can address these drawbacks, they are not typically capable of the performance levels achieved by deep neural networks. To improve performance, ensemble methods are used to combine multiple base learners. Super learning is an ensemble that finds the optimal combination of diverse learning algorithms. This paper proposes deep super learning as an approach which achieves log loss and accuracy results competitive to deep neural networks while employing traditional machine learning algorithms in a hierarchical structure. The deep super learner is flexible, adaptable, and easy to train with good performance across different tasks using identical hyper-parameter values. Using traditional machine learning requires fewer hyper-parameters, allows transparency into results, and has relatively fast convergence on smaller datasets. Experimental results show that the deep super learner has superior performance compared to the individual base learners, single-layer ensembles, and in some cases deep neural networks. Performance of the deep super learner may further be improved with task-specific tuning.

Keywords: Deep learning · Neural network · Ensemble learning

1 Introduction

Deep learning is a machine learning method that uses layers of processing units where the output of a layer cascades to be the input of the next layer and can be applied to either supervised or unsupervised learning problems [1, 2]. Deep neural networks (DNN) is an architecture of deep learning that typically

The work in this paper is conducted in the Capstone Project Course of the Certificate in Data Analytics, Big Data, and Predictive Analytics at Ryerson University.

has many connected units arranged in layers of varying sizes with information being fed forward through the network. DNN have been successfully applied to fields such as computer vision and natural language processing, having achieved accuracy rates similar or superior to humans in classification [3]. For example, Ciresan et al. using DNN achieved an error rate half the rate of humans in recognizing traffic signs. The multiple layers of a DNN allow for varying levels of abstraction and the cascade between the layers enables the extraction of features from lower to higher level layers to improve performance [4]. However, DNN also have drawbacks, listed below:

- DNN have many hyper-parameters, which are parameters where their values are set prior to training as opposed to parameter values that are set via training, that interact with each other in their relation to performance. Numerous hyper-parameters, together with infinite architectures, makes tuning of hyper-parameter and architecture difficult [5].
- With a large number of processing units, tracing through a DNN to understand the reasoning for classifications is difficult, leading to DNN being treated as black boxes [6].
- DNN typically require very large amounts of data to train and do not converge as fast, with respect to sample size, as traditional machine learning algorithms [7].

Traditional machine learning algorithms, on the other hand, are relatively simple to tune and their output may provide interpretable results leading to a deeper understanding of the problem, though they tend to underperform DNN in terms of accuracy.

The remainder of this paper is organized as follows: Sect. 1 introduces the motivation and background for this paper, Sect. 2 presents the overall procedure of the DSL approach, Sect. 3 describes the methodology of the experiment, Sect. 4 presents the results of a comparison of the performance of the DSL to the individual base learners and a selection of ensembles and DNN on various problems, and Sect. 5 concludes and describes future work.

1.1 Motivation

Given the drawbacks of DNN and the poor performance of traditional machine learning algorithms in some domains and/or prediction tasks, this paper investigates whether traditional machine learning algorithms can be used to address the drawbacks of DNN and achieve levels of performance comparable to DNN. A new ensemble method, named here as Deep Super Learner (DSL), seeks to have simplicity in setup, interpretability of results, fast convergence on small and large datasets with the power of deep learning.

1.2 Ensemble Methods

Ensemble methods are techniques that train multiple learning algorithms, which in combination yields significantly higher accuracy results than a single

learner [8]. Common methods include boosting, bagging, stacking, and a combination of base learners. Each of these methods are tested for performance in this paper. Boosting takes a model trained on data and incrementally constructs new models that focus on the errors in classifying made by the previous model. An example is XGBoost, which is an efficient implementation of gradient boosting decision trees [9]. Bagging involves training models on random subsamples and then each model votes with equal weight on the classification. Random forest uses a bagging approach to enable the selection of a random set of features at each internal node [10]. Stacking takes the output of a set of models and feeds them into another algorithm that combines them to make the final predictions. Any arbitrary set of base learners and combiner algorithm can be used. Combination takes the predictions of the models and combines them with a simple or weighted average. Super learner is a combination method that finds optimal weights to use when calculating the final prediction [11].

Super learning is an ensemble method proposed by Van der Laan et al. that optimizes the weights of the base component learners by minimizing a loss function given cross-validated output of the learners. Super learning finds the optimal set of weights for the learners and guarantees that performance will be at least as good as the best base learner [11]. The proposed algorithm, DSL, is an extension of the super learner ensemble.

When constructing an ensemble, having diversity among the component learners is essential for performance and a strong generalization ability [12]. The super learner adapts to various problems given a set of diverse base learners since the weights of the components are optimized for the problem as different learners perform differently on different problems. There is also flexibility in the set of base learners to use depending on requirements or constraints as dictated by the problem or computational resources.

1.3 Related Work

Very little research has been conducted on using traditional machine learning in a deep learning architecture. Zhou and Feng describe a tree-based deep learning architecture [5]. However, the use of only decision tree based algorithms and the use of a simple average to combine the results of the base learners may limit the ultimate performance of this approach and its adaptability to diverse sets of problems. Farrelly tested an architecture using traditional learners arranged in three hidden layers [7]. It is unclear if the implementation allowed iteration to continue the deep learning. Accuracy results from this architecture did not outperform the super learner.

2 The Proposed Deep Super Learner Approach

Deep learning consists of a layer by layer processing of features with a cascading hierarchy structure where the information processed by a layer is fed to the next layer for further processing. The deep super learner essentially uses a super learning ensemble for each layer. The overall training process and hyper-parameter

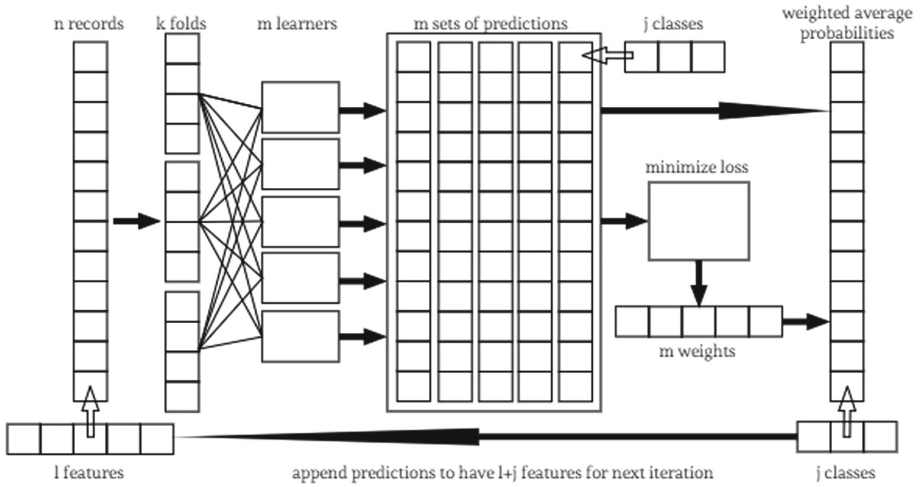


Fig. 1. Overall procedure of DSL with j classes, k folds, l features, m learners, n records

values used are described below (see Fig. 1 and Algorithm 1). Let there be j classes, k folds, l features, m base learners, and n records in the training set.

1. Cross-validation is used to generate out-of-sample predictions for the entire training set. Split the training set into k equal size, n/k , mutually exclusive groups to be used as validation sets. For each of the k validation sets form k folds where all the $n - n/k$ training records not in the validation set are used for training for that validation set. The number of folds can impact the degree of under and over-fitting of the algorithm as well as runtime since with a higher number of folds each fold contains a larger portion of the training data to train on leading to a better fit, all else being equal. However, with more folds there is a greater overlap in the training data between the folds leading to potential over-fitting and more runtime. Three folds are used in this paper as experimentation showed three folds to be a good balance of fit and runtime.
2. Build and train each of the m base learning algorithms on each fold. Each model outputs class probabilities for each fold. There are $k * m$ trained models. Bring the predictions on each validation set together for each learner to obtain predictions for the entire training set. This paper uses five base learners: logistic regression, k-nearest neighbors, random forest, extremely randomized trees, and XGBoost. Support vector machines was also tested but due to runtime is excluded here. Logistic regression, k-nearest neighbors, and random forest are used as they represent three different classification models with different philosophies and performance on different datasets [13]. Extremely randomized trees and XGBoost are used for additional diversity within tree-based approaches. The hyper-parameters of these learners are described below.

3. Optimize a loss function to find the linear combination of predictions from each of the m learners to minimize the objective against the true values for the training data. Save the value of the loss function and the optimized weights. Log loss, which is a convex function and therefore convex optimization is sufficient, is used here. Log loss is commonly used when calibration is important [14].
4. With the optimized weights, calculate the weighted average of the predictions across learners to obtain overall predictions for each record.
5. (Optional) Re-train each of the models on the entire training set to get m trained models. Super learning as described by Van der Laan et al. requires this step [11]. However, with a sufficient number of cross-validation folds, as described above, the trained models will be trained on a sufficient portion of the training data where additional training data does not improve goodness of fit. Re-training may also be computationally expensive. This step is recommended when the number of folds is low or when making predictions is more computationally expensive than training, such as for k-nearest neighbors. This step is not performed for this paper as experimentation showed no significant difference in performance on the tested datasets.
6. Append the overall predictions to the original training data. For example, if there are j classes, step 4 produces a j -dimensional vector containing class probabilities for each record. These vectors are concatenated as additional features to the original l -dimensional feature vectors for the records, resulting in a total of $l + j$ features.
7. Feed the training data augmented with the predictions through the steps above. Repeat this process until the optimized loss value no longer decreases with each iteration. Save the number of iterations after which the training process ends.

To make predictions on unseen test data, pass the data in its entirety through a similar process using each of the models trained and weights optimized at each iteration. If the models are trained on the entire training set, use these m models for each iteration. If the models are trained on the k folds, use each model trained on each fold to make predictions on all the unseen data and average across the k models to get predictions for each of the m learners. Using the optimum weights for the m learners found during training for the iteration, calculate the overall weighted average predictions for the iteration. Append the predictions to the original test data as additional features. Repeat the process for the same number of iterations used in training.

3 Methodology

The hyper-parameters and architectures for the DSL, base learners, benchmark ensembles, and benchmark DNN described below are kept constant between datasets. When necessary, adjustments are made for the different dimensionality of the datasets.

```

for iteration in 1 to max iterations do
  Split data into k folds each with train and validate sets;
  for each fold in k folds do
    for each learner in ensemble do
      Train learner on train set in fold;
      Get class probabilities from learner on validate set in fold;
      Build predictions matrix of class probabilities;
    end
  end
  Get weights to minimize loss function with predictions and true labels;
  Get average probabilities across learners by multiplying predictions with weights;
  Get loss value of loss function with average probabilities and true labels;
  if loss value is less than loss value from previous iteration then
    Append average probabilities to data;
  else
    Save iteration;
    Break for;
  end
end

```

Algorithm 1. A Pseudo-code of the Proposed Approach, DSL

3.1 Base Learners and Benchmark Ensembles

The same five base learners used in DSL are also tested individually and in the benchmark ensembles using identical hyper-parameter values. Hyper-parameter values are set to balance performance and runtime. If a hyper-parameter of a learner is not listed below, in Table 1, default values of the implementation of the algorithm are used.

Since random forest, extremely randomized trees, and XGBoost are themselves ensembles, three additional ensembles are tested for comparison: a simple equal weighted average of the base learners, a stacked ensemble where the output of the base learners is fed into XGBoost, and a single-layer super learner.

Table 1. Summary of hyper-parameters for base learning algorithms

Base learner	Hyper-parameters
Logistic regression	N/A
k-Nearest neighbors	Neighbors: 11
Random forest	Trees: 200; Depth: unlimited
Extremely randomized trees	Trees: 200; Depth: unlimited; Max features when splitting: 1
XGBoost	Trees: 200; Max depth: 3; Row subsampling: 0; Column subsampling: 0; Learning rate: 1

Table 2. Summary of architecture and hyper-parameter values used for benchmark deep neural networks.

Architecture	Hyper-parameters (Multi-layer perceptron)	Hyper-parameters (Convolutional neural network)
Convolutional layer	N/A	Filters: 32; Kernel size: 5 or (5, 5); Activation: RELU; Weight constraint: 4
Max pooling layer	N/A	Pool size: 2 or (2, 2)
Convolutional layer	N/A	Filters: 16; Kernel size: 3 or (3, 3); Activation: RELU; Weight constraint: 4
Max pooling layer	N/A	Pool size: 2 or (2, 2)
Dropout regularization	N/A	Drop rate: 0.2
Dense layer	Nodes: 128; Activation: RELU; Weight constraint: 4	Nodes: 128; Activation: RELU; Weight constraint: 4
Dense layer	Nodes: 64; Activation: RELU; Weight constraint: 4	Nodes: 64; Activation: RELU; Weight constraint: 4
Output layer	Nodes: number of classes; Activation: Softmax	Nodes: number of classes; Activation: Softmax
Optimizer: Adam	Learning rate: 0.001; Learning rate decay: $\text{Learning rate}/\sqrt{\text{Max epochs}}$	Learning rate: 0.001; Learning rate decay: $\text{Learning rate}/\sqrt{\text{Max epochs}}$
Batch size	200	200
Max epochs	50	50
Validation split	0.2	0.2
Early stopping patience	3	3

3.2 Benchmark Deep Neural Networks

DNN are used to establish benchmarks for performance. Some hyper-parameter tuning through experimentation is performed to achieve performance indicative of the capabilities of DNN. In Table 2, two DNN architectures are tested and described: a multi-layer perceptron (MLP) and a convolutional neural network (CNN).

3.3 Datasets

Sentiment Classification

The IMDB Movie reviews sentiment classification dataset contains 25,000 reviews for training and 25,000 for testing. The reviews are labelled as positive or negative [15]. The 2,000 most frequent words in the set are used to calculate the term frequency-inverse document frequency (TF-IDF) matrix.

Image Categorization

The MNIST database of handwritten digits is a commonly used dataset to test the performance of computer vision algorithms. It includes a training set of 60,000 images and a test set of 10,000 images of handwritten digits 0 to 9. The images are 28 pixels by 28 pixels in greyscale [16].

3.4 Performance Measures

Two metrics are used to evaluate the performance of the learning algorithms. One is *Accuracy*, which is the proportion of correctly classified records, and the other is *LogLoss*. Both *Accuracy* and *LogLoss* formulas are shown in Eqs. 1 and 2, respectively.

$$Accuracy = \frac{\sum_{x=1}^n \sum_{y=1}^j f(x, y)C(x, y)}{n} = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

Where n denotes the number of instances, j the number of classes, $f(x, y)$ the actual probability of instance x to be of class y . $C(x, y)$ is one if and only if y is the predicted class of x , otherwise $C(x, y)$ is zero. *Accuracy* is equivalently defined in terms of the confusion matrix, where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

$$LogLoss = \frac{-\sum_{y=1}^j \sum_{x=1}^n f(x, y)\log(p(x, y))}{n} \quad (2)$$

Where $f(x, y)$ is defined as above and $p(x, y)$ is the estimated probability of instance x is class y . Minimizing *LogLoss*, also known as cross entropy, is equivalent to maximizing the log likelihood of observing the data from the model. Both *Accuracy* and *LogLoss* are commonly used performance measures in machine learning [14].

4 Results

The following results are averaged across 10 repetitions of the experiment.

4.1 Sentiment Classification

Log loss and accuracy results of DSL, base learners, benchmark ensembles, and benchmark DNN on the IMDB sentiment classification dataset are shown in Table 3.

The DSL achieved statistically significantly lower loss and higher accuracy than all other algorithms. Since the TF-IDF matrix does not convey spatial or sequential relationships, DNN architectures like CNN may not be expected to perform as well on this test. The MLP, like DSL here, is set up to be more general purpose yet is outperformed by DSL. DSL outperforming a single-layer super learner indicates adding depth to the algorithm improves performance. Figure 2 shows the performance of DSL on the IMDB test data by iteration. DSL automatically terminates after the third iteration as log loss, calculated from the k folds during training, ceases to continue improving.

Table 3. Comparison of log loss and accuracy on IMDB test data

Test	Log loss	Accuracy
Deep Super Learner (DSL)	0.28	88.22%
Multi-layer perceptron	0.29	87.53%
Super learner	0.30	86.59%
XGBoost stack ensemble	0.31	87.19%
Convolutional neural network	0.31	86.40%
Logistic regression	0.32	87.78%
XGBoost	0.39	84.45%
Simple average ensemble	0.42	86.57%
Random forest	0.46	84.27%
Extremely randomized trees	0.60	79.20%
k-Nearest neighbors	0.72	68.22%

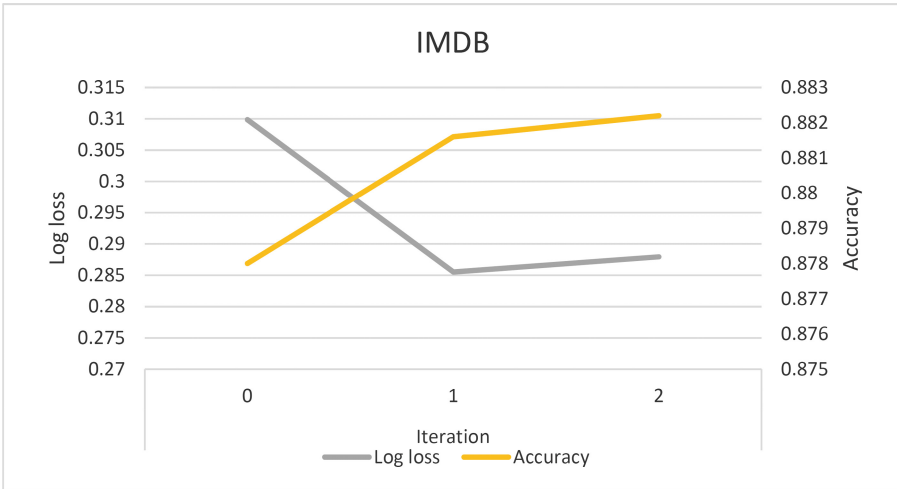


Fig. 2. Log loss and accuracy by iteration of the DSL on IMDB test data.

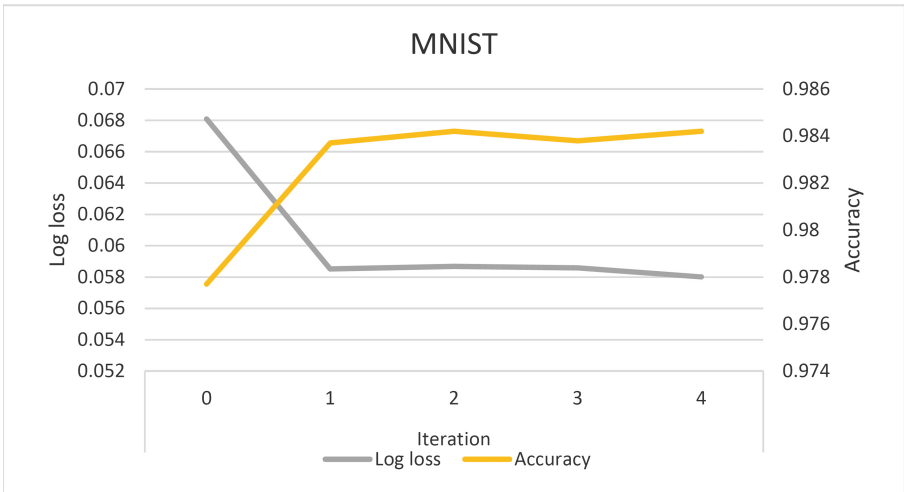
4.2 Image Categorization

Log loss and accuracy results of DSL, base learners, benchmark ensembles, and benchmark DNN on the MNIST handwritten digits dataset are shown in Table 4.

The DSL achieved statistically significantly lower loss and higher accuracy than all algorithms except for CNN. The design of CNN make them well suited to image processing. Again, DSL outperformed MLP and super learner showing the advantages of diversity in learners and depth. The order of the base learners by performance differs between the two datasets showing the importance of

Table 4. Comparison of log loss and accuracy on MNIST test data.

Test	Log loss	Accuracy
Convolutional neural network	0.03	99.17%
Deep Super Learner (DSL)	0.06	98.42%
Super learner	0.07	97.82%
Multi-layer perceptron	0.07	97.85%
XGBoost stack ensemble	0.08	98.24%
XGBoost	0.08	97.65%
Simple average ensemble	0.18	97.65%
Random forest	0.24	97.00%
k-Nearest neighbors	0.26	96.68%
Logistic regression	0.27	92.55%
Extremely randomized trees	0.43	95.87%

**Fig. 3.** Log loss and accuracy by iteration of the DSL on MNIST test data.

including a diverse set of learners when addressing various problems and the value of optimizing the component weights. Figure 3 shows the performance of DSL on the MNIST test data by iteration. DSL automatically terminates after the fifth iteration as log loss, calculated from the k folds during training, ceases to continue improving.

4.3 Runtime

All algorithms are implemented in Python using the scikit-learn library for logistic regression, k-nearest neighbors (KNN), random forest, and extremely randomized

trees, XGBoost library for XGBoost, SciPy for the convex optimizer, and Keras with a TensorFlow backend for MLP and CNN. Experiments are run on a desktop with an Intel Core i7-7700 with 16 GB of RAM. DSL on IMDB converges after three iterations running for a total of 50 min, 46 of which are spent in the prediction phase of KNN. MLP on IMDB converges after two epochs running for one minute. CNN on IMDB converges after six epochs running for a total of seven minutes. DSL on MNIST converges after five iterations, running for a total of 86 min, 70 of which are spent in the prediction phase of KNN. MLP on MNIST converges in 12 epochs running for two minutes. CNN on MNIST converges in 12 epochs running for a total of 12 min. DSL is inherently parallel across component learners. With optimized parallel processing and selection of base learners, the runtime of DSL can be dramatically reduced.

4.4 Threats to Validity

Threats to internal validity include errors in the experiments and implementations. While the experiments and implementations were carefully double checked, errors are possible. Threats to external validity include whether the results in this paper generalize to other datasets, sets of base learners and architectures. While the tests include two rather different datasets, only one set of base learners and deep architecture were tested. Applying the methods described here to additional datasets as well as varying base learners and architecture will reduce this threat. Threats to construct validity include the appropriateness of the benchmark algorithms and evaluation metrics. For the most part, benchmark ensembles outperformed their component learners and DNN outperformed ensembles of base learners on the same tasks as expected. The use of log loss and accuracy are common in machine learning studies to evaluate the performance of prediction algorithms.

5 Conclusion



Results for the deep super learner are encouraging. Using a weighted average of the base learners optimized to minimize log loss yields results superior to any individual base learner. Using a cascade of multiple layers of base learners where each successive layer uses the output from the previous layer as augmented features for input to add depth to the learning improved performance further. While still shy of the performance levels obtained by CNN on image data, the deep super learner using traditional machine learning algorithms outperformed MLP on image data and outperformed MLP and CNN on classification from a TF-IDF matrix while also having fewer hyper-parameters and providing interpretable and transparent results. Though still in the early stages of development of a deep super learning ensemble, particularly compared to DNN, further development of the architecture, for example to better capture spatial or sequential relationships, should be conducted. Research on the performance of deep super learner on various larger datasets will be conducted.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), 1798–1828 (2013)
2. Längkvist, M., Karlsson, L., Loutfi, A.: A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recogn. Lett.* **42**, 11–24 (2014)
3. Schmidhuber, J.: Multi-column deep neural networks for image classification. In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*, pp. 3642–3649. IEEE Computer Society, Washington (2012)
4. Bengio, Y.: Learning deep architectures for AI. *Found. Trends® Mach. Learn.* **2**(1), 1–127 (2009)
5. Zhou, Z.H., Feng, J.: Deep forest: towards an alternative to deep neural networks. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, Melbourne, Australia, pp. 3553–3559 (2017)
6. Sussillo, D., Barak, O.: Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput.* **25**(3), 626–649 (2013)
7. Farrelly, C.M.: *Deep vs. Diverse Architectures for Classification Problems* (2017)
8. Seni, G., Elder, J.F.: Ensemble methods in data mining: improving accuracy through combining predictions. In: Grossman, R. (ed.) *Synthesis Lectures on Data Mining and Knowledge Discovery*. Morgan & Claypool (2010)
9. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, San Francisco (2016)
10. Xie, J., Rojkova, V., Pal, S., Coggeshall, S.: A combination of boosting and bagging for KDD cup 2009 - fast scoring on a large database. *J. Mach. Learn. Res. (JMLR)* **7**(2009), 35–43 (2009)
11. van der Laan, M.J., Polley, E.C., Hubbard, A.E.: Super learner. *Stat. Appl. Genet. Mol. Biol.* **6**(1) (2007)
12. Zhou, Z.H.: *Ensemble Methods: Foundations and Algorithms*. Machine Learning & Pattern Recognition Series. Chapman & Hall/CRC, Boca Raton (2012). illustrate edn.
13. Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking classification models for software defect prediction: a proposed framework and novel findings. *IEEE Trans. Softw. Eng.* **34**(4), 485–496 (2008)
14. Ferri, C., Hernández-Orallo, J., Modroi, R.: An experimental comparison of performance measures for classification. *Pattern Recogn. Lett.* **30**(1), 27–38 (2009)
15. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, HLT 2011, Portland, Oregon*, pp. 142–150 (2011)
16. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)



One Single Deep Bidirectional LSTM Network for Word Sense Disambiguation of Text Data

Ahmad Pesaranghader¹(✉) , Ali Pesaranghader²(✉) , Stan Matwin¹,
and Marina Sokolova^{2,3}

¹ Institute for Big Data Analytics, Dalhousie University, Halifax, Canada
{ahmad.pgh,st837183}@dal.ca

² School of Computer Science, University of Ottawa, Ottawa, Canada
{apesaran,sokolova}@uottawa.ca

³ School of Epidemiology and Public Health, University of Ottawa, Ottawa, Canada

Abstract. Due to recent technical and scientific advances, we have a wealth of information hidden in unstructured text data such as offline/online narratives, research articles, and clinical reports. To mine these data properly, attributable to their innate ambiguity, a Word Sense Disambiguation (WSD) algorithm can avoid numbers of difficulties in Natural Language Processing (NLP) pipeline. However, considering a large number of ambiguous words in one language or technical domain, we may encounter limiting constraints for proper deployment of existing WSD models. This paper attempts to address the problem of one-classifier-per-one-word WSD algorithms by proposing a single Bidirectional Long Short-Term Memory (BLSTM) network which by considering senses and context sequences works on all ambiguous words collectively. Evaluated on SensEval-3 benchmark, we show the result of our model is comparable with top-performing WSD algorithms. We also discuss how applying additional modifications alleviates the model fault and the need for more training data.

Keywords: Natural Language Processing
Word Sense Disambiguation · Deep learning
Bidirectional Long Short-Term Memory · Text mining

1 Introduction

Word Sense Disambiguation (WSD) is an important problem in Natural Language Processing (NLP), both in its own right and as a stepping stone to other advanced tasks in the NLP pipeline, applications such as machine translation [1] and question answering [2]. WSD specifically deals with identifying the correct sense of a word, among a set of given candidate senses for that word, when it is presented in a brief narrative (surrounding text) which is generally referred to as *context*. Consider the ambiguous word ‘cold’. In the sentence “*He started to give me a cold shoulder after that experiment*”, the possible senses for cold can

be *cold temperature* (S1), *a cold sensation* (S2), *common cold* (S3), or *a negative emotional reaction* (S4). Therefore, the ambiguous word cold is specified along with the sense set {S1, S2, S3, S4} and our goal is to identify the correct sense S4 (as the closest meaning) for this specific occurrence of cold after considering - the semantic and the syntactic information of - its context.

In this effort, we develop our supervised WSD model that leverages a Bidirectional Long Short-Term Memory (BLSTM) network. This network works with neural sense vectors (i.e. *sense embeddings*), which are learned during model training, and employs neural word vectors (i.e. *word embeddings*), which are learned through an unsupervised deep learning approach called GloVe (Global Vectors for word representation) [3] for the context words. By evaluating our one-model-fits-all WSD network over the public gold standard dataset of SensEval-3 [4], we demonstrate that the accuracy of our model in terms of F-measure is comparable with the state-of-the-art WSD algorithms⁷.

We outline the organization of the rest of the paper as follows. In Sect. 2, we briefly explore earlier efforts in WSD and discuss recent approaches that incorporate deep neural networks and word embeddings. Our main model that employs BLSTM with the sense and word embeddings is detailed in Sect. 3. We then present our experiments and results in Sect. 4 supported by a discussion on how to avoid some drawbacks of the current model in order to achieve higher accuracies and demand less number of training data which is desirable. Finally, in Sect. 5, we conclude with some future research directions for the construction of sense embeddings as well as applications of such model in other domains such as biomedicine.

2 Background and Related Work

Generally, there are three categories of WSD algorithms: supervised, knowledge-based, and unsupervised. Supervised algorithms consist of automatically inducing classification models or rules from labeled examples [5]. Knowledge-based WSD approaches are dependent on manually created lexical resources such as WordNet [6] and the Unified Medical Language System¹ (UMLS) [7]. Unsupervised algorithms may employ topic modeling-based methods to disambiguate when the senses are known ahead of time [8]. For a thorough survey of WSD algorithms refer to Navigli [9].

2.1 Neural Embeddings for WSD

In the past few years, there has been an increasing interest in training neural word embeddings from large unlabeled corpora using neural networks [10, 11]. Word embeddings are typically represented as a dense real-valued low dimensional matrix \mathbf{W} (i.e. a *lookup table*) of size $d \times v$, where d is the predefined embedding dimension and v is the vocabulary size. Each column of the matrix is

¹ <https://www.nlm.nih.gov/research/umls/>.

an embedding vector associated with a word in the vocabulary and each row of the matrix represents a latent feature. These vectors can subsequently be used to initialize the input layer of a neural network or some other NLP model. GloVe [3] is one of the existing unsupervised learning algorithms for obtaining these vector representations of the words in which training is performed on aggregated global word-word co-occurrence statistics from a corpus.

Besides word embeddings, recently, computation of sense embeddings has gained the attention of numerous studies as well. For example, Chen et al. [12] adapted neural word embeddings to compute different sense embeddings (of the same word) and showed competitive performance on the SemEval-2007 data [13].

2.2 Bidirectional LSTM

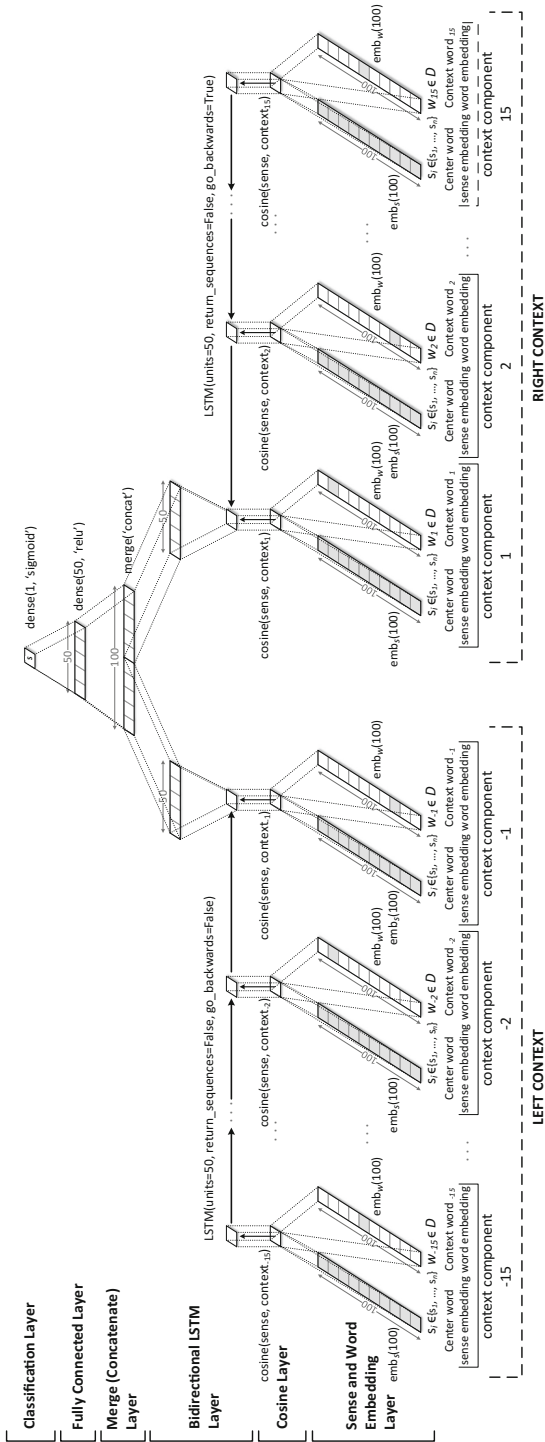
Long Short-Term Memory (LSTM), introduced by Hochreiter and Schmidhuber (1997) [14], is a gated recurrent neural network (RNN) architecture that has been designed to address the vanishing and exploding gradient problems of conventional RNNs. Unlike feedforward neural networks, RNNs have cyclic connections making them powerful for modeling sequences. A Bidirectional LSTM is made up of two reversed unidirectional LSTMs [15]. For WSD this means we are able to encode information of both preceding and succeeding words within context of an ambiguous word, which is necessary to correctly classify its sense.

3 One Single BLSTM Network for WSD

Given a document and the position of a target word, our model computes a probability distribution over possible senses related to that word. The architecture of our model, depicted in Fig. 1, consist of 6 layers which are a sigmoid layer (at the top), a fully-connected layer, a concatenation layer, a BLSTM layer, a cosine layer, and a sense and word embeddings layer (on the bottom).

In contrast to other supervised neural WSD networks in which generally a softmax layer - with a cross entropy or hinge loss - is parameterized by the context words and selects the corresponding weight matrix and bias vector for each ambiguous word's senses [16,17], our network shares parameters over all words' senses. While remaining computationally efficient, this structure aims to encode statistical information across different words enabling the network to select the true sense (or even a proper word) in a blank space within a context.

Due to the replacement of their softmax layers with a sigmoid layer in our network, we need to impose a modification in the input of the model. For this purpose, not only the contextual features are going to make the input of the network, but also, the sense for which we are interested to find out whether that given context makes sense or not (no pun intended) would be provided to the network. Next, the context words would be transferred to a sequence of word embeddings while the sense would be represented as a sense embedding (the shaded embeddings in Fig. 1). For a set of candidate senses (i.e. $\{s_1, \dots, s_n\}$) for an ambiguous term, after computing *cosine similarities* of each sense embedding



Single-classifier BLSTM architecture for WSD

Fig. 1. The single model of deep Bidirectional LSTM for Word Sense Disambiguation of text data. A series of (left and right) context components are centered around the ambiguous word. The cosine similarities between the context words and the examined sense as the outputs of the first two layers are fed to two LSTM networks with different directions. Then, the concatenated outputs of LSTMs is fed to a (binary) neural network sense classifier consisting of one fully-connected layer and a sigmoid unit. Finally, an argmax over the outputs of all the sigmoids for the set of candidate senses selects the true sense, confirming this sequence of cosine similarities is the best match for the correct sense based on the learned cosine similarities patterns during training of the network.

with the word embeddings of the context words, we expect the sequence result of similarities between the true sense and the surrounding context communicate a pattern-like information that can be encoded through our BLSTM network; for the incorrect senses this premise does not hold. Several WSD studies already incorporated the idea of sense-context cosine similarities in their models [18, 19].

3.1 Model Definition

For one instance (or one document), the input of the network consists of a sense and a list of context words (left and right) which paired together form a list of context components. For the context D which encompasses the ambiguous term T , that takes the set of predefined candidate senses $\{s_1, \dots, s_n\}$, the input for the sense s_i for which we are interested in to find out whether the context is a proper match will be determined by Eq. (1). Then, this input is copied (next) to $|D|$ positions of the context to form the first pair of the context components.

$$\mathbf{l}_i = \mathbf{W}_s^l \cdot \mathbf{v}_s(s_i), i \in \{1, \dots, n\}. \quad (1)$$

Here, $\mathbf{v}_s(s_i)$ is the one-hot representation of the sense corresponding to $s_i \in \{s_1, \dots, s_n\}$. A one-hot representation is a vector with dimension V_s consisting of $|V_s|-1$ zeros and a single one which index indicates the sense. The V_s size is equal to the number of all senses in the language (or the domain of interest). Equation (1) will have the effect of picking the column (i.e. sense embeddings) from \mathbf{W}_s^l corresponding to that sense. The \mathbf{W}_s^l (stored in the sense embeddings lookup table) is initialized randomly since no sense embedding is computed a priori.

Regarding the context words inputs that form the second pairs of context components, at position m in the same context D the input is determined by:

$$\mathbf{x}_m = \mathbf{W}_w^x \cdot \mathbf{v}_w(w_m), m \in \{-|D|/2, \dots, -2, -1, 1, 2, \dots, |D|/2\}. \quad (2)$$

Here, $\mathbf{v}_w(w_m)$ is the one-hot representation of the word corresponding to $w_m \in D$. Similar to a sense one-hot representation (V_s), this one-hot representation is a vector with dimension V_w consisting of $|V_w|-1$ zeros and a single one which index indicates the word in the context. The V_w size is equal to the number of words in the language (or the domain of interest). Equation (2) will choose the column (i.e. word embeddings) from \mathbf{W}_w^x corresponding to that word. The \mathbf{W}_w^x (stored in the word embeddings lookup table) can be initialized using pre-trained word embeddings; in this work, GloVe vectors are used.

On the other hand, the output of the network that is examining sense s_i is

$$\hat{y}_{s_i} = \sigma(\mathbf{W}_{out} \cdot \mathbf{h}_{cl} + \mathbf{b}_{out}), s_i \in \{s_1, \dots, s_n\} \quad (3)$$

where $\mathbf{W}_{out} \in R^{1 \times 50}$ and $\mathbf{b}_{out} \in R$ are the weights and the bias of the classification layer (sigmoid), and \mathbf{h}_{cl} is the result of the merge layer (concatenation).

When we train the network, for an instance with the correct sense and the given context as inputs, \hat{y}_{s_i} is set to be $\mathbf{1.0}$, and for incorrect senses they are

set to be $\mathbf{0.0}$. During testing, however, among all the senses, the output of the network for a sense that gives the highest value of \hat{y}_{s_i} will be considered as the true sense of the ambiguous term, in other words, the correct sense would be:

$$\arg \max_{s_i} \{\hat{y}_{s_1}, \dots, \hat{y}_{s_n}\}, s_i \in \{s_1, \dots, s_n\}. \quad (4)$$

By applying softmax to the result of estimated classification values, $\{\hat{y}_{s_1}, \dots, \hat{y}_{s_n}\}$, we can show them as probabilities; this facilitates interpretation of the results.

Further, the hidden layer h_{cl} is computed as

$$\mathbf{h}_{cl} = ReLU(\mathbf{W}_h \cdot [\mathbf{h}_{C-1}^L; \mathbf{h}_{C+1}^R] + \mathbf{b}_h) \quad (5)$$

where $ReLU$ means rectified linear unit; $[\mathbf{h}_{C-1}^L; \mathbf{h}_{C+1}^R]$ is the concatenated outputs of the right and left traversing LSTMs of the BLSTM when the last context components are met. \mathbf{W}_h and \mathbf{b}_h are the weights and bias for the hidden layer.

3.2 Validation for Selection of Hyper-parameters

SensEval-3 data [4] on which the network is evaluated, consist of separate training and test samples. In order to find hyper-parameters of the network 5% of the training samples were used for the validation in advance. Once the hyper-parameters are selected, the whole network is trained on all training samples prior to testing. As to the loss function employed for the network, even though is it common to use (*binary*) *cross entropy* loss function when the last unit is a sigmoidal classification, we observed that *mean square error* led to better results for the final *argmax* classification (Eq. (4)) that we used. Regarding parameter optimization, RMSprop [20] is employed. Also, all weights including embeddings are updated during training.

3.3 Dropout and Dropword

Dropout [21] is a regularization technique for neural network models where randomly selected neurons are ignored during training. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass, and any weight updates are not applied to the neuron on the backward pass. The effect is that the network becomes less sensitive to the specific weights of neurons, resulting in better generalization, and a network that is less likely to overfit the training data. In our network, dropout is applied to the embeddings as well as the outputs of the merge and fully-connected layers.

Following the dropout logic, *dropword* [22] is the word level generalizations of it, but in *word dropout* the word is set to zero while in dropword it is replaced with a specific tag. The tag is subsequently treated just like one word in the vocabulary. The motivation for doing dropword and word dropout is to decrease the dependency on individual words in the training context. Since by replacing word dropout with dropword we observed no change in the results, only word dropout was applied to the sequence of context words during training.

4 Experiments

In SensEval-3 data (*lexical sample task*²), the sense inventory used for nouns and adjectives is WordNet 1.7.1 [6] whereas verbs are annotated with senses from Wordsmyth³. Table 1 presents the number of words under each part of speech, and the average number of senses for each class.

As stated, training and test data are supplied as the instances of this task; and the task consist of disambiguating one indicated word within a context.

4.1 Experimental Settings

The hyper-parameters that were determined during the validation is presented in Table 2. The preprocessing of the data was conducted by lower-casing all the words in the documents and removing numbers. This results in a vocabulary size of $|V| = 29044$. Words not present in the training set are considered unknown during testing. Also, in order to have fixed-size contexts around the ambiguous words, the padding and truncating are applied to them whenever needed.

Table 1. Summary of senses in SensEval-3

Class	Number of words	Average senses
Nouns	20	5.8
Verbs	32	6.31
Adjectives	5	10.2
Total	57	6.47

Table 2. Hyper-parameter used for the experiments and the ranges that were searched during tuning. ‘-’ indicates no tuning was performed on that parameter.

Hyper-parameter	Range searched	Values used
Context size	[10, 100] [Left, Right]	[15 Left, 15 Right]
Embedding size	{50, 100, 200, 300}	100
BLSTM hidden layer size	[50, 300]	2 * 50
Dropout on sense/word embeddings	[0, 50%]	20%
Dropout on LSTM outputs	[0, 70%]	50%
Dropout on fully-connected layer	[0, 70%]	50%
Word dropout	[0, 50%]	20%
Sense embedding initialization	-	Random \in unif(-0.1, 0.1)
Word embedding initialization	-	GloVe ^a (uncased)

^aWikipedia and Gigaword (400 K vocab): <https://nlp.stanford.edu/projects/glove/>

² <http://www.senseval.org/senseval3>.

³ <http://www.wordsmyth.net/>.

4.2 Results

Between-all-models comparisons - When SensEval-3 task was launched 47 submissions (supervised and unsupervised algorithms) were received addressing this task. Afterward, some other papers tried to work on this data and reported their results in separate articles as well. We compare the result of our model with the top-performing and low-performing algorithms (supervised). We show our single model sits among the 5 top-performing algorithms, considering that in other algorithms for each ambiguous word one separate classifier is trained (i.e. in the same number of ambiguous words in a language there have to be classifiers; which means 57 classifiers for this specific task). Table 3 shows the results of the top-performing and low-performing supervised algorithms.

The first two algorithms represent the state-of-the-art models of supervised WSD when evaluated on SensEval-3. Multi-classifier BLSTM [16] consists of deep neural networks which make use of pre-trained word embeddings. While the lower layers of these networks are shared, upper layers of each network are responsible to individually classify the ambiguous that word the network is associated with. IMS+adapted CW [17] is another WSD model that considers deep neural networks and also uses pre-trained word embeddings as inputs. In contrast to Multi-classifier BLSTM, this model relies on features such as POS tags, collocations, and surrounding words to achieve their result. For these two models, softmax constitutes the output layers of all networks. htas3 [23] was the winner of the SensEval-3 lexical sample. It is a Naive Bayes system applied mainly to raw words, lemmas, and POS tags with correction of the a-priori frequencies. IRST-Kernels [24] utilizes kernel methods for pattern abstraction, paradigmatic and syntagmatic information and unsupervised term proximity on British National Corpus (BNC), in SVM classifiers. Likewise, nusels [25] makes use of SVM classifiers with a combination of knowledge sources (part-of-speech

Table 3. F-measure results for SensEval-3 (English lexical samples)

Rank	Method	F-measure(%)
1	Multi-classifier BLSTM [16]	73.4
1	IMS+adapted CW [17]	73.4
2	htas3 [23]	72.9
3	IRST-Kernels [24]	72.6
4	Our Single-classifier BLSTM	72.5
5	Nusels [25]	72.4
35	IRST-Ties	58.9
37	R2D2	57.2
39	NRC-Coarse	48.5
40	NRC-Coarse2	48.4
42	DLSI-UA-LS-SU	44.4

Table 4. WSD single-classifier BLSTM with other pieces or hyper-parameters

Network (Our single-classifier)	F-measure(%)
Full network in Fig. 1	72.5
BLSTM with reverse directions in Fig. 1	68.9
BLSTM with a shuffled context	67.3
Fully-connected layers instead of BLSTM layer	70.2
BLSTM without GloVe for the context (all weights are random)	65.6
BLSTM without word dropout	71.1
BLSTM with a larger context size [25 left, 25 right]	71.4

of neighboring words, words in context, local collocations, syntactic relations. The second part of the table lists the low-performing supervised algorithms [4]. Considering their ranking scores we see that there are unsupervised methods that outperform these supervised algorithms.

Within-our-model comparisons - Besides several internal experiments to examine the importance of some hyper-parameters to our network, we investigated if the sequential follow of cosine similarities computed between a true sense and its preceding and succeeding context words carries a pattern-like information that can be encoded with BLSTM. Table 4 presents the results of these experiments.

The first row shows the best result of the network that we described above (and depicted in Fig. 1). Each of the other rows shows one change that we applied to the network to see the behavior of the network in terms of F-measure. In the middle part, we are specifically concerned about the importance of the presence of a BLSTM layer in our network. So, we introduced some fundamental changes in the input or in the structure of the network. Generally, it is expected that the cosine similarities of closer words (in the context) to the true sense be larger than the incorrect senses' [18]; however, if a series of cosine similarities can be encoded through an LSTM (or BLSTM) network should be experimented. We observe if reverse the sequential follow of information into our Bidirectional LSTM, we shuffle the order of the context words, or even replace our Bidirectional LSTMs with two different fully-connected networks of the same size 50 (the size of the LSTMs outputs), the achieved results were notably less than 72.5%.

In the third section of the table, we report our changes to the hyper-parameters. Specifically, we see the importance of using GloVe as pre-trained word embeddings, how word dropout improves generalization, and how context size plays an important role in the final classification result (showing one of our experiments).

4.3 Discussion

From the results of Table 3, we notice our single WSD network, despite eliminating the problem of having a large number of WSD classifiers, still falls short when

is compared with the state-of-the-art WSD algorithms. Based on our intuition and supported by some of our preliminary experiments, this deficiency stems from an important factor in our BLSTM network. Since no sense embedding is made publicly available for use, the sense embeddings are initialized randomly; yet, word embeddings are initialized by pre-trained GloVe vectors in order to benefit from the semantic and syntactic properties of the context words conveyed by these embeddings. That is to say, the separate spaces that the sense embeddings and the (context) word embeddings come from enforces some delay for the alignment of these spaces which in turn demands more training data. Furthermore, this early misalignment does not allow the BLSTM fully take advantage of larger context sizes which can be helpful. Our first attempt to deal with such problem was to pre-train the sense embeddings by some techniques - such as taking the average of the GloVe embeddings of the (informative) definition content words of senses, or taking the average of the GloVe embeddings of the (informative) context words in their training samples - did not give us a better result than our random initialization. Our preliminary experiments though in which we replaced all GloVe embeddings in the network with sense embeddings (using a method proposed by Chen et al. [12]), showed considerable improvements in the results of some ambiguous words. That means both senses and context words (while they can be ambiguous by themselves) come from one vector space. In other words, the context would also be represented by the possible senses that its words can take. This idea not only can help to improve the results of the current model, it can also avoid the need for a large amount of training data since senses can be seen in both places, center and context, to be trained.

5 Conclusion

In contrast to common one-classifier-per-each-word supervised WSD algorithms, we developed our single network of BLSTM that is able to effectively exploit word orders and achieve comparable results with the best-performing supervised algorithms. This single WSD BLSTM network is language and domain independent and can be applied to resource-poor languages (or domains) as well. As an ongoing project, we also provided a direction which can lead us to the improvement of the results of the current network using pre-trained sense embeddings.

For future work, besides following the discussed direction in order to resolve the inadequacy of the network regarding having two non-overlapping vector spaces of the embeddings, we plan to examine the network on technical domains such as biomedicine as well. In this case, our model will be evaluated on MSH WSD dataset⁴ prepared by National Library of Medicine⁵ (NLM). Also, construction of sense embeddings using (extended) definitions of senses [26, 27] can be tested. Moreover, considering that for many senses we have at least one (lexically) unambiguous word representing that sense, we also aim to experiment with

⁴ <https://wsd.nlm.nih.gov/collaboration.shtml>.

⁵ <https://www.nlm.nih.gov/>.

unsupervised (pre-)training of our network which benefits form quarry management by which more training data will be automatically collected from the web.

References

1. Vickrey, D., Biewald, L., Teyssier, M., Koller, D.: Word-sense disambiguation for machine translation. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (2005)
2. Hung, J.C., Wang, C.S., Yang, C.Y., Chiu, M.S., Yee, G.: Applying word sense disambiguation to question answering system for e-learning. In: 19th International Conference on Advanced Information Networking and Applications, AINA 2005, vol. 1, pp. 157–162. IEEE (2005)
3. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
4. Mihalcea, R., Chklovski, T., Kilgariff, A.: The Senseval-3 English lexical sample task. In: Proceedings of SENSEVAL-3, The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (2004)
5. Zhong, Z., Ng, H.T.: It makes sense: a wide-coverage word sense disambiguation system for free text. In: Proceedings of the ACL 2010 System Demonstrations, pp. 78–83. Association for Computational Linguistics (2010)
6. Miller, G.A.: Wordnet: a lexical database for English. *Commun. ACM* **38**(11), 39–41 (1995)
7. Pesaranhader, A., Pesaranhader, A., Mustapha, N.: Word sense disambiguation for biomedical text mining using definition-based semantic relatedness and similarity measures. *Int. J. Biosci. Biochem. Bioinform.* **4**(4), 280 (2014)
8. Kim, S., Yoon, J.: Link-topic model for biomedical abbreviation disambiguation. *J. Biomed. Inf.* **53**, 367–380 (2015)
9. Navigli, R.: Word sense disambiguation: a survey. *ACM Comput. Surv. (CSUR)* **41**(2), 10 (2009)
10. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ACM (2008)
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781) (2013)
12. Chen, X., Liu, Z., Sun, M.: A unified model for word sense representation and disambiguation. In: EMNLP, pp. 1025–1035 (2014)
13. Navigli, R., Litkowski, K.C., Hargraves, O.: SemEval-2007 task 07: coarse-grained English all-words task. In: Proceedings of the 4th International Workshop on Semantic Evaluations, pp. 30–35. Association for Computational Linguistics (2007)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
15. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **18**(5), 602–610 (2005)
16. Kågebäck, M., Salomonsson, H.: Word sense disambiguation using a bidirectional LSTM. arXiv preprint [arXiv:1606.03568](https://arxiv.org/abs/1606.03568) (2016)
17. Taghipour, K., Ng, H.T.: Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In: HLT-NAACL, pp. 314–323 (2015)

18. McInnes, B.T., Pedersen, T.: Evaluating measures of semantic similarity and relatedness to disambiguate terms in biomedical text. *J. Biomed. Inform.* **46**(6), 1116–1124 (2013)
19. Pedersen, T., Kolhatkar, V.: Wordnet::Senserelate::Allwords: a broad coverage word sense tagger that maximizes semantic relatedness. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Demonstration Session*. Association for Computational Linguistics, pp. 17–20 (2009)
20. Hinton, G., Srivastava, N., Swersky, K.: RMSprop: divide the gradient by a running average of its recent magnitude. *Neural Networks for Machine Learning, COURSE-ERA Lecture 6e* (2012)
21. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
22. Iyyer, M., Manjunatha, V., Boyd-Graber, J., Daumé III, H.: Deep unordered composition rivals syntactic methods for text classification. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, pp. 1681–1691 (2015)
23. Grozea, C.: Finding optimal parameter settings for high performance word sense disambiguation. In: *Proceedings of SensEval-3 Workshop* (2004)
24. Strapparava, C., Gliozzo, A., Giuliano, C.: Pattern abstraction and term similarity for word sense disambiguation: IRST at SensEval-3. In: *Proceedings of SENSEVAL-3 Third International Workshop on Evaluation of Systems for the Semantic Analysis of Text*, pp. 229–234 (2004)
25. Lee, Y.K., Ng, H.T., Chia, T.K.: Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In: *SensEval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pp. 137–140 (2004)
26. Pesaranghader, A., Matwin, S., Sokolova, M., Beiko, R.G.: simDEF: definition-based semantic similarity measure of gene ontology terms for functional similarity analysis of genes. *Bioinformatics* **32**(9), 1380–1387 (2015)
27. Pesaranghader, A., Rezaei, A., Pesaranghader, A.: Adapting gloss vector semantic relatedness measure for semantic similarity estimation: an evaluation in the biomedical domain. In: Kim, W., Ding, Y., Kim, H.-G. (eds.) *JIST 2013. LNCS*, vol. 8388, pp. 129–145. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06826-8_11



MedFact: Towards Improving Veracity of Medical Information in Social Media Using Applied Machine Learning

Hamman Samuel^(✉)  and Osmar Zaiane 

Department of Computing Science, University of Alberta, Edmonton, Canada
{hwsamuel, zaiane}@ualberta.ca

Abstract. Since the advent of Web 2.0 and social media, anyone with an Internet connection can create content online, even if it is uncertain or fake information, which has attracted significant attention recently. In this study, we address the challenge of uncertain online health information by automating systematic approaches borrowed from evidence-based medicine. Our proposed algorithm, MedFact, enables recommendation of trusted medical information within health-related social media discussions and empowers online users to make informed decisions about the credibility of online health information. MedFact automatically extracts relevant keywords from online discussions and queries trusted medical literature with the aim of embedding related factual information into the discussion. Our retrieval model takes into account layperson terminology and hierarchy of evidence. Consequently, MedFact is a departure from current consensus-based approaches for determining credibility using “wisdom of the crowd”, binary “Like” votes and ratings, popular in social media. Moving away from subjective metrics, MedFact introduces objective metrics. We also present preliminary work towards a granular veracity score by using supervised machine learning to compare statements within uncertain social media text and trusted medical text. We evaluate our proposed algorithm on various data sets from existing health social media involving both patient and medic discussions, with promising results and suggestions for ongoing improvements and future research.

1 Introduction

Fake news on social media has garnered considerable attention recently. Our research looks at a related problem in the medical domain where consumption of inaccurate and uncertain medical information can have life-threatening consequences. For example, viral social media posts were recently used to falsely associate vaccinations with autism [1]. Articles supposedly written by medical professionals that linked autism and vaccinations were heavily shared on Facebook and other social networks, leading to a perception among many users that vaccinations are harmful. On the other hand, not getting vaccinated would give rise to more disease outbreaks and negatively affect public health overall. With the vast amount of information available online, certain information-seeking skill

sets are needed to locate credible information, especially for sensitive topics like health information. Online content about personal well-being, management of diseases, and other medical topics related to medicine and health care is referred to as health information [2]. In contrast, medical knowledge is health information verified through the scientific process and Evidence-Based Medicine (EBM).

Medical experts are able to determine trustworthiness of health information through EBM, a systematic approach for appraising health information on the basis of the best current evidence, clinical expertise, and patient needs in order to facilitate decisions about patient care [3]. EBM arranges pertinent information into a hierarchy of evidence based on methodological quality. From the most reliable Level I up to Level VII, evidence can be grouped into systematic reviews of randomized controlled trials, well-designed randomized controlled trials, quasi-experimental studies, cohort studies, meta-synthesis, single qualitative studies, and reports of expert committees [4].


This study explores how computing automation can be applied in conjunction with EBM to determine the credibility of online health information. We develop MedFact, an algorithm based on EBM and trusted medical information sources, in order to empower and educate online users to determine the credibility of health information. We also address the challenges of layperson versus technical vocabularies, and issues of effectively presenting credibility of information in simplified and non-technical formats. We also present our solution to the research question of granular phrase-level textual agreement. As a side effect of our proposed approach, various aspects of the EBM methodology are automated, including information retrieval and processing. We use the terms “credibility” and “veracity” interchangeably as referring to factual accuracy of information. These concepts are closely related to the notion of “trust”, involving a willing interaction between two or more entities with an implicit belief that the interaction will at least be self-beneficial in the worst case, and mutually beneficial to all entities involved in the best case [5].

2 Background

2.1 Current State of Computational Research in Trust and Health Social Media

Research into credibility in social media falls into two categories of empirical analysis and algorithmic contributions. Various studies have been conducted to measure the effectiveness of generic trust metrics in forums and online communities. These empirical studies can further be grouped into three categories looking at either the network structure, content, or behavioral signals from users. The network structure and its properties help to iteratively determine trust of a given user based on relationships to other trusted users [6]. Content has also been investigated as an indicator for trustworthiness. However, content assessment in current approaches relies on reputation assessment which is limited by user-based ratings. Collaborative content-based methods have been proposed to

determine user reputation. Other metrics such as frequency and sentiment of follow-up posts in relation to an original post have also been studied [7].

The popular approach for representing veracity is via ratings. There are various implicit and explicit metrics for trust requiring users to provide subjective feedback. Trust metrics provide an abstracted evaluation of the level of credibility or trust associated with content or users. Common trust metrics found in social networks are scaled unary ratings, such as Facebook “Like” , binary ratings such as up or down votes, ranked ratings such as Likert scale rankings, and reputation systems for measuring user trust using achievement levels, badges, and gamification [8]. Some drawbacks of ratings-based systems include inflation, bootstrapping, whitewashing, and cold start [9].

Research on pragmatic contributions to health information veracity are fewer. The seminal work by [10] on HealthTrust is one of the few health information-focused studies on trust. HealthTrust automatically assesses new health information based on a set of health web sites with known credibility. Comparison is based on link analysis and content-based analysis. In link analysis, the assumption is that trustworthy content will point to trustworthy web sites as an appeal for authority. Consequently, TrustRank is used to infer a ranking for new content based on inbound and outbound link analysis. In content-based analysis, topic discovery via the TAGME algorithm is used to classify new content as suspicious or trustworthy based on topic similarity with known content via affinity propagation clustering. Secondly, to improve content matching, Hidden Markov Models (HMM) are applied to an annotated training set in order to model trustworthy and suspicious sentences. A HealthTrust score is then assigned for each web site, which could then be iteratively exploited. However, there are no data sets available for use in training supervised learning models.

Veracity of specific health topics such as cancer treatments has also investigated using machine learning techniques such as the study by [11]. Using a bag of words representation as the feature set, web pages with medical advice were labeled as positive or negative based on whether they contained questionable content, and the trained model used to assign new labels to new web pages. This approach relied on keyword co-occurrences and correlations instead of cross-referencing trusted medical knowledge.

2.2 Psychological Viewpoint on Popularity of Fake Online Health Information

Various factors contribute to the present proliferation of unsafe health information online, which need to be taken into consideration when developing any approach for promoting credible information and preventing unsafe viral health campaigns. Apart from the development of technical solutions and effective trust metrics, the psychological biases of users consuming health information also need to be understood, including users’ preference for layperson health stories, perceived resistance to medical facts, and the perception of medical expertise among laypersons.

Neural Coupling. The information seeking behavior of laypersons and patients is based on story-telling rather than systematic medical and scientific methodologies. Patients tend to use personal experience and stories as a source of authoritativeness rather than scientific methodology [12]. This behavior is related to neural coupling, an effect observed in neuroscience between storytellers and listeners. Experiments have shown that when a storyteller is communicating with listeners, the listener's brain patterns will eventually mirror the storyteller's patterns. Neural coupling is an evolutionary trait to help human species to learn from each other through emotions [13]. The popularity of story-based narratives on health social media could also be attributed to these primal triggers. In the case of the "anti-vaxxers", even fake personal stories were effective in convincing people not to vaccinate because of the emotional format of the message [1].

Backfire Effect. Studies related to anti-vaxxers attempted to investigate the effectiveness of counter-messages promoting vaccinations for Measles Mumps Rubella (MMR) [14]. In the study, anti-vaxxer parents of children needing MMR vaccinations were presented with various interventions. Firstly, they were presented with information on the lack of evidence associating autism to vaccinations. Secondly, they were shown textual information on risks of not getting vaccinated. Thirdly, images of other children who had contracted MMR-related diseases were shown. And finally, parents were told a dramatic story of a child who did not get vaccinated for measles and almost died. Surprisingly, none of the interventions were statistically significant in convincing the parents. In some cases, the parents' belief that vaccinations are harmful was even strengthened, for instance when being shown the imagery of sick children who did not get vaccinated. These counterintuitive results could be explained by the backfire effect, wherein the presentation of contradictory evidence is not only ineffective in convincing people, but leads people to strengthen their belief [15]. Related to the backfire effect is confirmation bias, where users online tend to seek out and gravitate towards information supporting their beliefs and ignore opposing viewpoints [16].

Dunning-Kruger Effect. The Dunning-Kruger effect is attributed to unskilled persons having the illusion of superior competence [17], a trait that can be readily observed in the online health information communities, where laypersons eagerly and confidently provide medical advice to other laypersons. This phenomenon is clearer in the study of agnotology, where inaccurate or misleading scientific information is willfully promoted to induce ignorance about facts [18]. Essentially, online health information is saturated with information that is not credible, yet is being propagated due to users' willingness to look for quick solutions to complex health problems, such as autism [19].

3 Methodology

We define the task of determining the credibility of medical content as a five-step process. Given any textual document, such as a social media post, the first step is to extract health-related phrases $\{x_1, x_2, \dots, x_m \in X\}$. The veracity of these phrases is unknown. The second step uses automated information retrieval and processing to search trusted scientific and medical knowledge bases for each of the unknown phrases $x_i \in X$. In this step, each trusted source would yield zero or more relevant articles, providing a collection of trusted articles which are ranked and filtered by relevance. Moreover, each trusted article would have various related credible phrases that are identified in the third step to generate a collection of trusted phrases $\{t_1, t_2, \dots, t_n \in T\}$. The semantic similarity between a given trusted phrase, $t_j \in T$, and x_i is used for inferring an *agreement score*, $\Upsilon(x_i, t_j)$ between the two phrases. In the fourth step, an aggregated agreement score for a given unknown phrase is computed by comparing it with all trusted phrases and averaging the agreement score as formulated in Eq. 1. In the fifth step, an overall *veracity score* ϑ is computed for the social media post from the aggregated agreement scores of all unknown phrases as shown in Eq. 2.

$$\Upsilon(x_i) = \left(\sum_{p=1}^n \Upsilon(x_i, t_p) \right) / n \quad (1)$$

$$\vartheta = \left(\sum_{q=1}^m \Upsilon(x_q) \right) / m \quad (2)$$

Our methodology has parallels with the EBM five-step model: ask, acquire, appraise, apply, and analyze [20]. Overlapping MedFact with EBM, asking a question entails seeking to investigate the veracity of a social media post, while acquiring involves computationally gathering the available evidence related to the question. The overall pipeline for MedFact is depicted in Fig. 1.

Step 1. To extract relevant health phrases from a given social media posting, candidate phrases are extracted using key phrase extraction. The next stage identifies health-related phrases from among the candidate phrases. Extraction of key phrases is done using the TextRank algorithm¹. In the next stage, we use a supervised learning approach to build a binary classifier that for classifying a given phrase as medical or non-medical. The classifier is implemented as an artificial neural network, and medical phrases are input as word embeddings, with output of 0 if the phrase is non-medical or 1 if medical. In order to train our classifier, we use two categories of data sets. The first category corresponds to the “medical” label, including medical phrases from the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) database and layperson health terms from the Consumer Health Vocabulary (CHV) data set.

¹ The GenSim Python API includes the TextRank algorithm [21] implementation <https://radimrehurek.com/gensim/summarization/keywords.html>.

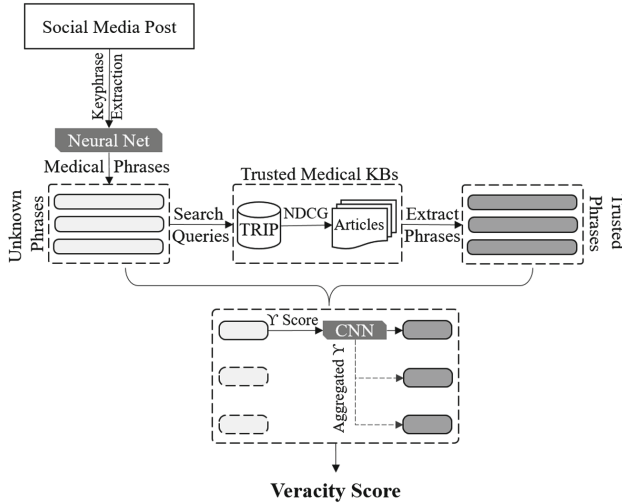


Fig. 1. Overview of MedFact algorithm

SNOMED CT² is a digital collection of medical terms provided by the U.S. National Library of Medicine [22]. The CHV data set³ provides mappings of common layperson medical terms to technical terms in the Unified Medical Language System (UMLS) [23]. The second category corresponds to the “non-medical” label and contains known non-medical corpora from the Simple English Wikipedia (SEW) data set⁴ [24]. From these data sets, a training sample is created by arbitrary selection of approximately 80% of the phrases from each data set. A test sample of 20% is kept for evaluation purposes. The phrases (hyphenated) are converted to word embeddings using the Word2Vec deep neural network model trained on medical corpora with skip-grams [25]. The phrases and their corresponding labels from the training sample are used to train our neural net. The arbitrary selection process is repeated a number of times to achieve non-exhaustive cross-validation and the best trained model is used.

Step 2. Credible medical knowledge can be queried from the Turning Research Into Practice (TRIP) database⁵. TRIP focuses on evidence-based medical literature from various trusted sources including the NLM’s MEDLINE and PubMed

² SNOMED CT data set available from U.S. National Library of Medicine (NLM) <https://nlm.nih.gov/healthit/snomedct>.

³ CHV data set available from the Consumer Health Vocabulary Initiative <http://consumerhealthvocab.org>.

⁴ SEW historical data set available via PIKES home page <http://pikes.fbk.eu/eval-sew.html>.

⁵ The TRIP database is accessible programmatically via web services that were most kindly made available to the authors by Jon Brassey, the TRIP database creator <https://tripdatabase.com/addtrip>.

articles, the Cochrane database of systematic reviews, the Database of Abstracts of Reviews of Effects (DARE), among others. Moreover, the TRIP database also searches within patient-friendly resources such as Cochrane Clinical Answers and WebMD’s Medscape [26]. Results are categorized into the levels of evidence and can be sorted by quality, relevance, or date. A publication score is used to assess and rank quality of the results by incorporating the levels of evidence, Level I receiving the highest weight and subsequent levels receiving progressively lower weights. We use TRIP’s quality metric to sort articles and incorporate strength of the evidence. We perform additional ranking of the articles in order to evaluate the usefulness of the top- n articles based on their position in the results using Normalized Discounted Cumulative Gain (NDCG) [27].

Step 3. In order to compare unknown phrases with trusted phrases, phrases are extracted from the ranked medical articles via phrase chunking. Firstly, each article’s text is split using sentence and word tokenization. Next, Part-Of-Speech (POS) tagging is performed on the tokens, followed by phrase chunking⁶ which segments the sentences into noun phrases. After that, each chunked phrase extracted from the medical articles is compared with the set of unknown phrases, and trusted phrases that do not correlate with unknown phrases are discarded because they will not be useful in the next steps.

Step 4. Given a phrase whose credibility needs to be ascertained, a corresponding set of phrases from a trusted source can be used as evidence for supporting or rejecting the unknown phrase as credible. We model this problem as that of predicting a class label over a pair of phrases, where two binary labels are possible: *Yes* and *No*. The former label implies that the two phrases have the same meaning, while the latter label means the phrases could contain incompatible propositions such as contradictions.

Given two phrases, we determine their agreement using deep learning, incorporating semantic similarity and sentiment analysis of the two phrases. Our feature set consists of the word embeddings of the two phrases, and sentiment information⁷ for each phrase, specifically polarity and subjectivity [28]. Polarity for a phrase is in the range $[-1.0, 1.0]$ where -1.0 implies very negative sentiment and 1.0 means very positive sentiment, while subjectivity values are in the range $[0.0, 1.0]$ where 0.0 means very objective and 1.0 implies very subjective. We also use the negation modifier from dependency parsing [29] of the related sentence containing the target phrases as an additional binary feature, where 1 implies the presence of the negation modifier and 0 means an absence⁸. For our deep learning neural network implementation, we use a shallow Convolutional Neural

⁶ POS tagging is done using the Penn Treebank tags set, all steps in this particular pipeline are programmed with the NLTK Python library <http://nltk.org>.

⁷ Sentiment analysis is performed using the TextBlob Python library <http://textblob.readthedocs.io>.

⁸ The spaCy Python library is used for generating dependency trees <https://spacy.io>.

Network (CNN) architecture⁹, which is more suitable for learning from smaller-sized labeled training data sets [30]. We build our training data set from Health Stack Exchange (HSE)¹⁰, an online question-answering community where users can post health-related questions¹¹. Within the HSE community, moderators can manually flag semantically equivalent posts as *Duplicate*.

Our training data set consists of pairs of phrases extracted from the duplicate posts' title and body using phrase chunking. The related medical phrase pairs extracted from these question pairs are assigned the *Yes* label. For question pairs that are not duplicates, the *No* label is assigned to the related phrase pairs in the training data set. We subsequently manually curate the training data set for accuracy of the initial labeling in order to verify whether the phrase pairs are in agreement or not. Ultimately, given two phrases, the *agreement score* is defined using the classifier's output label. If the *No* label is returned, agreement score is 0, while for the *Yes* label, the score is 1.

Step 5. The veracity score enables aggregation of the agreement scores of many pairs of unknown phrases and their respective trusted phrases, and provides a single metric for measuring the credibility of a given social media posting or document. This approach allows for a granular definition of veracity starting from phrase-level agreement to document-level aggregated agreement. Depending on the number of unknown and trusted phrase pairs, the overall veracity score is computed as an average, hence it is within the range [0.0, 1.0], and can be expressed as a simplistic percentage value.

4 Evaluation

4.1 Effectiveness of Key Phrase Extraction

We evaluated the performance of the key phrase extraction step using the HSE data set, which contains human-annotated tags per question. We compared our extracted key phrases with the annotated tags and used the recall metric to measure performance. For instance, if all the tags were found within the key phrases of a given question, the recall was recorded as 100%. The HSE data set contained 3,958 questions and 2,260 tag sets and the average recall for the key phrase extraction step was 81.28%.

4.2 Effectiveness of Medical Phrases Extraction

We used the SNOMED CT, CHV, and SEW databases to perform extraction of medical phrases from social media postings using a neural network for binary

⁹ We implement a shallow CNN with the ConText tool
<https://github.com/riejohnson/ConText>.

¹⁰ Health Stack Exchange's beta web site <https://health.stackexchange.com>.

¹¹ Data set curated from the Stack Exchange Data Dump from the Internet Archive
<https://archive.org/details/stackexchange>.

classification. We initially tested the performance of the classifier trained on combinations of medical and English data sets, and then evaluated the overall performance of the classifier using all three data sets, which provided the best precision and recall values of 74.00% and 67.10% respectively via 10-fold cross-validation.

4.3 Curation of Phrase Pairs for CNN Training

Using the best performance configuration for the medical key phrases extraction step, a total of 11,517 relevant medical phrases was extracted, averaging 2.91 phrases per HSE question. A total of 43 duplicate questions was recorded by the HSE moderators, and 175 phrases were extracted from these duplicates, out of which 83 pairs had agreement. We also manually inspected the rest of the data set to retrieve a total of 181 pairs that could be marked with the *Yes* label. The training data set was then balanced to arbitrarily include an equivalent number of phrase pairs with the *No* label. Overall, the average Fleiss Kappa for the curation process was 0.691, indicating “moderate agreement” for the corrections and additions made [31].

4.4 Appraisal of Phrase Agreement CNN

We explored the relationship between performance of the CNN used for determining phrase agreement and the feature set. The best precision and recall values of 0.606 and 0.448 respectively were achieved by including all features.

4.5 Feedback on Usage of Veracity Score

To assess the effectiveness of MedFact, we designed a short survey that was administered to 19 users. The survey contained polarizing social media postings on the link between vaccination and autism, apricot pits as a cure for cancer, and usefulness of flossing for dental care. Firstly, a posting supporting vaccination and autism was displayed, followed by a post debunking the notion. Similarly, users were then shown a posting supporting apricot pits as a cure for cancer, and then shown an opposing post. Lastly, posts supporting and opposing the need to floss were shown. For each posting shown, the veracity score expressed as a percentage (rounded-off) was visible. The top 3 trusted articles related to the posting were also displayed. After displaying each posting, users were asked three questions related to the veracity score and the linked articles. Each question required a Yes or No response for the related post being displayed. Firstly, they were asked “*Is the veracity score useful in this context?*”. Next, they were asked “*Is the veracity score accurate for this post?*”. Lastly, they were asked “*Are the links to the medical articles useful?*”. A summary of users’ responses recorded for the questions is shown in Fig. 2. At the end of the survey, users were optionally asked to give any general feedback in free text form.

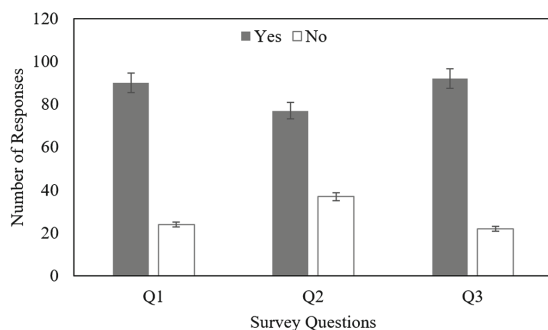


Fig. 2. Summary of veracity score survey responses

4.6 Veracity Score on Unproven Cancer Treatments

We randomly selected 30 articles on cancer from QuackWatch¹², a web site indexing unproven treatments [11]. These selected articles were input to MedFact to compute a veracity score in order to determine whether the score would align with experts' opinions. The veracity score for the selected articles is summarized in Fig. 3(a), showing an overall low score for the articles, hence a consensus with the opinions of the experts who identified the unproven claims, as well as comparable results with the study on predicting unproven cancer treatments by [11].

4.7 Online Medic Discussions Evaluated via Veracity Score

To further evaluate the performance and representative accuracy of MedFact's veracity score, we randomly sampled 30 answers posted on the DocCheck forums¹³. DocCheck allows verified medical professionals to ask questions and post answers. The results showed an average veracity score of 78.32%. A comparison of the veracity scores for medic posts versus QuackWatch averaged scores is presented in Fig. 3(b).

5 Discussion

Regarding the results of the survey, users did provide generally positive feedback to all three questions. However, regarding the accuracy of the veracity score, users gave less than expected positive feedback. Further analysis into the responses revealed that this was related to the second posting on apricot pits as a cure for cancer, accounting for 70.27% of the lower positive feedback. We investigated the free text feedback to further understand user perspectives. We discovered that the majority of survey participants viewed apricot pit treatments

¹² QuackWatch web site <http://quackwatch.org>.

¹³ DocCheck web site <http://doccheck.com>.

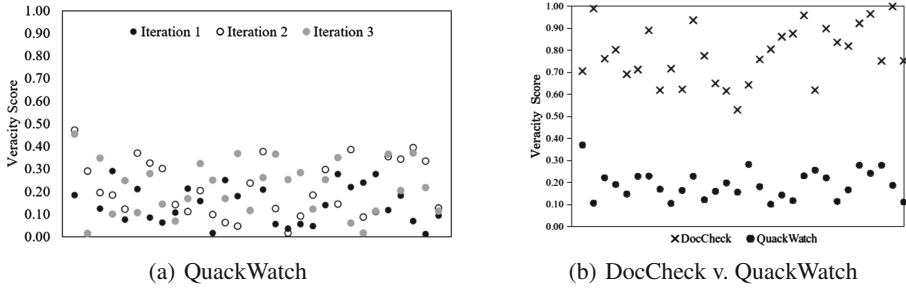


Fig. 3. Veracity score comparisons

as a homeopathic remedy that should not be covered by scientific literature. Overall, the survey recorded positive feedback from 67.54% of the responses regarding the veracity score accuracy. One area of improvement was the phrase pairs used for our CNN training data, which were limited in number based on the HSE data set. Currently, to the best of our knowledge, there are no medical data sets containing phrase pairs that are annotated for agreement, and involvement from medical experts is essential in order to improve the quality of the phrase pairs. Regarding the comparison between DocCheck and QuackWatch, the results were as expected, with the DocCheck veracity scores being significantly higher than QuackWatch. Moreover, a binary clustering effect can be observed between credible and untrustworthy posts by use of veracity score.

6 Conclusion

The modern patient desires self-education of medical concepts, and seeks to be part of the diagnosis process. However, there is a communication barrier when dealing with technical medical terminology, which could have led to patients seeking more personal and narrative-based sources of medical information. MedFact is an initial step towards bringing trusted and patient-friendly medical knowledge into social media discourse. In this study, we addressed the challenge of veracity in online health information by automating the evidence-based medicine methodology, thereby incorporating medical knowledge into social media discussions while taking into account layperson terminology and hierarchy of evidence.

We also demonstrated the MedFact algorithm, which models trustworthiness and reliability of online information using machine learning and facilitates recommendation of trusted medical articles, ultimately empowering online users to make informed decisions about health information they are consuming. Preliminary work towards a granular veracity score was demonstrated using practical and systematic state-of-the-art methods from deep learning, information retrieval, and text processing. We performed an in-depth experimental analysis of the accuracy and performance of MedFact using the Health Stack Exchange, QuackWatch, and DocCheck data sets via metrics such as precision and recall, as well as qualitative analysis via survey. MedFact improves

upon existing approaches towards determining credibility in health-related social media, where ratings, user reputations, and “wisdom of the crowd” are being used predominantly. Firstly, ratings and reputations are more subjective than cross-referencing trusted medical literature. Secondly, ratings require extensive community input, but MedFact relies on existing and comprehensive medical knowledge bases available via the TRIP database. Thirdly, MedFact presents a granular approach that computes veracity from the phrase-level to the document-level. For future work, an interesting aspect to explore is that of agreement between trusted medical articles. Various medical articles may occasionally contain contradictory facts which need to be resolved before incorporating these facts into the MedFact algorithm. A self-referencing veracity score between two trusted phrases can be determined using truth discovery algorithms.

Acknowledgement. We thank the Alberta Machine Intelligence Institute (Amii) for funding this research.

References

1. Kata, A.: Anti-vaccine activists, web 2.0, and the postmodern paradigm-an overview of tactics and tropes used online by the anti-vaccination movement. *Vaccine* **30**(25), 3778–3789 (2012)
2. Rippen, H., Risk, A.: e-Health code of ethics (May 24). *J. Med. Internet Res.* **2**(2) (2000)
3. Greenhalgh, T.: *How to Read a Paper: The Basics of Evidence-Based Medicine*. Wiley, Chichester (2010)
4. Ackley, B.J.: *Evidence-Based Nursing Care Guidelines: Medical-Surgical Interventions*. Elsevier Health Sciences, St. Louis (2008)
5. Child, J.: Trust-the fundamental bond in global collaboration. *Organ. Dyn.* **29**(4), 274–288 (2001)
6. Varlamis, I., Eirinaki, M., Louta, M.: A study on social network metrics and their application in trust networks. In: *Proceedings of the IEEE International Conference on Advances in Social Networks Analysis and Mining*, pp. 168–175 (2010)
7. Abdaoui, A., Azé, J., Bringay, S., Poncelet, P.: Collaborative content-based method for estimating user reputation in online forums. In: Wang, J., Cellary, W., Wang, D., Wang, H., Chen, S.-C., Li, T., Zhang, Y. (eds.) *WISE 2015. LNCS*, vol. 9419, pp. 292–299. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26187-4_26
8. Grant, S., Betts, B.: Encouraging user behaviour with achievements: an empirical study. In: *IEEE International Working Conference on Mining Software Repositories (MSR)*, pp. 65–68 (2013)
9. Aljazzaf, Z.M.: *Trust-Based Service Selection*. Ph.D. thesis. University of Western Ontario (2011)
10. Park, M.: *HealthTrust: Assessing the Trustworthiness of Healthcare Information on the Internet*. Ph.D. thesis. University of Kansas (2013)
11. Aphinyanaphongs, Y., Aliferis, C., et al.: Text categorization models for identifying unproven cancer treatments on the web. In: *World Congress on Medical Informatics (MedInfo)*, p. 968. IOS Press (2007)

12. Oliphant, T.: “I am making my decision on the basis of my experience”: constructing authoritative knowledge about treatments for depression. *Can. J. Inf. Libr. Sci.* **33**(3–4), 215–232 (2009)
13. Stephens, G.J., Silbert, L.J., Hasson, U.: Speaker-listener neural coupling underlies successful communication. *Proc. Natl. Acad. Sci.* **107**(32), 14425–14430 (2010)
14. Nyhan, B., Reifler, J., Richey, S., Freed, G.L.: Effective messages in vaccine promotion: a randomized trial. *Pediatrics* **133**(4) (2014)
15. Nyhan, B., Reifler, J.: When corrections fail: the persistence of political misperceptions. *Polit. Behav.* **32**(2), 303–330 (2010)
16. Plous, S.: *The Psychology of Judgment and Decision Making*. McGraw-Hill, New York (1993)
17. Dunning, D.: The dunning-kruger effect: on being ignorant of one’s own ignorance. *Adv. Exp. Soc. Psychol.* **44**, 247 (2011)
18. Proctor, R., Schiebinger, L.L.: *Agnotology: The Making and Unmaking of Ignorance*. Stanford University Press, Stanford (2008)
19. Henderson, J.: Expert and lay knowledge: a sociological perspective. *Nutr. Diet.* **67**(1), 4–5 (2010)
20. Straus, S.E., Richardson, S.W., Glasziou, P., Haynes, B.R.: *Evidence-Based Medicine: How to Practice and Teach EBM*. Elsevier/Churchill Livingstone, New York (2005)
21. Mihalcea, R., Tarau, P.: TextRank: bringing order into text. In: *EMNLP*, vol. 4, pp. 404–411 (2004)
22. Cornet, R., de Keizer, N.: Forty years of SNOMED: a literature review. *BMC Med. Inform. Decis. Mak.* **8**(1), S2 (2008)
23. Smith, C., Stavri, P.: Consumer health vocabulary. In: *Consumer Health Informatics*, pp. 122–128 (2005)
24. Corcoglioniti, F., Rospocher, M., Apro시오, A.P.: Extracting knowledge from text with PIKES. In: *International Semantic Web Conference* (2015)
25. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. *arXiv* (2013)
26. Brassey, J.: TRIP database: identifying high quality medical literature from a range of sources. *New Rev. Inf. Netw.* **11**(2), 229–234 (2005)
27. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2008)
28. Pang, B., Lee, L., et al.: Opinion mining and sentiment analysis. *Found. Trends® Inf. Retr.* **2**(1–2), 1–135 (2008)
29. De Marneffe, M.C., Manning, C.D.: *Stanford Typed Dependencies Manual*. Technical report, Stanford University (2008)
30. Johnson, R., Zhang, T.: Effective use of word order for text categorization with convolutional neural networks. In: *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)* (2015)
31. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. *Biometrics* **33**(1), 159–174 (1977)



Reranking Candidate Lists for Improved Lexical Induction

Laurent Jakubina and Philippe Langlais^(✉)

Université de Montréal, CP 6128 Succursale Centre-Ville,
Montreal, QC H3C3J7, Canada
felipe@iro.umontreal.ca
<http://rali.iro.umontreal.ca>

Abstract. Identifying translations in bilingual material—also referred to as the Bilingual Lexicon Induction (BLI) task—is a challenge that has attracted many researchers since a long time. In this paper, we investigate the reranking of two types of state-of-the-art approaches that have been used for the task. We test our reranker on four language pairs (translating from English), analyzing the influence of the frequency of the source terms we seek to translate. Our reranking approach almost invariably leads to performance gains for all the translation directions we consider.

1 Introduction

Identifying translations in bilingual material—also referred to as the Bilingual Lexicon Induction (BLI) task—is a challenge that has attracted many researchers since a long time. One of the earliest approaches [1] relies on the assumption that words in translation relation show similar co-occurrence patterns. Many variants of this approach have been investigated [2]. Some authors have for instance reported gains by considering syntactically motivated co-occurrences, either by the use of a parser [3] or simpler Part of Speech patterns [4]. Extensions to multiword expressions have also been proposed [5].

Recently, there has been a wealth of works dedicated to identify translations thanks to so-called word embeddings. In [6] the authors describe two models implemented in the popular `Word2Vec` toolkit, that efficiently train vector-representations of the words in a large monolingual collection of texts. In [7], it is further shown that a mapping between word embeddings learnt independently for each language can be trained by making use of a (small) seed bilingual lexicon. Since then, many practitioners have studied the BLI task as a mean to evaluate continuous word-representations [8–12]. While approaches differ in the type of data they can digest (monolingual data, word-aligned parallel sentence pairs, parallel sentence pairs) it is still fair to say that training monolingual word embeddings, then learning a mapping between the two vector spaces obtained is an extremely efficient solution that performs rather well on several BLI benchmarks. Read [13, 14] for two comparisons of several of those techniques.

Learning to discriminate good from bad translations has been investigated in [15]. In this work, the authors show that monolingual signals (orthographic,

temporal, topical, etc.) can be efficiently used to train in a supervised way a binary classifier. For this approach to work, some metadata is required, such as the time stamp of (pre-collected) news texts, or the list of inter-language linked Wikipedia pages.¹ We are more interested in this work to approaches that do not exploit such metadata.

Reranking the output of several BLI approaches has been investigated by some authors [16–18], mostly for translating terms of the medical domain, where dedicated approaches can be designed to capture correspondences at the morphemic level.² A similar idea has been proposed in [19] for translating noun-noun compounds in English and Japanese. In those works, the rerankers are prescribed (basically, a weighted average of native scores), while in [20], a reranker is trained in a supervised way to rerank or merge n-best lists. While interesting, this last approach has only been tested on the English-French language pair.

In this study, we revisit and extend the work of [20] by considering three other language pairs: English-Romanian, English-Spanish, and English-German. We provide evidences that the reranking approach is apt for all these language pairs. We distribute our datasets and bilingual lexicons, so that further experiments can be performed.³

2 Native BLI Methods

We consider three native approaches to BLI, all of them being projective in the sense that they all map mono-lingually acquired representations (trained or not), thanks to a (small) seed bilingual lexicon.

2.1 Plain Projective Approach (Rapp)

We tested variants of [1] where each word of interest is represented mono-lingually by a so-called context vector, that is, the words it co-occurs with. A co-occurrence can be encoded by a boolean (present/absent), by its frequency, or by a real measuring its association strength: point-wise mutual information (PMI), log-likelihood ratios (LLR) and discontinuous odd-ratios (ODR) are popular weighting schemas. Given a word to translate, its vector representation is projected using a seed bilingual lexicon: each co-occurrent word is simply looked-up in the bilingual dictionary, and the sanctioned translations are added to the vector representation in the target language. A similarity measure (typically cosine) is used for ranking target vectors according to their similarity to the projected vector.

We ran more than 110 configurations, varying typical hyper-parameters among which the context window size (6, 15, 20, 30), the association measure (PMI, LLR, ORD), as well as meta-parameters that control the way the target

¹ The authors demonstrate the potential of the approach on 22 language pairs, but the datasets used in their work are unfortunately not available.

² Similarly to the orthographic signal used in [15].

³ <http://rali.iro.umontreal.ca/rali/en/bli-dataset>.

vectors are computed. We used the cosine similarity measure in all configurations, and projected source vectors according to a large in-house bilingual lexicon with no overlap with the test material.

Since many contextual words are typically absent from this lexicon, the resulting projected vector can be rather sparse. To overcome this situation to some extent, words in the context vector unknown for the seed bilingual lexicon are added to the projected vector. We observe this strategy to be beneficial in practice since unknown words encompass proper names, numerical entities or acronyms that often are invariant across languages.

2.2 Linear Projection (Miko)

In [7], the authors propose to train a linear transformation between independently trained source and target embeddings, thanks to a seed lexicon. We reproduced variants of this approach, training monolingual embeddings with `Word2Vec`⁴ toolkit [7]. We ran over 160 configurations, varying the architecture of the model (Skip-gram (*skg*) or Continuous Bag-of-Words (*cbow*)), the optimization algorithm (Negative Sampling (5 or 10 samples) or Hierarchical Softmax), the context window size (6, 10, 20, 30). The largest embedding dimension for which we managed to train a model is 200 for the *cbow* architecture and 250 for the *skg* architecture.⁵ We learnt the projection matrix with the implementation described in [21]. We built 6 bilingual lexicons of different characteristics: 5 from our in-house bilingual lexicon, that is, 2k of low frequency word pairs, 5k, 15k and 30k of randomly picked word pairs, and 5k of highly frequent word pairs; as well as one lexicon populated with 5k of highly frequent word pairs, as in [7].

2.3 Canonical Correlation Analysis (Faru)

In [22], the authors propose a technique based on canonical correlation analysis which transforms two existing monolingual embeddings so as to maximize the correlation between representations of words paired in a bilingual lexicon. It has been shown to improve independently trained monolingual embeddings on a number of monolingual benchmarks.

This approach only changes the way monolingual embeddings are mapped, so we used the embeddings we identified the most suited for the *Miko* approach. We used the implementation accompanying [22].⁶ We ran a number of configurations, varying the bilingual lexicon used (as previously described), and tuning the *ratio* parameter over the values 0.5, 0.8 and 1.0.

3 Reranking

We used the RankLib⁷ library to access the implementation of 8 Learning to Rank Algorithms (MART, RankNet, RankBoost, AdaRank, Coordinate Ascent,

⁴ <https://code.google.com/p/word2vec/>.

⁵ Our cluster could accommodate up to 64 Gb of memory.

⁶ <https://github.com/mfaruqui/crosslingual-cca>.

⁷ <https://sourceforge.net/p/lemur/wiki/RankLib/>.

LambdaMART, ListNet, and Random Forests). We optimized each algorithm in a supervised way thanks to precision at rank 1. For a source term s and a candidate translation t , we compute 3 families of features:

Frequency features 4 features recording the (raw) frequency of s (resp. t) in the source (resp. target) corpus, the difference between those two frequencies as well as their ratio. Frequency is one of the signals used in [15].

String features 5 features recording the length (counted in chars) of s and t , their difference, their ratio, and the edit-distance between the two. Edit-distance has been repetitively reported to be a useful clue for matching terms. It has been also used as a useful signal in [15].

Rank features for each n -best list considered, we compute 2 features: the score of t in the list, as well as its rank. Whenever several n -best lists are reranked, we also add a feature which records the number of n -best lists in which t appears as a candidate translation of s .

Those features are straightforward, do not require any metadata, and are of course largely extensible. But as we shall see, they already yield gains, for the four language pairs we studied here.

4 Experimental Protocol

4.1 Monolingual Collections

We consider the task of identifying the translation of English terms into 4 languages: French, German, Romanian and Spanish. We extracted the text from those dumps thanks to WikiExtractor⁸. The monolingual datasets used for computing distributional representations are Wikipedia dumps, which means that a fair number of article pairs are indeed comparable⁹, although we do not use this information specifically. The main characteristics of the Wikipedia dumps we collected for each language are reported in Table 1. One specificity of our experimental protocol, is that we seek the translation of a source term among all the token types present in the target Wikipedia collection. As can be seen in the last line of Table 1, this represents a rather large set of candidate translations (in the order of millions), which poses technical challenges. This choice departs from most studies where heuristics are being used to reduce the list of candidate terms among which a translation is searched for. A typical experimental setting consists in considering only target words that happen frequently enough (say at least five times) in the target collection, as in [7] where the size of the target vocabulary they consider is in the order of a hundred thousand words, an order of magnitude less than in our setting. We believe our choice to be more faithful to the zipfian nature of (even massive) texts collections, where most token types actually occur rarely.

⁸ <https://github.com/attardi/wikiextractor>.

⁹ Two documents are said comparable if they address the same topic, without being in translation of each other.

Table 1. Main characteristics of the Wikipedia material. The French dump is from June 2013, while for the other languages we took the dumps of July 2017.

	English	German	French	Spanish	Romanian
#tokens	4075.7M	1425.6M	1220.3M	866.5M	130.8M
#types	8.6M	8.0M	3.7M	3.2M	1.2M

4.2 Test Sets

Following [20], we gathered for each language pair three reference lists of words and their translations, each one populated with source (English) terms of various frequency in Wikipedia. One named $\text{Wiki}_{\leq 25}$, is populated with English words occurring less than 26 times in English Wikipedia. Actually 92% of the tokens in the English collection have this property. Another set, named $\text{Wiki}_{> 25}$, gathers words with frequency in (English) Wikipedia higher than 25. Last, since most recent studies on BLI focus on translating highly frequent words, we reproduced this setting here, following the protocol described in [7]. Basically, it consists in translating 1 000 terms of the WMT11 dataset¹⁰, which rank between 5 000 and 6 000 when sorted in decreasing order of frequency (the first 5k top-frequent words being spared to train the projection). We name the resulting dataset Euro_{5-6k} hereafter.

For the English-French language pair, and for each test set, we randomly picked 1 000 English words among those belonging to an in-house general bilingual lexicon, and for which one of their sanctioned translations belongs to the French Wikipedia vocabulary. This way, we eliminate the numerous proper names present in our lists (especially for low-frequency words): translating proper names may involve transliteration [23], an interesting problem which is not the focus of this study.

For the other language pairs, we did not have such a general bilingual lexicon. Therefore, we followed [7] and resorted to **Google Translate**¹¹ to translate the English words of our test sets. We removed pairs where the does not belong to the target Wikipedia vocabulary. Often, this application produces a translation identical to the source word. While it might happen that a word and its translation are identical in a given language pair, most often, this is a mistake of the application. We are not aware of any study that paid attention to this phenomenon. We found it problematic enough in our case¹² that we decided to remove from our test sets all the pairs involving a translation identical to its source word. Further, it happens that **Google Translate** does not translate a word, or that the translation produced is not part of the Wikipedia vocabulary. We removed those entries from our test sets. The main characteristics of our datasets are presented in Table 2.

¹⁰ www.statmt.org.

¹¹ <https://translate.google.com/>.

¹² For instance, 32% (resp. 50.8%) of the rare words we submitted to Google Translate for the German (resp. Romanian) test set received a translation identical to the source term.

Table 2. Number of pairs of words per target language considered. ϕ stands for the number of English words we could not translate with **Google Translate** or for which the translation was not part of the target Wikipedia; = indicates the number of pairs where the translation was identical to the source term; and $\tau/test$ stands for the split of the remaining pairs into TRAIN and TEST datasets.

	German			Spanish			Romanian			French
	ϕ	=	$\tau/test$	ϕ	=	$\tau/test$	ϕ	=	$\tau/test$	$\tau/test$
Wiki _{>25}	79	208	500/213	272	26	500/202	201	148	500/151	700/300
Wiki _{≤25}	276	317	300/107	385	149	300/166	131	484	300/85	700/300
Euro _{5-6k}	239	78	500/183	299	28	500/173	319	57	500/124	700/300

4.3 Reranking Protocol

For reranking experiments, we kept a number of entries of our test sets for training the classifier, and the remaining ones for testing. Based on the size of our datasets, we kept 700 entries for French, 500 for German and Spanish, and 300 for Romanian. Because this represents small quantities of material, we resorted to a 3-fold cross-validation procedure and report the average performance over the 3 folds. The results across folds are actually fairly stable.

Each approach has been configured to produce a ranked list of (at most) 100 candidate translations. We measure performance with accuracy at rank i , $@i$, computed as the percentage of test words for which a reference translation is identified in the first i candidates proposed.

5 Experiments

5.1 Calibration

For the English-French language pair, we compared hundreds of variants and came to the conclusion that there is no free lunch: for better performance, each approach should be adjusted to the specificity of the test words. For instance, for the **Rapp** approach, the variant performing the best on Wiki_{>25} is one with a window size of 6 (3 words before and after), and using PMI, while on Wiki_{≤25}, the best configuration is obtained by considering a window size of 30 and the discontinuous odd-ratio association measure. Similarly, for the **Miko** approach, the best configuration on frequent words (Wiki_{>25} and Euro_{5-6k}) is obtained by training embeddings with the *cbow* architecture, *negative sampling* (10 samples), and a window size of 10. For learning the translation matrix, we found the best results with a bilingual lexicon of size 5k, containing frequent words¹³, as suggested in [7]. For Wiki_{≤25}, however, a *skip-gram* architecture with a *hierarchical softmax*, and a window size of 20 yields the best performance.

¹³ For Euro_{5-6k}, we took the top 5k frequent words of the Europarl corpus, while for Wiki_{>25}, we took the 5k most frequent words of Wikipedia intersected with our in-house lexicon. Using a single lexicon for both test sets markedly decreases performance.

For the French-English language pair, the best performing configuration for each test set has been selected in the figures reported in this study. While this overestimates the performance on this language pair, our focus in this study is on reranking. Therefore, we assume that native approaches have been tuned specifically. For the other language pairs however, we kept the best configurations identified on the English-French language pair, which might not be optimal. This provides some data points on what happens when fine tuning is not performed.

Table 3. @1 of the native approaches and their (individual) reranking.

	German		French		Spanish		Romanian	
	Native	Rerank	Native	Rerank	Native	Rerank	Native	Rerank
Euro _{5-6k}								
Rapp	0.2	0.2	16.6	34.6	0.9	3.1	0.9	2.5
Miko	38.1	39.0	42.0	47.0	36.8	37.0	24.6	26.9
Faru	5.2	21.6	30.6	41.2	6.5	22.9	0.1	3.2
Wiki _{>25}								
Rapp	3.6	1.8	20.0	36.3	4.3	8.0	4.1	5.1
Miko	19.1	20.2	17.0	38.1	18.1	21.5	5.4	9.5
Faru	4.0	14.6	13.3	34.3	3.1	15.5	0.2	3.2
Wiki _{≤25}								
Rapp	1.0	0.4	2.6	8.6	1.0	1.5	0.2	0.5
Miko	0.2	1.2	1.6	16.6	0.4	3.1	0.0	0.0
Faru	0.7	4.6	1.6	5.0	0.9	9.1	0.0	2.5

5.2 Native Approaches and Their Reranking

Table 3 reports the results of the native approaches, as well as their individual reranking. This table calls for several comments. A first thing to note is the overall good performance of Miko on the Euro_{5-6k} test set. Variations do occur with the target language, Romanian being the most difficult to deal with. One explanation for this is the relatively small size of the Wikipedia Romanian collection, where many target words are seen only a few times, which puzzles the approach somehow. Still, the precision at rank 1 of Miko is 24.6%, while the two other approaches culminate at less than 1%! Another observation is the overall bad performance of the Rapp approach. Only on the Wiki_{>25} corpus, for the English-French translation direction, does it outperform Miko by 3 absolute @1 points. Similarly, the results of Faru are disappointing. This contradicts the observation made by their authors who reported better results. This might be explained by the different nature of the tasks they tested.

A second striking observation is the very disappointing results of all the approaches on the Wiki $_{\leq 25}$ test set, where for as less as 1% of the test words, could a translation be identified at rank 1. Enlarging the candidate list raises this figure slightly¹⁴, but not to a satisfactory level. This clearly indicates that seeking the translation of unfrequent words in a large collection of texts is by far an unsolved problem.

A third, and more positive, observation is the overall stable performance of the reranking approach. Gains are not spectacular, but (to a few exceptions) consistent across test sets, language pairs, and approaches. Considering the very light features set we considered, this somehow comes to a surprise, and calls for more feature engineering. The gains observed while reranking the n -best list produced by the **Faru** approach are actually rather impressive. For instance, reranking increases @1 from 6.5% to 22.9% for the English-Spanish language pair.

Figure 1 shows the average rank of the reference translation before and after reranking, for all the approaches and data sets for both the English-German and English-Spanish translation directions (similar patterns are observed for the English-Romanian language pair). Terms for which the reference translation was not in the native 100-best list were not considered. It is clear that the average rank of the reference translation does decrease significantly after reranking, often drastically. There is one notable exception for German, with the **Rapp** approach on the Euro $_{5-6k}$ test set, where the rank increased from 9 to 44 after reranking. The reason why it is so still needs to be investigated¹⁵. This data point apart, we were rather astonished of the gains in rank obtained by our simple reranker. Further analysis is provided in Sect. 5.4.

5.3 Combining Native n -best Lists by Reranking

Despite the stable gains we obtained when reranking individual n -best lists, there is no reason not to rerank all the candidate lists produced for a given test word. This is what we investigate hereafter. Table 4 reports the results of the best native approach per test set (the one recording the best @1), its individual reranking, as well as the reranking of the candidates produced by the three approaches (R+M+F). The latter case leads overall to the best performance, with one notable exception for the English-German language pair, where reranking the three candidate lists leads to a slight loss in @1 (37.87% versus 38.1%). Most of the time, reranking the output of the three approaches is preferable to reranking the best approach only. This demonstrates the complementarity of the approaches, and the efficiency of the simple rank features we exploited (see Sect. 3). The most impressive gains are observed for the Wiki $_{\leq 25}$ and Wiki $_{> 25}$ data sets, a test case scenario we think more interesting, for the reason previously mentioned. For instance, on Wiki $_{\leq 25}$, when seeking French translations, the best native approach (**Rapp**) performs 2.6% @1 while R+M+F achieves 21.3%.

¹⁴ The highest @20 (10.4%) is recorded by **Faru** when translating into Spanish.

¹⁵ Will be done for the final version.

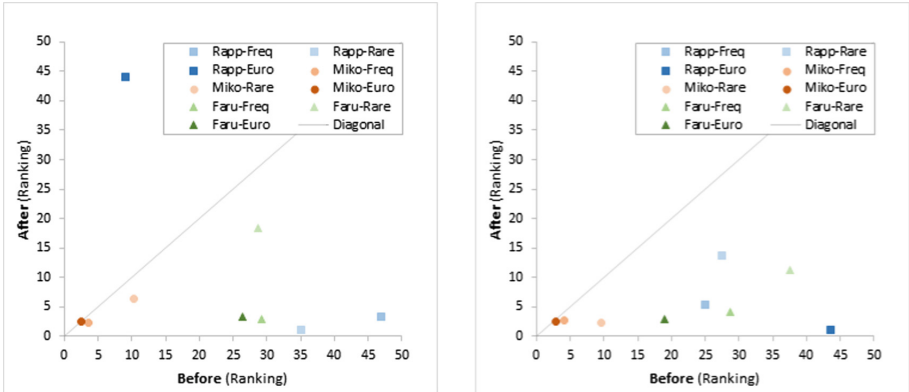


Fig. 1. Average rank of the reference translation before and after reranking, for each approach and test set for the English-German translation task (left) and the English-Spanish task (right). Average are computed on the only terms for which the reference translation was proposed in the top 100 candidate list of the native approach.

Looking at the performance of an oracle that picks the reference translation whenever it exists in one of the three n -best lists, we realize there is still a gap between the reranking gains obtained and the ones we could possibly achieve. The simplicity of the feature set we considered here is certainly to be blamed.

5.4 Analysis

We analyzed the output of the different approaches and their reranking. We observed in Sect. 5.2 that native approaches are weak when the source frequency is low. We also noted that native approaches are typically puzzled by cases where the frequency of the source term in the source collection differs significantly from the frequency of its translation in the target collection, especially when the source frequency is much lower than the target one. Overall, the approach of [7] is the one that delivers the most coherent candidate lists: candidates are often synonyms, antonyms or syntactic derivations of the expected translation.

We identified two patterns that characterize the reranker. First, it tends to prefer source and target pairs that have small edit-distance. Second, and to a less extent, it does prefer pairs of words where the difference between the source and target frequency is small.

Table 5 shows two examples of the top-4 candidates produced by each native approach, as well as their reranking. The gain of reranking can be important as in the second example where the reference translation gets a final rank of 3, while the best rank of a native approach was 44.

Table 4. @1 and @5 of the best native approach (according to @1), the best reranked native approach, as well as the reranking of the candidates produced by the 3 approaches (R+M+F). R stands for Rapp, M for Miko, and F for Faru. *oracle* picks the reference translation whenever it exists in one n -best list.

	German		French		Spanish		Romanian					
	@1	@5	@1	@5	@1	@5	@1	@5				
Euro _{5-6k}												
Best native	M	38.1	51.9	M	42.0	59.0	M	36.8	46.1	M	24.6	37.9
Best reranked	M	39.0	49.8	M	47.0	68.1	M	37.0	46.3	M	26.9	37.8
R+M+F		37.8	50.0		47.6	68.5		37.3	46.7		26.1	37.8
<i>Oracle</i>		<i>63.6</i>			<i>84.4</i>			<i>52.3</i>			<i>49.0</i>	
Wiki _{>25}												
Best native	M	19.1	27.8	R	20.0	33.0	M	18.1	26.0	M	5.4	10.1
Best reranked	M	20.2	27.9	M	38.1	49.0	M	21.5	27.2	M	9.5	13.3
R+M+F		21.1	29.6		45.6	59.6		23.9	30.7		13.0	17.8
<i>Oracle</i>		<i>49.6</i>			<i>69.3</i>			<i>42.1</i>			<i>26.0</i>	
Wiki _{≤25}												
Best native	R	1.0	2.6	R	2.6	4.3	R	0.9	1.6	R	0.2	0.7
Best reranked	F	4.6	6.0	M	16.6	19.0	F	9.1	9.4	F	2.5	3.2
R+M+F		5.0	6.4		21.3	24.4		8.9	10.2		3.2	4.1
<i>Oracle</i>		<i>14.1</i>			<i>28.6</i>			<i>14.1</i>			<i>5.7</i>	

Table 5. Top-4 translations produced by each native approach for two English-French term pairs (uncrushable/infroissable and brotherliness/confraternité), as well as their reranking. The last column indicates the rank of the reference translation in the top-100 candidates, or \emptyset if it is absent from a list.

uncrushable	infroissable				
Rapp	senente	imbroyables	pulvérix	attriteur	\emptyset
Miko	nattées	paumage	infroissable	mouillettes	3
Faru	roninson	ospovat	talanov	mouraviova	\emptyset
reranker	infroissable	incrustait	raquettistes	paludicroque	1
brotherliness	confraternité				
Rapp	qoudous	tâche	attentifs	crainte	44
Miko	joie	volonté	envie	intensité	68
Faru	observant	cueuillies	concordent	moisson	\emptyset
reranker	volonté	précepte	fraternel	désintéressé	3

6 Conclusion

We compared the distributional-based approaches of [1,7,22] for the task of identifying the translation of (English) words in different Wikipedia collections (German, French, Spanish and Romanian).

Our experiments suggest that we are terribly in need of approaches that are able to manage rare words, a test case scenario we feel is more useful, since frequent translation pairs are likely to be listed in existing bilingual lexicons. Overall, we found that the approach of [7] is more stable than the other two we tested. To our satisfaction, the reranking approach we developed, delivers stable and sometimes drastic gains over native approaches. Reranking several candidate lists is overall preferable. A reranker can be trained very rapidly on a few hundred pairs of translations, exploiting a very narrow feature set.

This work leaves open a number of issues that deserve further investigations. First, it is natural to extend the list of features we considered here. The work of [15] provides a list of promising ones that we want to consider, although some are specific to news texts or Wikipedia. Also, we only combined 3 approaches with our reranker, while others could be attempted. We noticed that for an approach to work, we should better adjust its meta-parameters to the task. Investigating the reranking of different variants of a given approach would be interesting, and might be a solution for avoiding to adjust one approach to a specific task.

Acknowledgments. This work has been partly funded by the FRQNT. We are grateful to reviewers for their insightful comments.

References

1. Rapp, R.: Identifying word translations in non-parallel texts. In: Proceedings of the 33rd ACL, pp. 320–322 (1995)
2. Sharoff, S., Rapp, R., Zweigenbaum, P.: Overviewing important aspects of the last twenty years of research in comparable corpora. In: Sharoff, S., Rapp, R., Zweigenbaum, P., Fung, P. (eds.) Building and Using Comparable Corpora, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-20128-8_1
3. Yu, K., Tsujii, J.: Extracting bilingual dictionary from comparable corpora with dependency heterogeneity. In: Annual Conference of the NAACL, Companion Volume: Short Papers, pp. 121–124 (2009)
4. Otero, P.G.: Learning bilingual lexicons from comparable English and Spanish corpora. In: Proceedings of MT Summit xI, pp. 191–198 (2007)
5. Daille, B., Morin, E.: Effective compositional model for lexical alignment. In: Proceedings of the 3rd IJCNLP, pp. 95–102 (2008)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of ICLR Workshop (2013)
7. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation. CoRR (2013)
8. Coulmance, J., Marty, J.M., Wenzek, G., Benhalloum, A.: Trans-gram, fast cross-lingual word-embeddings. In: Proceedings of EMNLP, pp. 1109–1113 (2015)

9. Vulic, I., Moens, M.F.: Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In: Proceedings of the 53rd ACL (2015)
10. Luong, T., Pham, H., Manning, C.D.: Bilingual word representations with monolingual quality in mind. In: Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, pp. 151–159 (2015)
11. Gouws, S., Bengio, Y., Corrado, G.: BilBOWA: fast bilingual distributed representations without word alignments. In: Proceedings of the 32nd ICML, pp. 748–756 (2015)
12. Duong, L., Kanayama, H., Ma, T., Bird, S., Cohn, T.: Learning crosslingual word embeddings without bilingual corpora. arXiv preprint [arXiv:1606.09403](https://arxiv.org/abs/1606.09403) (2016)
13. Upadhyay, S., Faruqui, M., Dyer, C., Roth, D.: Cross-lingual models of word embeddings: an empirical comparison. In: Proceedings of ACL (2016)
14. Levy, O., Søgaard, A., Goldberg, Y.: Reconsidering cross-lingual word embeddings. arXiv preprint [arXiv:1608.05426](https://arxiv.org/abs/1608.05426) (2016)
15. Irvine, A., Callison-Burch, C.: Supervised bilingual lexicon induction with multiple monolingual signals. In: Proceedings of NAACL-HLT, pp. 518–523 (2013)
16. Delpech, E., Daille, B., Morin, E., Lemaire, C.: Extraction of domain-specific bilingual lexicon from comparable corpora: compositional translation and ranking. In: Proceedings of COLING (2012)
17. Harastani, R., Daille, B., Morin, E.: Ranking translation candidates acquired from comparable corpora. In: Proceedings of the 6th IJCNL, pp. 401–409 (2013)
18. Kontonatsios, G., Korkontzelos, I., Tsujii, J., Ananiadou, S.: Combining string and context similarity for bilingual term alignment from comparable corpora. In: Proceedings of EMNLP, pp. 1701–1712 (2014)
19. Baldwin, T., Tanaka, T.: Translation by machine of complex nominals: getting it right. In: Proceedings of the Workshop on Multiword Expressions: Integrating Processing, pp. 24–31 (2004)
20. Jakubina, L., Langlais, P.: Reranking translation candidates produced by several bilingual word similarity sources. In: 15th Conference of the European Chapter of the Association for Computational Linguistics, April 2017
21. Dinu, G., Baroni, M.: Improving zero-shot learning by mitigating the hubness problem. CoRR (2014)
22. Faruqui, M., Dyer, C.: Improving vector space word representations using multilingual correlation. In: Proceedings of EACL (2014)
23. Li, H., Kumaran, A., Pervouchine, V., Zhang, M.: Report of news 2009 machine transliteration shared task. In: Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, NEWS 2009, pp. 1–18 (2009)



Analysis of Social Media Posts for Early Detection of Mental Health Conditions

Antoine Briand, Hayda Almeida, and Marie-Jean Meurs^(✉)

Université du Québec à Montréal, Montréal, QC, Canada
meurs.marie-jean@uqam.ca

Abstract. This paper presents a multipronged approach to predict early risk of mental health issues from user-generated content in social media. Supervised learning and information retrieval methods are used to estimate the risk of depression for a user given the content of its posts in reddit. The approach presented here was evaluated on the CLEF eRisk 2017 pilot task. We describe the details of five systems submitted to the task, and compare their performance. The comparisons show that combining information retrieval and machine learning methods gives the best results.

Keywords: Artificial intelligence · Classification
Information retrieval · Machine learning · Mental health
Natural language processing · Social media · Text mining

1 Introduction

Social media content has been commonly utilized to develop approaches towards the analyze of user behavior. Fields of application for such research are numerous, from advertisement targeting to user profiling or sentiment analysis [14]. Users constantly share their feelings, behaviors, and reactions in social networks, microblogs, forums, or communities (*e.g.* reddit¹). This rich content can be specially useful in health related applications, for example to detect patients likely to present mental health issues [25]. Since early risk detection is of the utmost importance for mental health practitioners, the analysis of user-generated content can help them providing better support to users in need [2]. In the research community, shared tasks are organized with the motivation of promoting the development of tools to support automatic health issue detection. Examples are the CLPsych Shared Task², and the CLEF eRisk pilot task³.

In this paper, we describe a prediction system based on an ensemble classification approach that was presented at the CLEF eRisk pilot task 2017. eRisk 2017 proposed the challenge of identifying as quickly as possible – *i.e.* using as little content as possible – the risks of depression arising from users' productions. The proposed system performs early risk detection based on user-generated content

¹ <https://www.reddit.com/>.

² <http://clpsych.org/shared-task-2017/>.

³ <http://early.irlab.org/>.

from `reddit`. This is achieved by combining supervised learning and information retrieval methods to identify `reddit` users susceptible to depression.

The following Section introduces some previous works. Section 3 describes our methodology, Sect. 4 provides details on the task dataset, and Sect. 5 presents the obtained results, which are then discussed in Sect. 6.

2 Previous Work

Detection of depression from content shared in social networks can be of critical importance in suicide prevention, especially regarding teens or young adults. The potential of automatic approaches towards this goal is huge, given the great popularity of social networks. For example, Facebook has more than 2 billion members posting several hundred million messages a day⁴. Industrial and academic initiatives are multiplying. Facebook already set up a suicide detection and prevention system on part of the platform⁵. Instagram is also often the scene of photo exhibitions or depressive behaviours. Based on the textual descriptions of users' photos, and relevant mentions such as `#selfharmmm` and associated tags, the authors of [21] have succeeded in identifying a large number of cases of non suicidal self-harm.

Several studies have been triggered by the CLPsych Shared Tasks [6, 11, 20] that require participants to predict users in eminent risk of depression or Post Traumatic Stress Disorder (PTSD) by analyzing their tweets or mental health forum posts. Based on sentiment lexicons and intensity, the authors of [12] built a rule-based sentiment analysis model providing promising results on social media content. In [22], the authors evaluated the usage of different features to analyze user posts from LiveJournal⁶, and compared discrepancies of posts from depression related versus control online communities. Based on the analysis of over 176 million tweets, the authors of [19] used a statistical method to identify communication patterns related to mental illness in Twitter, and to attempt predicting user behavioral patterns related to depression. Also in [7], tweets from 554 users that attempted to take their own life were analyzed for signs prior to suicide. The authors identified important differences in the language of such users, compared to the language used by neurotypicals.

Many studies make use of supervised learning methods. For instance, the work presented in [8] used a Support Vector Machine (SVM) classifier and behavioral features to predict depression from tweets, relying on over 2 million posts from 476 users. In [16], a four layer Deep Neural Network (DNN) was trained on tweets to identify user psychological stress.

Information retrieval techniques are also widely used to perform knowledge discovery in the field of mental health. In [10], the authors presented an interesting study to support mental health maintenance of U.S. army soldiers. The goal

⁴ <https://newsroom.fb.com/company-info/>.

⁵ <https://newsroom.fb.com/news/2017/03/building-a-safer-community-with-new-suicide-prevention-tools/>.

⁶ <http://www.livejournal.com>.

was to aid health practitioners to perform efficient follow-ups on soldiers, since the suicide attempt rate among them is known to be high. The approach made use of the Veterans Informatics and Computing Infrastructure (VINCI) resource to process mostly unstructured health information, such as clinical notes. The authors built a search engine indexing these textual data to predict the risk of suicide attempt among soldiers. Given the promising results achieved by all these approaches, we designed our system to combine supervised learning and information retrieval methods.

3 Methodology

In order to adopt a multipronged approach for early risk detection, our pipeline relies on two sub-systems: the production of users to classify is processed separately by a sub-system based on information retrieval and another sub-system based on supervised learning. The prediction output of each sub-system is then merged based on a decision algorithm, which provides a final prediction for content from a given user: *risk* or *non-risk*. Although our approach was developed utilizing the eRisk dataset, the sub-systems are generic enough to easily process new datasets. We will describe in Sect. 3.1 the approach used for the sub-system based on supervised learning system, while the approach used for the sub-system based on information retrieval will be presented in Sect. 3.2.

3.1 Supervised Learning Method

The Supervised Learning (SL) method relies on three classification algorithms, and four feature types (n-grams, Part-Of-Speech (POS), dictionary words, and user posting frequency).

Features. N-grams considered to train our models were Bag-Of-Words (BOW) and bigrams. Experiments were performed using trigrams, however the preliminary results did not show much improvement in performance compared to bigrams. Selected POS were adjective, noun, predeterminer (words preceding a determiner, such as *all* or *both* [26]), particle, and verb. Dictionary words were extracted based on a list of feelings⁷, as well as lists of drugs, medicine, and disease names related to depression⁸. Finally, we computed a posting frequency value for each user in the dataset. The posting frequency is a ratio that represents how often a user posts, by considering the oldest and the most recent writing dates, divided by the total number of writings of a user. Table 1 shows the feature statistics in the training dataset for the eRisk 2017 shared task.

Classifiers. The three supervised learning models were built based on Logistic Model Tree (LMT) [15] classifiers, an Ensemble of Sequential Minimal Optimization (SMO) [23] (ens_SMO) classifiers, and an Ensemble of Random Forests [3]

⁷ Obtained from a conceptual feature map in SenticNet [5].

⁸ Obtained from Wikipedia pages for “depression”, “psychoactive drugs”, and “list of antidepressants”.

Table 1. Number of unique features in the eRisk 2017 training dataset

	# Features
BOW	105, 161
Bigrams	1, 544, 714
Trigrams	3, 397, 459
Selected POS	118, 139
Feelings dic.	205
Medicine dic.	30
Drugs dic.	57
Diseases dic.	43

(ens.RF) classifiers. SVM, which is similar to SMO, and Random Forest classifiers are commonly applied in comparable tasks [20] showing good performance. LMTs are also found to perform well in these tasks, especially when handling imbalanced datasets [1]. The ensemble classifiers are composed by 30 single classifiers. The 30 classifiers composing the ens.RF are each Random Forest classifiers designed with iteration values from 10 to 50 (with increments of 10), and tree depth values from 2 to 10 (with increments of 2), or unlimited. The 30 classifiers composing the ens.SMO are each SMO classifiers designed with tolerance values from 0.001 to 0.005 (with increments of 0.001), and epsilon for round-off error values from 1 to 5 (with increments of 1).

3.2 Information Retrieval Method

The intuition behind the Information Retrieval (IR) based approach is that if unseen posts from a new user are semantically close to posts from risk users, this new user might be at risk as well. The sub-system based on information retrieval hence considers a new user posts as a query, and submits this query to a search engine that mines the training corpus. To build the search engine, we indexed the training corpus using Apache Solr⁹. The Okapi BM25 [13] algorithm is utilized to retrieve documents close to the submitted query. The classification decision is made as follows: All the posts d of a new user are sent to the search engine as a query. Then the top n retrieved documents are collected to compute the score $S_{IR}(d)$ as shown in Eq. 1. $S_{IR}(d)$ reflects how likely d has been produced by a depressed user.

$$S_{IR}(d) = \frac{1}{n} \sum_{i=1}^n \delta(d_i) \quad (1)$$

where d_i is the document retrieved by query d in position i , and

$$\delta(d_i) = \begin{cases} 1, & \text{if } d_i \text{ is labeled as } \textit{risk} \\ 0, & \text{otherwise} \end{cases}$$

⁹ <http://lucene.apache.org/solr/>.

Table 2. Pre-processing steps

	Preprocessing
Index	Tokenization
	Lowercasing
	Stemming
	Stopwords
	Punctuation

Table 3. Indexed fields

#	Indexed fields
1	Title
2	Content
3	User label
4	Text (fields 1 + 2)

Indexes. All the fields available in user posts from the training set were selected to build an index supporting the search engine. Prior to build this index, the documents were pre-processed using tokenization, lowercasing, stop word removal, stemming (by the Porter stemmer), and punctuation filtering. Table 2 summarizes these pre-processing steps. Table 3 presents the fields used in the index schema, *i.e.* all the fields available in the corpus (title, content, label). The “Text” field is a copy field, which contains both content and title, and which is used as the default search field.

Experimental Settings. The number of retrieved documents taken into account and the $S_{IR}(d)$ threshold above which a user is considered at risk were set after running several tests. The values were respectively set to 20 and 0.7, since it maximized the F1 score of the sub-system.

4 Dataset

Our experiments rely on the CLEF eRisk 2017 dataset [18]. This dataset has been built by retrieving the production of target users on `reddit`. Then, part of its content has been manually annotated by experts, classifying a user either as risk (depressed) or non-risk (non-depressed).

Table 4 presents the detailed statistics of the training and test datasets. Both training and test datasets are split in 10 chunks. Each chunk contains 10% of annotated user-generated content within a specific period in time, and is organized in a chronological order. Each document in a chunk refers to a specific user, and contains at least one `reddit` post. All training chunks were used to develop our multipronged approach. The test chunks were released, and therefore processed, weekly.

From observing the dataset statistics in Table 4, it is possible to notice the imbalance between the number of risk and non-risk posts. Such an imbalanced distribution may bias some classifiers, which can make a supervised learning task harder. In addition to this, the risk or non-risk user classification for eRisk is already of challenging nature since the content of user writings might seem rather ambiguous at times. The sentences below show a few samples of the user writing content for a non-risk user in Table 5, and a risk user in Table 6.

Table 4. Statistics on the eRisk 2017 pilot task dataset

	Training dataset		Test dataset	
	Non-risk	Risk	Non-risk	Risk
# users	403	83	349	52
# writings	264,172	30,851	217,665	18,706
Avg. # writings per user	655.5	371.7	623.7	359.7
Avg. length of writing period	626.6	572.7	623.2	608.31
Avg. # words per writing	21.3	27.6	22.5	26.9

Table 5. Sample sentences from a non-risk user

"It's been awesome! Although it was tough sometimes.
I had the best experience in my life!"

"My boyfriend has depression and I don't know
what to do anymore?"

"Thanks for the help. It's nice to hear
that you think I'm doing good."

Table 6. Sample sentences from a risk user

"The sun was shining in my face and I was thinking:
'This feels pretty good, I think I am really happy'."

"I am glad I see my therapist tomorrow, I certainly
need to vent to someone."

"I'd most likely be dead if it weren't
for my antidepressants."

5 Experiments and Results

The results achieved in the CLEF eRisk pilot task are presented hereafter. To evaluate and compare the performance of several approaches, we participated with five systems. The detailed configuration of each system is as follows:

- SYS-A: based on the decision algorithm described below.
- SYS-B: based on the information retrieval results only (our baseline).
- SYS-C: a LMT classifier, using either BOW or bi-grams separately, and BOW or bi-grams together with all the dictionary features.
- SYS-D: an ens_RF classifier, using either BOW or bigrams together with all the dictionary features.
- SYS-E: an ens_SMO classifier, using bigrams separately and together with all the dictionary features.

The user posting frequency was used by the five systems.

Decision Algorithm for system SYS-A. The decision algorithm merges the predictions from both SL and IR sub-systems. Each candidate is associated with a S_{IR} score. The SL-based classifiers are used to refine the list of existing candidates, and add new candidates if they were ranked first. A new user is classified at risk according to the value of $\Delta(d)$ with:

$$\Delta(d) = \mathbb{1}_{IR}(d) + \mathbb{1}_{SL}(d) + \mathbb{1}_{SLf}(d)$$

where d represents the new user posts, and $\mathbb{1}_{IR}$, $\mathbb{1}_{SL}$, $\mathbb{1}_{SLf}$ are the indicator function of the risk class respectively associated to the IR candidates, SL candidates, and SL first ranked candidates. Indicator functions return 1 if the decision is to classify a user as risk (of depression), or 0 if the user is to be classified as non-risk.

If $\Delta(d) \geq 2$, the user who produced d is assigned the risk class.

Evaluation Metrics. The Early Risk Detection Error (ERDE) [17] metric was one of the performance metrics used in the eRisk task to account for the point in time when the system properly predicts the user as risk, either rewarding an early classification or penalizing a late one. To account for the late decision of a given system, the ERDE metric considers the number of chunks and writings that were needed to output a decision for a given user.

Additionally, the errors between the non-risk and risk classes are weighted differently, since these classes are imbalanced in the dataset. This is achieved by using a specific cost function, which balances the cost of false positive outputs to be penalized according to the number of risk writings in the test set.

In [17] the authors define the ERDE metric for a decision dec taken by a given system with delay k as:

$$ERDE_o(dec, k) = \begin{cases} c_{fp} & \text{if } dec = \text{positive AND ground truth} = \text{negative (FP)} \\ c_{fn} & \text{if } dec = \text{negative AND ground truth} = \text{positive (FN)} \\ lc_o(k) \cdot c_{tp} & \text{if } dec = \text{positive AND ground truth} = \text{positive (TP)} \\ 0 & \text{if } dec = \text{negative AND ground truth} = \text{negative (TN)} \end{cases}$$

where c_{fp} and c_{fn} are the costs of false positives and false negatives respectively, $lc_o(k) \in [0, 1]$ is a cost that increases according to the value of k , and computes a cost for late decision of true positives. This cost can be seen as a penalty that grows faster after a number of o user's writings have been observed. $lc_o(k)$ is defined as follows:

$$lc_o(k) = 1 - \frac{1}{1 + e^{(k-o)}}$$

Depending on the value of o , taking a late decision is more or less expensive. For the eRisk task, values of 5 and 50 were used.

Standard evaluation metrics as Precision (P), Recall (R) and F-measure (F1) on the positive class (risk) were also used, though not providing any insight into the time/number of writings needed to make a decision.

Table 7. Performance results of our systems on the eRisk 2017 test set

	ERDE 5	ERDE 50	F1	P	R
SYS-A	14.03%	12.29%	0.53	0.48	0.60
SYS-B	13.78%	12.78%	0.48	0.49	0.46
SYS-C	13.58%	12.83%	0.42	0.50	0.37
SYS-D	13.23%	11.98%	0.38	0.64	0.27
SYS-E	13.68%	12.68%	0.39	0.45	0.35
Avg. 30 systems	14.68%	12.76%	0.40	0.37	0.51

Table 8. Best performance for each metric across teams on the eRisk 2017 pilot task

	ERDE 5	ERDE 50	F1	P	R
FHDOB	12.70%	10.39%	0.55	0.69	0.46
UNSLA	13.66%	9.68%	0.59	0.48	0.79
FHDOA	12.82%	9.69%	0.64	0.61	0.67
UArizonaC	17.93%	12.74%	0.34	0.21	0.92

Results. The performance achieved by each of our systems is shown in Table 7. The results are presented in terms of $ERDE_5$, $ERDE_{50}$, F-measure (F1), Precision (P), and Recall (R). The last line in the table shows the average scores for the 30 systems participating in the task. In order to evaluate our models, we focused on Recall. Since the goal of the task is to identify users at risk of depression, it is preferable to identify more users as risk, who are actually non-risk (false positives), than to miss any of the risk users while aiming for Precision.

For the sake of comparison, we also present in Table 8 the best performance for each metric across all teams participating in the eRisk pilot task. The highest F1 score obtained among all submissions was 0.64, while the highest Precision score was 0.69. According to the task organizers, the overall rather low performance results are due mainly to two reasons [18]. First, the task difficulty in general, and second the way the dataset was organized, since it contained a large variety of user content including users who were only interested in the depression topic, or who had relatives suffering from depression.

Our best results are obtained by the SYS-A system, both in terms of F1 and Recall. This system combines supervised learning and information retrieval approaches, paving the way for more investigation. Moreover, the best Precision is achieved by SYS-D, which is designed based on an ens_RF classifier, and could probably be used with different parameters to enhance the precision of SYS-A.

Evaluation on Related Tasks. Automatic detection of mental health issues from user data is a topic explored by several authors, and shared tasks. Apart from the CLEF eRisk task, another well known effort in the community is the

CLPsych workshop¹⁰. In CLPsych, user generated content in a mental health forum is labeled according to one of four categories (crisis, red, amber, green). Similarly to eRisk, the CLPsych task is also challenging due to the natural imbalance between categories in the data. Usually, the categories of most interest (such as *risk* for eRisk, and *red* or *crisis* for CLPsych) are the least represented in the data.

Common approaches applied by CLPsych participants involve Logistic Regression, Random Forest, SVM or classifier ensembles. For the CLPsych 2016 task, the baseline system was composed of unigrams, bigrams, and a default Logistic Regression classifier, and it was compared to the task participant systems using a macro-averaged F1, which considers the performance of all four categories together. The baseline achieved a macro-averaged F1 of 0.31.

Even though eRisk and CLPsych tasks have different aspects, analyzing the performance achieved from CLPsych in this context is useful to provide insights on the difficulty such tasks can present.

Insights from Posts Written by Risk Users. By analyzing the content of risk user productions selected by our systems, we observed that two major topics were often discussed: “video games”, and “sexuality or relationship issues”. The co-occurrence of these two topics and depression was intuitively noticed, and could indicate very interesting and fairly realistic associations between these topics. As a matter of fact, the connection between these topics has been studied from a clinical perspective in several recent works [4,9,24,27]. We found very exciting the convergence between clinical investigation and findings based on natural language processing approaches regarding these topics. This is one of the research directions we will further explore.

6 Conclusion and Future Work

This paper presented a multipronged approach to perform early risk detection of depression in social media posts. The five systems described here were presented at the CLEF eRisk 2017 pilot task. The best performance was achieved by the system that combined supervised learning and information retrieval approaches.

The ongoing improvement of the presented approach currently relies on the contribution of a recurrent neural network added to the list of classifiers, and the indexing of features related to depression to boost the capabilities of the search engine.

Reproducibility. To ensure full reproducibility and comparisons between systems, our source code is publicly released as an open source software in the following repository: <https://github.com/BigMiners/eRisk2017>.

¹⁰ <http://clpsych.org/>.


References

1. Almeida, H., Queudot, M., Meurs, M.J.: Automatic triage of mental health online forum posts: CLPsych 2016 system description. In: Proceedings of the Third Workshop on Computational Linguistics and Clinical Psychology, pp. 183–187 (2016)
2. Ayers, J.W., Althouse, B.M., Allem, J.P., Rosenquist, J.N., Ford, D.E.: Seasonality in seeking mental health information on Google. *Am. J. Prev. Med. (AJPM)* **44**(5), 520–525 (2013)
3. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
4. Brunborg, G.S., Mentzoni, R.A., Frøyland, L.R.: Is video gaming, or video game addiction, associated with depression, academic achievement, heavy episodic drinking, or conduct problems? *J. Behav. Addict.* **3**(1), 27–32 (2014)
5. Cambria, E., Olsher, D., Rajagopal, D.: SenticNet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, pp. 1515–1521. AAAI Press (2014)
6. Coppersmith, G., Dredze, M., Harman, C., Hollingshead, K., Mitchell, M.: CLPsych 2015 shared task: depression and PTSD on Twitter. In: Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology (CLPsych): From Linguistic Signal to Clinical Reality, pp. 31–39 (2015)
7. Coppersmith, G., Ngo, K., Leary, R., Wood, A.: Exploratory analysis of social media prior to a suicide attempt. In: Proceedings of the 3rd Workshop on Computational Linguistics and Clinical Psychology (CLPSych), pp. 106–117 (2016)
8. De Choudhury, M., Gamon, M., Counts, S., Horvitz, E.: Predicting depression via social media. In: Proceedings of the 7th International AAAI Conference on Weblogs and Social Media (ICWSM), p. 2 (2013)
9. Granic, I., Lobel, A., Engels, R.C.: The benefits of playing video games. *Am. Psychol.* **69**(1), 66 (2014)
10. Hammond, K.W., Laundry, R.J., OLeary, T.M., Jones, W.P.: Use of text search to effectively identify lifetime prevalence of suicide attempts among veterans. In: Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS), pp. 2676–2683. IEEE (2013)
11. Hollingshead, K., Ireland, M.E., Loveys, K.: Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology—From Linguistic Signal to Clinical Reality (2017)
12. Hutto, C.J., Gilbert, E.: VADER: a parsimonious rule-based model for sentiment analysis of social media text. In: Proceedings of the 8th International AAAI Conference on Weblogs and Social Media (ICWSM), June 2014
13. Jones, K.S., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Inf. Process. Manag.* **36**(6), 809–840 (2000)
14. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web (WWW), pp. 591–600. ACM (2010)
15. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. *Mach. Learn.* **59**(1–2), 161–205 (2005)
16. Lin, H., Jia, J., Guo, Q., Xue, Y., Li, Q., Huang, J., Cai, L., Feng, L.: User-level psychological stress detection from social media using deep neural network. In: Proceedings of the 22nd ACM International Conference on Multimedia, pp. 507–516. ACM (2014)

17. Losada, D.E., Crestani, F.: A test collection for research on depression and language use. In: Fuhr, N., Quaresma, P., Gonçalves, T., Larsen, B., Balog, K., Macdonald, C., Cappellato, L., Ferro, N. (eds.) CLEF 2016. LNCS, vol. 9822, pp. 28–39. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44564-9_3
18. Losada, D.E., Crestani, F., Parapar, J.: eRISK 2017: CLEF lab on early risk prediction on the internet: experimental foundations. In: Jones, G.J.F., Lawless, S., Gonzalo, J., Kelly, L., Goeuriot, L., Mandl, T., Cappellato, L., Ferro, N. (eds.) CLEF 2017. LNCS, vol. 10456, pp. 346–360. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65813-1_30
19. McClellan, C., Ali, M.M., Muttter, R., Kroutil, L., Landwehr, J.: Using social media to monitor mental health discussions - evidence from Twitter. *J. Am. Med. Inform. Assoc. (JAMIA)* (2016). <https://doi.org/10.1093/jamia/ocw133>
20. Milne, D.N., Pink, G., Hachey, B., Calvo, R.A.: CLPsych 2016 shared task: triaging content in online peer-support forums. In: CLPsych@ HLT-NAACL, pp. 118–127 (2016)
21. Moreno, M.A., Ton, A., Selkie, E., Evans, Y.: Secret society 123: understanding the language of self-harm on Instagram. *J. Adolesc. Health* **58**(1), 78–84 (2016)
22. Nguyen, T., Phung, D., Dao, B., Venkatesh, S., Berk, M.: Affective and content analysis of online depression communities. *IEEE Trans. Affect. Comput.* **5**(3), 217–226 (2014)
23. Platt, J.: Sequential minimal optimization: a fast algorithm for training support vector machines. Technical report MSR-TR-98-14, Microsoft, April 1998
24. Ramrakha, S., Paul, C., Bell, M.L., Dickson, N., Moffitt, T.E., Caspi, A.: The relationship between multiple sex partners and anxiety, depression, and substance dependence disorders: a cohort study. *Arch. Sex. Behav.* **42**(5), 863–872 (2013)
25. Rice, S.M., Goodall, J., Hetrick, S.E., Parker, A.G., Gilbertson, T., Amminger, G.P., Davey, C.G., McGorry, P.D., Gleeson, J., Alvarez-Jimenez, M.: Online and social networking interventions for the treatment of depression in young people: a systematic review. *J. Med. Internet Res. (JMIR)* **16**(9), e206 (2014)
26. Santorini, B.: Part-of-speech tagging guidelines for the Penn Treebank project, 3rd revision. Technical reports (CIS), p. 570 (1990)
27. Schou Andreassen, C., Billieux, J., Griffiths, M.D., Kuss, D.J., Demetrovics, Z., Mazzone, E., Pallesen, S.: The relationship between addictive use of social media and video games and symptoms of psychiatric disorders: a large-scale cross-sectional study. *Psychol. Addict. Behav.* **30**(2), 252 (2016)



Motor Bearing Fault Diagnosis Using Deep Convolutional Neural Networks with 2D Analysis of Vibration Signal

M. M. Manjurul Islam and Jong-Myon Kim^(✉) 

School of Electrical, Electronics and Computer Engineering,
University of Ulsan, Ulsan, South Korea
m.m.manjurul@gmail.com, jongmyon.kim@gmail.com

Abstract. Bearings are critical components in rotating machinery, and it is crucial to diagnose their faults at an early stage. Existing fault diagnosis methods are mostly limited to manual features and traditional artificial intelligence learning schemes such as neural network, support vector machine, and k -nearest-neighborhood. Unfortunately, interpretation and engineering of such features require substantial human expertise. This paper proposes an adaptive deep convolutional neural network (ADCNN) that utilizes cyclic spectrum maps (CSM) of raw vibration signal as bearing health states to automate feature extraction and classification process. The CSMs are two-dimensional (2D) maps that show the distribution of cycle energy across different bands of the vibration spectrum. The efficiency of the proposed algorithm (CSM+ADCNN) is validated using benchmark dataset collected from bearing tests. Experimental results indicate that the proposed method outperforms the state-of-the-art algorithms, yielding 8.25% to 13.75% classification performance improvement.

Keywords: Convolutional neural network · Cyclostationary signal analysis
Feature extraction · Fault diagnosis · Vibration analysis
Data-driven diagnostic

1 Introduction

Induction motors have undeniable presences in pivotal industrial systems such as wind turbine, aircraft, automotive, and power generator [1]. A motor is composed of bearing, stator and rotor, however bearing is the most frequent failing component, and responsible for 50% of all failures [1]. Thus, fault diagnosis of bearings is paramount importance to keep the machines operating normally and reduce breakdown time.

Fortunately, the rapid development of data mining, data acquisition technique, and machine learning technique since 1990 has provided the ability collect and store a vast amount process data to extract useful information inherent in such a significant amount of recorded data (*e.g.*, vibration, acoustic emission and current) for the purpose of reliable fault diagnosis in the large-scale industries [2, 3]. Therefore, it is needed to develop fault diagnosis methods that can efficiently process massive data to self-learn fault features and intelligently obtain accurate diagnosis results. Traditional data-driven based fault diagnosis methods consist of two main categories: manual fault feature

extraction by signal processing techniques and identification of faults using the extracted features [4, 5]. These manual-crafted features require careful engineering and substantial domain expertise that transform the raw data (*e.g.*, each time samples values of a one-dimensional (1D) bearing fault signal) into an appropriate feature vector or internal representation from which learning system, for instance, a classifier, could classify patterns in the input. For example, Jack and Nandi implemented neural network and support vector machine models on extracted time-domain features for bearing fault detection and diagnosis [5]. In 2016, Islam et al. explained the utilization of parameter combinations through a hybrid feature extraction model that aimed at extracting as much as information about bearing defect to define each bearing fault condition uniquely and selected features subset is further utilized with a k -NN classifier to identify fault types [4]. Therefore, the data-driven diagnosis methods, in [4, 6, 7], are facing some major challenges in the age of big-data. For examples, features are selected for a specific diagnostic problem and might not be appropriate for different fault diagnostic problems. Another challenge is that the neural networks used in most approaches are constructed with only a single hidden layer. Such a shallow structure confines their capability to learn and explore coveted non-linear information adaptively. Therefore, we incorporate a deep learning technique to resolve these problems.

Recently, manifold literature has reported rotating machinery fault diagnosis using deep learning approaches. Janssens et al. proposed a fault diagnosis method using convolutional neural networks (CNN) and raw vibration signal pre-processing [6]. In that approach, first, one-second windows of raw vibration signals are extracted. For each window of obtained samples, the discrete Fourier transform (DFT) is calculated. The amplitudes of the DFT decompositions are then used as training data samples for the CNN model. However, that kind of uncorrelated reshaping may not represent distinct bearing health states, and this process is very similar to use a 1D raw signal with a domain change. The issues with the existing method are that they do not consider the appropriate bearing health conditions and the physical characteristics of the faults, which might be the reason of degraded classification performance.

In this paper, we develop an appropriate 2D visualization tool to represent bearing health states. Whenever any bearing fault occurs at any of its raceways (*e.g.*, outer, inner, and ball), it excites a resonance in the system at a high frequency. The impulse frequency of the failure modulates this resonant frequency. Consequently, a demodulation technique is often required to efficiently reveal the frequency of impulse repetition since bearing fault signals are 2nd order cyclostationary signals [8]. Most of the existing demodulation approaches are adversely affected by noise and interferences vibration. However, the cyclostationary analysis that is capable of showing the signal characteristics regarding cyclic frequencies and spectral frequencies [8]. Thus, the visualization of cyclic frequencies and spectral frequencies in 2D cycle spectrum map (CSM), using an improved cyclostationary analysis, which is highly efficient to represent each health state distinctly.

As CSM is highly efficient to visualize each bearing health condition, but most of the existing studies apply no classifier at all to diagnose fault types [9, 10]. Once we obtain a CSM about the bearing health state, an adaptive convolutional neural network (ADCNN), a variant of LeNet5 [11] architecture, is proposed to automate the feature extraction and optimal feature selection processes by recognizing patterns from the

image pixels. Furthermore, we apply an adaptive learning rate for training ADCNN. The proposed diagnosis methodology (CSM+ADCNN) is validated on publicly available benchmark bearing vibration data from Case Western Reserve University (CWRU) lab [12, 13].

The remainder of this paper is structured as follows. Section 2 provides proposed fault diagnosis scheme including data acquisition system, proposed cycle spectrum map (CSM), and offered ADCNN architecture for fault classification. Section 3 validates the proposed method's effectiveness and compares its performance with the state-of-the-art methods. Finally, Sect. 4 summarizes the conclusion of this paper.

2 The Methodology

The main steps of the proposed ADCNN-based bearing fault diagnosis are shown in Fig. 1. The proposed method consists of three significant steps. After data acquisition, the first step is to obtain 2D cycle spectrum map (CSM) using an improved cyclostationary analysis to pre-process the vibration data. The main advantage of using such pre-processing is that the valuable information about how rotating components are distributed within vibration spectrum. The second step is to feed the pre-processed 2D matrix (CSM) into an ADCNN for an optimized deep learning model. The final step is to diagnose faulty bearings using that optimized model.

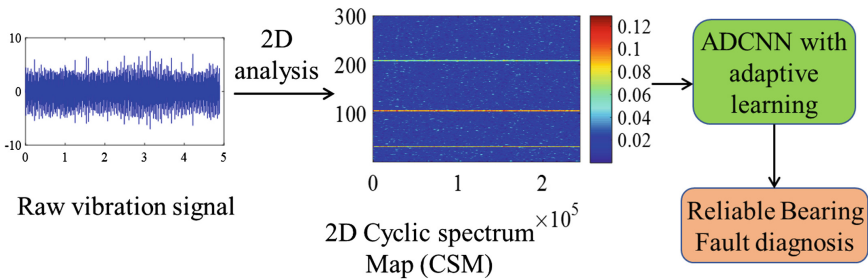


Fig. 1. An overall structure of the ADCNN-based method for bearing fault diagnosis

2.1 Experimental Data

The proposed methodology is tested on the publicly available seeded fault test data of CWRU Lab [13]. The data were collected using a 2-horsepower (hp) motor with a dynamometer and a torque transducer. The dynamometer is used to apply 1hp loads on the bearing. Figure 2 illustrates the detailed location of each component. In this study, the vibration acceleration signals were recorded at 48,000 sampling/second (Hz) for the drive-end bearings for analysis. The vibration signals are recorded at 1730 rpm speed for a healthy bearing condition (HBC) and bearings seeded with three types of defects, i.e., outer raceways fault (ORF), inner raceway fault (IRF), and ball raceway fault (BRF) for fault classification. A total of 110 vibration signals are configured for each fault type. Table 1 presents the details description of the dataset.

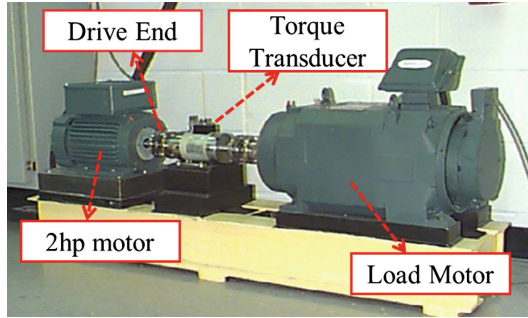


Fig. 2. The seeded bearing test ring for recording fault data

Table 1. The details description of dataset used in this paper

Fault types	Rotational speed (rpm)	No. of signals	Motor load (hp)
Outer raceway fault (ORF)	1730	110	1
Inner raceway fault (IRF)	1730	110	1
Ball raceway fault (BRF)	1730	110	1
Healthy bearing condition (HBC)	1730	110	1

2.2 CSM for Bearing Health Condition Visualization

As explained in Sect. 1, bearing fault signal are inherently cyclostationary signals. Unlike the stationary signal, the cyclostationary signal contains additional information due to their hidden periodicities in their structure [8]. Traditional, Fourier transform is not useful in describing cyclostationary signal while it is needed cyclostationary analysis (CA) [8]. The aim of CA is describing the cyclostationary components using two frequency variables named as cyclic frequency and spectral frequency. The spectral frequency band represents the parts of fault signal that corresponds to the resonance areas of the bearing mechanical system while the cyclic frequency components of the phenomenon describe the periodicity of the faults.

The CA is based on two Fourier transform (FT) steps. In the first step, a series of short-time FTs are applied for the transformation to the spectral frequency domain and for the transformation of the time axis of the spectrogram to the cyclic frequency domain, and then the second step FT is used after demodulation by taking the square of the spectrum. CA might be a fast calculation since it takes the power of signal based on squaring the spectrum. However, CA-based spectral frequency vs. cyclic frequency visualization may increase false alarm.

To reduce the shortcomings of CA, cycle spectrum map (CSM) is proposed based on short-time envelope analysis. To calculate the Fourier transform of an input signal $x[n]$, we define $X[N]$ as follows,

$$X[N] = \sum_{n=0}^{K-1} x[n] e^{-2\pi j \frac{kn}{K}} \quad (1)$$

Instead of the power signal calculation in [8], we take the envelope spectrum of the signal, $x[n]$, which is defined as follows,

$$y = |a(x)|^2 \quad (2)$$

Where,

$$a(t) = x(t) + i\bar{x}(t), \quad i = \sqrt{-1} \quad (3)$$

$$\text{and, } \bar{x} = \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau \quad (4)$$

where, \bar{x} is the Hilbert transform of the signal [1].

Now, to define cycle spectrum map (CSM), a narrow-band cycle spectrum is calculated in envelope spectrum as follow,

$$y[k, t] = \{S_x[k, t]\} \quad (5)$$

Here, S_x is the spectrogram in time, and t represents the short-time spectral component of the signal (*i.e.*, envelope spectral frequency). This CSM is highly effective to detect induction motor bearing faults.

2.3 Proposed ADCNN Architecture for Fault Diagnosis

CNNs are biogenically inspired variants of the multi-layer perceptron that have proved useful in areas such as, image recognition and classification. CNN is designed to take advantage of the 2D structure of an input image. But, the original bearing fault signal is a 1D time-domain signal where it is difficult to observe any pattern of bearing defect. Therefore, this study applies an appropriate transformation technique in the above section for attaining bearing health states in 2D visualization tool. Once we obtain a 2D CSM of the raw signal using an improved cyclostationary analysis, we use an ADCNN for multi-fault classification. A typical ADCNN architecture for fault diagnosis is shown in Fig. 3, which is inspired by LetNet-5 in [11]. This proposed ADCNN architecture with adaptive learning rate is composed of 7 layers, excluding input, each of which contains trainable parameters (weights). In Fig. 3, layer CX is the convolutional layers which extract features through convolutional operations, SX is the sub-sampling layers that perform the max-pooling operations, FX is the fully connected layer which connects the network to output layer enabling to perform classification, where X is the layer index.

In this study, we use the back-propagation (BP) algorithm [14] to update all trainable parameters on all layers in the proposed ADCNN. In the proposed architecture, convolutional layers are placed subsequently with subsampling layers to build up spatial invariance progressively and to reduce computational complexity, but fully connected layers are placed in the last stage of the architecture to make consistency with the generic structure, as can be seen in Fig. 3.

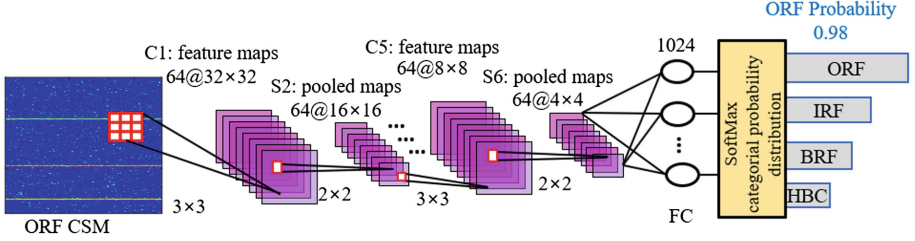


Fig. 3. The proposed ADCNN architecture used to classify different fault types. A CSM containing the bearing health state feeds the deep neural network as input.

On a convolution layer, the nearby receptive fields of the previous layer's feature maps are convolved with learnable kernels, and the result is fed to an activation function to generate the output feature map. Each output feature map can be created by combining the convolution implementations of multiple input maps. Let, l is the current convolutional layer in a network. In general, the output feature map in that layer is calculated as follows:

$$z_j^l = f \left(\sum_{i \in M_j} z_i^{l-1} * k_{ij}^l + b_j^l \right), \quad (6)$$

Where, j is the j^{th} output feature map in a convolutional layer, i is the i^{th} input feature map, M_j defines a selection of input feature maps in the layer $l - 1$, k_j represents a convolutional kernel of feature map j^{th} , and f is an activation function. b is an additive basis for each output map. With different output feature maps, the input feature maps are convolved with different kernels k .

The pooling layer aids to reduce the spatial size of the representation, to steadily reduce the number of parameters, amount of computation in the network, and also to control overfitting. The pooling layer is defined as follows:

$$z_j^l = f \left(\varphi_j^l ss(z_j^{l-1}) + b_j^l \right), \quad (7)$$

Where $ss(\cdot)$ defines a subsampling function. Typically, this function computes the max of each distinct 2-by-2 block in the input feature map, so the output feature map is two times smaller than the number of rows and columns in the input feature map. Each output map is given its trainable coefficient, (φ) and trainable bias, (b) .

The final step of the proposed ADCNN architecture is fully connected (FC) layer: the last few layers (closest to the outputs) are FC one-dimensional (1D) layers. The output of this layer is:

$$y^l = f(z^l w^l + b^l), \quad (8)$$

Where, the output activation function, $f(\cdot)$, is a logistic function, and w is trainable weights.

BP algorithm is applied to train ADCNN, so the gradients of the loss function for all trainable weights in all layers are calculated during BP operation. However, it is vital to define an appropriate objective function for BP algorithm. Thus, a squared-error loss function is used to address the objective function. Equation (9) defines the loss function as follows:

$$\text{cost}(w) = \frac{1}{2} \sum_{i=1}^m (y_i^l - t_i^l)^2, \quad (9)$$

Where, t_i^l represents the target output value of the i^{th} pattern. Suppose a point w to find the next weight point ($w + 1$) to find a minimizer, and it is started from w , and moves by $\alpha \frac{\partial}{\partial w} \text{cost}(w)$ as in Eq. (10), where α is a definite scalar step size.

$$w := w - \alpha \frac{\partial}{\partial w} \text{cost}(w) \quad (10)$$

That weight update process in Eq. (10) is called a stochastic gradient descent (SGD) algorithm [14]. It is supposed to assume that the gradient in SGD will vary as the search continues and tends toward zero as it approaches the minimizer. So, the steps size can be either small or large. The first approach (*i.e.*, a small step size) is computationally complicated to reach a minimizer; however, the second approach (*i.e.*, a large step size) could result in a more zigzag path to the minimizer. This approach is computationally inexpensive and straightforward, and also it could be trapped in a local minimizer that means there is no optimum value of step size.

To elevate the above issues in SGD, this study applies an adaptive moment estimation that combines the advantages of the adaptive gradient (AdaGrad) — working well with sparse gradients — and root-mean-square propagation (RMSProp) — working well in non-stationary settings [15]. The main idea is that it maintains exponential moving averages of the gradient and its square in each update proportional to the average gradient and its square as follows:

$$w := w - \alpha \frac{M_t}{\sqrt{R_t + \epsilon}} \quad (11)$$

$$\text{where, } M_t = \beta_1 M_{t-1} + (1 - \beta_1) \frac{\partial}{\partial w} \text{cost}(w) \quad (12)$$

$$\text{and, } R_t = \beta_1 R_{t-1} + (1 - \beta_2) \frac{\partial^2}{\partial w^2} \cos t(w) \tag{13}$$

Where, M_t is the 1st-moment bias correction, R_t is the 2nd-moment bias correction, and the decay rates are small (i.e., β_1 and β_2 are close to 1). Weight update process in Eq. (11) provides a signification result about near optimum learning rate selection.

3 Result and Discussion

To investigate the effect of CSM-based health state visualization of bearing fault and its application to fault diagnosis using ADCNN are presented in this section.

To validate the performance, this paper utilizes benchmark dataset with four fault types namely ORF, IRF, BRF, and HBC at various operating conditions (see Table 1). Figure 4 illustrates recorded example vibration signals of each bearing conditions. As this study provides an improved health state visualization using cyclic spectrum map (CSM), Fig. 5 presents the results of CSM four bearing health conditions. According to the results in Fig. 5, it is seen that proposed CSM represents a unique health condition for each fault types. An interesting point to note that outer fault defect frequency (103 Hz in this paper for 1730 rpm), inner fault defect frequency (157 Hz) and ball fault defect frequency (136 Hz), and up to 2nd harmonics each of them can be seen in Fig. 5(a), (b), and (c) respectively. Furthermore, it is evident that there is no defect frequency found for healthy bearing condition (HBC) as in Fig. 5(d).

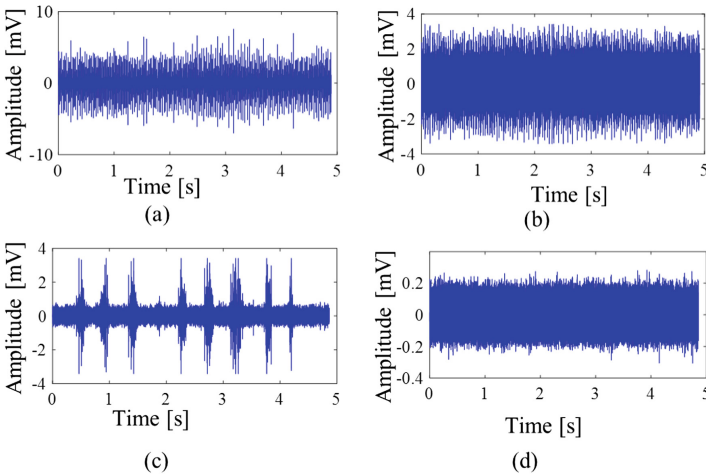


Fig. 4. Raw time-domain vibration signals of each bearing condition, (a) ORF, (b) IRF, (c) BRF, and (d) HBC bearings, respectively

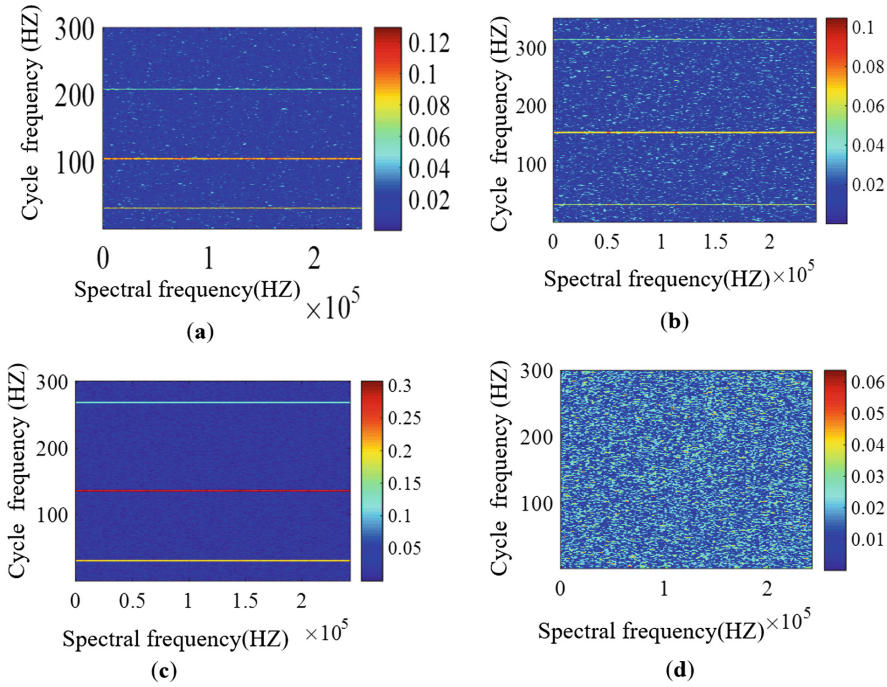


Fig. 5. Cyclic spectrum maps (CSMs) of each bearing fault condition, (a) ORF, (b) IRF, (c) BRF, and (d) HBC, in which bearing conditions are unique depending on fault type.

As CSM is highly capable of representing bearing health state, this study further utilizes CSM for fault classification. The main difference between the CNN-based method and traditional analysis-based methods is the manner for extracting features.

To validate the efficiency of the proposed method (CMS+ADCNN), we conducted experiments in comparison with two state-of-the-art fault diagnosis methods: first method [4] that extract features by calculating statistical time and frequency measurements, along with complex envelope power spectrum features. The extracted features are then evaluated and selected by a feature selection algorithm. Selected features are finally used to detect bearing defects using a k -NN classifier. Our second comparative model is deep learning-based CNN that classifies faults using the 1D raw signal [6].

In ADCNN, an appropriate learning rate is a crucial parameter in training stage optimization. It is essential to report the performance of the optimization scheme in proposed ADCNN to ensure a higher classification accuracy. Figure 6 presents the convergence rate of the proposed ADCNN and conventional deep learning with a fixed learning rate. The result is significant because the proposed scheme is converged almost to zero, which is expected to forecast a better classification accuracy.

To calculate the classification accuracy, we divide the dataset into two groups: 50 signals for training and 60 signals for testing for each faults type (see Table 1). Therefore, we have 200 samples training and 240 samples for testing for four fault

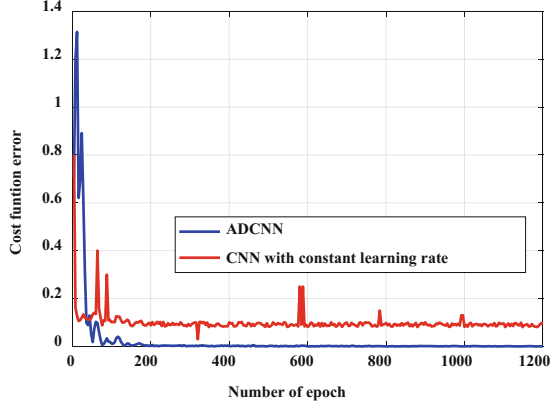


Fig. 6. Cost function convergence rate of conventional CNN and ADCNN at an initial learning rate of $1e^{-2}$.

types (e.g., ORF, IRF, BRF, and HBC). The training samples are kept lower than that of testing to ensure the reliability of the generalized performance. For a reliable comparison of classification accuracy, we use sensitivity and average classification accuracy (ACA) as in [4] as follows.

$$\text{Sensitivity} = \left(\frac{N_{\text{True_positive}}}{N_{\text{True_positive}} + N_{\text{False_negative}}} \right) \times 100(\%), \quad (14)$$

where $N_{\text{True_positive}}$ is the total number of samples in class l that are correctly classified as a class l , and $N_{\text{False_negative}}$ is the number of samples within the class l that are not recognized as a class l .

$$\text{ACA} = \left(\frac{\sum_{j=1}^L N_{\text{True_positive}}}{N_{\text{samples}}} \right) \times 100(\%), \quad (15)$$

Where L defines the number of fault classes (i.e., 4 in this paper) and N_{samples} is the total number of samples represented in a particular testing subset.

Table 2 presents the classification accuracy in terms of sensitivity and ACA. The proposed method delivers better classification performance than the methods from [4, 6]. It yielded a classification accuracy of 98%, 93%, 95% and 97% for ORF, IRF, BRF, and HBC respectively. According to result in Table 2, the proposed method outperforms two state-of-the-art algorithms, yielding an enhancement of 8.25% to 13.75% in ACA compared with the methods in [4, 6], respectively.

Also, the confusion matrices for the proposed framework and referenced methods are presented in Fig. 7. The confusion matrix is a robust technique that provides a visualization of the performance of a classifier algorithm where actual versus the predicted deviation can be observed. According to the results in Fig. 7(a), the proposed method can correctly identify all fault types with a small misclassification rate in comparison with its two counterparts: [6] in Fig. 7(b) and in [4] Fig. 7(c).

Table 2. Experimental results of identify bearing conditions

Methodology	Average sensitivity of each fault class (%)				ACA (%)
	ORF	IRF	BRF	HBC	
Proposed	98	93	95	97	95.75
[6]	83	88	87	92	87.5
[4]	87	70	83	88	82

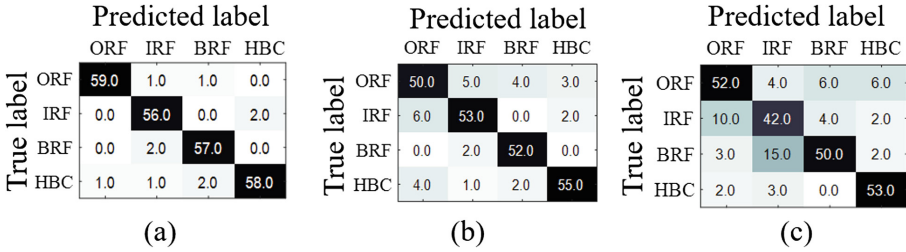


Fig. 7. The confusion matrices for showing classification results of the (a) proposed, (b) [6], and (c) [4] for each bearing condition.

4 Conclusion

This paper proposed a reliable bearing fault diagnosis scheme using a two-dimensional visualization tool for bearing health state and an adaptive deep convolutional neural network. First, we applied an improved cyclostationary analysis method for cyclic spectrum map (CSM) for representing bearing health condition. CMSs were highly effective that represent unique fault information about the bearing health state. Once we obtained the CMS, we fed it into an ADCNN. Using CSM as input in ADCNN, useful bearing fault features were then self-learned automatically and the bearing faults were diagnosed with a high precision. The presented method was validated with benchmark vibration data collected from bearing tests. The results showed that the proposed approach outperforms its referenced methods regarding classification accuracy. Thus, the combined effects of CMS-based bearing health state and adaptive learning in ADCNN contributed to achieving higher precision.

Acknowledgements. This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea (Nos. 20162220100050, 20161120100350, 20172510102130). It was also funded in part by The Leading Human Resource Training Program of Regional Neo-Industry through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2016H1D5A1910564), and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1D1A3B03931927).

References

1. Kang, M., Kim, J., Kim, J.M.: High-performance and energy-efficient fault diagnosis using effective envelope analysis and denoising on a general-purpose graphics processing unit. *IEEE Trans. Power Electron.* **30**, 2763–2776 (2015)
2. Dai, X., Gao, Z.: From model, signal to knowledge: a data-driven perspective of fault detection and diagnosis. *IEEE Trans. Industr. Inform.* **9**, 2226–2238 (2013)
3. Islam, M.M.M., Kim, J.-M.: Reliable multiple combined fault diagnosis of bearings using heterogeneous feature models and multiclass support vector machines. *Reliab. Eng. Syst. Saf.* (2018)
4. Islam, R., Khan, S.A., Kim, J.-M.: Discriminant feature distribution analysis-based hybrid feature selection for online bearing fault diagnosis in induction motors. *J. Sens.* **2016**, 16 (2016)
5. Jack, L.B., Nandi, A.K.: Fault detection using support vector machines and artificial neural networks, augmented by genetic algorithms. *Mech. Syst. Signal Process.* **16**, 373–390 (2002)
6. Janssens, O., Slavkovikj, V., Vervisch, B., Stockman, K., Loccufer, M., Verstockt, S., Van de Walle, R., Van Hoecke, S.: Convolutional neural network based fault detection for rotating machinery. *J. Sound Vib.* **377**, 331–345 (2016)
7. Islam, M.M.M., Islam, M.R., Kim, J.-M.: A hybrid feature selection scheme based on local compactness and global separability for improving roller bearing diagnostic performance. In: Wagner, M., Li, X., Hendtlass, T. (eds.) *ACALCI 2017. LNCS (LNAI)*, vol. 10142, pp. 180–192. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51691-2_16
8. Antoni, J.: Cyclostationarity by examples. *Mech. Syst. Signal Process.* **23**, 987–1036 (2009)
9. Antoni, J.: Fast computation of the kurtogram for the detection of transient faults. *Mech. Syst. Signal Process.* **21**, 108–124 (2007)
10. Wang, D., Tse, P.W., Tsui, K.L.: An enhanced kurtogram method for fault diagnosis of rolling element bearings. *Mech. Syst. Signal Process.* **35**, 176–199 (2013)
11. LeCun, Y.: LeNet-5, Convolutional neural networks (2015). <http://yann.lecun.com/exdb/lenet>
12. Case Western Reserve University. Seeded Fault Test Data. <http://csegroups.case.edu/bearingdatacenter/home>
13. Haidong, S., Hongkai, J., Xingqiu, L., Shuai peng, W.: Intelligent fault diagnosis of rolling bearing using deep wavelet auto-encoder with extreme learning machine. *Knowl. Based Syst.* **140**, 1–14 (2018)
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**, 436–444 (2015)
15. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv preprint [arXiv: 1412.6980](https://arxiv.org/abs/1412.6980) (2014)



Mobile App for Detection of Counterfeit Banknotes

Tamarafinide V. Dittimi^(✉) and Ching Y. Suen^(✉)

CENPARMI, Concordia University, Montreal, QC H3G 1M8, Canada
{t_dittim, suen}@encs.concordia.ca

Abstract. Mobile phone usage has become very common. The market continues to grow as more advanced functionalities are incorporated in mobile phones. They possess sufficient computational capability that is needed for the identification and authentication of banknotes. This paper presents a mobile application for the recognition of banknote denominations and detection of counterfeit Nigerian Naira notes using Unity 3D – which is a multiplatform mobile application development system. The system extracted the face value of the banknotes and evaluated its performance using a combination of several KNN distant measures based on a Cascaded Ensemble approach. It was tested on the Android and iOS platforms using a Samsung Galaxy S6 and an iPhone 6 respectively. The experimental results presented a 99.27% recognition rate, a 94.70% detection rate, at an average processing time of 0.02 ms.

Keywords: Counterfeit banknote · Mobile phone · Region of interest
Unity 3D

1 Introduction

Smartphone usage has become part of our daily life, and their development is still snowballing. Most phones possess advanced functionalities that go beyond making calls and text messaging. They also serve as portable computing devices used in running e-mails, agendas, and storing data [2]. Furthermore, mobile devices have at least one high-resolution camera that can be used to capture images and also hold sufficient computational capability to process the image and execute algorithms for the recognition of banknotes [19]. People identify currencies using several methods like inspecting the pictures, words, and numerals as well as distinctive signs, note consistency, and size. However, these techniques are prone to human error or omissions caused by visual impediments.

According to the World Health Organization (WHO), there were about 285 million visually impaired people worldwide, of which 39 million were visually impaired, and 246 million had low vision. Furthermore, one of the most significant problems faced is the ability to detect the value of a banknote [8], since most people differentiate banknote denominations by folding them in different distinct modes. Although this works with banknotes already in their possession, it would be difficult for them to identify notes received during daily transactions.

Presently, no country is exempted from the problem faced because of the counterfeiting of its banknotes. It is a significant issue that can cripple a country's economy; note forgery increases the circulation of paper currency leading to higher inflation and devaluation of the money, reduce people savings, loss in national economy and public confidence [1]. Furthermore, counterfeit Nigerian currencies are very much in circulation in Nigeria and although the regulatory body does not have a record of the incidences of fake banknotes that are in circulation since the deposit-taking institutions which intercept such notes do not keep the records and report same. It is still a fact that counterfeit currencies are very much in circulation in Nigeria today, and should be a matter of great worry for the government because of the grave consequences they tend to produce on the economy [4].

This research presents a cellphone-based banknote recognition application to assist people with reading disabilities, and non-native speakers to hear audio output of the banknote value, in addition to determining their authenticity. This paper is organized as follows. State of the art is discussed in Sect. 2; Sect. 3 describes the proposed banknote reader. Experimental results and evaluations are presented in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 Literature Survey

Several Mobile phone-based banknote recognition and detection systems have been designed and reported to aid visually impaired individuals. [17] Some of these methods include LookTel Money Reader and IDEAL Currency Identifier. Both of which are intended to recognize any banknote. Country-specific applications have also been developed like the Money Talker for Australian dollar recognition and Note Teller 2 and K-NFB mobile reader, both designed for US dollar recognition. Similarly, a camera phone based banknote recognition system was also proposed to identify the denomination of a U.S note [11]. It utilized a background subtraction and perspective correction algorithm and trained the currency reader using an efficient Ada-boost framework. The system works on some of the leading smartphone platforms like Symbian and Windows Mobile. However, the system is yet to be evaluated with state-of-the-art methods and the recognition time needs to be improved even though the system works in real time.

Furthermore, another research also employed a real-time portable object recognition system [14]. The system is based on the bio-inspired image analysis of visual cues to increase the autonomy of blind people. The method was built to recognize the 5€, 10€, 20€ and 50€ bills. The device could detect banknotes regardless of its orientation. However, the notes were captured using a webcam and processed on an Ultra Mobile PC using Spikenet and had yet to be adapted to run on mobile phones. Likewise, [10] developed a technique for Myanmar currency recognition, which was based on the KNN classification of extracted Gray Level Co-occurrence Matrix (GLCM). The dataset used consisted of 500 images grouped into five classes of banknotes. The limitation of the system is its poor accuracy in determining the denomination of real-world Myanmar notes.

Correspondingly, [4] described a technique for recognizing the US Dollar using an image recognition approach called Eigenfaces based on the Principal Component Analysis (PCA). The system was tested on two different Nokia Smartphones, and it had a precision rate of 99.8% and a processing speed of at least seven frames per second. The system faced issues with the image acquisition illumination and differentiating the background from the banknote. In the same view, [13] suggested an approach for the blind that was built to run on smartphones to identify notes like the Euro, Zloty and the US Dollar denominations. The method used computation based on the height and width, which is not ideal because they change due to wrinkling, wear, and tear of the banknotes.

Similarly, another work introduced a paper currency identification method based on a mobile phone as an assistive aid for the visually impaired [12]. The approach utilized the k-means clustering technique to recognize US banknotes captured by blind users. The system extracted the SIFT features from the notes, which are computationally demanding, thus increasing the recognition time. Also, a researcher [18] implemented a recognition system based on a bionic eyeglass. This system extracted the Zernike moment and employed K-means and CNN for notes identification. It was designed using a Samsung Galaxy S mobile phone, and five visually impaired subjects participated in the experiment. But, the method used a server/client architecture to transmit the image to a PC for recognition, thus making the mobile phone act just as an input/output device with no processing capability.

Additionally, [16] developed an Indian note recognition system for mobile phones. The system engaged the visual Bag of Words (BOW) recognition method and used the GrabCut algorithm to separate the foreground from the background. It was tested on 2584 images and had an accuracy rate of 96.7%. However, the system was prone to failure based on image illumination and inclusion of background in the captured banknote image. Finally, [6] designed a method for a currency recognition system. Their system extracted and compared the recognition rate of SIFT features from both grayscale and colored images. It was tested on 400 Jordanian currencies (coins and banknotes). Still, the presented approach had illumination difficulty, was dependent on the capturing distance of the images, and was relatively slow when adapted to a mobile phone.

Most of the applications employ a server-client model that uses an internet connection to communicate with a backend processor on a computer with the mobile device executing little or no processing tasks; inherently limiting the role of the phone to an input/output device. Furthermore, the proposed banknote recognition system makes use of features extracted from the binary face value of the notes. Additionally, the approach is simple and comparatively less time consuming which makes it suitable for real-time applications. Finally, the system is a dedicated banknote recognition system running on a smartphone that communicates its recognition output to visually impaired people by using pre-recorded verbal and text message.

3 Proposed Banknote Reader

Banknote recognition system was designed to recognize banknote face value captured using a smartphone camera. The system was entirely based on a mobile device and did not need any additional hardware as our software is wholly dependent on the processing capability of the smartphone. Finally, the system can be used on either side of the banknote and outputs detection results in text and audio. Figure 1 presents the flow diagram of the banknote reader. The reader acquires the banknote image, activates the banknote recognition system and then outputs the value or/and authenticity of the bill.

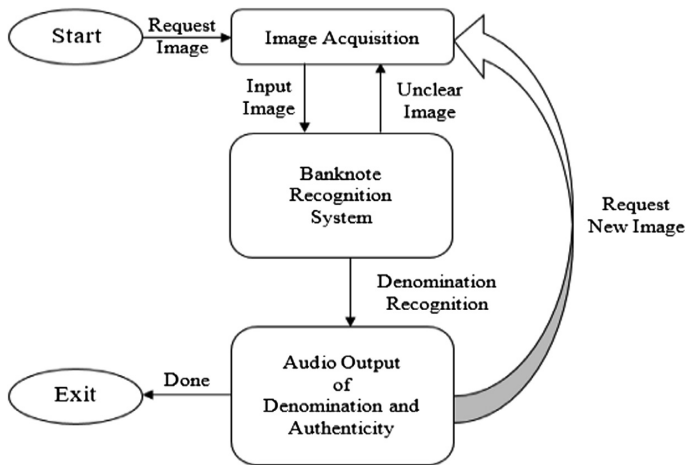


Fig. 1. Flow diagram of banknote reader

3.1 Image Acquisition

When the Mobile app is launched, the system provides two options for banknote acquisition. In the first choice, the user selects an existing Naira image on the smartphone. While with the second alternative; the user activates the cell phone camera, and takes a snapshot of the note for recognition. Also, this phase includes an object tracker module to handle banknote detection issues that occur when using this system.

3.2 Region of Interest (ROI) Extraction

The first step carried out during this phase was separating the Region of Interest (ROI) from the rest of the image. The banknote is converted into grayscale, and ROI extraction achieved by dividing the image into partitions. The partition containing the

ROI is extracted using the detected edges. Computation time is reduced by using the knowledge that the face value can only be found in two specific partitions in a banknote. We optimized the system by configuring it to check just two partitions, depending on the orientation of the Naira note. It then locates the face value by identifying all the blobs in these partitions and selects the most prominent blobs of the banknotes; crops and utilizes it as the face value [5].



Fig. 2. Extracted ROI image

Also, the extracted ROI contained noise that was reduced using a combination of gamma correction and histogram equalization to enhance the image contrast. Furthermore, the filtered ROI is then binarized, and stored in a matrix with W representing a white (background image) pixel and B for a black (banknote value) pixel [15]. Additionally, Slant correction using the image histogram is employed. The approach computes the Horizontal and vertical projection histograms, shears it by the angle that maximizes the height of peaks in the vertical projection of the figures, Zhang-Suen Thinning algorithm was used. Finally, the image was normalized to uniform the dataset. The result of the region of interest extraction is shown in Fig. 2; the figure presents a distribution of good to worn out banknotes contained in the dataset.

3.3 Feature Extraction

The feature extraction process involves removing features from the preprocessed image to examine and categorize the distinctive and peculiar characteristics of each banknote denomination to aid in the classification of our note image. In this research, 16 features were extracted and combined as feature vectors: projection, crossing, distance, direction, geometric moment, variance, skewness, kurtosis, eccentricity, orientation, Euler number, centroid, area, intersection, start and end point.

3.4 Classification

In the classification of this system, the first phase involves constructing the base classifiers using the five distance measures of the KNN method to obtain a very good generalization of the banknote predictions. Four distance measurements were utilized namely: Euclidean, Manhattan, Bray-Curtis, and Canberra distance where x and y are vectors in n -dimensional space in Eqs. 1, 2, 3, and 4 respectively [3]. During the process, the lapses of the existing KNN variants were identified. Therefore, a variation to adapt to the errors led to the development of a new Distance Measure.

The proposed distance measure method computes the absolute variance between the extracted features of the template image and the test image using Eq. 5. The differences

are multiplied together, and the various products are compared. The class with the lowest value is selected since it has the closest distance to the new image. The multiplication of the absolute variance was introduced to avoid cases where two or more classes have an equal number of images nearest to the test image. Also, the KNN phase also includes a benchmark for rejecting images that are not recognized as a Naira note.

$$\text{Distance } (x, y) = \sqrt{\sum_{i=1}^d (x_i - y_i)^2} \quad (1)$$

$$\text{Distance } (x, y) = \sum_{i=1}^d |x_i - y_i| \quad (2)$$

$$\text{Distance } (x, y) = \frac{\sum_{i=1}^d (x_i - y_i)}{\sum_{i=1}^d (x_i + y_i)} \quad (3)$$

$$\text{Distance } (x, y) = \sum_{i=1}^d \frac{|x_i - y_i|}{|x_i + y_i|} \quad (4)$$

$$\text{Distance } (x, y) = \sqrt{\prod_{i=1}^d |x_i - y_i|} \quad (5)$$

For further evaluation, the results of the best three distance measures were then combined using the Stacking and Weighted Majority Algorithm (WMA) ensemble learning methods that led to the proposing of a Cascaded Ensemble algorithm. Weighted Majority Algorithm (WMA) is a heterogeneous ensemble learning algorithm employed in constructing a compound algorithm from a pool of prediction algorithms. Furthermore, the technique is a binary decision problem and builds a compound algorithm by assigning a positive weight to each base KNN classifier and computes the weighted votes of all the base models in the pool and finally assigns the sample to the prediction with the highest majority. Lastly, the WMA technique assumes no prior knowledge of the detection rate of the KNN algorithms in the pool but instead believes that one or more of the base classifiers will perform well. While stacking involves an ensemble classifier learning to combine multiple KNN based classifiers generated by different distance measures on a single dataset, the approach produces a set of base-level classifiers and train a linear regression classifier to combine the outputs of the base KNN models.

Furthermore, the individual classifiers were trained using the whole training dataset; before linear regression is applied to fit using the outputs of each distance measure in the ensemble [7]. The Cascaded Ensemble approach employs a two-level classification; the first phase is based on the distance measure of the KNN classifier with the highest accuracy (true positive) and recall (false negative). Which is used in the classification/detection of the banknote by relating the unknown data to the known data based on a calculated distance or similarity measure. While the second stage employs WMA and Stacking algorithm as meta-classifiers using the remaining KNN methods as the base classifier. The flowchart of the approach is presented in Fig. 3.

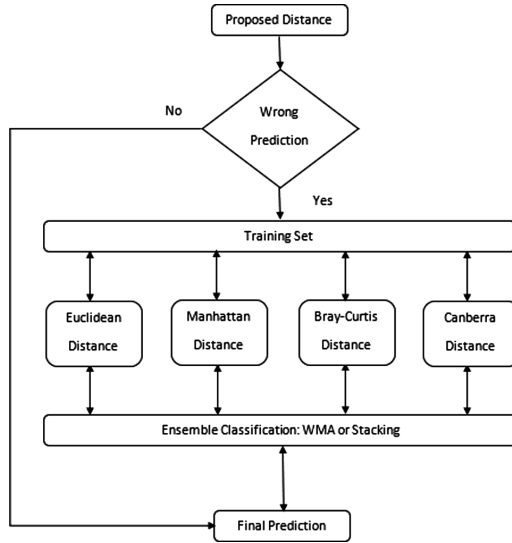


Fig. 3. Proposed ensemble KNN algorithm

4 Result and Evaluation

The proposed approach evaluates the viability of natively designing a mobile-based banknote recognition and authentication technique rather than scaling down existing systems to be compatible with the processing capability of mobile devices. The KNN and Ensemble classifiers were constructed on the Unity platform from scratch rather than scaled down from other existing software. During the process, the lapses of the current KNN and Ensemble variants were identified. Therefore, a variation to adapt to the errors led to the development of a new Distance measure and a Cascaded Ensemble based approach. The Unity system is multiplatform, so we tested the system on the Android and iOS platforms using a Samsung Galaxy S6 and an iPhone 6 respectively and recorded the average recognition time. The application was designed as a dedicated banknote recognition system running on a smartphone that communicates the recognition output to the visually impaired by using a pre-recorded text and verbal message.

4.1 Hardware and Software

This research utilized Unity 3D as its development platform. It is an application designed by the Danish Unity Technologies Company in 2005. It consists of a fully developed multimedia system with built-in features like rendering, illumination, behavior, audio, video and sound provisions. Unity 3D also has a powerful user interface development facility. It is supported by multiple platforms (Android, iOS, and Windows) and deployed on mobile phones, computers, virtual reality systems, gaming consoles, web applications and smart TVs. Additionally, it aids in bypassing the use of conventional programming languages like C++ and Java, in addition to working around

platform dependent applications like Android or iOS and their related toolboxes. Furthermore, two platforms were chosen for this work: Android using Samsung S6, and iOS via iPhone 6S. Lastly, developing a solution in Unity 3D is quick, since the Unity Editor has many powerful convenience tools and has an integrated Asset Store with many free or easy-to-purchase plugins for any system [9].

4.2 Data Collection

The dataset images were captured using two acquisition methods. One using a Canon T2i digital camera and the second using an iPhone 6. Pictures were taken from different angles. The digital camera images were reused physical banknotes taken with the mobile phone. The iPhone 6 captured real-time photos used for the training and testing processes. The Naira notes were gathered from existing Nigerian banknotes from the north, south and eastern parts of the country. They were collected and labeled by the CENPARMI research group and contained different distortions like stains, mutilations, fading, and tears representing the reality of circulated paper money in Nigeria. Ten-fold cross-validation was employed for analyzing the dataset. For this research, 2310 Genuine and 2048 fake Nigerian banknotes were collected ranging from Five Naira to One Thousand Naira (₦5, ₦10, ₦20, ₦50, ₦100, ₦200, ₦500 and ₦1000) and the system had 16 classes: 8 for genuine notes and 8 for fake bills. Figure 4 displays the Naira note dataset employed in this work.

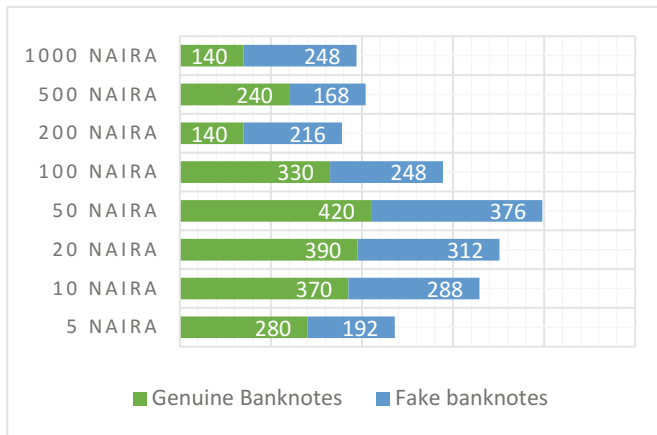


Fig. 4. Naira note dataset

4.3 Results

Figure 5 presents the recognition rates for Canberra, Manhattan, Bray-Curtis, Euclidean, Proposed Distance Measure, Stacking, WMA, Cascaded Stacking, and Cascaded WMA. In general, all distance measure achieved relatively high recognition rate. But, Bray-Curtis had the lowest recognition rate of 95.86%, and our proposed Cascaded WMA Ensemble Classifier presented the highest recognition rate of 99.27%.

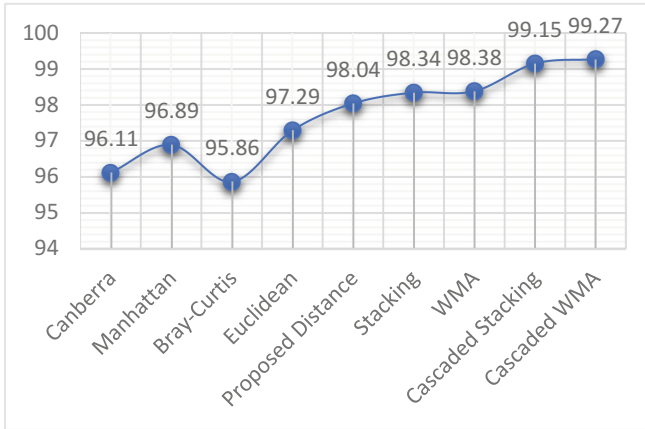


Fig. 5. Recognition rate (%) of classifiers

Table 1. Comparison of proposed method with state of the art

Distance measure	Technique	Recognition rate (%)	Recognition time
Liu 2008	AdaBoost Framework	97.30	0.11 s
Singh <i>et al.</i> 2014	Bag of Words Sift Descriptor K-mean	95.95	1.08 s
Doush <i>et al.</i> 2016	SIFT KNN	72.4	72.6 s
Proposed method	Face Value (OCR) Cascaded Ensemble Classifier	99.27	0.02 ms

Furthermore, Table 1 displays existing state of the art mobile-based systems trained using our dataset. Although, some existing research also achieved comparative results without outperforming the proposed method. They had considerably longer processing time as they either used a client-server architecture or were tailored down to fit the computational capability of mobile devices. Finally, these previous works are yet to extend their approach to incorporate authentication of genuine and fake notes.

In the same view, Fig. 6 shows the detection rate of the eight Nigerian banknote denominations using Canberra, Manhattan, Bray-Curtis, Euclidean, Proposed Distance Measure, Stacking, WMA, Cascaded Stacking, and Cascaded WMA. The lowest denomination detection was recorded in the 1000 Naira notes using Bray-Curtis distance while the highest was also achieved on 1000 Naira notes by our Cascaded Ensemble Classifier. Furthermore, all distance measure performed better with notes 100 Naira and above; as they contained three individual characters while 50 Naira and below which included one or two distinct characters had a lower recognition rate. Additionally, the proposed distance measure outperformed all other existing techniques employed in this research.

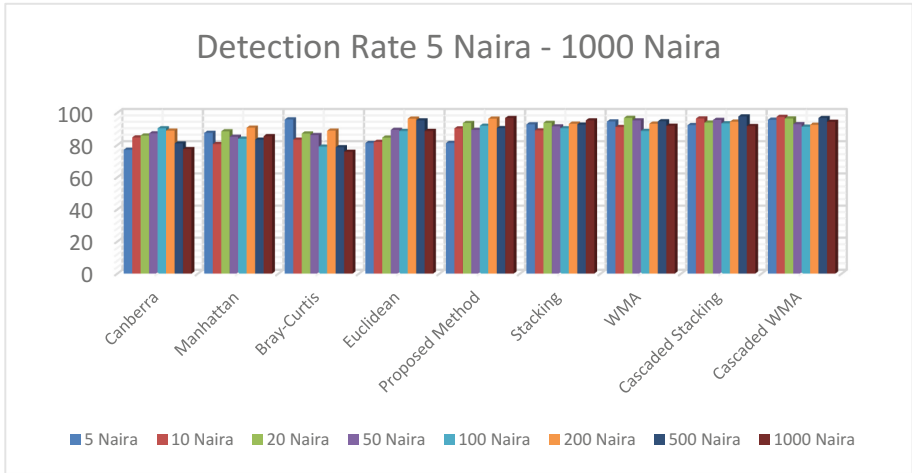


Fig. 6. Detection rate of banknotes on class/distance measures

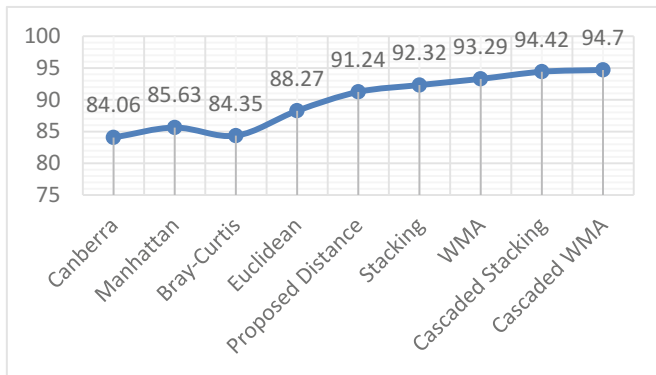


Fig. 7. Detection rate (%) of classifiers

Figure 7 presents the detection rate for Canberra, Manhattan, Bray-Curtis, Euclidean, Proposed Distance Measure, Stacking, WMA, Cascaded Stacking, and Cascaded WMA (Classifier) utilized in this research; Canberra had the lowest detection rate of 84.06% while our proposed distance measure offered the highest detection rate of 91.24%. Also in Fig. 8, images of genuine and fake banknotes that was rejected by the system are shown. Thus, this study presents a mobile application for paper currency identification and authentication using OCR feature in conjuncture with an adaptation of the Nearest Neighbor and Ensemble Classifier. The proposed system had a 99.27% recognition rate, a detection rate of 94.70% and processing time of 0.02 ms.

During the classification phase, the mobile app first checks to see if the face value of the test image meets the set threshold value, which was based on the average face value feature set. If the feature set used doesn't fit the image threshold, it gives an audio



Fig. 8. Rejected (a) Genuine and (b) fake banknotes

and text output requesting the user to recapture or select the image again, in addition to providing the necessary steps needed to ensure the system works correctly. Furthermore, the system also has a help menu that gives a detailed explanation and guidance in audio and text informing users of the steps to go through for the recognizing and authenticating of banknotes.

Lastly, to utilize the system for another currency, the banknote image can be uploaded into the system via existing pictures or by activating the mobile phone camera on the application. If bulk training/testing is to be done, then the existing image option is utilized. However, if a single image training or testing approach is required, the mobile phone camera is activated to capture the note in real time. Furthermore, the system must be configured to select the partitions that include the face value of the currency of choice. The system can also be configured to use all or some of the extracted features. The text input can also be updated for a new currency of choice, and since the system was built to employ a text to audio converter upgrading the recognition output text to the original banknote of choice automatically updates its audio output to reflect the new currency being tested by the mobile application.

5 Conclusion

This research describes the development, implementation, and appraisal of a mobile phone banknote recognition system based on Unity 3D. The designed system is very general and with little modifications, it would work on any currency worldwide. The system uses several KNN classifier as base learners to sort data on a calculated distance measure and employs Ensemble classifiers to combine the base KNN approach to create a strong classifier. It uses a two-tier prediction approach; the first level identifies the denomination of the banknote and the second level determines the authenticity of the currency. The system could recognize note values, detect fake notes and reject banknotes that are not Naira notes. The proposed Cascaded approach presented significantly higher recognition and detection accuracy than the base KNN classifiers with statistically significant improvement and thus can be concluded that the proposed Cascaded system is superior to individual KNN, stacking and WMA method for banknote recognition and detection. Furthermore, it is essential to note that our proposed banknote recognition system only made use of features extracted from the Face value. Future research would implement a feature fusion system that combines other security features such as security thread, serial number, micro-letters, signature, and

face on the banknote to improve the detection rate of the system. Also, usability testing would be conducted with the participation of visually impaired individuals.

Acknowledgment. The Center for Pattern Recognition and Machine Intelligence (CENPARMI) Research Group of Concordia University and Natural Sciences and Engineering Research Council of Canada (NSERC) has partially sponsored this research, and the authors sincerely recognize the assistance and cooperation of the CENPARMI members especially Nicola Nobile at Concordia University.

References

1. Ahmed, Z., Yasmin, S., Islam, M.N., Ahmed, R.U.: Image processing based feature extraction of Bangladeshi banknotes. In: 2014 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), pp. 1–8. IEEE (2014)
2. Angsupanich, S., Matayong, S.: Applying the mental model for real-time recognition of Thai banknotes: the blinds' mobile application. In: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), pp. 86–90. IEEE (2016)
3. Chomboon, K., Chujai, P., Teerarassammee, P., Kerdprasop, K., Kerdprasop, N.: An empirical study of distance metrics for k-nearest neighbor algorithm. In: The 3rd International Conference on Industrial Application Engineering 2015 (ICIAE 2015) (2015)
4. Daniel, B.P., Idoko, C.U.: The macro-economic consequences and regulatory challenges of currency counterfeiting in Nigeria (2014)
5. Dittimi, T.V., Hmood, A.K., Suen, C.Y.: Multi-class SVM based gradient feature for banknote recognition. In: 2017 IEEE International Conference on Industrial Technology (ICIT), pp. 1030–1035, March 2017
6. Doush, I.A., Sahar, A.B.: Currency recognition using a smartphone: comparison between color SIFT and gray scale SIFT algorithms. *J. King Saud Univ. Comput. Inf. Sci.* **29**, 484–492 (2016)
7. Govindarajan, M.: Evaluation of ensemble classifiers for intrusion detection. *World Acad. Sci. Eng. Technol. Int. J. Comput. Electr. Autom. Control Inf. Eng.* **10**(6), 876–884 (2016)
8. Hasanuzzaman, F.M., Yang, X., Tian, Y.: Robust and effective component-based banknote recognition for the blind. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **42**(6), 1021–1030 (2012)
9. Harshfield, N., Chang, D.J.: A Unity 3D framework for algorithm animation. In: *Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*, pp. 50–56. IEEE (2015)
10. Hlaing, K.N.N., Gopalakrishnan, A.K.: Myanmar paper currency recognition using GLCM and k-NN. In: 2016 Second Asian Conference on Defence Technology (ACDT), pp. 67–72. IEEE (2016)
11. Liu, X.: A camera phone based currency reader for the visually impaired. In: *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 305–306. ACM (2008)
12. Paisios, N., Rubinsteyn, A., Vyas, V., Subramanian, L.: Recognizing currency bills using a mobile phone: an assistive aid for the visually impaired. In: *Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology*, pp. 19–20. ACM (2011)
13. Papastavrou, S., Hadjiachilleos, D., Stylianou, G.: Blind-folded recognition of bank notes on the mobile phone. In: *ACM SIGGRAPH 2010 Posters*, p. 68. ACM (2010)

14. Parlouar, C.R., Dramas, F., Macé, M.M., Jouffrais, C.: Assistive device for the blind based on object recognition: an application to identify currency bills. In: Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 227–228. ACM (2009)
15. Sathisha, K.: Bank automation system for Indian currency-a novel approach. In: 2011 IEEE Recent Advances in Intelligent Computational Systems (2011)
16. Singh, P., Budhiraja, S.: Feature extraction and classification techniques in OCR systems for handwritten Gurmukhi Script—a survey. *Int. J. Eng. Res. Appl. (IJERA)* (2011). ISSN 2248-9622
17. Singh, S., Choudhury, S., Vishal, K., Jawahar, C.V.: Currency recognition on mobile phones. In: 22nd International Conference on Pattern Recognition (ICPR), pp. 2661–2666. IEEE (2014)
18. Solymár, Z., Stubendek, A., Radványi, M., Karacs, K.: Banknote recognition for visually impaired. In: 20th European Conference on Circuit Theory and Design (ECCTD), pp. 841–844. IEEE (2011)
19. Stein, C., Nickel, C., Busch, C.: Finger-photo recognition with smartphone cameras. In: 2012 BIOSIG-Proceedings of the International Conference of the Biometrics Special Interest Group (BIOSIG), pp. 1–12. IEEE (2012)



A Multiagent Framework for Understanding Addiction

Wasif Khan^(✉) and Robin Cohen^(✉)

University of Waterloo, Waterloo, ON, Canada
{w23khan, rcohen}@uwaterloo.ca

Abstract. In this paper, we provide a framework for examining the problem of addiction that considers both internal factors like self-control as well as external factors denoted as the environment. We do this by considering the Prisoner’s Dilemma, a game-theoretic concept examined by competitive multiagent systems researchers. In particular, we devise an iterated Prisoner’s Dilemma involving you and future you. We devote considerable effort in defining the notion of selfhood from previous literature in economics, as this is critical in examining addiction. The main contribution is a framework that enables calibration of alternate scenarios of behaviour for addicted individuals: in essence an application of artificial intelligence for the important social problem of modeling addiction, yielding some intuitive and explanatory results. We briefly comment on the main underlying assumptions and biases as well as mention future work that could be derived from this research, including commentary on how reasoning about both current and future reflections of self may be useful in general for multiagent decision making.

Keywords: AI applications · Multiagent systems · Addiction
Prisoner’s Dilemma · Game theory · Identity · Economics

1 Introduction

Addiction is a complex issue that has received heightened attention over the past decade causing major changes in how we think about the matter. Historically, the explanation for the cause of addiction was centered around the paradigm of an individual’s inability to express aspects of human nature like self-control. Over the past decade there has been a drastic shift that has looked at addiction through the lens of external factors. Researchers have shown with some success that the primary cause for addiction is heavily related to the environment and conditions imposed on the individual rather than anything intrinsic. In this paper, we provide a framework for examining addiction that aligns with the artificial intelligence paradigm of competitive multiagent systems. We build on existing research looking into addiction from an abstract mathematical perspective by formalizing some previously defined notions, providing a comprehensive framework that offers intuitive and explanatory insights into addictive behaviour.

We define addiction as “compulsive engagement in rewarding stimuli despite adverse consequences” [1]. To discuss addiction on a general level, we build on the

idea that addiction can be modeled using the Prisoner's Dilemma (PD) as an intertemporal conflict between your current self and your future self. These two identities of self can be viewed as competing intelligent agents. We then explain how basic game-theoretic principles can be leveraged in order to provide a formal understanding of addictive states. As our PD is a conflict between your current self and future self, we will ground our utility estimates in previously defined notions of "the self". Behavioral economics will provide us with critical insights into this which we will use as a foundation for our utility calculations and the results that follow. There is growing evidence to show that addiction is largely influenced by your environment and social conditions [2]. After constructing our basic model, we will formalize the notion of the environment and show how it can be applied to our model and yield results that seem intuitive and explanatory. The main contribution of this paper is to formalize previously defined notions related to addiction into a preliminary yet comprehensive framework that yields some promising results.

2 Related Work

There has been substantial evidence to show humans do not behave rationally. Specifically, the branch of economics known as behavioral economics shows humans do not evaluate alternatives consistently over time. This sub-field studies the effects of psychological and emotional factors involved in a rational agents decision making process [6] and this area is where most literature discussing the decision making of addiction is found. There are two prominent theories that we will consider.

Theory of Multiple Selves. Thaler et al. provide us with an elaborate model for considering self-control [5]. They claim that the idea of self-control and refraining from addictive behavior seem paradoxical unless we consider the individual as an organization or committee of agents attempting to work together. This committee consists of a farsighted *planner* and several myopic *doers*. The self can then be thought of as several relatively independent self-interested entities (doers) each with potentially competing utility functions and a planner that discounts each doers utility depending on farsighted effects. After discounting the doers utilities, the planner picks actions for the doer with the maximal utility.

A simplified example is as follows. An average university student could consist of a doer whose utility is a function of doing well in school. There could also be doers consisting of utilities as a function of maintaining an active social life, a physically healthy lifestyle and managing finances effectively. If allowed complete control, these doers would seek purely self-interested goals, and oftentimes these would conflict. The doer for managing money can often have conflicting utility functions with the doer that wants to maintain a healthy social life (assuming most social activities require some financial investment). Similarly, the doer for doing well in school could have conflicting utility with the doer for maintaining a healthy social life when faced with the situation of studying for an exam or going out with friends. The job of the planner is to consider the utility functions of all the doers and evaluate how valuable the exam is relative to the night out with friends and discount the utilities accordingly followed by

picking actions from the doer with the higher utility. In the case of addiction, there is a doer interested in the addictive activity, and given the adverse consequences for partaking in the activity - there are several other doers within the committee of the self that have conflicting utility functions as compared to the doer interested in the addictive activity. We will use this notion of the self in defining utility functions for the PD presented in the next section.

Hyperbolic Discounting of Delayed Rewards. The theory of multiple selves allows us to consider addiction as an independent agent (doer) within the committee that is the individual. It tells us that different agents may have different and sometimes competing utility functions but doesn't elaborate on the shape or properties of these functions. Hyperbolic discounting allows us to give a specific shape to the utility function for doers. There is substantial evidence to show that humans discount future rewards subject to a discount factor that is hyperbolic in the time of the reward [4]. A visual representation of hyperbolic discounting of two rewards A and B is shown in Fig. 1. In short, when rewards are imminent, there is a spike in utility which can potentially lead to a momentary preference reversal. Think of waiting an additional day for a reward that is a year off as compared to waiting a day for a reward that is expected immediately. It should be evident that the additional day for the immediate reward would be harder to withstand than the additional day for the reward a year away. This is due to the spike in value as one gets temporally closer to a reward. There is a moment where an action whom utility is regularly lower than other actions has a reward spike and surpass a further but greater utility creating a reversal of preferences. This is evidence of humans not being rational agents as we often times don't make decisions consistent through time.

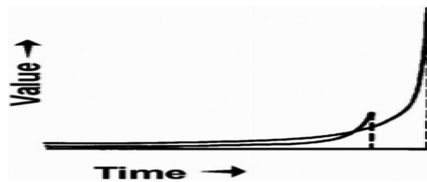


Fig. 1. Hyperbolic discounting of two rewards

Consider the previous example of an average university student. He could have two doers - one concerned with a successful university career and another, lower utility reward function concerned with a healthy social life; this can be modeled by Fig. 1. Upon waking up on a Friday morning, a person could tell themselves they will study the weekend for the test on Monday. The utility of studying and the successful university career would be marginally greater as both rewards are relatively far away hence the person would go about their day in a manner conducive of studying for the weekend. However - come Friday night when the opportunity to socialize is imminent and the reward for studying relatively far away, one can have a preference reversal in

utility and opt to socialize even when the reward for studying would be greater. Now it is the planner's job to influence the utility of the doers in such a way that spikes in sooner, imminent rewards doesn't surpass the utility of further, greater rewards. However in the case of addiction where there is some sort of inherent reward, the utility can sometimes be heightened for the doer seeking the addictive activity. It is not difficult to consider the previous example with the socialization doer replaced with one solely concerned with drinking.

Given these two theories, addiction can be considered as a disconnect between the doers and the planner. More specifically - as the planners inability to influence the utility of the doer interested in the addictive activity, just as the hyperbolic function spikes for that doer. Therefore the addict partakes in the activity as the utility for partaking is higher than refraining at the immediate moment when one considers whether to partake - due to the planners inability to discount the utility for the addictive activity. Behavioral economics also shows us humans don't behave rationally as we do hyperbolic discounting of future rewards which leads to inconsistent actions over time. Although not directly related - there appears to be a common theme of human decision making not being rational. We will use this idea when constructing utility estimates and argue that the primary factor in determining cooperation is not evaluating between different utilities, but rather determining how connected you feel to your future self.

3 The Prisoner's Dilemma

The Prisoner's Dilemma is an abstract representation of a situation where two rational agents may not cooperate even if it is in their best interest to do so [5]. The standard example is as follows. You and a fellow gang member are caught at a crime scene and the police take you to separate rooms not allowing for any communication. They have evidence to convict each of you for a minor crime that serves a one year prison sentence if you stay silent. The principal crime has a three year prison sentence but the police have insufficient evidence to convict either of you for that - unless there is a verbal testimony against the other person. The police offer each of you the option to testify against the fellow gang member and in return, they will let you off the one year sentence and go free. The catch is - if both of you testify against each other, then both are found guilty and will serve a reduced two year prison sentence for the principal crime. This is represented mathematically with $T = \text{Temptation} = 0$, $C = \text{Cooperation} = -1$, $D = \text{Defection} = -2$ and $S = \text{Sucker} = -3$. This can be shown through the payoff matrix presented in Fig. 2. An interesting result that arises is the fact that two rational self-interested agents will opt to defect even when it is clear that mutual cooperation would yield a greater individual and global outcome.

Although this theory was originally developed with reference to two individuals, Ainslie et al. mention how addiction can be viewed as an intertemporal Prisoner's Dilemma (PD) with your current self and future self [6]. In this context - cooperation would refer to not partaking in the addictive activity while defecting would be to relapse. It is easy to see that not defecting today and in the future would be the best outcome as one would no long be addicted. The case where one abstains today but relapses in the future is the worst outcome as the individual spent substantial effort

		Prisoner A	
		Silent	Betray
Prisoner B	Silent	-1 -1	0 -3
	Betray	-3 0	-2 -2

Fig. 2. The standard Prisoner’s Dilemma

making an attempt to quit, but this effort was wasted as the individual is still addicted. Relapsing today and in the future is slightly better than the previous outcome. The individual is still addicted but at least they didn’t spend effort in making an attempt to quit. The final case where one relapses today but abstains in the future is familiar to anyone who has struggled with addiction.

There are obvious benefits in defecting today as one gets to partake in the addictive activity while also not being addicted in the future - without having to spend any effort right now trying to quit. However - when the individual encounters this situation in the future, they can reason in a similar manner leading to an endless string of defection. The challenge is that starting to cooperate takes substantial effort on the part of current self while the rewards of not being addicted are reaped only by the future self. Therefore in any individual situation - if one does not feel strongly connected to their future self, the utility for partaking in the addictive activity may be higher than the utility for sobriety for future self due to hyperbolic discounting.

In the next section we will formalize the notion of addiction as an intertemporal PD with your current self and future self. This aligns with work that shows that the identity of the interactant is a large predictor of cooperation within the PD [3].

4 Framework Fundamentals

We will now formalize the above description by introducing a model that attempts to capture the different relevant factors pertaining to addiction. To provide a formal framework, we start by defining three variables that aim to formalize the notions mentioned in the previous Section.

1. *Effortless?*: Note that the explanation above often references the notion of self-control or will-power. Whether or not the individual has to spend effort in quitting is essential for our discussion so we will introduce this notion within the binary variable which is 1 if no effort was spent trying to quit and 0 otherwise.
2. *Satisfaction?*: Whether or not we are partaking in the addictive activity is relevant to the discussion so we define a second variable which is 1 if we are partaking in the activity and 0 otherwise.

3. *Sobriety?*: We also note that the notion of whether or not we are addicted is of prime importance. Therefore we define a third variable which is 1 if we are sober and 0 otherwise.

We will use these variables as the primary parameters in constructing utility estimates within our model. Note that they have been derived from high level discussion of addiction within the PD. Before proceeding any further, we want to highlight the fact that these variables are defined as basically as possible as our goal is to simply introduce the framework. There should be extensive work that considers how these variables should be formulated. We also only list three variables however we believe there are several more that need to be considered. Lastly, different doers will have different factors that affect their utility function. A doer concerned with academic success will be more sensitive to the experience of receiving a good grade than a doer concerned with physical fitness.

Our notion of the self will be motivated by ideas from the philosophers Locke and Hume which claim that the self is merely a bundle of the experiences so far as we are aware of the experience [7]. We also have the planner-doer model from behavioral economics that suggest an individual is an organization consisting of a farsighted planner and several myopic doers. Using these theories, we can formally define the self. We begin by defining an experience E as a set of inputs received from the five senses - sight S_i , hearing H , taste T_a , smell S_m and touch T_o . We then define a doer D with some utility U as a set of N experiences.

$$E = \{S_i, H, T_a, S_m, T_o\} \quad (1)$$

$$D = \{E_1, \dots, E_N\} \quad (2)$$

Consider an individual that sits down to watch a movie on the meat industry or global warming. The experience(s) of watching a movie can create a doer that is concerned about being vegetarian or the environment respectively. Similarly, the experience of a new sensation from an addictive activity can create a doer that seeks that addictive activity. Experiences from addictive activities are especially prone to create and reinforce doers due to the inherent rewards in addictive activities. We define the planner P as a function that takes all M doers created from all the experiences the individual has experienced until the present moment as input, and outputs a weight associated with each doer.

$$P : \{D_1, \dots, D_M\} \rightarrow \{w_1, \dots, w_M\} \text{ with } w_1 + \dots + w_M = 1 \quad (3)$$

The sum of the weights being 1 allows the planner to influence all doers within the self while taking into consideration all other doers. From this, we can define the self S as a set of all the doers created from all the experiences the individual has been exposed to each discounted by the planner.

$$S = \{w_1D_1, \dots, w_M D_M\} \tag{4}$$

At any given moment - an individual will simply pick actions from the doer with the highest utility after being discounted by the planner. After an action is picked - the individual receives a new experience and the process repeats. As time progresses, doers which had rewards that were far off become imminent and preference reversal may be created due to hyperbolic discounting of future rewards if the planner cannot adequately discount the imminent reward. This can be highlighted in our previous example with a student waking up in the morning with intentions of studying. However, as the evening approaches and the rewards for socializing become imminent, a preference reversal may be created if the planner cannot adequately discount the imminent reward and stick to the original plan of studying.

We now have all the tools to set up our model. To formally consider addiction as a Prisoner’s Dilemma between your current self and future self - we consider the current self as the doer D_1 concerned with partaking in the addictive activity while the future self can be thought of as the aggregate of all other doers D_2 concerned with rewards further in time. We make the critical assumption that $w_1 + w_2 = 1$ as we isolate the addictive activity in D_1 and group all other doers in D_2 . For simplicity, we will denote this variable as α and define it as how connected you feel to future self and how invested you are in focusing on the future.

$$\alpha = 1 - w_1 \tag{5}$$

This equality also makes the added assumption that the planner believes the addictive activity has conflicting utility with all other doers within the individual and we feel this is an accurate assumption. Given that the utility for D_1 explicitly comes from the inherent reward of the addictive activity - this generally doesn’t complement utilities for any other doers. This excludes situations where the utilities may be complementary as would be the case for a skilled gambler making an income from poker or an athlete taking performance enhancing drugs prior to a competition for example. We do not consider this addictive behavior as per our original definition. For an activity to be considered addictive, the utility must only be derived from the inherent reward partaking in the activity provides - and in this case, it is generally conflicting with all other utilities.

Now let’s consider each of the four quadrants within the PD and see what the utility for cooperation and defection may look like. We will use the variables *Effortless?* E , *Satisfaction?* Sa , and *Sobriety?* So to calculate utility. Note that the standard PD has two utilities per cell - one for each person. However, as this is an intertemporal PD - we aggregate the utilities as the individual is concerned with both utilities for current self D_1 and future self D_2 . Therefore the utility U for any cell

$$\begin{aligned} U &= w_1U_1(D_1) + w_2U_2(D_2) \\ U &= (1 - \alpha)U_1(D_1) + \alpha U_2(D_2) \end{aligned} \tag{6}$$

D₁ = Cooperate & D₂ = Cooperate
 D₁ spent effort trying to quit (E = 0)
 D₁ did not get satisfaction from the activity (Sa = 0)
 D₁ is not addicted as D₂ cooperated (So = 1)
 Therefore U₁(D₁) = (0 + 0 + 1) = 1
 D₂ did not spend effort trying to quit (E = 1)
 D₂ isn't concerned with satisfaction (Sa = 0)
 D₂ is not addicted (So = 1)
 Therefore U₂(D₂) = (1 + 0 + 1) = 2
U(D₁=D₂=Cooperate) = (1-α)(1) + α(2)
= α + 1

D₁ = Defect & D₂ = Cooperate
 D₁ did not spend effort trying to quit (E = 1)
 D₁ did get satisfaction from the activity (Sa = 1)
 D₁ is still not addicted as D₂ cooperated (So = 1)
 Therefore U₁(D₁) = (1 + 1 + 1) = 3
 D₂ did spend effort trying to quit (E = 0)
 D₂ isn't concerned with satisfaction (Sa = 0)
 D₂ is not addicted (So = 1)
 Therefore U₂(D₂) = (1 + 0 + 0) = 1
U(D₁=Defect & D₂=Corporate) = (1-α)(3) + α(1)
= -2α + 3

D₁ = Cooperate & D₂ = Defect
 D₁ spent effort trying to quit (E = 0)
 D₁ did not get satisfaction from the activity (Sa = 0)
 D₁ is still addicted as D₂ defected (So = 0)
 Therefore U₁(D₁) = (0 + 0 + 0) = 0
 D₂ did not spend effort trying to quit (E = 1)
 D₂ isn't concerned with satisfaction (Sa = 0)
 D₂ is still addicted (So = 0)
 Therefore U₂(D₂) = (1 + 0 + 0) = 1
U(D₁=Cooperate & D₂=Defect) = (1-α)(0) + α((1)
= α

D₁ = Defect & D₂ = Defect
 D₁ did not spend effort trying to quit (E = 1)
 D₁ did get satisfaction from the activity (Sa = 1)
 D₁ is still addicted as D₂ defected (So = 0)
 Therefore U₁(D₁) = (1 + 1 + 0) = 2
 D₂ did not spend effort trying to quit (E = 1)
 D₂ isn't concerned with satisfaction (Sa = 0)
 D₂ is still addicted (So = 0)
 Therefore U₂(D₂) = (1 + 0 + 0) = 1
U(D₁=D₂=Defect) = (1-α)(2) + α(1)
= -α + 2

We considered different values for α and observed when it was rational to cooperate and when to defect. A correlation between α and the decision to cooperate arose that seemed intuitive when considering the phenomena of addiction (Fig. 3).

In the general form, we see that cooperation for D₁ is positively correlated to α while defection for D₁ is negatively correlated to α which is what we expect. Notice for some values of α - it is rational for a person to defect. Most importantly, notice when $\alpha \leq 0.5$ - the cell with the highest utility is that where current self defects and future self cooperates. Also notice that only when α is substantially high is it rational to cooperate with future self. This means that we are only inclined to not partake in addictive behavior so far as we feel connected to the rewards of our future selves.

With these results, we have shown addiction is a failure to adequately discount immediate rewards. If the planner can discount the immediate reward enough, then it is rational to cooperate and an individual won't be stuck in an endless string of defection. This should make intuitive sense; the more connected we feel to future rewards - the less inclined we are to partake in an addictive activity. The next section examines the dynamics of the planner function and how this can be leveraged to get an individual in and out of addiction.

5 The Environment Factor

In the previous section we showed that the higher the value for α , the more utility an individual received for cooperating with their future self and vice versa. To consider how α fluctuates, we need to consider the environment as an important factor in this framework.

As α is loosely defined as how connected your current self feels to your future self, a natural question that arises is what characteristics determine how connected one feels

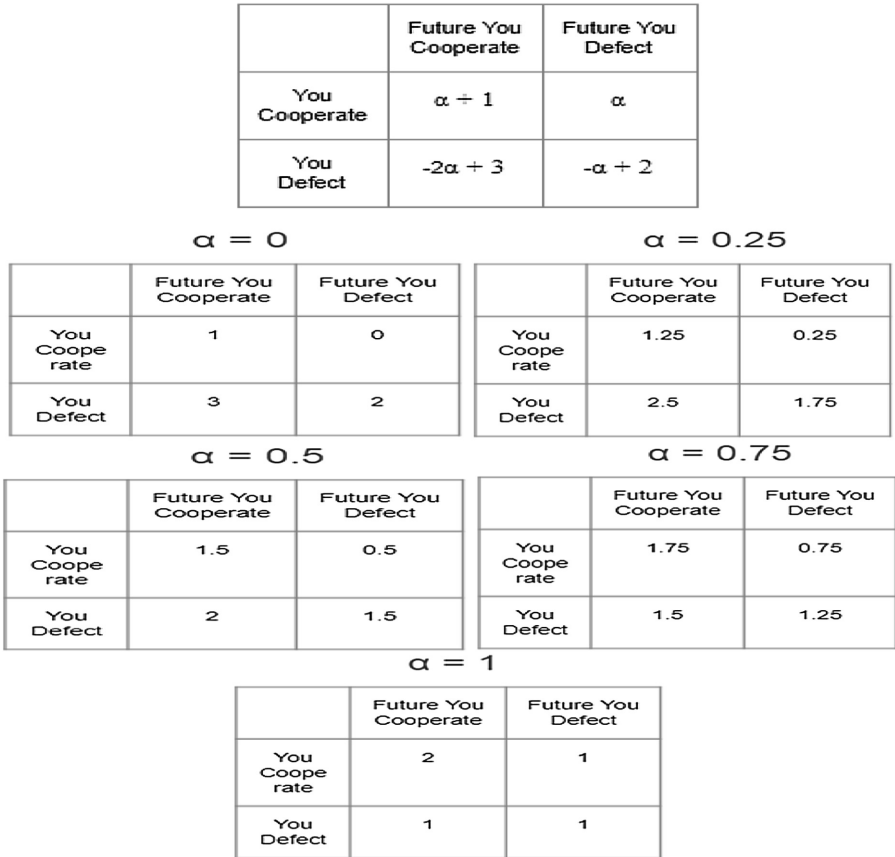


Fig. 3. Summary of addiction Prisoner’s Dilemma

to themselves through time. Recall that α is simply a weight assignment of the planner over doers D_1 whose utility is derived from partaking in the addictive activity and D_2 whose utility is derived from future rewards. We claim that the value of α is determined by how accurately the planner predicted future events in the past. If in the past - the planner assigned weights in a manner that executed actions that resulted in future experiences better than the planner expected, then α is increased. If the future yields experiences worse than the planner planned, α is decreased. In essence, if the planner is ineffective at predicting future experiences - then there is not much benefit in discounting immediate rewards for future ones as the individual cannot trust these future rewards will come to fruition given that he had past experiences where expected future rewards didn’t come to fruition.

Consider our previous example of a university student trying to manage his interests in academics and a healthy social life. If in the past, the student has experienced situations where they had an upcoming exam and hence the planner influenced the utilities of the doers leading to the individual studying. If in the future, the student

got a mark back that was better than they expected, they will be more inclined to study in the future as they have experienced the fruition of future rewards that they planned for. However if the student studied and then received a mark far below what they expected - this makes the individual less inclined to study in the future as they have spent considerable effort in resisting immediate temptation and studying without any reward from it.

Situations like this are common for individuals to fall into addictive behaviors. It is not uncommon to hear about people who get into a habit of drinking after an end of a relationship or some other unfortunate event. In essence - an individual has taken several actions over months or years in accordance to doers whose reward comes to fruition in the future, but as these rewards didn't get actualized in the future - the individual has much less incentive to seek farsighted rewards and hence turns to immediate inherent rewards.

To consider this formally, we define the notion of "the environment" which we define as all external influences and events that occur to an individual. At any point in time, an individual feels connected to future rewards at value $\alpha = \alpha_0$. As time progresses and they receive new experiences and the value of α increases or decreases depending on whether the experiences were better or worse than what the individual predicted. This can be conceptualized as an iterative game where you predict whether the environment will cooperate or defect with you and the environment sends you experiences as defined in (2) which communicate whether it cooperated or defected. If you predict cooperation and the environment cooperates or if you predict defection and the environment defects, then α increases as you predicted correctly. Similarly, if you predict the environment will defect and it cooperates, α increases as the future was better than expected. However if you predict the environment will cooperate and it defects, α decreases.

Therefore if a person is stuck in a string of defection, we need to change the environment of the individual such that they receive experiences they expect. Only after some amount of positive interactions with the environment can an individual encounter the PD with current self versus future self and reasonably expect to cooperate with future self. As humans are social creatures we have a natural desire to bond with other humans to some degree. If these connections deteriorate, we often try to bond with other things that may give us relief, whether that be gambling, drugs, sex or any other addictive activity. Environments that are more conducive to healthy social interactions should prove to be helpful in this realm.

The First Step to Recovery is Acceptance. Alcoholics anonymous have a popular saying stated above. One of the most important insights from this paper comes from examining the dynamics of α in the context of self control and discovering results that are conducive to the popular statement above. We previously stated that α increases or decreases depending on how accurately the individual predicted the future. Within each PD, an individual takes an action that a previous self predicted either correctly or incorrectly. Specifically, α increases in the C/C and D/D quadrant as the individual predicted the future correctly, but decreases in the C/D and D/C quadrant as the individual failed to predict the future correctly. We have already shown how addiction is described as an endless string of defection stuck in the D/C quadrant as the individual

will consistently tell themselves that they will quit in the future. If one simply accepts that they are an addict and will defect in the future - effectively putting them in the D/D quadrant, this will in time cause α to increase which will modify the utilities of the PD and eventually lead to ones where cooperation for current self yields a higher utility than defection would. In short, we see that getting in and out of addiction is directly related to the effectivity of predicting future experiences, whether that is from the environment or from internal perceptions of one's future actions. With this rudimentary setup of an addiction scenario, we see our model is conducive to natural results that we expect to arise given various parameters in the framework.

6 Limitations, Future Work and Conclusions

As our goal was to simply introduce this framework, we used very rough estimation of utility through the three factors 'Effortless?', 'Satisfaction?' and 'Sobriety?' with boolean values. Further work that investigates what factors are involved in these utility estimates and the domain of these factors would be extremely beneficial to add onto this work. We also left out any neurological discussion but further work that draws connections between this model and neuroanatomical explanations of addiction could be fruitful. We commented on how humans discount rewards hyperbolically in the time of the reward but didn't include it in the formalization of the framework due to requiring time indexing. A more comprehensive simulation with time indexing that factors in hyperbolic discounting could potentially yield insights that the current model fails to capture.

As this work is primarily within the context of behavioral economics, drawing connections to prospect theory could be insightful. Prospect theory examines how humans make decisions involving risk and claims humans are more sensitive to losses than gains. Our model provides results that are conducive to this as we show that only for high values of α (above 75%) is it rational to cooperate. This implies that for each negative interaction with the environment, an individual needs at least three positive interactions to return α to previous levels. Investigating the link between these descriptive frameworks could potentially allow for a learning opportunity for either model to adapt ideas from the other. This would align well with effort within the competitive multiagent systems community to adjust their models according to whether agents are risk seeking or risk averse [8, 9]. Lastly, given the iterative nature of the formalization of the human experience, this can be modelled by a Markov Decision Process (MDP). Further work that maps this formalization into a MDP and determines which policy ensues could yield interesting results when considering the characteristics of the policy.

In summary, addiction is an issue many people struggle with and researchers are investigating the causes and effects of it every day. This paper builds on previous research that considers addiction as an intertemporal Prisoner's Dilemma with your current self and future self. The main contribution of this paper is to formalize previously defined notions into a comprehensive framework that yields results that seem intuitive and explanatory. These results come from formally defining the notion of "the self" as well as various parameters involved in utility estimation. We provide three


basic parameters that can be used as a starting point for utility estimation and comment on how further research into these parameters is critical in improving the predictive power and usefulness of this framework. Finally, we comment on the main underlying assumptions and biases and mention future work that can build from this research. In all, this research constitutes an application of artificial intelligence to the crucial social problem of addiction.

References

1. Koob, G., Volkow, N.: Neurocircuitry of addiction. *Neuropsychopharmacology* **35**, 217–238 (2010). <https://doi.org/10.1038/npp.2009.110>
2. Lander, L., Howsare, J., Byrne, M.: The impact of substance use disorders on family and children: from theory to practice. *Soc. Work Public Health* **28**, 194–205 (2013). <https://doi.org/10.1080/19371918.2013.759005>. National Institute of Health
3. Khan, W., Hoey, J.: How different identities affect cooperation. In: Proceedings of the Humaine Association Conference on Affective Computing and Intelligent Interactions, San Antonio, Texas (2017)
4. Heshmat, S.: Behavioral economics of self-control. *Yale J. Biol. Med.* **88**, 333–337 (2015)
5. Thaler, R., Shefrin, H.: An economic theory of self-control. *J. Polit. Econ.* **89**(2), 392–406 (1981). <https://doi.org/10.1086/260971>
7. Monterosso, J., Ainslie, G.: The behavioral economics of will in recovery from addiction. *Drug Alcohol Depend.* (2007). <https://doi.org/10.1016/j.drugalcdep.2006.09.004>. National Institute of Health
7. Ayer, A.: *Hume: A Very Short Introduction*. Oxford University Press, Oxford (2000)
8. Hines, G., Larson, K.: Preference elicitation for risky prospects. In: 9th International Conference on Autonomous Agents and Multiagent Systems, Toronto, vol. 1–3, pp. 889–896 (2010)
9. Robu, V., Poutre, H.: Designing bidding strategies in sequential auctions for risk averse agents. *Multiagent Syst. Grid Syst.* **6**(5–6), 437–457 (2010). <https://doi.org/10.3233/MGS-2010-0160>



Infusing Domain Knowledge to Improve the Detection of Alzheimer's Disease from Everyday Motion Behaviour

Chao Bian¹, Shehroz S. Khan²✉ , and Alex Mihailidis^{1,2}

¹ Institute of Biomaterials and Biomedical Engineering,
University of Toronto, Toronto, Canada

chao.bian@mail.utoronto.ca, alex.mihailidis@utoronto.ca

² Department of Occupational Science and Occupational Therapy,
University of Toronto, Toronto, Canada
shehroz.khan@utoronto.ca

Abstract. Alzheimer's disease can severely impair the independent lifestyle of a person. Dem@Care is an European research project that conducted a study for timely diagnosis of Alzheimers disease by collecting everyday motion data from couples (or dyads), with one of the person in the couple having AD. Their results suggest that AD can be detected using everyday motion data from accelerometers. They did evaluation based on leave-one-person-out cross-validation. However, this evaluation can introduce bias in the classification results because one of the person from the dyad is present in the training set while the other is being tested. In this paper, we revisit the Dem@Care study and propose a new evaluation method that performs leave one-dyad-out cross-validation to remove the dataset selection bias. We then introduce new domain specific features based on dynamic and static intervals of motions that significantly improves the classification results. We further show increase in performance by combining the proposed features with new time, frequency domain and baseline features used in the Dem@Care study.

Keywords: Machine learning · Cross-validation · Accelerometer
Alzheimer's disease · Feature extraction

1 Introduction

As the population of elderly people increase, more cases of Alzheimer's disease (AD) and dementia are reported. According to Alzheimer's Association, one in 10 Americans age 65 and older has AD in 2017 [1]. AD is listed as the sixth-leading cause of death in the United States [1]. AD destroys brain cells that impairs the thinking ability and deteriorates memory, which in turn impairs everyday activities of daily living (ADL) by significantly changing the temporal structure of these activities [2]. An early detection and prediction of such behaviours may allow the application of required interventions to deal with this

cognitive degeneration problem. However, typical clinical assessment approaches have an episodic nature [3] and tend to introduce a high level of subjectivity [4]. Clinicians also lack a comprehensive view of the person’s ADL, which can only be provided by continuous monitoring [4].

With the advancements in sensor technology and cost effective sensing devices, it is feasible to perform continuous and longitudinal monitoring of elderly people in smart homes [2, 3]. Dem@Care is an European research project for timely diagnosis, assessment, maintenance and promotion of self-independence of people with dementia [5]. One of their studies is related to detecting the effects of AD from everyday motion behavior [2]. This study used machine learning methods on motion data collected from community-dwelling individuals with and without AD. Their results suggest that changes of everyday behavior in people with AD are detectable in motion data collected using accelerometer. They collected motion data from several dyads (couples) with one partner diagnosed with AD and one partner being a healthy control. They extracted frequency domain and principal component analysis (PCA) features from this data, tested several machine learning algorithms using leave-one-subject-out cross-validation (LOSOCV) and reported high classification accuracy. The LOSOCV methodology adopted in this study suffers from a drawback that one participant of the dyad is always present in the training set while the other participant of that dyad is being tested. Since both the participants in a dyad lived together, they may bear resemblance in their lifestyle and ADL. During the LOSOCV, an inadvertent bias may be introduced while training the model leading to high detection rates. In such a case, the classifier could be mapping a ‘similar’ participant present in the training set to a participant being tested, rather than identifying the patterns related to AD. To circumvent the problems associated with the above cross-validation method, we propose to use leave-one-dyad-out cross-validation (LODOCV) that keeps the dyad for testing separate from the training test. Then, we utilize domain knowledge about the motor behaviour of people with AD and infuse that knowledge to the classifiers with corrected evaluation method. Our results show improved detection rate by combining motion based domain features with new evaluation methodology.

The rest of the paper is structured as follows. Section 2 presents literature review on using sensor technology and machine learning techniques to identify AD or dementia. In Sect. 3, we describe the Dem@Care dataset, data pre-processing, and feature extraction methods. In Sect. 4, we present the new dynamic and static interval based features along with new time and frequency features to handle this problem. Section 5 presents the classification results under both LOSOCV and LODOCV framework and different feature regimes. Section 6 concludes the paper with pointers to future research directions.

2 Literature Review

Researchers have been using various sensors and machine learning techniques to detect dementia in older adults. Accelerometer is the most commonly used

body-worn sensor for dementia detection [6,7]. Researchers have also adopted heterogeneous sensors to create smart ambient environment, or smart homes, for assessing cognitive health and detecting dementia related symptoms. For instance, passive infrared motion sensors were installed in home environments to detect motion patterns of older adults at home [8]. Sensors were attached on objects or items at home such as door, telephone, broom, tea cup, kettle, etc., to monitor activities that involved interaction with these objects [9,10]. Wristband that contains sensors to monitor motion intensity and vital signs were also used in several research [3,8,11]. Vision sensors that are either in the form of camera or kinect are also used for this task [4,12,13].

The aforementioned sensors used in dementia detection generated heterogeneous data, which are used to extract features of monitored targets. Examples of those features include motion, physical activity, use of appliances, etc. Normally, the extracted features from sensor data are either statistical, temporal, frequency based, or domain specific. For instance, features extracted from accelerometers includes zero crossing rate [6,14], area under magnitude of the Fast Fourier Transform (FFT) [15], peak acceleration and energy [16], etc. Kinect-based vision sensor provided features like relative and absolute joint angles and distance to the hip center [17]. Smart home sensors usually generate event-based or triggering-based features such as binary, change-point and last-fired features [11]. These features were then fed into various algorithms for dementia detection. There were mainly two type of algorithms, namely, statistical [18] and machine learning [3,9,19]. Commonly used statistical methods were analysis of variance and correlation analysis. These methods were used to study the correlations between sensor data and dementia-related parameters such as apathy [20], agitation [14] and gait [21]. Various machine learning algorithms have been used to detect dementia-related symptoms or abnormal activities, for example, Support Vector Machine algorithm has been used for agitation detection [15,16], mobility assessment [22] and activities recognition [23].

Most of the studies were conducted in an instrumented lab environment [17,22], however, a free-living environment may be more naturalistic to test the real effect of proposed methods [2,23]. The study by Kirste et al. [2] involved two partners (a dyad) living in the same residence/home. In each dyad, one partner was diagnosed with AD and the other one was a healthy control. Moreover, the number of study participants in different studies varied greatly from less than 10 to more than 100 [6,14]. Some of the studies only recruited healthy adults and simulated behaviors of older adults based on defined protocols [3,16]. These proposed sensor systems and algorithms reported various detection accuracy for dementia. Alam et al. [23] and Dawadi et al. [24] both investigated automated assessment of activity performed by older adults with dementia to detect function decline due to reduced task quality. Alam [23] claimed sequencing and interruption scores that were higher than Dawadi et al.'s [24] in measuring correlation of supervised task quality scoring with observed activity performance of dementia patients using supervised machine learning methods. Marcén et al. reported 78.86% accuracy of classifying AD and healthy controls through detecting nocturnal agitation [16], whereas a similar study revealed a much higher accuracy of

94.1% on detecting psycho-motor agitation [15]. The varieties in reported accuracy can be explained from different settings of the studies, including the choice of sensors, data collection methods and evaluation criteria. Machine learning algorithms with different cross-validation evaluation methods can yield different accuracy even if other settings are kept same. K-fold [3, 23] and leave-one-out cross-validation techniques [19] were commonly used in the reviewed studies. Kirste et al. [2] and Chikhaoui et al. [17] used leave-one-subject-out cross-validation whereas Riboni et al. [9] used leave-one-day-out cross-validation.

In this paper, we revisit the study performed by Kirste et al. [2] to detect AD from everyday motion data using accelerometers. We will use their data, test it with a new cross-validation method and introduce new dynamic and static interval based domain features and compare with their method of evaluation.

3 Dem@Care Study

3.1 Dataset

The study by Kirste et al. [2] attached an ankle-mounted accelerometer on each participant to collect their everyday motion behavior data. The raw data was the magnitude value of the normalized acceleration measured by the 3 axes accelerometer. The data set included motion data from 78 subjects (39 dyads). In the dataset, there was no data for 2 subjects; therefore, the number of subjects for this paper is 76 (38 dyads). An average of 53.4 h of data were collected in the original study. We were shared with 5 types of data sets that were prepared by processing the data using a low pass filter with different frequencies (0.5, 5, 25, 50 and 250 mHz). The data sets were named F1 to F5 corresponding to the five frequencies. For instance, the F1 data set was created from the raw data using a 0.5 mHz low pass filter.

3.2 Data Processing

As each subject were sampled at different times using the same sampling rate, the timestamps for each subject were slightly different even if we selected the same time window, for instance, between 22:00 on Day 1 and 22:00 on Day 2. Table 1 shows an example for the first two timestamps after 22:00 on Day 1 for two subjects. It can be seen that the data were sampled at different starting time after 22:00, one was at 22:02 and the other one was at 22:08. As all subsequent data were sampled in 50 mHz (every 20 s) in this setting, the data for different subjects were therefore not sampled at the same time points. To make the samples comparable with each other, an interpolation of data is needed to synchronize data for all subjects. Ignoring this step may lead to undesirable results. To make the data points consistent with timestamps, we generated a reference array of timestamps between 22:00 on Day 1 to 22:00 on Day 2 based on a sampling rate used to generate the corresponding data set. The starting time for all 5 data sets was exact 22:00. Thus, an array containing reference

Table 1. Example of timestamps in different samples

Sample	Motion data	
	Timestamp	Value
Participant 1	22:02	1.7084
	22:22	1.8173
Participant 2	22:08	2.1343
	22:28	2.0467

Table 2. Model settings for the best classification accuracy

Category	Setting
Feature	4320 FFT
Reduced features	5 PCA features
Time window for data collection	22:00–22:00
Low pass filter on datasets	50 mHz low pass filter
Machine learning classifier	Quadratic discriminant analysis

timestamps can be obtained. For example, for the data set sampled in 50 mHz, we obtained an array of 4320 timestamps ($24\text{ h} * 60\text{ min} * 60\text{ s} * 0.05$). Then, we used MATLAB’s *interp1* function [25] to generate the interpolated data. To prepare for interpolation, data of every subject were trimmed to 24-h data. Only the data that has timestamps between 22:00 on Day 1 to 22:00 on Day 2 was kept. The rest of data were discarded in this paper.

3.3 Model Settings

The machine learning model that gave the highest classification accuracy in the study by Kirste et al. [2] applied the model settings shown in Table 2. PCA was used to reduce the dimension of features generated by applying FFT on the original data. Six different time windows were tested in the original model. The 24-h time window, which is from 22:00 on Day 1 to 22:00 on Day 2, was reported to give the best results. The better results on 24-h time windows over other sub-windows of smaller duration may be because the time window includes a full cycle of a person’s motion patterns from getting up from bed to going to sleep. This will cover different types of activities that a person would perform in a typical day. Therefore, we use the 24-h window to build our models. They tested different machine learning classifiers and found that Quadratic Discriminant Analysis (QDA) classifier with the data set processed using 50mHz filter showed the best classification accuracy. To evaluate the classifiers, they adopted LOSOCV methodology.

4 Revised Evaluation and Infusing Domain Knowledge

In the Dem@Care study, LOSOCV is employed for the evaluation of classifiers. This means that one of the person from the dyad will be present in the training set. This can bias the performance because both the person in the dyad lived together and may have done similar activities. To avoid such evaluation bias, we present LODOCV. In LODOCV, if there are N dyads present (or $2N$ total number of subjects), we train a classifier on $(N - 1)$ dyads (or $(2N - 2)$ subjects) and test on N^{th} dyad (or 2 subjects). We repeat the process N times and present the average performance. Performing LODOCV ensures that one dyad for testing will not be observed during training and biases due to similar actions of the subjects may be eliminated. To evaluate the effect of the new LODOCV evaluation on the Dem@Care dataset (discussed in Sect. 3.1), we extracted 4320 FFT features from the raw data and computed 5 PCA features on the entire feature data. Then, we trained different machine learning algorithms on the 5 data sets corresponding to different low pass filters (F1–F5). The machine learning classifiers used in this paper were Support Vector Machine (SVM), Random Forest (RF), and Quadratic Discriminant Analysis (QDA). For RF classifier, the number of trees was 100. LODOCV were performed with all machine learning classifiers along with LOSOCV to compare different evaluation methods.

4.1 Time and Frequency Features

In order to investigate the performance of revised evaluation methodology, we extract the following additional 10 time and frequency domain features [26] from the different accelerometer datasets within a time window of 24 h:

- Mean, maximum, minimum, standard deviation, inter-quartile range,
- Total average power of power spectral density,
- Spectral entropy,
- DC component of energy,
- Normalized energy, and
- FFT entropy

These features have been used earlier for activity recognition tasks. We compare and combine the time/frequency domain features with the proposed features, which are described in the next section.

4.2 Dynamic/Static Interval Based Features

People with dementia/AD may exhibit different behaviours than healthy older adults, especially at different times of the day. This is referred to as ‘sundowning’ effect, where a person with dementia/AD can become more agitated, aggressive or confused during late afternoon or early evening [27]. Hsu et al. [21] show that gait and balance based quantitative measurements can be good indicators for

Table 3. The extracted domain features

Number of points in each dynamic and static intervals in 24 h
Number of dynamic and static intervals in 24 h
Number of dynamic and static intervals from 12:01 am–6:00 am
Number of dynamic and static intervals from 6:01 am–12:00 pm
Number of dynamic and static intervals from 12:01 pm–6:00 pm
Number of dynamic and static intervals from 6:01 pm–12:00 am

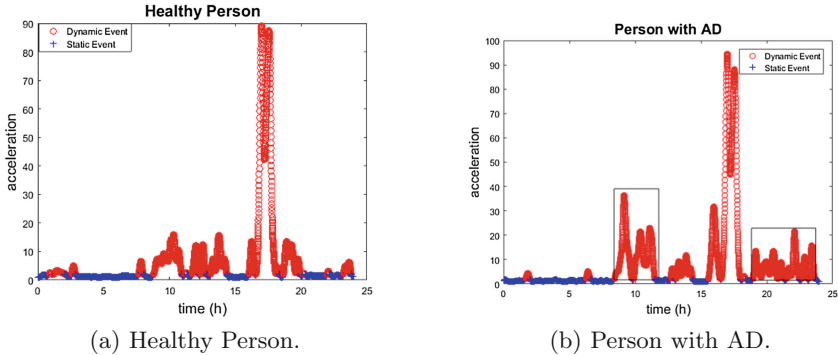


Fig. 1. Dynamic and Static intervals of a dyad that comprises of a healthy person and with AD. (Color figure online)

early diagnosis of AD. We extend this idea to measure dynamic and static intervals from the accelerometer readings. A dynamic or static interval is defined as the total time period during which a person has more motor activities or not. The level of activity was deduced from the acceleration magnitude values that was consecutively higher or lower than a pre-defined value. This pre-defined threshold was empirically set to 4 in the unit of acceleration magnitude. The choice of the value 4 was based on the best accuracy obtained from multiple experiments using values range between 1–5. This threshold will determine dynamic and static intervals in the overall motion activities of a person during the whole day. Figure 1a and b shows the dynamic intervals (in red color) and static intervals (in blue color) for a dyad that has a healthy person and another with AD at different times of the day. We can visually observe that the person with AD is more active during the later part of the day. We extracted 12 features based on these dynamic and static intervals. The features were extracted from the entire 24 h data to obtain an overall motion pattern. Then, the day’s data was divided into four 6-h periods to capture motion information that may arise due to sun-downing effect or other behaviours in specific quarters of the day. The following is the table of the extracted features (Table 3):

5 Results

To compare the performance of various machine learning algorithms on different versions of the Dem@Care datasets, we did the following experiments:

1. Compare the LOSOCV and LODOCV on the original 5 PCA features. This will act as a baseline for our analysis.
2. Compare the LOSOCV and LODOCV on time/frequency features, dynamic and static interval based features and their combinations with PCA features. This experiment will investigate the effect of new features in improving the baseline.

We used area under the ROC curve (AUC) as the performance metric, averaged across different folds.

5.1 PCA Features Baseline

Table 4 shows the AUC values on different datasets with 5 PCA features using LOSOCV and LODOCV evaluations. The number of PCA components is set to be the same as the work of Kirste et al. [2] for fair comparison. The best AUC obtained by LOSOCV is 0.73 for datasets F4 and F5 and 0.64 for LODOCV for dataset F4 using the QDA classifier. It is to be noted that in the LOSOCV evaluation 75 participants are used for training and for LODOCV 74 participants

Table 4. Performance of different classifiers and evaluation methods using 5PCA features.

Dataset	Methods	LOSOCV	LODOCV
F1	SVM	0.45	0.43
	RF	0.46	0.45
	QDA	0.56	0.51
F2	SVM	0.5	0.45
	RF	0.42	0.4
	QDA	0.54	0.49
F3	SVM	0.44	0.43
	RF	0.39	0.38
	QDA	0.73	0.64
F4	SVM	0.46	0.45
	RF	0.42	0.36
	QDA	0.73	0.58
F5	SVM	0.45	0.38
	RF	0.42	0.39
	QDA	0.55	0.5

(or 37 dyads) were used for training. There is a difference of only ‘one’ participant during the training phase; however, the effect of performance is significant. This experiment highlights the bias that may be induced by including a person from the dyad to be tested in the training set as it may lead to identifying similar activities rather than patterns to discriminate AD and healthy persons. The drop in AUC by using LODOCV is expected as both the participants in the dyad were not observed during training. These results serve as a baseline for both types of evaluation methods when comparing the performance with other types of features.

5.2 Time, Frequency and Dynamic/Static Interval Based Features

The second experiment deals with evaluating different classifiers on the time, frequency and dynamic/static interval based features and their combinations with PCA features. We evaluated the classifiers on the following different feature regimes:

- 10 time and frequency features (10TF, discussed in Sect. 4.1).
- 12 new domain features (12D, discussed in Sect. 4.2).
- Combination of 10 time and frequency features and 5 PCA features (10TF + 5PCA).
- Combination of 12 domain features and 5 PCA features (12D + 5PCA).
- Combination of 12 domain features and 10 time and frequency features (12D + 10TF).
- Combination of 12 domain features, 10 time and frequency features and 5 PCA features (12D + 10TF + 5PCA).

The classification results using 10TF, 10TF + 5PCA and 12D features are shown in Table 5. The best AUC obtained for LOSOCV is 0.76 for F3 dataset using QDA classifier with combining time, frequency and PCA features, which is better than only using PCA features. This shows that adding additional time and frequency features improves the performance in the case of LOSOCV evaluation. For LODOCV, the best AUC is 0.71 achieved by using F3 dataset, RF classifier with domain features. The AUC outperforms the combinations of PCA, time and frequency features for the case of LODOCV evaluation. Therefore, this experiment suggests that domain specific features improve the performance of LODOCV evaluation.

To analyze the effect of domain features together with time, frequency and PCA features, we combine these features in different ways and compare their results in Table 6. With LOSOCV evaluation, the highest AUC of 0.74 is obtained for dataset F3 with QDA classifier using 12D + 10TF + 5PCA features. This AUC is less than the best AUC obtained by combining 10TF and 5PCA. This shows that dynamic/static interval based domain features do not help much for the LOSOCV evaluation. On the contrary, the best AUC of 0.73 is obtained using dataset F4, RF classifier, 12D + 10TF + 5PCA feature combination and LODOCV. This AUC is better than that of using domain features only (as shown

Table 5. Performance of different classifiers and evaluation methods using 10TF, 10TF + 5PCA and 12D features.

Feature	Methods	LOSO CV			LODO CV		
		10TF	10TF + 5PCA	12D	10TF	10TFD + 5PCA	12D
F1	SVM	0.48	0.48	0.52	0.49	0.49	0.51
	RF	0.52	0.45	0.46	0.46	0.46	0.56
	QDA	0.61	0.61	0.54	0.5	0.53	0.46
F2	SVM	0.45	0.45	0.52	0.45	0.45	0.55
	RF	0.55	0.54	0.41	0.48	0.42	0.62
	QDA	0.69	0.64	0.56	0.54	0.52	0.47
F3	SVM	0.52	0.52	0.52	0.52	0.52	0.67
	RF	0.58	0.48	0.36	0.49	0.45	0.71
	QDA	0.68	0.76	0.60	0.53	0.61	0.46
F4	SVM	0.53	0.53	0.42	0.52	0.52	0.65
	RF	0.42	0.44	0.37	0.35	0.40	0.64
	QDA	0.68	0.73	0.42	0.52	0.57	0.57
F5	SVM	0.44	0.44	0.49	0.38	0.45	0.53
	RF	0.4	0.4	0.46	0.34	0.40	0.58
	QDA	0.58	0.52	0.57	0.49	0.46	0.44

Table 6. Results for domain features combined with statistical and spectral features

Dataset	Method	LOSO CV			LODO CV		
		12D + 5PCA	12D + 10TF	12D + 10TF + 5PCA	12D + 5PCA	12D + 10TF	12D + 10TF + 5PCA
		AUC	AUC	AUC	AUC	AUC	AUC
F1	SVM	0.41	0.48	0.48	0.51	0.51	0.51
	RF	0.37	0.49	0.49	0.61	0.55	0.58
	QDA	0.58	0.58	0.59	0.46	0.51	0.49
F2	SVM	0.62	0.45	0.45	0.55	0.55	0.55
	RF	0.52	0.52	0.53	0.54	0.57	0.48
	QDA	0.54	0.62	0.57	0.49	0.44	0.48
F3	SVM	0.38	0.52	0.52	0.67	0.48	0.48
	RF	0.38	0.45	0.47	0.64	0.59	0.58
	QDA	0.67	0.69	0.74	0.41	0.39	0.37
F4	SVM	0.38	0.53	0.53	0.65	0.48	0.48
	RF	0.36	0.35	0.35	0.72	0.69	0.73
	QDA	0.46	0.54	0.56	0.58	0.53	0.52
F5	SVM	0.54	0.44	0.44	0.53	0.55	0.55
	RF	0.45	0.49	0.46	0.54	0.59	0.56
	QDA	0.55	0.60	0.58	0.47	0.46	0.49

in Table 5). This shows that combining domain features with time, frequency and PCA features improves the classification performance in patients with AD and healthy controls.

Through these experiments, we corrected the evaluation strategy of the Dem@Care study, and improved the classification performance in detecting AD from everyday motion data by infusing domain knowledge features along with time, frequency and PCA features.

6 Conclusion

Detecting AD from everyday motion behaviour is a challenging problem. In this paper, we revisited the Dem@Care study that identifies people with AD by capturing their everyday motion data. Their evaluation method uses a subject from the dyad in training while the other is being tested, which can induce bias in the overall performance of the machine learning methods. To handle this problem, we proposed an evaluation methodology that keeps a full dyad out for testing without influencing the training set. Then, we propose new dynamic and static intervals features that are derived from different motor behaviours of persons with AD at different intervals of the day. By infusing domain features along with time, frequency features and the features used in Dem@Care study, significant improvement in performance was obtained using LODOCV evaluation strategy. In future, we will work on automating the threshold to distinguish static and dynamic intervals and extract more discriminatory features.

References

1. Association, A.: 2017 Alzheimer's disease facts and figures. *Alzheimer's & Dementia J. Alzheimer's Assoc.* **13**(4), 325–373 (2017)
2. Kirste, T., Hoffmeyer, A., Koldrack, P., Bauer, A., Schubert, S., Schröder, S., Teipel, S.: Detecting the effect of Alzheimer's disease on everyday motion behavior. *J. Alzheimer's Dis.* **38**(1), 121–132 (2014)
3. Janjua, Z.H., Riboni, D., Bettini, C.: Towards automatic induction of abnormal behavioral patterns for recognizing mild cognitive impairment. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 143–148. ACM (2016)
4. Karakostas, A., Meditskos, G., Stavropoulos, T.G., Kompatsiaris, I., Tsolaki, M.: A sensor-based framework to support clinicians in Dementia assessment: the results of a pilot study. *Adv. Intell. Syst. Comput.* **376**, 213–221 (2015)
5. Dem@Care: Dem@Care Project. <http://www.demcare.eu/>. Accessed on 13 Aug 2017
6. Tung, J., Semple, J., Woo, W., Hsu, W.: Ambulatory assessment of lifestyle factors for Alzheimer's Disease and related Dementias. In: *AAAI Spring Symposium*, pp. 50–54 (2011)
7. Ando, B., Baglio, S., Lombardo, C.O., Marletta, V.: A multisensor data-fusion approach for ADL and fall classification. *IEEE Trans. Instrum. Meas.* **65**(9), 1960–1967 (2016)
8. Akl, A., Taati, B., Mihailidis, A.: Autonomous unobtrusive detection of mild cognitive impairment in older adults. *IEEE Trans. Biomed. Eng.* **62**(5), 1383–1394 (2015)

9. Riboni, D., Bettini, C., Civitarese, G., Janjua, Z.H., Helaoui, R.: SmartFABER: recognizing fine-grained abnormal behaviors for early detection of mild cognitive impairment. *Artif. Intell. Med.* **67**, 57–74 (2016)
10. Dawadi, P.N., Cook, D.J., Schmitter-Edgecombe, M.: Automated cognitive health assessment using smart home monitoring of complex tasks. *IEEE Trans. Syst. Man Cybern. - Part C: Appl. Rev.* **43**(6), 1302–1313 (2013)
11. Arifoglu, D., Bouchachia, A.: Activity recognition and abnormal behaviour detection with recurrent neural networks. *Procedia Comput. Sci.* **110**, 86–93 (2017)
12. Sacco, G., Joumier, V.V., Darmon, N., Dechamps, A., Derreumaux, A., Lee, J.H.H., Piano, J., Bordone, N., Konig, A., Teboul, B., David, R., Guerin, O., Bremond, F.F., Robert, P.: Detection of activities of daily living impairment in Alzheimer's disease and mild cognitive impairment using information and communication technology. *Clin. Interventions Aging* **7**, 539–549 (2012)
13. Crispim-Junior, C.F., Joumier, V., Hsu, Y.L., Pai, M.C., Chung, P.C., Dechamps, A., Robert, P., Bremond, F.: Alzheimer's patient activity assessment using different sensors. *Gerontechnology* **11**(2), 266–267 (2012)
14. Nagels, G., Engelborghs, S., Vloeberghs, E., Van Dam, D., Pickut, B.A., De Deyn, P.P.: Actigraphic measurement of agitated behaviour in Dementia. *Int. J. Geriatr. Psychiatry* **21**(4), 388–393 (2006)
15. Pedro, S., Quintas, J., Menezes, P.: Sensor-based detection of Alzheimer's disease-related behaviors. In: *The International Conference on Health Informatics, IFMBE Proceedings* (2014)
16. Marcén, A.C., Carro, J., Monasterio, V.: Wearable monitoring for the detection of nocturnal agitation in Dementia. In: *Proceedings of the 6th International Joint Conference on Pervasive and Embedded Computing and Communication Systems*, pp. 63–69, January 2016
17. Chikhaoui, B., Ye, B., Mihailidis, A.: Ensemble learning-based algorithms for aggressive and agitated behavior recognition. In: *Proceedings of the 10th International Conference on Ubiquitous Computing & Ambient Intelligence (UCAMI 2016)* (2016)
18. David, R., Rivet, A., Robert, P.H., Mailland, V., Friedman, L., Zeitzer, J.M., Yesavage, J.: Ambulatory actigraphy correlates with apathy in mild Alzheimer's disease. *Dementia* **9**(4), 509–516 (2010)
19. Miranda, D., Favela, J., Ibarra, C.: Detecting State Anxiety When Caring for People with Dementia. In: Bravo, J., Hervás, R., Villarreal, V. (eds.) *AmIHEALTH 2015*. LNCS, vol. 9456, pp. 98–109. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26508-7_10
20. Kuhlmei, A., Walther, B., Becker, T., Müller, U., Nikolaus, T.: Actigraphic daytime activity is reduced in patients with cognitive impairment and apathy. *Eur. Psychiatry* **28**(2), 94–97 (2013)
21. Hsu, Y.L., Chung, P.C., Wang, W.H., Pai, M.C., Wang, C.Y., Lin, C.W., Wu, H.L., Wang, J.S.: Gait and balance analysis for patients with Alzheimer's disease using an inertial-sensor-based wearable instrument. *IEEE J. Biomed. Health Inform.* **18**(6), 1822–1830 (2014)
22. Richter, J., Wiede, C., Hirtz, G.: Mobility assessment of demented people using pose estimation and movement detection an experimental study in the field of ambient assisted living. In: *4th International Conference on Pattern Recognition Applications and Methods, Proceedings, ICPRAM 2015*, vol. 2, pp. 22–29 (2015)

23. Alam, M.A.U., Roy, N., Holmes, S., Gangopadhyay, A., Galik, E.: Automated functional and behavioral health assessment of older adults with Dementia. In: First IEEE Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), Washington D.C., May–June 2016
24. Dawadi, P.N., Cook, D.J., Schmitter-Edgecombe, M.: Automated cognitive health assessment from smart home-based behavior data. *IEEE J. Biomed. Health Inform.* **20**(4), 1188–1194 (2016)
25. The MathWorks Inc.: MATLAB R2016b
26. Khan, S.S., Karg, M.E., Kuli, D., Hoey, J.: Detecting falls with X-Factor hidden Markov models. *Appl. Soft Comput.* **55**, 168–177 (2017)
27. Alzheimer's Society (GB): Sundowning - behaviour changes. https://www.alzheimers.org.uk/info/20064/symptoms/87/behaviour_changes/8. Accessed 11 Jan 2018



An Incremental Machine Learning Algorithm for Nuclear Forensics

Chris Drummond(✉)

Data Technology Research Center, National Research Council Canada,
Ottawa, ON K1A 0R6, Canada
Chris.Drummond@nrc-cnrc.gc.ca

Abstract. This paper presents an incremental machine learning algorithm that identifies the origin, or provenance, of samples of nuclear material. This is part of work being undertaken by the Canadian National Nuclear Forensics Library development program, which seeks to build a comprehensive database of signatures for radioactive and nuclear materials under Canadian regulatory control. One difficulty with this application is the small ratio of the number of examples over the number of classes. So, we introduce variants to a basic generative algorithm, based on ideas from the robust statistics literature and elsewhere, to address this issue and to improve robustness to attribute noise. We show experimentally the effectiveness of the approach, and the problems that arise, when adding new examples and classes.

1 Introduction

Canada, amongst other nations, is in the process of developing a National Nuclear Forensics Library cataloging signatures for radioactive and nuclear materials under Canadian regulatory control. Although sources are distributed throughout the world, the samples in question were, at some time, within the Canadian regulatory system. Each library entry is produced by analyzing a sample of material using the tools of analytical chemistry: spectrometry, chromatography, microscopy, physical morphology etc. This is used to establish the origin, or provenance, of the material and thereby provide useful information to an investigator to aid discovery of how the material was lost from regulatory control.

In this paper, we discuss early steps in that process. Our first task is to identify the source of samples of Uranium ore concentrate. Although a material of relatively low radioactivity, it is the first stage of the nuclear fuel cycle and also acts as a good initial test case. Each sample comes from a single mine and is processed by a single mill. When analyzed through mass spectrometry, compositions are determined for around 60 elements, later reduced to 23 for model building, that make up the sample. For reasons of confidentiality, we are only given a label which signifies a particular combination of mine and mill, termed a producer. To achieve identification, we train a classification model on a collection of samples. We have at present some 300 plus examples from 20 distinct producers. It is anticipated that both numbers will grow with time.

© Crown 2018

E. Bagheri and J. C. K. Cheung (Eds.): Canadian AI 2018, LNAI 10832, pp. 194–207, 2018.

https://doi.org/10.1007/978-3-319-89656-4_16

Here, we detail the development and testing of an incremental classifier that can update the model as new examples and new classes, the producers here, become available and are added to the library.

One way to divide machine learning algorithms is into discriminative and generative types. A discriminative algorithm constructs a decision boundary between classes, so as to totally partition the input space into separate regions. As a simple example, a linear discriminator might be used to separate positive and negative examples. In the two dimensional case, this is simply a line. For higher dimensional cases, it would be a hyper-plane. When in use, if the sample falls in a region associated with one particular class it is labeled with that class name. The shape of the decision boundary is determined by the algorithm used and is typically much more complex than a hyper-plane. It is made even more complex, in this application, by having twenty rather than two classes. Each one must be separated from all the others by a decision boundary.

A generative algorithm learns a representation of the full joint probability distribution of all classes across the input space. Sampling from that distribution would generate new, albeit artificial, samples, hence the name. In practice though, decision theory is often applied to the distribution to classify new examples, effectively partitioning the input space. It might seem, therefore, that there is little advantage to generative algorithms if the end task is simply to classify new samples. In fact, generally, if the data set is static, it is commonly held that discriminative algorithms have higher classification accuracy. The intuition being that focusing strongly on the particular task, i.e. classifying the examples, is better than learning more than is needed, i.e. a full distribution [1]. Yet, some have shown that the choice between them, even for a discriminative task, is dependent on the number of examples [2].

Certainly, it is when things are subject to change, such as for incremental learning, that generative algorithms come into their own. To make a discriminative algorithm be able to add new examples and classes, it is often necessary to store all the old examples. In some cases it may be necessary to completely retrain the model when new data are received. This is certainly a counter-intuitive way to produce an incremental algorithm. As the number of examples increases, it also becomes computationally expensive. Whereas for a generative algorithm, it is often sufficient to simply update the parameters of the model. If the number of parameters is much smaller than the number of examples then processing is much faster.

The contributions of this paper are threefold. The application, itself, is an interesting one where machine learning has a lot to offer. Generally, analytical chemistry applications would often benefit from such expertise. The paper introduces one way of using shrinkage estimators for generative algorithms to improve performance when the number of samples is small. It also shows how to construct an ensemble generative algorithm, based on random forests, which is more robust to noise, than the basic version, when classifying new examples.

2 Related Work

There are two types of work most closely related to that presented here. That which deals with identifying the provenance of radioactive and nuclear materials and that which addresses making learning algorithms incremental.

Research in the area of nuclear forensics has been around for more than a decade. Analytical chemists have typically applied their expertise to select elements in the Uranium ore concentrate that are effective in identifying the source of the material. Different factors have been used. One is the ratio of isotopic variants of the same element. This may involve isotopes of uranium, say $^{235}\text{U}/^{238}\text{U}$, but other elements such as lead [3] and neodymium [4] have also been used. Another source of information is the trace elements, ones found in relatively low concentration in the sample. Perhaps the strongest focus of attention recently has been on the rare earth elements [5]. Not only are they good geolocators but also they are relatively unaffected by the milling process. This work uses data for around 23 elements with some feature generation to expand the number of attributes to 32. Future work will add isotopic ratios of lead, uranium and other elements, plus values such as those for the physical characteristics of the sample material.

So far, the main analytic tools that have been used, for forensics, are those found in chemometrics [6]. These largely come from multivariate statistics. These include Principal Components Analysis and the closely related Partial Least Squares Discriminant Analysis. More recently, though, algorithms from machine learning such as neural networks and support vector machines have become more popular [7]. In some of our own earlier experiments, we found that random forests worked best for this problem. Incremental versions of this algorithm have been developed [8]. Indeed, variants of other standard discriminative learning algorithms have been made incremental since the early days of machine learning [9]. However, to make discriminative algorithms, such as decision trees, loss-less, i.e. for there to be no loss of information when adding new data, often all examples had to be stored [10]. Even the most theoretically justified SVM must keep “reserved instances”, the greater the number the less likely for errors in the margin [11]. There are some types of fitting schemes, such as least squares, which can also be made incremental and loss-less. However, for most discriminative algorithms this is not possible. Which is why this paper focuses on a type of generative algorithm.

3 An Incremental Algorithm

To use a generative algorithm, we need a way to represent the joint probability distribution. Intuitively, a Bayesian network would fit the bill. However, at present, all our attributes are continuous variables whereas such networks have typically assumed discrete probabilities [12]. We could discretize continuous values or look for alternatives. For continuous variables, some versions of Naive Bayes use either a normal distribution or kernels, a parametric or non-parametric

representation of the probability distribution. One disadvantage of the latter is that complexity is linear in the number of examples, one kernel is needed for every data point. Although strictly, it doesn't have to be retrained, unlike the discriminative algorithm, it does effectively remember all the examples. Again, counter to intuition as to what an incremental algorithm should be. The number of components for a normal distribution will be constant determined by the number of classes and variables. At present, we have only a small number of examples and a non-parametric representation would be very competitive. Yet, the point of this work is to deal with an ever increasing amount of data. Thus, the parametric form should come into it own.

3.1 The Basic Algorithm

For a uni-variate normal distribution, the sufficient statistics are the mean and variance, i.e. together they completely characterize the distribution. For a multivariate normal distribution, the equivalent are the mean vector and the covariance matrix. The Naive Bayes algorithm only needs the mean and variance for each attribute. Covariances are not modeled due to the assumption of the independence of attributes given the class. Using the full matrix gives greater flexibility but considerably increases the number of parameters ($O(n) \rightarrow O(n^2)$). Future work will determine what is gained by the extra degrees of freedom.

The mean is simply the cumulative sum of the values for each dimension, normalized by the number of values n , see Eq. 1. As we have one mean for each dimension, the result is a vector whose length is the number of dimensions, each attribute being a dimension, see Eq. 2. For each attribute, we can store the sum and n separately (strictly, we only need one n for all dimensions). When a new example is added, all cumulative sums are updated and n incremented.

$$\mu_x = \frac{1}{n} \underbrace{\sum_i^n x_i}_{\text{cumulative sum}} \tag{1}$$

$$\mu = [\mu_1, \mu_2, \dots, \mu_n] \tag{2}$$

$$\begin{aligned} COV(X, Y) &= E[(X - \mu_x)(Y - \mu_y)] \\ &= E[XY] - \mu_x \mu_y \\ &= \frac{1}{n} \underbrace{\sum_i^n x_i y_i}_{\text{cumulative sum}} - \mu_x \mu_y \end{aligned} \tag{3}$$

$$\Sigma = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \sigma_{13}^2 & \dots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \sigma_{23}^2 & \dots & \sigma_{2n}^2 \\ \dots & \dots & \dots & \dots & \dots \\ \sigma_{n1}^2 & \sigma_{n2}^2 & \sigma_{n3}^2 & \dots & \sigma_{nn}^2 \end{bmatrix} \tag{4}$$

Although slightly more complex, we can do the same thing for the covariance matrix. The covariance, given by Eq. 3, can be decomposed into three parts; again a cumulative sum and the normalizing factor n but also a product of the means. We already know how to calculate the means. So, if we simply store this cumulative sum then we can easily calculate the covariance. The covariance matrix is a symmetric square matrix with the number of rows and columns equal to the number of dimensions, as seen in Eq. 4. Each entry can be calculated by using Eq. 3 (NB: when the indices are identical, i.e. σ_{ii}^2 , it represents the variance). So, instead of storing the mean and covariance, we store a vector and matrix of the relevant cumulative sums and a scalar for the number of samples. Using this information, we can generate the moments when they are used during classification. We need a scalar, vector and matrix for each class. If new samples, for an existing class, are introduced then simply adding to the cumulative sums and n will update the model. To add a new class, we need to construct a new normal distribution through storing the same information as for other classes. The change in classification occurs as the new density function enters into the calculation of the posterior probability.

3.2 The Modified Learner

In the previous section, the basic algorithm was outlined but there are further considerations guided by the application itself. The distribution of the values of individual chemical elements are clearly not Gaussian. For one thing, they are typically positive. Some papers in the geological literature contend that a log-normal might be a better fit [13]; although this view is not universally accepted [14]. So, a log transform might be useful. However, when the concentration of an element in the sample is below the method detection limit, one can sometimes obtain a negative value. This could be replaced by the detection limit, or zero. However, some initial experiments suggested these substitutions had an undesirable effect. There are a number of methods of transformation that produce a closer to normal distribution. The Box-Cox transformation is the most well known and includes the log function. It is however only for use with positive numbers. An alternative that also works with negative numbers is the Yeo-Johnson transformation [15]. Applying a simplified version to the attribute values, see Eq. 5, considerably improves the accuracy of the classifier.

$$x_i = \begin{cases} \log(x_i + 1) & \text{if } x \geq 0 \\ -[(-x_i + 1)^2 - 1]/2 & \text{if } x < 0 \end{cases} \quad (5)$$

Another difficulty is that we have relatively few samples per class as compared with the number of attributes. So, although we have 307 examples over 33 attributes, when the class is taken into account we only have between 11 and 17 examples from which to learn the model. The idea of shrinkage estimators, introduced by James and Stein [16], have been around for some time. They bias an estimated value towards a fixed value, sometimes zero. These estimators are proven to have lower mean squared error than the normal least square sample

estimates. They are particularly important when the number of attributes is bigger than the number of samples, as in this application.

We use a shrinkage estimator for the covariance matrix for each class. This is estimated by the sum, strictly a convex combination, of the sample estimate S , as discussed in the previous section, and some target matrix T , see Eq. 6. The choice of target matrix is an area of study but alternatives have been proposed in the literature: an identity matrix, a diagonal matrix of the sample variances, a diagonal matrix with values an average of all variances. We use the last as it seems the most popular and empirically does well, indicated in Eq. 6 by $c\hat{\sigma}I$, a product of a constant, the average variance and the identity matrix. One problem is choosing the value for c . We could do this by a number of methods but many are data dependent. That would negate our ability to use cumulative sums and, therefore, producing a cumulative algorithm. Instead, a simple equation is used, the ratio of the number of attributes over the number of examples squared. This embeds the idea that as the number of examples increases the bias towards the target matrix should decrease and works empirically well.

$$\begin{aligned} COV(X, Y) &= \alpha S + (1 - \alpha)T \\ &= \alpha S + (1 - \alpha)c\hat{\sigma}I \end{aligned} \tag{6}$$

As noted before, the best discriminative algorithm for this application is the random forest. Although, the overall performance is not significantly better than other algorithms, it is much more robust to added attribute noise. The main idea borrowed from the random forest algorithm is the use of multiple models learned on different attribute subsets. We repeatedly randomly select a subset of attributes, learn a normal model based solely on these and then combine the output of each model to make a probability estimation. Here, a robust statistic, the trimmed mean that rejects the most extreme 20% of values, gave the best results. 100 normal distributions of 8 randomly chosen attributes for each class was found empirically effective. The number of parameters is still fixed although it has increased over the basic algorithm. The probability of each class is assumed to be identical. Although the number of examples per class differs, this is more likely an artifact of how the data was collected rather than a true prior distribution.

4 Experimental Evaluation

This section begins by showing the robustness to attribute noise obtained by using multiple normal distributions defined on small random subspaces. The first row of Table 1 is for a single one for all 32 attributes. The second is for 100, the third for 500 both on random subsets of 8 attributes. As the standard deviation of the $\mathcal{N}(0, \sigma^2)$ noise increases, the accuracy of the first drops off faster than the others. The more distributions used the more robust the result, though at a cost of more parameters.

The rest of this section discusses experimentally evaluating the algorithm for two possible additions: a new instance and a new producer.

Table 1. Robustness to noise

σ	0	0.2	0.4	0.6	0.8	1
1 by 32	0.977	0.988	0.873	0.633	0.454	0.308
100 by 8	0.977	0.971	0.882	0.694	0.527	0.403
500 by 8	0.979	0.977	0.889	0.707	0.536	0.416

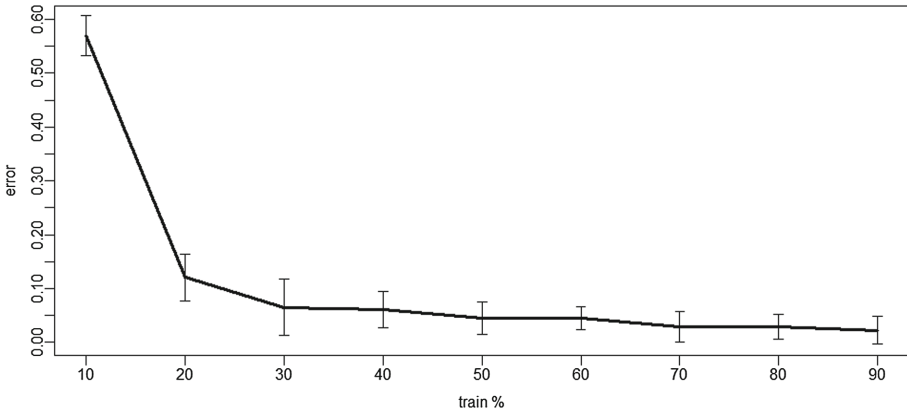


Fig. 1. The learning curve

4.1 Adding Instances

Figure 1 is the learning curve when adding instances, error bars show the standard deviation. The data is split into 10 subsets, as is normal for 10 fold cross validation, with one subset, or fold, is held back for testing. This is repeated until all folds have been used. Here, instead of the training data being composed of all the remaining folds, we train on one fold at a time. Thus, progressively more data is added to the model. Then, we determine the error rate of the models on the test fold. As the x-axis shows, the experiment starts with 10% of the data and then progressively adds more until 90% of the original data set is reached.

Unsurprisingly, the more samples used in training the lower the error rate. At only 10% of the samples, the far left hand side of Fig. 1, we have close to 60% error. However, this still a substantial improvement over a random guess; with 20 classes, that error would be 95%. The error drops quickly, to about 12%, after the next 10% of examples is added. It then decreases more slowly down to the final value of about 3%. Table 2 shows the confusion matrix for 10% of the data set: the rows the true class, the columns the predicted class. Some classes are predicted very well even when the model is trained on this very small set

Table 2. Confusion matrix for 10%

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
A	16	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	13	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
C	0	0	1	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
D	0	0	0	14	0	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	1	0	0	0	0	7	9	0	0	0	0	0	0	0	0
F	0	0	0	0	0	1	0	2	0	0	14	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	1	0	0	0	12	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	2	0	3	0	0	8	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	1
J	0	0	0	0	0	0	0	0	0	3	14	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	2	14	0	0	0	0	0	0	0	0	1
L	0	0	0	0	0	0	0	0	0	0	1	16	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	14	0	1	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	1
O	0	0	0	0	0	1	0	0	0	0	10	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	1	0	0	5
R	0	2	0	0	0	1	0	0	0	0	12	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	10	0
T	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	11

of samples: no misclassifications for P, only 1 for classes A, D and L. On the other hand, the majority of the others do very poorly: C, E, F, G, I, M, N, O, Q and R. What is also noteworthy is that when a mistake is made, class K is preferred choice. Further work would be needed to determine exactly why that is the case. Table 3 is a confusion matrix for 90% of the data. In other words, for the normal size training set used in 10 fold cross validation. Here, we can see that the values on the diagonal, i.e. correct classification, dominate the matrix. This gives an error rate of about 3%, an accuracy of 97% which is comparable to the non-incremental algorithms. The overall point to make here is that adding data incrementally had no effect on the final classification accuracy.

4.2 Adding Classes

The experiments here are much like those of the previous section but instead of adding new folds of data we add new classes. For simplicity, classes were added

Table 3. Confusion matrix for 90%

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
A	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
D	0	0	0	16	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	15	2	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	6	11	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15

in pairs, AB, CD etc. in alphabetic order. Table 4 shows the results for when the first two classes are added. The test set still contains all the classes. However, the model lacks distributions associated with the other classes so will make no probability estimate. Here, an extra column is added to the right hand side of the table to show the classification of an instance as belonging to new class. This is determined by a fixed threshold that is compared against the probability estimates of the other classes. The first thing to notice is that the instance of classes A and B are all correctly classified. As we can see, the majority of instances of the remaining classes are indeed classified as New. However, a few are classified as class A and somewhat more as class B. We will explore why this is true in the next section. Increasing the threshold might exclude some of these but as we will discuss later there seems to be no perfect threshold.

Table 5 is the result of adding all pairs of classes. Looking at the diagonal, we can see that most instances are correctly classified. There are, however, three that are incorrectly classified as new. Comparing these results to those in Table 2, we see that for classes C and D this does not increase the error; those instances were misclassified anyway. So, the overall error rate is only slightly bigger than

Table 4. Confusion matrix for classes A and B

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	Nw
A	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
G	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
H	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
I	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
L	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16
M	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
R	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15

that of the previous section due to the one additional error for class O. For the complete set of classes, if we slightly lowered the threshold we might achieve an exact match in error rates. However, to reduce the misclassification of new classes to old ones, as shown in Table 3, we would have to raise the threshold. So clearly, at present there is no single threshold that would achieve the desired effect. It should also be noted that exactly how things are misclassified will depend of the order at which the classes are added. We would need further experimentation to see if adding, or indeed removing, attributes would improve the situation.

4.3 Difficulties in Identifying a New Class

In the previous sections, we showed that new data can be easily added, be it an instance or a producer. Yet, we also found that new classes cannot always be readily distinguished from existing ones. In this section, we explore why this may be so. Tables 6 and 7 are recordings of the likelihood function for each class.

Table 5. Confusion matrix for all classes

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	Nw
A	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
D	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
E	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	1	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	5	12	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	1
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	14	0

The x-axis is the class, the y-axis is just the count of the number of instances in each class. As we can see, class A has 17, the maximum, whereas class C has 11, the minimum. In the normal classification procedure when a new instance is received, each class distribution is applied to the instance and the likelihoods generated. The likelihoods are the values of each class’s probability distribution function at the point defined by the attributes. Multiplying these by the class prior and normalizing gives the posterior probability of each class. When carrying out cross-validation, every instance will be evaluated.

Tables 6 and 7 give the likelihood values when the instance class matches that of the distribution. In Table 6, we have 17 values for the instances of class A. Looking at the exponents, the range is from +5 to -10, 15 orders of magnitude. This is not as extreme as those for classes C and D, 60 and 45 orders of magnitude, respectively. In Table 7, class O has a range of 61 orders of magnitude. These are the points that were misclassified as new in the previous section. In these examples, there is a single instance much further from the mean of the distribution (in terms of the Mahalanobis distance which accounts for covariance). Although not as extreme, many classes have one exponent which is quite a bit lower than the norm: A -10; H -17; J -30, and -27; K -11; O -19; S -17; T -13. So, at least half the classes have examples which are quite different from the rest. This might be due to changes in the mining or milling process. It may simply be variance, that we would regard as noise, from another yet unknown source.

Table 6. Likelihood classes A-J

No.	A	B	C	D	E	F	G	H	I	J
1	1e-04	3e-02	1e-11	4e-06	7e+00	2e-10	4e+03	1e-07	1e+01	2e-01
2	6e-06	7e-02	7e-04	2e-05	1e+01	2e-09	2e+05	1e-05	2e-02	2e-01
3	6e-01	2e-02	2e-04	8e-08	3e+01	3e-13	1e+06	2e-05	9e+01	3e+02
4	1e+03	2e-03	1e-04	1e-06	3e-02	2e-10	6e+03	6e-08	5e-04	3e-01
5	3e-10	4e+02	6e-05	4e-14	1e+03	1e-08	3e-09	2e-08	2e+00	4e-27
6	3e+02	6e-02	3e-07	1e-50	2e+06	5e-11	5e+04	4e-05	8e+00	6e+01
7	2e+01	9e+00	1e-03	3e-07	2e+03	2e-10	2e+06	3e-05	3e-17	9e+02
8	4e+04	6e-01	1e-08	6e-06	7e+04	4e-08	1e+07	2e-05	8e+03	8e+02
9	2e+04	4e-01	1e-04	2e-03	9e+04	2e-12	1e+07	2e-05	9e+03	4e-02
10	5e+03	7e-01	7e-63	4e-06	6e+00	1e-11	2e+05	1e-17	2e+02	1e+00
11	3e-01	4e-02	2e-07	2e-04	9e+04	3e-10	8e-01	1e-05	1e-02	1e+01
12	1e+04	3e-01	NA	1e-03	7e+02	6e-17	5e+06	3e-05	1e+04	1e-30
13	5e+03	1e-01	NA	2e-06	1e-01	7e-10	3e+06	2e-05	6e+02	2e+04
14	4e-04	9e-03	NA	1e-05	9e+03	2e-11	NA	NA	1e+03	2e+05
15	4e+02	3e-03	NA	1e-06	9e+04	1e-10	NA	NA	5e+02	1e+03
16	6e+03	NA	NA	8e-09	6e+04	4e-08	NA	NA	NA	4e+03
17	4e+03	NA	NA	2e-07	2e+05	7e-10	NA	NA	NA	5e+03

Table 7. Likelihood classes K-T

No.	K	L	M	N	O	P	Q	R	S	T
1	8e-12	3e-01	5e-02	2e+00	4e-10	6e+06	1e-02	2e+02	1e-01	1e-03
2	3e-03	3e-03	9e-03	1e+00	3e-09	4e+03	2e-04	4e+02	2e+01	2e-02
3	3e+01	2e+00	8e-08	3e-03	5e-09	2e+00	4e-03	2e+03	8e-02	6e-05
4	7e+00	3e-03	2e+00	6e+00	5e-09	4e+01	8e-03	4e+02	9e-04	9e-05
5	5e+01	1e+00	4e+01	7e-02	1e-19	1e+02	2e-02	6e+02	5e-09	1e-03
6	2e+01	2e-04	6e-01	8e+00	1e-68	8e+07	5e-03	2e+00	4e-14	1e+00
7	1e+00	8e-02	5e+01	9e-04	4e-09	6e+08	9e-03	2e-01	6e-01	3e-03
8	4e-02	6e-03	9e-01	1e+02	2e-08	3e+07	1e-04	1e-03	2e+00	5e-01
9	1e-01	2e-03	9e-01	7e-07	4e-07	3e+08	5e-02	2e-01	4e-01	3e-07
10	1e-11	2e-05	3e-02	2e+01	3e-06	1e+07	1e-01	8e+02	2e+00	4e-05
11	9e+00	6e-07	3e-03	2e+01	8e-09	2e+07	8e-02	5e-04	7e-01	1e-02
12	2e-01	6e-03	5e+01	7e+01	1e-08	4e+03	7e-03	3e-01	2e-17	1e-12
13	2e+03	2e-02	6e-04	2e+02	NA	4e+02	9e-06	2e-01	3e+00	5e-06
14	1e-01	1e+00	2e+00	3e+02	NA	1e+07	9e-05	5e-02	3e-03	2e-02
15	9e-03	4e-03	2e-02	4e+00	NA	3e+07	2e-06	2e-01	3e-07	3e-03
16	3e+01	2e-02	NA	NA	NA	4e+08	NA	NA	8e+00	NA
17	5e+00	6e-01	NA	NA	NA	5e+06	NA	NA	2e-03	NA

5 Conclusions

This paper discussed an incremental algorithm that can be easily updated when new samples including those from new producers become available. This algorithm achieves accuracy that is indistinguishable from that produced by the discriminatory algorithms used previously. It can also indicate when class is sufficiently different from the old model to be labeled as new class. This is not always correct but we have shown that the mistakes are made on examples that are quite different from others in that class and may be worth flagging, as a concern, anyway. As part of this application, we are still looking at what impact new attributes, say isotopic ratios, might have on the algorithms performance. Another issue worth exploring is the induction of sparsity on the covariance matrix to reduce the degrees of freedom. This would embody the independence of certain attributes as is captured in a Bayesian network.

Acknowledgments. I would like to thank the Canadian Nuclear Safety Commission not only did they finance this work but also supplied considerable domain knowledge and leadership through Ali El-Jaby. I would also like to thank colleagues within NRC, Josette El Haddad and Alain Blouin, for their analytical chemistry expertise.

References

1. Vapnik, V.: *Statistical Learning Theory*. Wiley, New York (1998)
2. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: a comparison of logistic regression and Naive Bayes. In: *Proceedings of 14th International Conference on Neural Information Processing Systems*, pp. 841–848 (2001)
3. Keegan, E., Richter, S., Kelly, I., Wong, H., Gadd, P., Kuehn, H., Alonso-Munoz, A.: The provenance of Australian uranium ore concentrates by elemental and isotopic analysis. *Appl. Geochem.* **23**, 765–777 (2008)
4. Krajčák, J., Varga, Z., Yalcintas, E., Wallenius, M., Mayer, K.: Application of neodymium isotope ratio measurements for the origin assessment of uranium ore concentrates. *Talanta* **129**, 499–504 (2014)
5. Mercadier, J., Cuney, M., Lach, P., Boiron, M.C., Bonhoure, J., Richard, A., Leisen, M., Kister, P.: Origin of uranium deposits revealed by their rare earth element signature. *Terra Nova* **23**(4), 264–269 (2011)
6. Sirven, J.B., Pailloux, A., M'Baye, Y., Coulon, N., Alpettaz, T., Gosse, S.: Towards the determination of the geographical origin of yellow cake samples by laser-induced breakdown spectroscopy and chemometrics. *J. Anal. At. Spectrom.* **24**(4), 451–459 (2009)
7. Balabin, R.M., Lomakina, E.I.: Support vector machine regression an alternative to neural networks for analytical chemistry? Comparison of nonlinear methods on near infrared spectroscopy data. *Analyst* **136**, 1703–1712 (2011)
8. Ristin, M., Guillaumin, M., Gall, J., Gool, L.V.: Incremental learning of random forests for large-scale image classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(3), 490–503 (2016)
9. Schlimmer, J.C., Fisher, D.: A case study of incremental concept induction. In: *Proceedings of 5th National Conference on Artificial Intelligence*, pp. 496–501 (1986)

10. Utgoff, P.E.: Incremental induction of decision trees. *Mach. Learn.* **4**, 161–186 (1989)
11. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: *Proceedings of 13th International Conference on Neural Information Processing Systems*, pp. 388–394 (2000)
12. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. *Mach. Learn.* **29**(2–3), 131–163 (1997)
13. Ahrens, L.: The log-normal distribution of the elements (a fundamental law of geochemistry and its subsidiary). *Geochim. Cosmochim. Acta* **5**, 49–73 (1954)
14. Bürger, S., Boulyga, S.F., Penkin, M.V., Bostick, D., Jovanovic, S., Lindvall, R., Rasmussen, G., Riciputi, L.: Quantifying multiple trace elements in uranium ore concentrates: an interlaboratory comparison. *J. Radioanal. Nucl. Chem.* **301**(3), 711–729 (2014)
15. Yeo, I.K., Johnson, R.A.: A new family of power transformations to improve normality or symmetry. *Biom. Ser. B* **87**(4), 954–959 (2000)
16. James, W., Stein, C.: Estimation with quadratic loss. In: *Proceedings of 4th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 361–379 (1961)

Short Papers



MML-Based Approach for Determining the Number of Topics in EDCM Mixture Models

Nuha Zamzami^{1,2}  and Nizar Bouguila¹

¹ Concordia Institute for Information Systems Engineering,
Concordia University, Montreal, QC, Canada
n_zamz@encs.concordia.ca, nizar.bouguila@concordia.ca

² Faculty of Computing and Information Technology,
King Abdulaziz University, Jeddah, Saudi Arabia

Abstract. This paper proposes an unsupervised algorithm for learning a finite mixture model of the exponential family approximation to the Dirichlet Compound Multinomial (EDCM). An important part of the mixture modeling problem is determining the number of components that best describes the data. In this work, we extend the Minimum Message Length (MML) principle to determine the number of topics (clusters) in case of text modeling using a mixture of EDCMs. Parameters estimation is based on the previously proposed deterministic annealing expectation-maximization approach. The proposed method is validated using several document collections. A comparison with results obtained for other selection criteria is provided.

1 Introduction

The Dirichlet Compound Multinomial (DCM) [1] has shown to be a more effective generative model than the traditional Multinomial usually used for text mining. However, the parameter estimation of DCM can not be done quickly in high dimensional spaces. Taking into account the sparsity and high-dimensional representation of real text, the exponential family approximation of DCM called (EDCM) [2] has been proposed to reduce the computation time. Speeding up the learning process is critical for modeling large document collections.

Finite mixture modeling provides a formal approach for clustering high-dimensional text data [3]. Determining the number of topics (mixture components) that best describes the associated document collection is a crucial aspect in mixture modeling [4]. In this matter, several approaches have been proposed in the literature. The present paper considers MML and the EDCM mixture as an efficient unsupervised learning algorithm for clustering high-dimensional textual data. Implementing MML as a model selection criterion has shown to give good results with mixtures in previous works (e.g. mixture of Gaussians [4]).

The rest of the paper is organized as follows: in Sect. 2, we review the exponential-family approximation to the Dirichlet Compound Multinomial

(EDCM) distribution. We determine the MML expression for the EDCM mixture in Sect. 3 and give the complete algorithm of estimation and selection. Section 4 reports the experimental results, and finally Sect. 5 concludes the paper.

2 The Exponential-Family Approximation to DCM

Given a document vector $\mathbf{X} = (x_1, \dots, x_W)$ with W dimensions equal to the vocabulary size, where x_w represents the frequency of the word w appearing in that document, the Dirichlet Compound Multinomial (DCM) distribution, given a set of parameters $\varphi = (\varphi_1, \dots, \varphi_W)$, is defined as [1]:

$$\mathcal{DCM}(\mathbf{X}|\varphi) = \frac{n!}{\prod_{w=1}^W x_w!} \frac{\Gamma(s)}{\Gamma(s+n)} \prod_{w=1}^W \frac{\Gamma(x_w + \varphi_w)}{\Gamma(\varphi_w)} \quad (1)$$

where $n = \sum_{w=1}^W x_w$ is the document length, and $s = \sum_{w=1}^W \varphi_w$ is the sum of the parameters.

For high dimensional data sets, the parameters estimation for DCM is very slow. Thus, a new distribution that is a close approximation to the DCM has been derived as a member of the exponential family of distributions and was called EDCM [2]. In such approximation, only non-zero word counts x_w are used for computation efficiency, considering that most words do not appear in most documents. In other words, we retain only the sufficient statistic for the purpose of estimating the parameters to reduce the computation time [5]. The EDCM, as an approximation to DCM, is given by [2]:

$$\mathcal{EDCM}(\mathbf{X}|\varphi) = n! \frac{\Gamma(s)}{\Gamma(s+n)} \prod_{w:x_w \geq 1} \frac{\varphi_w}{x_w} \quad (2)$$

3 MML Criterion for EDCM Mixture

Minimum Message Length (MML) is a selection criterion based on evaluating statistical models according to their ability to compress a message containing the data [6]. Let $\mathcal{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_N\}$ be a set of data controlled by a mixture of EDCM distributions with parameters Θ . According to the information theory, the optimal number of clusters M is the candidate value which minimizes the amount of information, measured in *nats* using the natural logarithm, to transmit \mathcal{X} efficiently from a sender to a receiver [7]. The formula for the message length in case of mixture modeling is given by [4]:

$$\text{MessLen} \simeq -\log(h(\Theta)) - \log(P(\mathcal{X}|\Theta)) + \frac{1}{2} \log(|F(\Theta)|) + \frac{N_p}{2} (1 + \log(MN_p)) \quad (3)$$

where $h(\Theta)$ is the prior probability, $P(\mathcal{X}|\Theta)$ is the likelihood for the complete data set, $|F(\Theta)|$ is the determinant of the expected Fisher information matrix. Moreover, the number of free parameters to be estimated N_p for a mixture of

EDCM with M components is $(W+1)M$, while MN_p is the optimal quantization lattice constant for \mathbb{R}^{N_p} [8]. As N_p grows, MN_p tends to the asymptotic value given by $\frac{1}{2\pi e} \simeq 0.05855$ which can be approximated by $\frac{1}{12}$ [9]. Next sections show the detailed calculations for the determinant of Fisher information matrix $|F(\Theta)|$, and the prior pdf $h(\Theta)$ for a mixture of EDCM.

3.1 Fisher Information for a Mixture of EDCM $F(\Theta)$

In mixtures, the complete-data Fisher information matrix has a block-diagonal structure and its determinant is given as the product of the determinant of the Fisher information of φ_j for each component and the determinant of the Fisher information of mixing parameters vector $\boldsymbol{\pi}$ [4, 10].

As the mixing proportions satisfy the requirement $\sum_{j=1}^M \pi_j = 1$, it is possible to consider a series of trials, each has M possible outcomes. In this case, the number of trials of the j th cluster is a multinomial distribution with parameters (π_1, \dots, π_M) . Hence, the determinant of the Fisher information of the mixing parameters vector with N documents is [4]:

$$|F(\boldsymbol{\pi})| = \frac{N}{\prod_{j=1}^M \pi_j} \quad (4)$$

The $|F(\varphi_j)|$ is the determinant of the Fisher information matrix of the expected second partial derivatives of the negative log-likelihood [4]. It can be computed after the data vectors have been assigned to their respective clusters [10]. The data elements in the j th cluster is considered to be $\mathcal{X}_j = \{\mathbf{X}_l, \dots, \mathbf{X}_{l+\eta_j-1}\}$, where $l \leq N$ and η_j the number of the documents generated by j th topic with parameters φ_j . We remark that the Fisher information matrix $F(\varphi_j)$ can be written as: $F(\varphi_j) = D_j + \gamma_j \mathbf{A} \mathbf{A}^T$, where D_j is the diagonal, γ_j is the off-diagonal entries and $\mathbf{A}^T = \mathbf{1}$. Thus, its determinant is given by (Theorem 8.4.3) [11]:

$$|F(\varphi_j)| = \left(1 + \gamma_j \sum_{w=1}^W \frac{a_{jw}^2}{D_{jw}}\right) \prod_{w=1}^W D_{jw} \quad (5)$$

Then, the log of the Fisher information matrix determinant for a mixture of EDCM is given by:

$$\begin{aligned} \log(|F(\Theta)|) \simeq & \log(N) - \sum_{j=1}^M \log(\pi_j) + \sum_{j=1}^M \log \left(\left| 1 + \left(\eta_j (-\Psi'(s_j)) + \sum_{d=l}^{l+\eta_j-1} \Psi'(s_j + n_d) \right) \right. \right. \\ & \left. \left. \sum_{w=1}^W \frac{1}{\sum_{d=l}^{l+\eta_j-1} \mathcal{I}(x_{dw} \geq 1) \frac{1}{\varphi_{jw}^2}} \right| \right) + \sum_{j=1}^M \sum_{w=1}^W \log \left(\sum_{d=l}^{l+\eta_j-1} \mathcal{I}(x_{dw} \geq 1) \frac{1}{\varphi_{jw}^2} \right) \quad (6) \end{aligned}$$

3.2 Prior Distribution $h(\Theta)$

The MML criterion capability is controlled by the choice of prior distribution $h(\Theta)$ for the mixture parameters considering a general assumption that the parameters of the different components as a prior are independent from the mixing probabilities [12]. Knowing that the mixing proportions vector $\boldsymbol{\pi}$ is following a multinomial distribution, a symmetric Dirichlet distribution with parameter vector $\boldsymbol{\alpha}$ is a natural choice as a prior for the mixing probabilities. The choice of $\alpha_1 = \dots = \alpha_M = 1$ gives a uniform prior as follows [4]:

$$h(\boldsymbol{\pi}) = \Gamma(M) = (M - 1)! \quad (7)$$

In the absence of other knowledge about φ_{jw} , we assume that the components prior distributions $h(\boldsymbol{\varphi}_j)$ are independent as well. We choose to use a simple uniform prior, which is known to give good results, in accordance with Ockham's razor [13] as: $h(\varphi_{jw}) = \frac{e^{-6\hat{\varphi}_{jw}}}{\hat{\varphi}_j}$, where $\hat{\varphi}_j$ is the estimated vector of j th component parameters. Thus, the prior distribution for the components is:

$$h(\boldsymbol{\varphi}) = e^{-6MW} \prod_{j=1}^M \frac{\prod_{w=1}^W \hat{\varphi}_{jw}}{\hat{\varphi}_j^W} \quad (8)$$

Then, the log of the complete prior distribution $h(\Theta) = h(\boldsymbol{\pi})h(\boldsymbol{\varphi})$, is given by:

$$\log(h(\Theta)) = \sum_{j=1}^M \log(j) - 6MW - W \sum_{j=1}^M \log(\hat{\varphi}_j) + \sum_{j=1}^M \sum_{w=1}^W \log(\hat{\varphi}_{jw}) \quad (9)$$

The expression of MML for a finite mixture of EDCM distributions, given a candidate value for M , is then obtained by substituting Eqs. (9) and (6) in Eq. (3). For estimating the EDCM mixture parameters, the Deterministic Annealing Expectation Maximization (DAEM) approach has been suggested by Elkan [2]. The complete algorithm of estimation and selection is then as follows:

Algorithm for Estimation and Selection. For each candidate value M :

1. Estimate the parameters of the EDCM mixture using the algorithm in [2].
2. Calculate the associated criterion $MML(M)$ using (3).
3. Select the optimal M^* such that: $M^* = \arg \min_M MML(M)$.

4 Experimental Results

We compare the results from the MML approach with those obtained for the same model parameters using other model selection techniques. The methods that we compare the MML to are Akaike's Information Criterion (AIC) [14], the Minimum Description Length (MDL)[15], and the Mixture MDL (MMDL) [10] where we select the M that yields the minimum value. Another considered criterion is the Partition Coefficient (PC) [16], where we select the M that yields the

Table 1. Clustering results of the data sets (N : number of documents, \bar{n}_d : average document length, W : vocabulary size, M : true number of classes, AC : accuracy \pm standard error, MI : mutual information \pm standard error)

Data set	N	\bar{n}_d	W	M	AC	MI
WebKB4	4199	49.7	7,786	4	84.31 ± 0.02	77.94 ± 0.03
7sectors	4,581	433.2	4,500	7	81.41 ± 0.05	83.21 ± 0.07
NIPS	391	1332.4	6,871	9	90.28 ± 0.03	84.06 ± 0.05
Reuters-21578	9,033	193.3	19,119	10	88.63 ± 0.01	78.20 ± 0.04

maximum value of PC . The performance of each method is evaluated according to whether the selected M agrees with the pre-specified number of clusters in each data set. We use text data sets that have been considered previously (see for example, [2, 17]), namely NIPS¹, Reuters-21578², WebKB4 and 7Sectors³.

Some statistical properties of the used data sets are summarized in Table 1, where we also reported the performance of the model is evaluated on each document collection using the average accuracy, and mutual information [2] based on 100 independent runs with different random initialization for each document collection. The number of clusters M found by the five criteria used is shown in Fig. 1 for the different tested data sets. We can see clearly that the MML

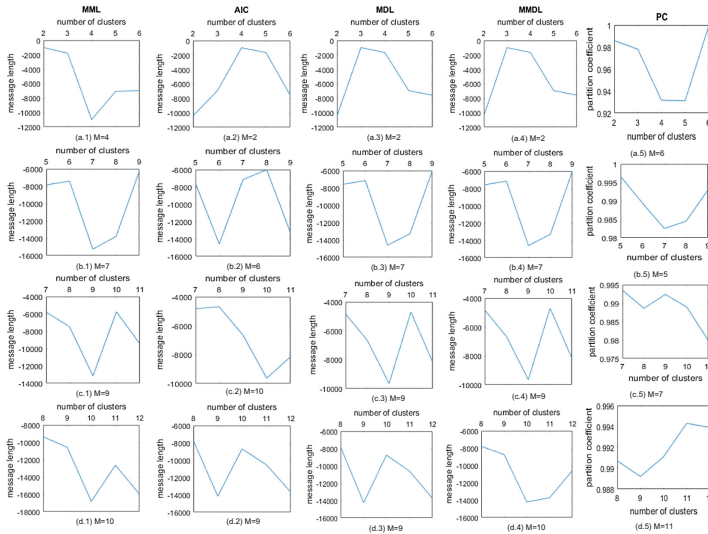


Fig. 1. Number of clusters found by the different criteria for the different text data sets (a. WebKB4, b. 7sectors, c. NIPS, d. Reuters-21578)

¹ <http://www.datalab.uci.edu/author-topic/NIPs.htm>.

² <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>.

³ <http://www.cs.cmu.edu/~webkb/>.

criterion found the correct number of clusters each time. For WebKB4, only MML select $M = 4$ which corresponds to the true number of classes. MML and MMDL found the true number of classes $M = 7$ and $M = 10$ for the 7sectors and Reuters data sets, respectively. The number of classes found using MML, MDL and MMDL with NIPS documents collection is $M = 9$ also agrees with the pre-specified number of classes.

5 Conclusion

We have presented an MML-based criterion to determine the number of topics in a document collection modeled by a mixture of EDCM distributions. The results presented clearly indicate that the MML model selection criterion outperforms other selection criteria. The good results of MML can be explained by the prior information term which are not considered in the other methods. The validation was based on real well-known text data sets. We can conclude that the mixture of EDCM and the MML approach offer strong modeling capabilities for high dimensional text data.

References

1. Bouguila, N., Ziou, D.: Improving content based image retrieval systems using finite multinomial dirichlet mixture. In: Proceedings of the 14th IEEE Signal Processing Society Workshop, pp. 23–32. IEEE (2004)
2. Elkan, C.: Clustering documents with an exponential-family approximation of the dirichlet compound multinomial distribution. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 289–296. ACM (2006)
3. McLachlan, G., Peel, D.: Finite mixture models. Wiley, New York (2004)
4. Baxter, R.A., Oliver, J.J.: Finding overlapping components with MML. *Stat. Comput.* **10**(1), 5–16 (2000)
5. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with bregman divergences. *J. Mach. Learn. Res.* **6**, 1705–1749 (2005)
6. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, New York (2012)
7. Wallace, C.S.: Statistical and Inductive Inference by Minimum Message Length. Springer, New York (2005). <https://doi.org/10.1007/0-387-27656-4>
8. Conway, J.H., Sloane, N.J.A.: Sphere Packings, Lattices and Groups, vol. 290. Springer, New York (1993). <https://doi.org/10.1007/978-1-4757-2249-9>
9. Bouguila, N., Ziou, D.: High-dimensional unsupervised selection and estimation of a finite generalized dirichlet mixture model based on minimum message length. *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(10), 1716–1731 (2007)
10. Figueiredo, M.A.T., Jain, A.K.: Unsupervised learning of finite mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(3), 381–396 (2002)
11. Graybill, F.A.: Matrices with Applications in Statistics. Wadsworth, Belmont, CA (1983)
12. Wallace, C.S.: Classification by minimum-message-length inference. In: Akl, S.G., Fiala, F., Koczkodaj, W.W. (eds.) ICCI 1990. LNCS, vol. 468, pp. 72–81. Springer, Heidelberg (1990). <https://doi.org/10.1007/3-540-53504-7.63>

13. Jefferys, W.H., Berger, J.O.: Ockham's razor and Bayesian analysis. *Am. Sci.* **80**(1), 64–72 (1992)
14. Akaike, H.: A new look at the statistical model identification. *IEEE Trans. Autom. Control* **19**(6), 716–723 (1974)
15. Rissanen, J.: Modeling by shortest data description. *Automatica* **14**(5), 465–471 (1978)
16. Bezdek, J.C.: *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, Boston (1981). <https://doi.org/10.1007/978-1-4757-0450-1>
17. Lin, Y.S., Jiang, J.Y., Lee, S.J.: A similarity measure for text classification and clustering. *IEEE Trans. Knowl. Data Eng.* **26**(7), 1575–1590 (2014)



Constrained Bayesian Optimization for Problems with Piece-wise Smooth Constraints

Aliakbar Gorji Daronkolaei^(✉), Amir Hajian, and Tonya Custis

Centre for Cognitive Computing, ThomsonReuters, Toronto, ON M5J0A8, Canada
{ali.gorji, amir.hajian, tonya.custis}@thomsonreuters.com

Abstract. This paper proposes a new formulation of Gaussian process for constraints with piece-wise smooth conditions. Combining ideas from decision trees and Gaussian processes, it is shown that the new model can effectively identify the non-smooth regions and tackle the non-smoothness in piece-wise smooth constraint functions. A constrained Bayesian optimizer is then constructed to handle optimization problems with both noisy objective and constraint functions.

Keywords: Constrained optimization · Gaussian regression
Bayesian optimization

1 Introduction

Bayesian optimization has recently attracted a lot of attention among researchers [1]. The noisy nature and lack of closed-form representation for the objective function, and expensive cost of generating samples from the objective function have urged people to propose efficient algorithms to deal with noisy and expensive-to-sample cost functions.

While the majority of initial Bayesian optimizers were developed for unconstrained problems, an extension to the constrained version has been made in [2]. The main idea in these works is to extend the traditional acquisition function to a weighted constrained alternative where constraints are each represented by a Gaussian process [3]. A comprehensive review of state-of-art methods for constrained Bayesian optimization can be found in [4]. The main contribution in [5] is to apply a Quasi Monte Carlo (QMC) algorithm to find an unbiased and low-variance estimate of the EI function.

Optimization problems with non-smooth constraints arise in scenarios where designed constraints show non-smooth patterns in certain ranges of parameters space [6]. A hybrid training based method using Artificial Neural Network (ANN) as an intermediate feature-generation step has been proposed in [7] to deal with modeling non-smooth functions using the Gaussian process. The idea of input warping is also used in [8] to tackle non-stationary functions. A Hamiltonian Monte Carlo method is also proposed in [9] to deal with non-stationary functions

where it is assumed that signal, noise-variance and length-scale are all data-dependent and contributions are made to capture the resulting non-stationarity. In [10], a hierarchical tree-based Gaussian process is proposed to handle the non-stationarity and seasonality in time-series. It is also shown in [11] how a tree-based Gaussian process could be applied to a variety of real-world problems such as classification and clustering. The seasonality and trend-change in time-series has been tackled using compositional kernels in [12].

This paper proposes a novel framework to deal with a class of non-smooth functions that meet a piece-wise smoothness condition. The algorithm proposed in this paper does neither require an additional intermediate training phase [7] nor applies any sophisticated input transformation [8]. The proposed framework extends the work in [10] to tackle Bayesian optimization problems with non-smooth constraints and demonstrates the power of the proposed piece-wise Gaussian process.

The rest of the paper is organized as follows. Problem formulation is given in Sect. 2. Section 3 presents piece-wise smooth functions and the new algorithm to handle the aforementioned property. The Bayesian optimization framework is also discussed in this section. Simulated results are given in Sect. 4.

2 Problem Formulation

A standard optimization problem can be explicitly defined in the form of the following set of equations:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_c(\mathbf{x}) \leq 0, \forall c \in \{1, \dots, C\} \end{aligned} \tag{1}$$

where \mathbf{x} denotes an N -dimensional vector of parameters, $f(\cdot)$ corresponds to the objective function, $g_c(\cdot)$ represents the c -th constraint function, and C is the overall number of constraints. In many real-world problems, the functions in (1) may not be available in a closed-form, and/or may be contaminated with a random noise. For the optimization problem given by (1), the n -th parameter \mathbf{x}_n falls in the interval $[x_n^{min}, x_n^{max}]$ with both min/max values being known. Both objective and constraint functions in (1) are also noisy with the following expressions:

$$\begin{aligned} f(\mathbf{x}) &= \bar{f}(\mathbf{x}) + n \\ g_c(\mathbf{x}) &= \bar{g}_c(\mathbf{x}) + m \end{aligned} \tag{2}$$

where \bar{f} and \bar{g} both denote noiseless expressions of the objective and constraint functions, respectively, and n and m are assumed to be additive Gaussian noises with zero-mean and variances σ_f^2 and σ_g^2 , respectively.

3 Piece-wise Smooth Functions

Definition 3.1. A function $f(x)$ is called **piece-wise continuous** for an interval $a \leq x \leq b$ if there is a finite partition, $a = t_0 < t_1 < \dots < t_n = b$, such

that $f(x)$ is continuous for $x \in (t_{i-1}, t_i)$, $i = 1, \dots, n$ and $\lim_{x \rightarrow t_{i-1}^+} f(x)$ and $\lim_{x \rightarrow t_i^-} f(x)$ exist and are finite.

Definition 3.2. Assuming $f(x)$ is divided into n intervals with $f_i(x)$ as the function assigned to the i -th interval, $f(x)$ is called **piece-wise smooth** if all $f_i(x)$ s have continuous first derivatives.

3.1 Piece-wise Gaussian Process

Assumption 3.1. Given a piece-wise smooth function $f(\mathbf{x})$, it is assumed that the domain space $\mathbf{x} \in \mathcal{A}$ is split into N finite disjoint subsets S_n where $\cup_{i=1, \dots, N} S_i = \mathcal{A}$. In this case, the function can be represented as

$$f(\mathbf{x}) = \cup_{i=1, \dots, N} (f_i(\mathbf{x}), \mathbf{x} \in S_i) \quad (3)$$

Assumption 3.2. Given the representation in (3), each piece-wise smooth function $f_i(\mathbf{x})$ is assumed to be represented as a Gaussian process over an uncountable set of pairs $\{(\mathbf{x}_1, f_i(\mathbf{x}_1)), \dots, (\mathbf{x}_M, f_i(\mathbf{x}_M))\}$ where $\mathbf{x}_m \in S_i$. It is then concluded that $f_i(\mathbf{x}) \sim \mathcal{N}(\mu_i(\mathbf{x}), \Sigma_i(\mathbf{x}))$.

Given the above representations for the piece-wise functions, it can be shown that the function $f(\mathbf{x})$ follows the distribution below:

$$f(\mathbf{x}) \sim \frac{1}{N} \sum_i \mathcal{N}(\mu_i(\mathbf{x}), \Sigma_i(\mathbf{x})) \mathcal{I}_{\mathbf{x}}(S_i) \quad (4)$$

where $\mathcal{I}_{\mathbf{x}}(S_i) = \begin{cases} 1 & \mathbf{x} \in S_i \\ 0 & \text{otherwise} \end{cases}$. Given the function representation in (4), an expectation over piece-wise Gaussian process can be defined and calculated as follows:

$$E_f\left(\Delta(f(\mathbf{x}))\right) = \frac{1}{N} \sum_i E_{f_i}\left(\Delta(f_i(\mathbf{x})) | \mathbf{x} \in S_i\right) \quad (5)$$

3.2 Decision Trees for Piece-wise Gaussian Process

In this section, leveraging the ideas from decision trees, a systematic algorithm is proposed to deal with piece-wise smooth functions.

Definition 3.3. The n -th node in the constructed tree is defined as follows:

$$node_n = \{x_{best}, i_{best}, \mathcal{M}_n, S_n\} \quad (6)$$

with x_{best} and i_{best} being the best decision boundary and the feature index (for multi-dimensional cases), respectively, \mathcal{M} is the fitted Gaussian process and S_n corresponds to the subset of parameter values falling in the current node.

To evaluate the efficacy of the Gaussian process, the weighted average uncertainty over the posterior estimates of function values is computed as follows:

$$Q(\text{node}_n) = \text{avg}(\sigma_{1:n_{\text{node}_n}}) \quad (7)$$

with $\sigma_m, m \in \{1, \dots, n_{\text{node}_n}\}$ being the standard deviation of posterior distribution of function estimates provided by \mathcal{M}_n . The quality of a node split is then found by finding an average over any left and right nodes. Given the definitions for the node and the assigned quality, the algorithm for creating a piece-wise Gaussian process can be summarized in Algorithm 1.

Algorithm 1. Decision tree for piece-wise Gaussian process construction

0: **Inputs:** set of parameter values and noisy function values $D = \left\{ (\mathbf{x}_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_L, f(\mathbf{x}_L)) \right\}$

0: **Outputs:** tree-structured set of estimated nodes

0: **Initialization:** fit a Gaussian process to the dataset D and form nodes $\text{nodeSet} = [\text{node}(\text{None}, \text{None}, \mathcal{M}_0, S_0)]$ where \mathcal{M}_0 denotes the fitted model

while nodeSet is not empty **do**

$\text{currentNode} = \text{nodeSet.pop}()$

set $i_{\text{best}} = \text{None}, Q_{\text{best}} = \text{None}, \text{node}_{\text{left}}^* = \text{None}, \text{node}_{\text{right}}^* = \text{None}, n_x = \text{dim}(\mathbf{x})$

for i in $(0, n_x)$ **do**

find x_{best} (best split) and balanced node quality $Q^* = \frac{N_{\text{left}} \times Q(\text{node}_{\text{left}}) + N_{\text{right}} \times Q(\text{node}_{\text{right}})}{N_{\text{left}} + N_{\text{right}}}$ with N_{left} and N_{right} being the number of instances in the left and right nodes, respectively, and $\text{node}_{\text{left}}$ and $\text{node}_{\text{right}}$ being the left and right nodes obtained by the best split

if $Q^* < Q_{\text{best}}$ **then**

$Q_{\text{best}} = Q^*$ and $i_{\text{best}} = i$

$\text{node}_{\text{left}}^* = \text{node}_{\text{left}}$ and $\text{node}_{\text{right}}^* = \text{node}_{\text{right}}$

end if

end for

if $\text{currentNode}.Q / Q_{\text{best}} \geq \mu$ **then**

$\text{nodeSet.add}(\text{node}_{\text{left}}^*)$

$\text{nodeSet.add}(\text{node}_{\text{right}}^*)$

end if

end while=0

4 Simulation Results

To validate the efficacy of the proposed framework, the performance of the algorithm is studied in modeling a noisy rectangular function given by:

$$f(x) = \begin{cases} 1 & -\frac{T}{2} \leq x \leq \frac{T}{2} \\ -1 & \text{otherwise} \end{cases} + \mathcal{N}\left(0, \sigma^2\right) \quad (8)$$

For the training part, 500 samples are generated in the interval $[-10, 10]$ with choice of $T = 4$ and three algorithms as Gaussian process with non-stationary

RBF and Metren kernels, and piece-wise Gaussian process proposed in this paper are considered and trained using the simulated data. As the goal is to capture the true and noiseless rectangular function given by (8), the Root Mean Squared Error (RMSE) of the function estimation is calculated for all the models versus different values of the Signal-to-Noise Ratio (SNR) and, then, an average over 50 Monte-Carlo runs is calculated and depicted in Fig. 1a. To visualize the mean values, Fig. 1b also shows the mean estimates for piece-wise Gaussian process and non-stationary RBF-kernel in a low SNR (15dB) scenario. RMSE results show both non-stationary kernels attain the same performance with the RBF-kernel becomes slightly more accurate in higher SNRs. The piece-wise Gaussian process shows almost 10 times lower RMSE compared to both kernels with the performance gap being preserved even for lower SNR scenarios. The visual results in Fig. 1b also justify the superiority of the piece-wise technique in handling the strong additive noise and detecting the discontinuity region when compared to the RBF-kernel.

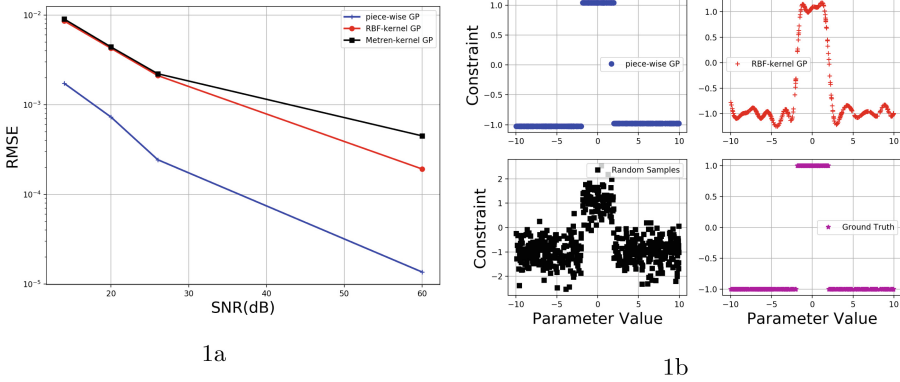


Fig. 1. Performance of Gaussian processes in estimating a rectangular function. (a) The estimation RMSE for different SNR values and (b) Mean estimates for piece-wise and RBF-kernel Gaussian process.

5 Conclusions

Constrained optimization for problems with piece-wise smooth constraints arise in many real-world applications. This paper proposed an easy-to-implement framework for estimating piece-wise smooth constraints using a Gaussian process technique. Using the expected improvement as an acquisition function, the designed constrained optimization framework was applied to a simulated problem with a noisy step-function constraint. The results verified the superiority of the designed technique over the traditional Bayesian optimizer with a regular Gaussian process, especially in dealing with noisy piece-wise constraints.

References

1. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., Freitas, N.D.: Taking the human out of the loop: a review of Bayesian optimization. *Proc. IEEE* **104**, 148–175 (2016)
2. Gardner, J.R., Kusner, M.J., Xu, Z., Weinberger, K.Q., Cunningham, J.P.: Bayesian optimization with inequality constraints. In: 31st International Conference on Machine Learning, Beijing, China, pp. 937–945 (2014)
3. Rasmussen, C.E., Williams, C.: *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge (2006)
4. Hernandez-Lobato, J., Gelbart, M., Adams, R., Hoffman, M., Ghahramani, Z.: A general framework for constrained Bayesian optimization using information-based search. *J. Mach. Learn. Res.* **17**, 1–53 (2016)
5. Letham, B., Karrer, B., Ottoni, G., Bakshy, E.: Constrained Bayesian optimization with noisy experiments (2017). <https://arxiv.org/abs/1706.07094>
6. Zlupko, T.: Portfolio optimization with discontinuous constraint. In: SAS Global Forum 2016, Las-Vegas, USA (2016)
7. Calandra, R., Peters, J., Rasmussen, C., Deisenroth, M.: Manifold Gaussian processes for regression (2014). <http://arxiv.org/abs/1402.5876>
8. Snoek, J., Swersky, K., Zemel, R., Adams, R.: Input warping for Bayesian optimization of non-stationary functions. In: International Conference on Machine Learning, Beijing, China, pp. 1674–1682 (2014)
9. Heinonen, M., Mannerstrom, H., Rousu, J., Kaski, S., Lahdesmaki, H.: Non-stationary Gaussian process regression with Hamiltonian Monte Carlo. In: 19th International Conference on Artificial Intelligence and Statistics, Cadiz, Spain, pp. 732–740 (2016)
10. Gramacy, R., Lee, H.: Bayesian treed Gaussian process models with an application to computer modeling. *J. Am. Stat. Assoc.* **103**, 1119–1130 (2007)
11. Broderick, T., Gramacy, R.: Classification and categorical inputs with treed Gaussian process models. *J. Classif.* **28**, 244–270 (2011)
12. Duvenaud, D., Liyod, J., Grosse, R., Tenenbaum, J., Ghahramani, Z.: Structure discovery in nonparametric regression through compositional kernel search. In: 30th International Conference on Machine Learning, Atlanta, USA, pp. 1166–1174 (2013)



Dimensionality Reduction and Visualization by Doubly Kernelized Unit Ball Embedding

Behrouz Haji Soleimani^{1,2}(✉) and Stan Matwin^{1,2}

¹ Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada
{behrouz.hajisoleimani,st837183}@dal.ca

² Institute for Big Data Analytics, Halifax, NS, Canada

Abstract. In this paper, we present a nonlinear dimensionality reduction algorithm which is aimed to preserve the local structure of data by building and exploiting a neighborhood graph. The cost function is defined to minimize the discrepancy between the similarities of points in the input and output spaces. We propose an effective way to calculate the input and output similarities based on Gaussian and polynomial kernel functions. By maximizing the within-cluster cohesion and between-cluster separation, KUBE remarkably improves the quality of clustering algorithms on the low-dimensional embedding. Our experiments on image recognition datasets show that KUBE can learn the structure of manifolds and it significantly improves the clustering quality.

Keywords: Dimensionality reduction · Manifold learning
Embeddings

1 Introduction

Data visualization and knowledge discovery using machine learning is of particular interest since most of the data being generated everyday is unlabeled and unstructured (e.g. text data). Dimensionality reduction and embedding algorithms are one of the core components in such unsupervised applications. The low-dimensional representation allows us to train learning algorithms more efficiently as well as reducing the chances of overfitting. It also enables us to visualize and make sense of the data more easily.

If $\mathbf{X}_{n \times p}$ is the input data matrix with n datapoints in a p -dimensional space, the aim of dimensionality reduction methods is to find an embedding $\mathbf{Y}_{n \times d}$ such that each input observation is represented by a d -dimensional vector. [4] provides a comprehensive survey of linear dimensionality reduction methods. There are two main categories in unsupervised dimensionality reduction: global and local approaches. Global methods such as Multi-Dimensional Scaling (MDS) [3] and Sammon [9] try to retain the global structure and configuration of points by preserving pairwise distances while local methods such as Locally Linear Embedding (LLE) [8], Laplacian eigenmaps [1], t-distributed Stochastic

Neighbor Embedding (t-SNE) [7] and its accelerated version [12] try to preserve the local neighborhood structure.

Global approaches have several drawbacks and disadvantages. They require notably more resources than local methods both in terms of computational complexity and memory complexity mostly because of calculating, storing, and maintaining all pairwise distances. Moreover, they are less accurate than local methods since they care the same amount for preserving small and large distances. Having to preserve large distances makes the algorithm to lose the ability to model the local neighborhoods' structure. Local approaches also have their own weaknesses. They all depend on the neighborhood graph and connectivities. There are certain situations where local approaches may fail: noise around the manifold, high curvature of manifolds, high intrinsic dimensionality of manifolds, and lastly presence of many manifolds in the data [2].

The proposed embedding algorithm, Kernelized Unit Ball Embedding (KUBE), is a local dimensionality reduction method which tries to preserve the neighborhood structure and address some of the aforementioned drawbacks of local approaches. It is an extension of the Unit Ball Embedding (UBE) method [10] with 2 major improvements: (1) Proposing an adaptive way of calculating input similarities based on densities around points and (2) Proposing a kernelized and effective approach for similarity calculations in output space. Modeling the input and output similarities effectively generally yields to higher quality embeddings. The source code of the algorithm is available at: <https://github.com/behrouzhs/kube>.

2 Kernelized Unit Ball Embedding (KUBE)

KUBE uses a nearest neighbor graph to capture the local structure and define an objective function that preserves the structure. The objective function is defined in a way to minimize the discrepancy between similarities of points in the input space and similarities of points in the transformed feature space. In the following, we first propose an effective way to determine the structure of input data using a sparse representation for input similarities. We then propose an objective function aiming at minimize the squared difference between similarities of points in input and output spaces. Afterwards, we propose a doubly kernelized version of the algorithm which enhances the similarity calculations in the low-dimensional embedding.

2.1 Calculating Input Similarities

The proposed method starts by calculating k_{max} nearest neighbors for each point using Euclidean distance. The set of neighboring points for \mathbf{x}_i is represented as $\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \dots, \mathbf{x}_{k_{max}}^{(i)}\}$ where $\mathbf{x}_j^{(i)}$ denotes the j -th nearest neighbor of \mathbf{x}_i . Distances to the nearest neighbors are then converted to affinity values by centering a Gaussian function on top of each datapoint. We adaptively choose the number of neighbors as well as the variance of the Gaussian for each datapoint since the

density around points can be very different. This will ensure that the algorithm can model the local neighborhood structure even when having various density levels in the dataset. This is a major improvement over UBE which uses a binary adjacency matrix with fixed number of neighbors. The authors in [11] have shown that having different density levels in the data can have a huge impact on the clustering quality.

One of the most important factors in choosing the right number of neighbors is the intrinsic dimensionality of manifolds. If the data lies on a line or plane in a local neighborhood, a few neighbors is enough to reasonably represent the structure in that region. But as the intrinsic dimensionality of manifolds increases, more neighbors are needed to effectively capture the structure. In this work, we propose an elbow based technique on the distances to find a good cut-off neighbor/distance for each datapoint. The elbow distance can be a good indication of moving to another cluster or manifold in the data which we exploit in order to maximize the separation of different manifolds. We choose the variance of the Gaussian for each datapoint in a way that its elbow point falls under the 3σ of the Gaussian, $\sigma_i = \frac{\|\mathbf{x}_i - \mathbf{x}_{elbow}^{(i)}\|}{3}$. More formally, the input similarities are calculated as follows:

$$w_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}) & \mathbf{x}_j \in \{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \dots, \mathbf{x}_{elbow}^{(i)}\} \\ 0 & otherwise \end{cases} \quad (1)$$

2.2 Cost Function

We propose a constrained variant of the objective function used in UBE [10] that is aimed to minimize the sum of squared differences between similarities in the input space and the similarities in the transformed feature space.

$$\mathcal{J}(\mathbf{Y}) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (w_{ij} - \mathbf{y}_i^T \mathbf{y}_j)^2 \quad \text{subject to : } \mathbf{y}_i^T \mathbf{y}_i = 1 \quad \forall i \quad (2)$$

where w_{ij} is calculated using Eq. 1. Here, the objective function forces the points to be on the surface of a hypersphere. Based on the discussion in [10] the unit length constraints is not restrictive. In the constrained form of the objective function, the dot product of output vectors gives the cosine similarity and is used as a kernel in the transformed feature space to calculate the affinities in a nonlinear manner.

The similarities of the points in the input and output spaces are calculated using Gaussian and Cosine kernels, respectively. We normalize the cosine kernel in $[0, +1]$ by using $\frac{1}{2}(\mathbf{y}_i^T \mathbf{y}_j + 1)$ to be in the same range as the Gaussian. This way, if the Gaussian similarity of two points in the input space is high, the method tries to place them as close as possible. And if they are not connected in the input space, $w_{ij} = 0$, the method tries to place them as far as possible.

2.3 Optimization

We have employed stochastic gradient descent algorithm to minimize the objective function. We initialize the datapoints randomly on the surface of a hypersphere and iteratively change the configuration of points to get a better objective value. We have used the general projected gradient descent framework [6] in order to satisfy the constraints. In this setting, the solution is projected onto the feasible region after each iteration. The gradient of the cost function with respect to a datapoint \mathbf{y}_i is:

$$\nabla \mathcal{J}(\mathbf{y}_i) = \sum_{j=1}^n \left[-w_{ij} \mathbf{y}_j + (\mathbf{y}_i^T \mathbf{y}_j) \mathbf{y}_j \right] = \overbrace{-\sum_{j=1}^n w_{ij} \mathbf{y}_j}^{v_1} + \overbrace{\sum_{j=1}^n (\mathbf{y}_i^T \mathbf{y}_j) \mathbf{y}_j}^{v_2} \quad (3)$$

The gradient consists of two components v_1 and v_2 . The former, v_1 , defines the sum of attractive forces being applied to the point while the latter, v_2 , defines the sum of weighted repulsive forces being applied to the point. The computational complexity of the optimization is $O(in^2d)$ where i is the number of iterations in the optimization. Therefore, it is equal to that of t-SNE.

2.4 Doubly Kernelized Objective Function

Looking at Eq. 3, the repulsive force v_2 , consists of a dot product of output vectors which measures their similarity. Here, we can apply a kernel to calculate the similarities of vectors in an implicit high-dimensional feature space. If $\phi(\cdot)$ is the implicit mapping function to the high-dimensional space, the kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ will compute the inner product of those vectors in an efficient way $K(\mathbf{y}_i, \mathbf{y}_j) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle$. The gradient of the objective function is then:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{y}_i} = -\sum_{j=1}^n w_{ij} \mathbf{y}_j + \sum_{j=1}^n K(\mathbf{y}_i, \mathbf{y}_j) \times \mathbf{y}_j \quad (4)$$

Here we apply the kernel just in the repulsive force and not in the objective function directly. Gaussian input similarities have proven to be able to capture the input data relationships and structure very well [7]. Therefore, we are only interested in enhancing the affinity calculations in the output space. Moreover, applying the kernel in the repulsive force will simplify the formulation and consequently, simplify the numerical optimization by preventing the derivative of $\phi(\cdot)$ to appear in the gradient.

Since our output vectors, \mathbf{y}_i , are constrained to have unit length, their dot product (i.e. linear kernel) is equivalent to cosine kernel. In this work, we use a polynomial kernel on top of that to adjust the nonlinearity and further strengthen the effect of closer points in negative force.

$$K(\mathbf{y}_i, \mathbf{y}_j) = (\mathbf{y}_i^T \mathbf{y}_j + 1)^l \quad (5)$$

where l is the degree of polynomial. t-SNE [7] uses t-student kernel to calculate the similarities of points in the transformed feature space. It is considered as a heavy-tailed distribution compared to Gaussian kernel used in [5]. Our case is different in nature as the domain of inputs to the kernel function is bounded. We consider the cosine kernel as heavy-tailed in our spherical representation. However, the tails of the distribution are being adjusted by applying a polynomial kernel on top of that.

3 Experiments and Results

We have executed the proposed method on 9 different image datasets to find a low-dimensional embedding for them. We have used face recognition, handwritten digit recognition and object recognition as benchmark datasets. They have different dimensionality ranging from 256 in USPS handwritten digits to 15386 in CMUfaces dataset.

For evaluating the quality of mapping, we have run clustering algorithms on the output of each embedding method and we associate the quality of clustering algorithms on transformed feature space to the embedding algorithm. In particular, we have used k-means clustering on low-dimensional data to see how good

Table 1. Results of k-means clustering on the embeddings evaluated by Normalized Mutual Information (NMI). The first row for each dataset shows the mean and standard deviation of NMI values in 50 runs and the second row shows the p-value of t-test against the best performing method on that dataset.

Dataset	Method					
	KUBE	t-SNE	Raw data	LLE	PCA	Sammon
Yale	61.9 ± 1.2 1.0000	60.5 ± 1.9 $\approx 10^{-5}$	53.3 ± 3.4 $\approx 10^{-29}$	52.1 ± 1.9 $\approx 10^{-50}$	49.2 ± 1.6 $\approx 10^{-65}$	48.7 ± 1.3 $\approx 10^{-71}$
Olivetti	79.8 ± 0.6 0.0007	80.5 ± 1.1 1.0000	73.8 ± 1.6 $\approx 10^{-41}$	71.8 ± 0.7 $\approx 10^{-65}$	63.3 ± 0.7 $\approx 10^{-94}$	64.5 ± 0.7 $\approx 10^{-91}$
Umist	84.4 ± 1.4 1.0000	74.2 ± 1.4 $\approx 10^{-56}$	64.5 ± 2.0 $\approx 10^{-75}$	58.0 ± 0.5 $\approx 10^{-107}$	58.4 ± 1.3 $\approx 10^{-97}$	59.5 ± 1.0 $\approx 10^{-99}$
CMUfaces	89.4 ± 0.8 1.0000	82.8 ± 2.6 $\approx 10^{-30}$	75.7 ± 3.0 $\approx 10^{-51}$	74.3 ± 1.5 $\approx 10^{-78}$	59.5 ± 1.2 $\approx 10^{-113}$	59.0 ± 1.2 $\approx 10^{-114}$
COIL20	92.6 ± 1.3 1.0000	84.9 ± 2.3 $\approx 10^{-36}$	75.4 ± 1.6 $\approx 10^{-78}$	75.4 ± 1.3 $\approx 10^{-82}$	71.9 ± 1.0 $\approx 10^{-94}$	71.2 ± 1.7 $\approx 10^{-85}$
Opltdigits	86.0 ± 0.8 1.0000	84.5 ± 4.7 0.0307	73.6 ± 2.4 $\approx 10^{-55}$	75.3 ± 1.7 $\approx 10^{-61}$	61.3 ± 1.5 $\approx 10^{-100}$	52.8 ± 1.7 $\approx 10^{-109}$
COIL100	90.1 ± 0.3 1.0000	88.4 ± 0.5 $\approx 10^{-31}$	76.0 ± 0.5 $\approx 10^{-117}$	74.2 ± 0.5 $\approx 10^{-123}$	71.4 ± 0.2 $\approx 10^{-145}$	72.4 ± 0.3 $\approx 10^{-139}$
USPS	84.9 ± 1.1 0.0706	85.8 ± 3.2 1.0000	60.9 ± 0.9 $\approx 10^{-73}$	34.8 ± 0.6 $\approx 10^{-104}$	41.3 ± 1.0 $\approx 10^{-97}$	44.7 ± 0.5 $\approx 10^{-95}$
MNIST	81.2 ± 1.8 0.1657	81.8 ± 2.4 1.0000	50.9 ± 1.7 $\approx 10^{-87}$	60.4 ± 1.0 $\approx 10^{-76}$	38.9 ± 0.3 $\approx 10^{-109}$	36.7 ± 0.7 $\approx 10^{-109}$

the clustering results will be on the transformed feature space. The output of clustering is evaluated using Normalized Mutual Information (NMI).

Table 1 represents the results of k-means clustering on the embeddings found by different algorithms and evaluated by NMI. K-means clustering is run 50 times to get the average and standard deviation of accuracies. The third column, “Raw Data”, shows the results of clustering on the original high-dimensional data. As we can see from the table, the proposed method is significantly better than all other methods in six datasets. In Olivetti face dataset, t-SNE is significantly better than others. In USPS and MNIST handwritten digits, our proposed algorithm and t-SNE are performing equally well. It is also noteworthy to mention that some of the embeddings such as LLE, PCA, and Sammon are usually leading to a lower quality clustering compared to the clustering on the original data. Please refer to [10] to see the results of UBE algorithm on the same datasets.

4 Conclusion

In this paper, we have proposed two major improvements over UBE to effectively model the local neighborhoods in the input data as well as using a polynomial kernel to effectively calculate and weight the affinities in the output space. The KUBE method can be used for visualization, vector embedding applications, or as a pre-step for clustering. It is a local approach aiming to preserve the neighborhood structure with minimal parameters to tune. We have defined the objective function as to minimize the squared difference between the similarities in the input and output spaces. Our experiments on 9 different image clustering tasks show that the proposed method, KUBE, significantly improves the quality of clustering by maximizing the separability of clusters.


References

1. Belkin, M., Niyogi, P.: Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **15**(6), 1373–1396 (2003)
2. Bengio, Y., Monperrus, M.: Non-local manifold tangent learning. *Adv. Neural Inf. Process. Syst.* **17**(1), 129–136 (2005)
3. Cox, T.F., Cox, M.A.: *Multidimensional Scaling*. CRC Press, Boca Raton (2000)
4. Cunningham, J.P., Ghahramani, Z.: Linear dimensionality reduction: survey, insights, and generalizations. *J. Mach. Learn. Res.* **16**(1), 2859–2900 (2015)
5. Hinton, G.E., Roweis, S.T.: Stochastic neighbor embedding. In: *Advances in Neural Information Processing Systems*, pp. 833–840 (2002)
6. Lin, C.J.: Projected gradient methods for nonnegative matrix factorization. *Neural Comput.* **19**(10), 2756–2779 (2007)
7. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**(2579–2605), 85 (2008)
8. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**(5500), 2323–2326 (2000)
9. Sammon, J.W.: A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.* **5**, 401–409 (1969)

10. Soleimani, B.H., Matwin, S.: Nonlinear dimensionality reduction by unit ball embedding (UBE) and its application to image clustering. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 983–988. IEEE (2016)
11. Soleimani, B.H., Matwin, S., De Souza, E.N.: A density-penalized distance measure for clustering. In: Barbosa, D., Milios, E. (eds.) Canadian Conference on Artificial Intelligence, pp. 238–249. Springer, Cham (2015)
12. Van Der Maaten, L.: Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.* **15**(1), 3221–3245 (2014)



Accelerated Gradient and Block-Wise Gradient Methods for Big Data Factorization

M. Reza Peyghami¹ , Kevin Yang², Shengyuan Chen², Ziji Yang¹,
and Masoud Ataei²

¹ School of Information Technology, York University, Toronto, ON, Canada
{rpeygham, zyang}@yorku.ca

² Department of Mathematics and Statistics,
York University, Toronto, ON, Canada
{yangzw, chensy, mataei}@mathstat.yorku.ca

Abstract. A problem often encountered in analysis of the large-scale data pertains to approximation of a given matrix $A \in \mathbf{R}^{m \times n}$ by UV^T , where $U \in \mathbf{R}^{m \times r}$, $V \in \mathbf{R}^{n \times r}$ and $r < \min\{m, n\}$. The aim of this paper is to tackle this problem through proposing an accelerated gradient descent algorithm as well as its stochastic counterpart. These frameworks are suitable candidates to surmount the computational difficulties in computing the SVD form of big matrices. On the other hand, big data are usually presented and stored in some fixed-size blocks, which is an incentive to further propose a block-wise gradient descent algorithm for their low-rank approximation. A stochastic block-wise gradient method will further be suggested to enhance the computational efficiencies when a large number of blocks are presented in the problem. Under some standard assumptions, we investigate the convergence property of the block-wise approach. Computational results for both synthetic data as well as the real-world data are provided in this paper.

Keywords: Gradient descent algorithm · Big data
Singular value decomposition · Stochastic optimization

1 Introduction

In this paper, for a given $A \in \mathbf{R}^{m \times n}$ and $0 < r \leq \text{rank}(A)$, the aim is to approximate A by a same size matrix X having rank r . This could be done through solving the following problem

$$\min_{X \in \mathbf{R}^{m \times n}, \text{rank}(X)=r} f(X) = \frac{1}{2} \|A - X\|_F^2. \quad (1)$$

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting this work.

Even though imposing such a rank constraint, for the general case of the objective function f , may result in an NP-hard problem, (1) could potentially be solved in an efficient manner, as discussed in [1], through the original rank constraint or the convex nuclear norm relaxation of the problem; see [2–4] for more details.

Analysis of the large-scale data is often limited by the need to store them in a matrix format for performing techniques like regression and classification on one hand, and by the inability to store the data matrix completely in memory due to the size of the matrix on the other hand [5]. Due to these facts and computational complexity of SVD, there is a growing interest to factorize the rank r matrix A as UV^T through solving the following non-convex optimization problem:

$$\min_{U \in \mathbf{R}^{m \times r}, V \in \mathbf{R}^{n \times r}} \frac{1}{2} \|A - UV^T\|_F^2 \quad (2)$$

In this setting, U and V are called factor matrices. This factorization contains far fewer variables to store and optimize than the original X , especially when r is small. Besides, it automatically encodes the rank constraint.

For the convex function $f(X)$, replacing $X = UV^T$ leads to the non-convex function $f(U, V) = \frac{1}{2} \|A - UV^T\|_F^2$, with respect to U and V . Moreover, we have

$$\nabla_U f(U, V) = UV^T V - AV, \quad \nabla_V f(U, V) = VU^T U - A^T U.$$

A crucial observation in optimizing f over the factored space is the existence of non-unique possible factorizations. Here, we focus on the set of equally-footed factorizations [1], i.e., the set of all (U, V) in which $A = UV^T$ and $\sigma_i(U) = \sigma_i(V) = \sqrt{\sigma_i(A)}$, for all $i = 1, \dots, r$, where $\sigma_i(\cdot)$ stands for the i -th singular value. Note that (U, V) satisfies equally-footed conditions if and only if $U = \hat{U} \Sigma^{0.5} P$ and $V = \hat{V} \Sigma^{0.5} P$, where $\hat{U} \Sigma \hat{V}^T$ is the SVD of A , and P is an orthogonal matrix. To force the solution of (2) to be balanced, the following regularized objective function is optimized instead:

$$\min_{U \in \mathbf{R}^{m \times r}, V \in \mathbf{R}^{n \times r}} g(U, V) := f(U, V) + \lambda R(U, V), \quad (3)$$

where $R(U, V) := \|U^T U - V^T V\|_F^2$ is the regularized term and $\lambda > 0$.

The *alternating minimization* and *gradient descent* algorithms are two popular approaches to tackle (3). In [6], authors proposed an alternating minimization algorithm for matrix sensing and matrix completion problems. First-order methods are used in several works for solving the problems similar to (3); see [7, 8]. Recently, an efficient first-order method, referred to as BFGD, was proposed in [1] that operates on the U and V factors. Starting from proper estimates U^0 and V^0 , BFGD proceeds towards an updating scheme $U^{t+1} = U^t - \eta \nabla_U g(U^t, V^t)$ and $V^{t+1} = V^t - \eta \nabla_V g(U^t, V^t)$, where $\eta \in (0, 1]$ is the step-size which is chosen properly so that

$$\eta \leq \frac{1}{12 (\|U^0\|_2^2 + \|V^0\|_2^2)}. \quad (4)$$

Nesterov in his seminal work [9] proposed an accelerated gradient method for solving a class of convex optimization problems. This approach was later on modified and customized slightly to make it applicable for solving other types of optimization problems [10]. Some stochastic variants of the (accelerated) gradient descent methods have also been developed in the literature.

In this paper, firstly, we develop the core idea of accelerated gradient descent algorithm as well as its stochastic variant for solving (3). In the large-scale data matrix, it is often difficult, if not impossible, to get the SVD form of a matrix by resorting to the existing packages like MATLAB. To this end, we develop a procedure to obtain SVD forms. Secondly, we establish an approach for block-wise low rank factorization of a given matrix A with predefined blocks. To do so, a gradient descent scheme and its (stochastic) accelerated version are developed. Under suitable assumptions, we establish the convergence property of the block-wise algorithm. Empirical computations of the suggested approaches on both synthetic and real-world data are provided.

In what proceeds, the (stochastic) accelerated version of the BFGD algorithm and an approach for computing SVD form is presented in Sect. 2. In Sect. 3, the (stochastic) block-wise low-rank matrix factorization are discussed. Also, the computational results are provided in Sect. 4.

2 Accelerated Gradient Descent for Factorization

The accelerated gradient descent algorithm in [9] has been generalized to the wide classes of optimization problems in [10]. In this section, we take advantage of the mentioned idea and develop the BFGD method. Details of the accelerated BFGD algorithm are summarized in Algorithm 1, shortly named as AccBFGD. In this algorithm, η is chosen properly so that (4) is satisfied. For the initial matrices U^0 and V^0 , one may use random or specific initialization, as given in [1]. In our implementations, we use the random initialization.

Now, assume that $m \gg n$. This is the case which is often encountered in real world applications. At each iteration of AccBFGD, computing full gradient with respect to U is time consuming. As a remedy for the posed problem, we propose a sampling technique for updating U and V at each iteration. Let us assume that $s < N$ is the sample size. With (U^t, V^t) at hand, instead of updating all the m rows belonging to U , we randomly modify only s rows. Let $k = 1, \dots, s$. For each $k \geq 1$, we first randomly pick row $i_k \in \{1, 2, \dots, N\}$, and update $U_{i_k}^{t+1}$. Note that for a matrix C , $C_{i\cdot}$ stands for the i -th row of C . After updating s rows of U and keeping the rest unchanged, the matrix V^{t+1} is updated accordingly. The details are outlined in Algorithm 2, referred to as SAccBFGD.

Clearly, the advantage of SAccBFGD is that, at each iteration, it only updates s rows of U , thus the computational cost per iteration is only s/m that of the AccBFGD. However, due to the variance introduced by random sampling, one may employ some variance-reduced sampling techniques [11]. Now, our aim is to employ either AccBFGD or SAccBFGD to construct the truncated SVD form of a big data matrix $A_{m \times n}$ with $m \gg n$. Due to the role of the regularized term,

the resulting factorization is equally-footed. That is, matrices U and V have the same singular values which are equal to the square root of singular values of A . Algorithm 3 provides a way to obtain SVD form of A by using matrix factorization.

3 Block-Wise Low-Rank Factorization

A block is defined as a collection of several rows/columns. Let matrix $A_{m \times n}$, with $m \gg n$, consists of N block submatrices $A_i \in \mathbf{R}^{I_i \times n}$, for $i = 1, \dots, N$, with $\sum_{i=1}^N I_i = m$. One can consider the blocks to be predefined due to natural constraints or cost, such as data partitioning in a distributed computer cluster. Such a structure is found in the coefficient matrices of the least squares problems arising in CP-ALS algorithm [12]. Our aim is to form a block UV^T factorization for A , in which $U \in \mathbf{R}^{m \times r}$ consists of N block submatrices $U_i \in \mathbf{R}^{I_i \times r}$, for $i = 1, \dots, N$, and $V \in \mathbf{R}^{n \times r}$. We assume that $r \leq \min\{I_1, \dots, I_N, n\}$.

For $i = 1, \dots, N$, let us define $f_i(U_i, V) = \frac{1}{2} \|A_i - U_i^T V\|_F^2$. Then, we have $f(U, V) = \tilde{f}(U_1, \dots, U_N, V) := \sum_{i=1}^N f_i(U_i, V)$, $R(U, V) = \tilde{R}(U_1, \dots, U_N, V) := \left\| \sum_{i=1}^N U_i^T U_i - V^T V \right\|_F^2$ and

$$g(U, V) = \tilde{g}(U_1, \dots, U_N, V) := \tilde{f}(U_1, \dots, U_N, V) + \lambda \tilde{R}(U_1, \dots, U_N, V).$$

Starting from a proper initial matrices U_i^0 's and V^0 , the block gradient descent procedure is hence modified to update the matrices U_i 's one block at a time, instead of all at once, and V thereafter. More precisely, for the t -th iteration, the k -th block U_k at the $t + 1$ iteration is updated by

$$U_k^{t+1} = U_k^t - \eta \nabla_{U_k} f_k(U_k^t, V^t) - \eta \lambda \nabla_{U_k} \tilde{R}(U_1^t, \dots, U_N^t, V^t). \tag{5}$$

Now, after computing U_j^{t+1} 's, we can update V as follows:

$$V^{t+1} = V^t - \eta \nabla_V \tilde{f}(U_1^t, \dots, U_N^t, V^t) - \eta \lambda \nabla_V \tilde{R}(U_1^t, \dots, U_N^t, V^t). \tag{6}$$

This procedure is repeated successively until some stopping criteria are met. Due to [1], the convergence within the factored space is established under the condition that $(U_1^0, \dots, U_N^0, V^0)$ is in a narrow neighborhood of $(U_1^*, \dots, U_N^*, V^*)$. Here, we use a modified version of random initialization as in [1].

For given pair (U, V) with $U \in \mathbf{R}^{m \times r}$ and $V \in \mathbf{R}^{n \times r}$, the distance to rank r matrix $X_r \in \mathbf{R}^{m \times n}$ is defined by $Dist(U, V; X_r) = \min_{\bar{U}, \bar{V}} \| (U - \bar{U}; V - \bar{V}) \|_F$, where the minimization is taken over all equally-footed factors (\bar{U}, \bar{V}) . Assume that X_r^* is the best rank r approximation of A . Let the singular values of X_r^* be in non-increasing order. Similar to [1], one can establish the following local convergence property of the block-wise gradient descent algorithm.

Theorem 1. Let $(U^0, V^0) = (U_1^0, \dots, U_N^0, V^0)$ be chosen so that $\text{Dist}(U^0, V^0; X_r^*) \leq \frac{\sqrt{2\sigma_r(X_r^*)}}{20}$. Let X^* be the unique minimizer of $f(X) = \frac{1}{2}\|A - X\|_F^2$ under $\text{rank}(X^*) = r$. Furthermore, let $\bar{\eta}$ and λ be chosen so that $\bar{\eta} \leq \frac{1}{4(N+1)(\|U^t\|_2^2 + \|V^t\|_2^2)}$ and $\lambda \leq \frac{N+1}{8}$, respectively. Then, the block-wise gradient descent algorithm converges linearly to X_r^* according to the following recursion:

$$\text{Dist}(U^{t+1}, V^{t+1}; X_r^*) \leq \gamma \text{Dist}(U^t, V^t; X_r^*) + \bar{\eta} \|X^* - X_r^*\|_F^2,$$

where $\gamma \in \left[1 - \frac{\lambda\sigma_r(X_r^*)}{20(N+1)\sigma_1(X_r^*)}, 1\right)$.

This theorem states that if X^* is approximately low-rank, then the iterates converge to a close neighborhood of X_r^* . Likewise AccBFGD and SAccBFGD, we can develop the block-wise AccBFGD and SAccBFGD as well, which are outlined respectively in Algorithms 4 and 5. We refer to these algorithms as BaccBFGD and SBaccBFGD, respectively.

4 Numerical Experiments

In this section, we compare the computational performance of the proposed schemes benchmarked over the BFGD algorithm in [1]. The computational experiments were conducted both on synthetic data as well as a real world data set. We report the relative-error of the factorization (i.e., $\text{rel.error} = \frac{\|A - UV^T\|_F}{\|A\|_F}$) and the CPU time of the algorithm for each data set. The synthetic data were generated by $A = UV^T$ in which $U \in \mathbf{R}^{m \times r}$ and $V \in \mathbf{R}^{n \times r}$ would denote random matrices having i.i.d. Gaussian random entries, resulting in a low rank matrix A .

Algorithms 1–5 were performed on matrices having sizes $m \times n$ ($m \gg n$), target rank r , and the number of the blocks equals to N . The same procedure was repeated employing the traditional BFGD algorithm. It is worth mentioning that the sampling rate of each iteration used in Algorithms 2 and 5 was set to be equal to $0.7N$. The tested algorithms were all terminated once obtaining $\text{rel.error} < 10^{-8}$ or in case the number of the iterations exceeded 20000. Note that in case the later termination condition is met, we may declare divergence of the algorithm.

All implementations were deployed on Matlab R2016b using an Intel core i-7 6700 HQ 2.6 GHz with 16 GB RAM. Tables 1 and 3 depict the computational results corresponding to the synthetic data, where n_{iter} and t_{tot} denote the number of iterations and the total CPU time, respectively. Moreover, in Table 3, N_{Samp} represents number of the sampled blocks. For randomly generated matrices, their SVD were computed using two variants of Algorithm 4. Table 2 compares the performance of Algorithm 3 benchmarked against the `svd` implementation of MATLAB. It is noteworthy to mention that other comparable implementations of SVD could also be found in other programming packages, e.g. see [13], however we opted to choose MATLAB’s implementation due to its wide applicability. Furthermore, the terms SVD-AccBFGD and SVD-BFGD

Table 1. AccBFGD vs BFGD

(m, n, r)	AccBFGD		BFGD	
	n_{iter}	t_{tot}	n_{iter}	t_{tot}
$(10^3, 200, 20)$	41	0.039	48	0.041
$(10^3, 400, 40)$	42	0.129	67	0.17
$(10^4, 200, 20)$	31	0.365	48	0.502
$(10^4, 400, 40)$	34	0.779	40	0.908
$(10^5, 200, 20)$	34	3.386	52	4.904
$(10^5, 400, 40)$	39	10.212	45	10.828
$(10^6, 200, 20)$	49	45.771	56	48.428
$(10^6, 400, 40)$	47	10.335	54	108.11

Table 2. CPU time for SVD form

(m, n, r)	SVD-AccBFGD	SVD-BFGD	SVD-MATLAB
	t_{tot}	t_{tot}	t_{tot}
$(10^3, 200, 20)$	0.039	0.045	0.024
$(10^4, 200, 20)$	0.403	0.592	0.676
$(10^5, 200, 20)$	4.083	4.371	-
$(10^6, 200, 20)$	43.881	45.959	-

Table 3. BAccBFGD vs SBAccBFGD

$(m, n, r, N, N_{\text{Samp}})$	SBAccBFGD		BAccBFGD	
	n_{iter}	t_{tot}	n_{iter}	t_{tot}
$(10^4, 200, 20, 10, 7)$	44	0.764	58	0.981
$(10^5, 200, 20, 10, 7)$	59	10.267	73	12.279
$(10^6, 200, 20, 100, 70)$	64	104.468	74	114.592

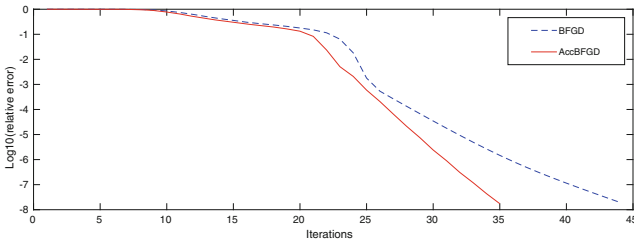


Fig. 1. Decrease of relative error for a matrix of size $10^5 \times 200$ with $r = 20$

refer to Algorithm 3 whose Step 1 includes AccBFGD and BFGD algorithms, respectively.

At a glance to Tables 1, 2 and 3, one may observe that both n_{iter} and t_{tot} are lower in case of AccBFGD being employed. To further visualize the mentioned consistency, Fig. 1 is provided which demonstrates the decrease of the relative error in terms of number of the iterations for a matrix having size $10^5 \times 200$ with $r = 20$.

Besides, the merit of our proposed schemes was confirmed through conducting experiments on a real-world dataset. The dataset used for this purpose, MiniBooNE [14], contains 130065 instances and 50 attributes in which the primary goal is to study the neutrino mass utilizing experimental data available on neutrino oscillations. Algorithm 3 (with AccBFGD) generates the corresponding SVD form of the MiniBooNE in less than two minutes, whereas svd was incapable of solving the problem.

Algorithm 1. Accelerated Bi-Factored Gradient Descent (AccBFGD)

Input
 $r > 0, \theta \in (0, 1), \eta \in (0, 1], \lambda > 0, t_{\max} > 0, A \in \mathbf{R}^{m \times n},$
 $U^0 \in \mathbf{R}^{m \times r}, V^0 \in \mathbf{R}^{n \times r}, T_{U_i}^0 = \mathbf{0}_{m \times r}, T_{V_i}^0 = \mathbf{0}_{n \times r}$

For $t = 0 : t_{\max}$
 $\hat{U}^t = U^t + (1 - \theta)T_{U_i}^t, \quad \hat{V}^t = V^t + (1 - \theta)T_{V_i}^t$
 $U^{t+1} = \hat{U}^t - \eta \nabla_{U_i} g(\hat{U}^t, \hat{V}^t), \quad V^{t+1} = \hat{V}^t - \eta \nabla_{V_i} g(\hat{U}^t, \hat{V}^t)$
 $T_{U_i}^{t+1} = U^{t+1} - U^t, \quad T_{V_i}^{t+1} = V^{t+1} - V^t$

Algorithm 2. Stochastic Accelerated Bi-Factored Gradient Descent (SAccBFGD)

Input
 $r > 0, 0 < \alpha < m, \theta \in (0, 1), \eta \in (0, 1], \lambda > 0, t_{\max} > 0, A \in \mathbf{R}^{m \times n},$
 $U^0 \in \mathbf{R}^{m \times r}, V^0 \in \mathbf{R}^{n \times r}, T_{U_i}^0 = \mathbf{0}_{m \times r}, T_{V_i}^0 = \mathbf{0}_{n \times r}$
 Set $\tilde{U}^0 = U^0, \tilde{V}^0 = V^0$

For $t = 0 : t_{\max}$
For $l = 1 : s$
 Randomly pick $i_l \in \{1, 2, \dots, m\}$
 $\hat{U}_{i_l}^t = U_{i_l}^t + (1 - \theta)T_{U_{i_l}}^t$
 $\hat{U}_{i_l}^{t+1} = \hat{U}_{i_l}^t - \eta \nabla_{U_{i_l}} g(\hat{U}^t, \hat{V}^t)$
 $T_{U_{i_l}}^{t+1} = U_{i_l}^{t+1} - U_{i_l}^t$
 $\hat{V}_{i_l}^t = V_{i_l}^t - U_{i_l}^t$
end
 $\hat{V}^t = V^t + (1 - \theta)T_{V_i}^t$
 $V^{t+1} = \hat{V}^t - \eta \nabla_{V_i} g(\hat{U}^t, \hat{V}^t)$
 $T_{V_i}^{t+1} = V^{t+1} - V^t$

Algorithm 3. SVD form of a matrix by factorization

Input Given $r > 0, A \in \mathbf{R}^{m \times n}$
Step 1 Compute factor matrices $U_{m \times r}$ and $V_{n \times r}$
 by either AccBFGD or SAccBFGD so that $A = UV^T$.
Step 2 Compute the reduced QR decomposition of V as $V = Q_1 R_1$, where
 $Q_1 \in \mathbf{R}^{n \times r}$ and $R_1 \in \mathbf{R}^{r \times r}$.
Step 3 Compute the SVD form of R_1 as $R_1 = U_R \Sigma_R V_R^T$.
Step 4 Set $\tilde{V} = Q_1 U_R, \tilde{\Sigma} = \Sigma_R^2$ and $\tilde{U} = U V_R \Sigma^{-1}$.
Output $A = \tilde{U} \tilde{\Sigma} \tilde{V}^T$ is the truncated SVD of A .

Algorithm 4. Block-wise Accelerated BFGD (BAccBFGD)

Input
 $r > 0, N > 0, \theta \in (0, 1), \eta \in (0, 1], \lambda > 0, t_{\max} > 0, A \in \mathbf{R}^{m \times n},$
 $U_i^0 \in \mathbf{R}^{i \times r}, V^0 \in \mathbf{R}^{n \times r}, T_{U_i}^0 = \mathbf{0}_{i \times r}, T_{V_i}^0 = \mathbf{0}_{n \times r}$

For $t = 0 : t_{\max}$
For $i = 1 : N$
 $\hat{U}_i^t = U_i^t + (1 - \theta)T_{U_i}^t$
end
For $i = 1 : N$
 $U_i^{t+1} = \hat{U}_i^t - \eta \nabla_{U_i} f_i(\hat{U}_i^t, V^t) - \eta \lambda \nabla_{U_i} \tilde{R}(\hat{U}_1^t, \dots, \hat{U}_N^t, V^t)$
 $T_{U_i}^{t+1} = U_i^{t+1} - U_i^t$
end
 $\hat{V}^t = V^t + (1 - \theta)T_{V_i}^t$
 $V^{t+1} = \hat{V}^t - \eta \nabla_{V_i} \tilde{f}(\hat{U}_1^t, \dots, \hat{U}_N^t, \hat{V}^t) - \eta \lambda \nabla_{V_i} \tilde{R}(\hat{U}_1^t, \dots, \hat{U}_N^t, \hat{V}^t)$
 $T_{V_i}^{t+1} = V^{t+1} - V^t$

Algorithm 5. Stochastic block-wise AccBFGD (SBAccBFGD)

Input
 $r > 0, 0 < \alpha < m, N > 0, \theta \in (0, 1), \eta \in (0, 1], \lambda > 0, t_{\max} > 0, A \in \mathbf{R}^{m \times n},$
 $\hat{U}_i^0 = U_i^0 \in \mathbf{R}^{i \times r}, \tilde{V}^0 = V^0 \in \mathbf{R}^{n \times r}, T_{U_i}^0 = \mathbf{0}_{i \times r}, T_{V_i}^0 = \mathbf{0}_{n \times r}$

For $t = 0 : t_{\max}$
For $i = 1 : N$
 $\hat{U}_i^t = U_i^t + (1 - \theta)T_{U_i}^t$
end
For $l = 1 : s$
 Randomly pick $i_l \in \{1, 2, \dots, m\}$
 $\hat{U}_{i_l}^t = \hat{U}_{i_l}^t - \eta \nabla_{U_{i_l}} f_{i_l}(\hat{U}_{i_l}^t, V^t) - \eta \lambda \nabla_{U_{i_l}} \tilde{R}(\hat{U}_1^t, \dots, \hat{U}_N^t, V^t)$
 $T_{U_{i_l}}^{t+1} = U_{i_l}^{t+1} - U_{i_l}^t$
end
 $\hat{V}^t = V^t + (1 - \theta)T_{V_i}^t$
 $V^{t+1} = \hat{V}^t - \eta \nabla_{V_i} \tilde{f}(\hat{U}_1^t, \dots, \hat{U}_N^t, \hat{V}^t)$
 $T_{V_i}^{t+1} = V^{t+1} - V^t$

References

1. Park, D., Kyrillidis, A., Caramanis, C., Sanghavi, S.: Finding low-rank solutions via non-convex matrix factorization, efficiently and provably. arXiv preprint [arXiv:1606.03168](https://arxiv.org/abs/1606.03168) (2016)
2. Cai, J.F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **20**(4), 1956–1982 (2010)
3. Lee, K., Bresler, Y.: Admira: atomic decomposition for minimum rank approximation. *IEEE Trans. Inf. Theory* **56**(9), 4402–4416 (2010)
4. Tanner, J., Wei, K.: Normalized iterative hard thresholding for matrix completion. *SIAM J. Sci. Comput.* **35**(5), S104–S125 (2013)
5. Oswal, U., Jain, S., Xu, K.S., Eriksson, B.: Block cur: Decomposing large distributed matrices. arXiv preprint [arXiv:1703.06065](https://arxiv.org/abs/1703.06065) (2017)
6. Jain, P., Netrapalli, P., Sanghavi, S.: Low-rank matrix completion using alternating minimization. In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 665–674. ACM (2013)
7. Chen, Y., Wainwright, M.J.: Fast low-rank estimation by projected gradient descent: general statistical and algorithmic guarantees. arXiv preprint [arXiv:1509.03025](https://arxiv.org/abs/1509.03025) (2015)
8. Tu, S., Boczar, R., Simchowitz, M., Soltanolkotabi, M., Recht, B.: Low-rank solutions of linear matrix equations via procrustes flow. arXiv preprint [arXiv:1507.03566](https://arxiv.org/abs/1507.03566) (2015)
9. Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN USSR* **269**, 543–547 (1983)
10. Ghadimi, S., Lan, G.: Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Math. Program.* **156**(1–2), 59–99 (2016)
11. Wang, L., Zhang, X., Gu, Q.: A universal variance reduction-based catalyst for nonconvex low-rank matrix recovery. arXiv preprint [arXiv:1701.02301](https://arxiv.org/abs/1701.02301) (2017)
12. Battaglino, C., Ballard, G., Kolda, T.G.: A practical randomized CP tensor decomposition. arXiv preprint [arXiv:1701.06600](https://arxiv.org/abs/1701.06600) (2017)

13. Sanderson, C., Curtin, R.: Armadillo: a template-based C++ library for linear algebra. *J. Open Source Softw.* **1**, 26 (2016)
14. Roe, B.P., Yang, H.J., Zhu, J., Liu, Y., Stancu, I., McGregor, G.: Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nucl. Instrum. Methods Phys. Res., Sect. A* **543**(2–3), 577–584 (2005)



Learning Belief Revision Operators

Aaron Hunter^(✉)

British Columbia Institute of Technology, Burnaby, Canada
aaron_hunter@bcit.ca

Abstract. The beliefs of an agent change in response to new information. Formal belief change operators have been introduced to model this change. Although the properties of belief change operators are well understood, there has been little work on specifying exactly where these operators come from. In this paper, we propose that belief revision operators can be learned from data. In other words, by looking at the behaviour of an agent, we can use basic machine learning algorithms to determine exactly how they revise their beliefs. This is a preliminary paper advocating a particular approach, and demonstrating its feasibility. Fundamentally, we are concerned with the manner in which machine learning techniques can be used to learn formal models of knowledge and belief. We suggest that this kind of advance will be important for future applications of AI.

1 Introduction

Belief revision operators are a formal tool for determining how the beliefs of an agent should change in response to new information. While the formal properties of such operators have been explored extensively, there has been relatively little focus on where these operators actually come from. In this paper, we explore the idea that belief revision operators can be learned from data. Towards this end, we view belief revision as a *classification problem*; we classify inputs to revision in terms of the result of belief revision. We then use standard learning algorithms to find a suitable revision operator that matches the inputs and generalizes well to new data.

This work makes several contributions to existing research. First, it makes a direct contribution to the belief revision literature by proposing how operators can be derived from data. Second, from a high-level perspective, this paper shows how two distinct areas of Artificial Intelligence (AI) can be effectively combined to improve problem solving in natural examples. Note that this is a position paper intended to open a discussion rather than provide a presentation of completed results.

1.1 Motivation

Recent advances in AI, particularly on the practical side, have been driven through Machine Learning (ML) algorithms that discover patterns in data. While this has been very successful, the premise of the present paper is that some

aspects of reasoning actually rely on drawing logical conclusions from structured data in the tradition of Knowledge Representation (KR) [5]. More precisely, we suggest that it can actually be beneficial to start with a formal model of reasoning and then *learn* the details of agent-level knowledge and belief. Combining ML and KR in this manner is an important step that will lead to fundamental advances in AI.

As an illustrative example, we will look at learning in the context of *belief revision*. Belief revision is concerned with the way that a rational agent should change their beliefs when new information about the world is obtained. We are interested in applying techniques of ML to look at how an agent's beliefs have changed in the past in order to predict how their beliefs will change in the future.

We introduce a motivating example, to be used throughout the paper.

Example 1. Consider a robot that is trying to learn how to provide food for a particular human to consume. Each time the robot prepares a meal, the human either eats it or does not eat it. Using standard ML methods, the robot could look at some measurable properties of the food, then observe which foods the human has eaten in the past. For example, suppose that we can observe the following properties:

$$\{\textit{colour}, \textit{temperature}, \textit{smell}, \textit{meat}\}.$$

Each of these has a small set of admissible values, such as *green*, *hot*, *strong* or *beef*.

There are two different ways that we could try to learn about revision in this scenario. First, we could simply have some information about how the human's beliefs have changed in related situations. For example, we could have evidence that the agent always believes food is hot when it is beef. Given information about past revisions, we can then try to learn an underlying revision operator.

We could also try to learn a revision operator to explain how the human's behaviour changes over time. For example, they may have eaten hot-green foods in the past, but now they always reject them. In order to understand this behaviour, we do not just want to predict what will be eaten. We also want to learn how the agent's beliefs about food change; we want to learn how they revise their beliefs as they gain experience.

Why does the robot need to learn how beliefs are revised? There are actually many possible reasons. Suppose, for instance, that the robot is interested in manipulating the human's diet to include more green vegetables. In order to do this effectively, we need to know how much the human cares about food colour. Consider two possibilities:

- Whenever the human eats something they do not like, they revise their beliefs so that they expect not to like all foods that share the colour of the disliked food.
- The human never changes their beliefs about food likeability due to colour.

In case (1), the robot needs to be careful about the introduction of any new green foods; in case (2), other factors are more important.

2 Preliminaries

2.1 Belief Revision

The dominant model of belief revision is the so-called AGM approach [1]. In this approach, we assume an underlying propositional vocabulary \mathbf{V} that consists of variables that take true-false values in order to represent properties of the world. A *belief set* is a set of propositional formulas over \mathbf{V} , using the usual syntax of propositional logic. A *belief revision operator* is a function that takes a belief set K and a new formula ϕ as input, then returns a new belief set $K * \phi$. An AGM revision operator is one that satisfies the AGM postulates, which is a collection of rules constraining the revision process. For example, the so-called *success postulate* indicates that $K * \phi \models \phi$ for every belief set K and formula ϕ . This captures that fact that new information is assumed to be correct.

It should be noted that the AGM model of belief revision is *normative*; it is intended to capture how a rational agent *should* revise their beliefs. The success of this model is partially due to the fact that revision operators can be characterized by orderings over propositional interpretations. In particular, it has been shown that an operator $*$ satisfies the AGM postulates if and only if each K can be associated with a total pre-order \prec_K such that $K * \phi$ is the set of \prec_K -minimal models of ϕ [2].

2.2 Classification Problems

One important kind of problem that has been studied extensively using ML techniques is a *classification problem*. In a classification problem, we are given *instances* that consist of property-value pairs. A standard example would involve a person trying to determine if some new food is going to make them sick. By looking at past meals and occurrences of sickness, we can try to predict what meals will lead to sickness in the future.

There are several different way to learn solutions to classification problems. In this paper, we focus on using the ID3 algorithm to learn a decision tree for classifying new instances [6]. A decision tree in this context is just a tree where the branches are labelled with property-value pairs, and the leaves classify instances. Given a set of instances, the ID3 algorithm tries to find a small decision tree that classifies everything correctly. The idea is to branch on properties that reduce the entropy of the data. Hence, at the first step, we look for the property that divides all cases into subsets with the smallest possible entropy. For example, if someone has been sick every time they have eaten green food, we would branch on this property at the top. At each subsequent step of the algorithm, we branch on the remaining attribute that reduces the entropy as much as possible. Normally, we use 75% of the known instances to build the tree and then we use the remaining instances to test it. In other words, we check if the remaining instances are classified correctly by the tree built from the training data.

We emphasize that there are other, more powerful ML techniques to learn classification problems. For the moment, however, we look only at ID3 because it is a simple algorithm that tends to build small decision trees that are easy to understand.

3 Revision as Classification

3.1 Basic Question

We frame belief revision as a classification problem. Informally, we proceed as follows. Suppose that we know the initial belief set K and we have a particular goal formula G . We also know that some AGM operator $*$ is used for belief revision, but we do not know anything more specific about the operator. Given an input sentence ϕ , we would like to determine if $K * \phi \models G$.

There are certainly cases where this is obvious. For example, if $\phi \models G$, then $K * \phi \models G$ as well. But there are also cases where we need more information. In the present paper, we assume that we do have more information in the form of *instance data* about the behaviour of $*$. In other words, for some finite collection of sentences ψ_1, \dots, ψ_n , we are given that either:

$$K * \psi_i \models G \text{ or } K * \psi_i \not\models G.$$

We can think of this as historical information about how revision has occurred in the past, or in similar scenarios. We use a simplified version of our example to illustrate.

Example 2. Assume the vocabulary $\{\textit{green}, \textit{hot}, \textit{strong}, \textit{beef}, \textit{eat}\}$. Each of these variables can be true or false. The variable *eat* is understood to be true just in case the food is considered edible by the agent in question. The initial belief state K is $\{\textit{beef}\}$; so the agent initially believes the food is beef, and everything else is false. Suppose we have the following information about the revision operator in the form of past instances:

$$\begin{aligned} K * \textit{green} &\not\models \textit{eat} \\ K * \neg\textit{beef} \wedge \textit{hot} &\models \textit{eat} \end{aligned}$$

We can then ask questions. For example, is $K * \neg\textit{beef} \models \textit{eat}$ true? Informally, this asks if the non-beef foods will be eaten.

While this example is very simple, the idea is clear. Given a set of instances that constrain the revision operator $*$, can we predict how it will behave for new examples?

3.2 Basic Data Sets

Assume an initial belief set K over the vocabulary \mathbf{V} . Recall that a *literal* is either a propositional variable in \mathbf{V} or the negation of a propositional variable. As in the previous section, let G denote a special *goal* sentence.

Definition 1. A basic instance (for G) is a pair (L, b) where L is a conjunction of literals over \mathbf{V} and b is either 0 or 1.

Informally, a basic instance $(L, 1)$ is interpreted to be evidence that $K * L \models G$. On the other hand, a basic instance $(L, 0)$ is interpreted to be evidence that $K * L \not\models G$. We refer to a set of basic instances as a *basic data set*.

Example 3. Continuing the example from the previous section, the collection of constraints is captured by the basic data set $D = \{(green, 0), (\neg beef \wedge hot, 1)\}$.

After framing the basic data set as a collection of instances, we can use learning techniques to determine if $K \models G$. For example, for any formula ϕ , we can use the basic instance data to define a Naive Bayes's Classifier to determine if $K * \models \phi$ or not.

We can also use the ID3 algorithm to learn a decision tree T from the instance data. The edges on each complete branch of the tree correspond to a propositional interpretation over $V - \{G\}$. For example, suppose that the tree T has a branch with edge labels L_1, \dots, L_n and a leaf labeled G . This is interpreted to mean $K * L_1, \wedge \dots \wedge L_n \models G$. Hence, the tree gives a collection of constraints on the underlying revision operator. Let $const(D)$ denote the set of constraints defined by a given data set D . The following result is straightforward.

Proposition 1. For some data sets D , there is no AGM revision operator T satisfying every constraint in $const(D)$.

One reason for this negative result is simply because the set of instances need not be consistent. Moreover, even if the constraints are logically consistent, there is a risk that they introduce circular constraints that can not be captured by an AGM revision operator. However, it would be possible to define a precise class of *admissible* basic data sets; these would be sets that correspond to the results of consistent observations.

Claim. The tree obtained by applying ID3 to an admissible basic data set is consistent with a non-empty set of AGM revision operators. Moreover, for every revision operator $*$, the restriction of $*$ to conjunctions of literals is consistent with the tree learned by ID3 for some admissible basic data set.

Of course this claim can only be proved with a precise definition of admissibility. We leave a full specification on admissible data sets for future work.

The preceding comments are concerned only with the problems associated with decision trees to represent constraints on revision. There is also a question about the suitability of the ID3 algorithm for selecting amongst possible decision trees. Do we have any reason to believe that ID3 will select particularly useful decision trees for capturing revision? This is a question that deserves investigation.

Claim. *In practice, decision trees generated by the ID3 algorithm will be more likely to correspond to the actual revision operators being learned.*

This claim is not admissible to proof, but it could be supported by evidence. The ID3 algorithm branches on properties that reduce entropy. In the case of revision operators, this means the algorithm tries to branch on literals L such that $K * L \models G$ and $K * \neg L \not\models G$. It follows that the tree produced by ID3 will define a revision operator where shorter formulas lead to differences in revision. If we have defined a language that captures significant distinctions in the world, then this should indeed be the case.

3.3 Beyond Basic

We now consider revision by formulas that need not be conjunctions of literals.

Definition 2. *An instance (for a goal formula G) is a pair (ϕ, b) where ϕ is a formula over \mathbf{V} and b is either 0 or 1.*

A set of instances is a *data set*. Using the same approach from above, we can define a classification problem to learn a revision operator from a general data set. However, in this case, the tree branches on arbitrary formulas rather than propositional variables. As a result, the notion of admissibility required will be more complex.

We can also move to an even more general setting, where we no longer have a goal formula G . In this case, a *general revision instance* would just be a pair of formulas (ϕ, ψ) interpreted to mean that $K * \phi \models \psi$. These are very general constraints that simply indicate what was believed after previous (or hypothetical) revisions. In this case, we no longer have a classification problem that can be solved with ID3. In order to do useful learning on this kind of problem, one would need large data sets and a more sophisticated learning model.

4 Conclusion

In this short paper, we have advocated for an approach to learning AGM revision operators from data. From a high-level perspective, the important point here is that we are bringing together techniques from ML and KR. Although the practical applications of ML continue to impress, there are certainly applications where a precise model of agent knowledge can be useful. However, defining knowledge representation schemes by hand is tedious and ineffective. If we can learn AGM revision operators from data, then we can have the best of both worlds: a precise model of knowledge that captures some aspect of a complicated practical domain.

Similar ideas have appeared in both the KR and ML literature. On the KR side, some work has been done on learning revision operators from past history [3]. However, this work does not introduce any real ML techniques. On the other hand, mind-changes have been introduced as a way to evaluate learning

mechanisms [4]. In other words, it has been recognized that formalizing the way an agent changes its mind can be important in developing powerful ML systems. As a speculative position paper, we have simply outlined a basic idea and argued that it may be useful. In future work, we will develop the formal tools more precisely.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet functions for contraction and revision. *J. Symbolic Logic* **50**(2), 510–530 (1985)
2. Katsuno, H., Mendelzon, A.O.: Propositional knowledge base revision and minimal change. *Artif. Intell.* **52**(3), 263–294 (1992)
3. Liberatore, P.: Revision by history. *J. Artif. Intell. Res.* **52**, 287–329 (2015)
4. Luo, W., Schulte, O.: Mind change efficient learning. *Inf. Comput.* **204**(6), 989–1011 (2006)
5. McCarthy, J., Lifschitz, V.: *Formalizing Commonsense: Papers by John McCarthy*. Greenwood Publishing Group Inc., Westport (1990)
6. Mitchell, T.: *Machine Learning*. McGraw Hill, New York (1997)



Solving Constraint Satisfaction Problems Using Firefly Algorithms

Mahdi Bidar¹(✉), Malek Mouhoub¹, Samira Sadaoui¹,
and Mohsen Bidar²

¹ University of Regina, Regina S4S 0A2, Canada

{bidar20m, mouhoubm, sadaouis}@uregina.ca

² Iran University of Science and Technology, Tehran, Iran

Abstract. Constraints Satisfaction Problems (CSPs) are known to be hard to solve and require a backtrack search algorithm with exponential time cost. Metaheuristics have recently gained much reputation for solving complex problems and can be employed as an alternative to tackle CSPs even if, in theory, they do not guarantee a complete solution to the problem. This paper proposes a new Discrete Firefly Algorithm (DFA) and investigates its applicability for dealing with CSPs. To assess the performance of the proposed DFA, experiments have been conducted on CSP instances, randomly generated based on the Model RB. The results of the experiments clearly demonstrate the significant performance of the proposed method in dealing with CSPs. For all the instances tested, DFA is successful to find a complete solution that satisfies all constraints in a reasonable amount of time.

Keywords: Constraint Satisfaction Problems (CSPs) · Metaheuristics
Evolutionary algorithms

1 Introduction

A Constraint Satisfaction Problem (CSP) consists of a set of variables, each defined on a discrete and finite set of values (also called variable domain) and a set of constraints that restrict the values that variables can simultaneously take. A solution to a CSP is an assignment of values to variables from their domain so that all constraints are satisfied [1]. Due to the fact that almost all real-world problems come with constraints that restrict the number of acceptable solutions, dealing with CSPs in an appropriate way has become a center of consideration. There are countless examples of CSPs including graph coloring problems [2], N-queen problems [3] as well as real-world applications like scheduling and planning, configuration, timetabling, gear train design, tension/compression spring design, pressure vessel design and welded beam design problems [4]. CSPs are known to be NP-complete problems, meaning that solving them with classical methods like systematic search requires exponential time cost. A CSP with n variables and a domain size d , will need a backtrack search algorithm with $O(d^n)$ time complexity in the worst case [1]. Backtrack search is the most known systematic method for solving CSPs. It suffers however from thrashing which basically means that the algorithm cannot identify and remember the real source of conflict, so repeatedly

fails due to the same reason. This has motivated the research community in the constraint solving area to develop metaheuristics, including evolutionary techniques that run faster but do not guarantee to return a complete solution [5]. Evolutionary techniques are inspired by physical, chemical and biological processes [6] and have shown successful results when solving constrained problems [5]. This paper investigates the applicability of Firefly algorithm, in dealing with CSPs. The main challenge in applying these algorithms to CSPs is that they must be adapted to deal with a discrete problem space. While evolutionary techniques such as Genetic Algorithms (GAs) are originally developed to deal with chromosome-like solutions, which make them easily applicable to CSPs, it is not the case for swarm-intelligence based metaheuristics like Firefly algorithms. Metaheuristic algorithms like genetic algorithm [7], ant colony optimization [8] and particle swarm optimization [9] have been recently successfully employed by different researchers for solving constraint satisfaction problems. In this work, we propose the different steps needed to apply the Firefly algorithm to CSPs. To evaluate the performance, in practice, of the proposed algorithm in dealing with CSPs, we have performed several experimental tests on problem instances randomly generated with the known model RB [10]. This latter has the ability to generate those hard instances near the phase transition. Results of the experiments prove the high performance of the proposed DFA in dealing with CSPs.

2 The Firefly Algorithm

Proposed by Yang in [5], the Firefly algorithm is one of the well-known nature-inspired algorithms inspired by social behavior of special flashing insect namely Firefly in nature. Three governing rules of the algorithm are, (a) the fireflies are unisexual, (b) brighter fireflies attract less brighter ones and (c) brightness of each firefly shows its solution's quality. Features like distance $r_{i,j}$, attractiveness β and the movement behavior of the fireflies are described as follows. $random()$ is a function that produces uniformly distributed vectors of $[0,1)$.

Distance ($r_{i,j}$): Measures the distance between two fireflies i and j at locations x_i and x_j respectively.

$$r_{i,j} = x_i - x_j = \sqrt{\sum_{k=1}^n (x_{i,k} - x_{j,k})^2} \quad (1)$$

Attractiveness: Closely depends on the light intensity I_0 one can see from distance r , absorption coefficient $\gamma \in [0, \infty)$ and I_0 the light intensity at source that can be defined as $I_0 = f(x_i)$ or $I_0 = 1$.

$$\beta = \frac{I_0}{1 + \gamma \cdot r^2} \quad (2)$$

Movement: Movement of Firefly i toward Firefly j (at x_i and x_j) that is brighter and more attractive. Here, $\alpha \in [0, 1)$ is the size of the random step.

$$x_i = x_i + \beta \cdot (x_i - x_j) + \alpha \cdot (\text{random}() - \frac{1}{2}) \quad (3)$$

Movement of the brightest Firefly: The brightest Firefly at location x_{min} moves stochastically.

$$x_{min} = x_{min} + \alpha \cdot (\text{random}() - \frac{1}{2}) \quad (4)$$

The brightness of the fireflies, representing the solution quality, is calculated with a fitness function that is defined according to the problem being solved [5]. The brightest Firefly is the optimal solution.

3 Proposed Discrete Firefly Algorithm for Solving CSPs

The Firefly algorithm has been introduced to deal with continuous optimization problems ($X \in R^n$). To apply it to discrete optimization problems, this algorithm must be well adapted to deal with a discrete space ($X \in S_n$) which is the space of all possible combinations of variables values. To do so, we need to redefine the initialization to produce and distribute the initial population, the distance between fireflies in discrete problem spaces, the attractiveness and the movement of less brighter fireflies toward brighter ones. The steps of discretization of the Firefly algorithm are presented below.

3.1 Solution Representation

In the standard Firefly algorithm, fireflies are uniformly distributed over the problem space (R^n). However, in discrete problem spaces (S_n) the set of randomly generated permutations of variables values is considered as initial population.

3.2 Fitness Function

One of the challenging issues in discretization of metaheuristic algorithms is to define the appropriate fitness function that exactly reflects the quality of the solutions. In the case of CSPs, this quality is measured as the number of violated constraints.

3.3 Updating Solutions

The process of updating solutions is performed through several steps listed below.

- *Distance*

Equation 1 calculates the Cartesian distance between two fireflies which is an appropriate measure to deal with continuous problem spaces. To deal with discrete problem spaces new measures have to be employed and in this regard, the following is

considered: Hamming distance [11] and swap distance. Hamming distance ($Hd_{i,j}$) between two CSP solutions is the number of the variables that have different values.

- *Attractiveness*

β in the Firefly algorithm can be considered as a probability value for replacing a value of a parameter with a new value [12]. From the definition of beta in continuous problems, we know that the bigger β is the bigger steps fireflies make toward brighter ones which implies faster convergence (encourages exploitation). In CSPs, to move from one solution toward a better one, the closer weaker Firefly moves, the more identical values must be shared to reduce the distance (according to the definition of distance and fitness functions for CSPs). To fit this definition, β must determine the probability of changing the value of a variable in weaker Firefly with the value of the corresponding variable in the brighter Firefly. So the bigger the β is the more likely the weaker solution is to replace its variables values by the corresponding variables values of the better solution. For instance $\beta = 0.4$ determines a probability equal to 0.4 for replacing the values of a solution with the values of the better ones.

- *Movement*

The most challenging issue in converting continuous Firefly to discrete Firefly is to adopt its movement to suit the discrete problem spaces. Considering Eq. 3, the movement of fireflies includes two different movements, local movement toward brighter fireflies and random movement.

a. Local movement (also called exploitation)

From Eq. (3) we know that β encourages convergence of fireflies and brings them together. Also, according to our discussion in the previous Section, the movement of a Firefly towards the brighter one in discrete problem spaces means sharing more values with the brighter Firefly by the less brighter one. Therefore, we need to define a different model for movement to bring fireflies together in discrete problem spaces.

1. Find the variables that share the same value. These values will definitely remain unchanged in less brighter Firefly and the rest are subject to replacement (gaps).
2. For filling the gaps considering the β value (probability value between [0,1]), two different cases may happen:
 - With probability β , the value of the corresponding variable in the better solution is chosen to fill a gap in less quality solution.
 - With probability $1 - \beta$, a gap is not filled by the value of the corresponding variable of the better solution. In this case, this gap will be filled by the previous value. This step continues until all gaps are filled.

b. Random Movement (also called Exploration)

For random search, after filling all gaps the mutation operator is applied to the complete assignments. We consider the four random mutations including Random Resetting Mutation (RRM), Swap Mutation (SwM), Scramble Mutation (ScM) and Inversion Mutation (IM).

4 Experimentation

4.1 Problem Instances and Experimentation Environment

We use the model RB to randomly generate CSP instances [10] due to the fact that it has exact phase transition and has therefore the ability to generate hard instances. Each CSP instance is generated as follows using the parameters n , p , α and r where n is the number of variables, p ($0 < p < 1$) is the constraint tightness, and r and α ($0 < r, \alpha < 1$) are two positive constants used by the model RB [10].

1. Select with repetition $rn \ln n$ random constraints. Each random constraint is formed by selecting without repetition 2 of n variables.
2. For each constraint we uniformly select without repetition pd^2 incompatible pairs of values, where $d = n^\alpha$ is the domain size of each variable.
3. All the variables have the same domain corresponding to the first d natural numbers ($0 \dots d - 1$). According to [10], the phase transition pt is calculated as follows: $pt = 1 - e^{-\alpha/r}$. Solvable problems are therefore generated with $p < pt$.

Through the model RB, we generate CSP instances with different tightness ranging from 0.05 to 0.6. All the methods used for the experiments have been implemented using MATLAB R2010a and all experiments have been performed on a PC with Intel Core i7 1.6 GHz processor and 1 GB RAM.

4.2 Test Results

We investigate the performance of the proposed DFA in terms of running time, Number of Violated Constraints (V) and also the Number of Constraint Checks (NCC). In Table 1 T is the tightness of the problems. The parameters of the proposed DFA are tuned to their best by trial and error, and their values are as follow: $\gamma = 0.009$ and *population size* = 30. For this experiment we generate CSP instances with 50 variables and tightness ranging from 0.05 to 0.6.

From the results of these experiments, reported in Table 1, one can see that the overall results are very promising and in all experiments the proposed DFA is successful in finding complete solutions (0 violated constraints) which prove the high performance of this method in dealing with constraint satisfaction problems. Table 1 also shows that SwM, ScM and IM mutations are better operators than RRM in terms of running time and NCC. Although instances with tightness equal 0.6 are the most hard to solve, the proposed method dealt very effectively with that and achieved the best result in very acceptable amount of time. It is worth mentioning that diversification has significant impact on the performance of the metaheuristics and enables the search algorithms to access diverse areas of the search space thanks to the random mutation operators. According to our discussion, the more diverse solutions one algorithm can produce the higher performance that algorithm would have and so the more likely that would be to find the best solution.

Table 1. Comparative results achieved by different DFA on problems with 50 variables

T	RRM			SwM			ScM			IM		
	Time	V	NCC	Time	V	NCC	Time	V	NCC	Time	V	NCC
.05	0.0168	0	780	0.0034	0	234	0.0136	0	1716	0.0081	0	780
.1	0.5016	0	302640	0.1283	0	295776	0.0306	0	3276	0.1595	0	296400
.15	0.8287	0	647712	1.1515	0	699192	1.4387	0	707616	0.8482	0	683748
.2	2.9135	0	1665300	2.5138	0	1572480	3.9337	0	1909440	2.5874	0	1581060
.25	3.5304	0	2230488	4.5499	0	2252952	3.8288	0	2042352	4.4962	0	2287584
.3	4.5006	0	2754024	5.4292	0	3015012	4.4682	0	2616432	4.8989	0	2802072
.35	7.1665	0	3635424	6.9642	0	3664128	6.0769	0	3379584	6.8982	0	3494400
.4	6.8800	0	3894696	6.7295	0	3722004	5.5146	0	4105296	6.6876	0	3825900
.45	9.7628	0	4848480	8.4080	0	4283760	8.4934	0	4684680	8.9119	0	4566120
.5	10.6226	0	5388240	8.6248	0	4784208	8.5342	0	4782492	8.8798	0	4677816
.55	11.4786	0	5625672	12.1430	0	6432816	11.6386	0	6219876	9.9110	0	5585112
.6	14.3486	0	6818448	11.4256	0	6189456	13.3762	0	6591312	9.1275	0	5639088

5 Conclusion

In this paper, we propose a new Firefly algorithm, that we call DFA, for dealing with CSPs. In order to prevent the proposed method from being trapped in local optimum, we consider several random mutation methods. To evaluate the performance of the proposed DFA, we conducted several comparative experiments on randomly generated CSP instances. The results are very promising and clearly demonstrate the high performance of the proposed algorithm.

References

1. Dechter, R.: *Constraint Processing*. Morgan Kaufmann, San Francisco (2003)
2. Eiben, Á.E., Van Der Hauw, J.K., van Hemert, J.I.: Graph coloring with adaptive evolutionary algorithms. *J. Heuristics* **4**(1), 25–46 (1998)
3. Brailsford, S.C., Potts, C.N., Smith, B.M.: Constraint satisfaction problems: algorithms and applications. *Eur. J. Oper. Res.* **119**(3), 557–581 (1999)
4. Cagnina, L.C., Esquivel, S.C., Coello Coello, C.A.: Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica* **32**(3), 319–326 (2008)
5. Yang, X.-S.: *Nature-Inspired Metaheuristic Algorithms*. Luniver press, Bristol (2010)
6. Gandomi, A.H.: Interior search algorithm (ISA): a novel approach for global optimization. *ISA Trans.* **53**(4), 1168–1183 (2014)
7. Abbasian, R., Mouhoub, M.: A new parallel ga-based method for constraint satisfaction problems. *Int. J. Comput. Intell. Appl.* **15**(03), 1650017 (2016)
8. Solnon, C.: Ants can solve constraint satisfaction problems. *IEEE Trans. Evol. Comput.* **6**(4), 347–357 (2002)
9. Clerc, M.: Discrete particle swarm optimization: a fuzzy combinatorial black box, 31 May 2006 (2000). <http://clerc.maurice.free.fr/PSO>
10. Xu, K., Li, W.: Exact phase transitions in random constraint satisfaction problems. *J. Artif. Intell. Res. (JAIR)* **12**, 93–103 (2000)
11. Li, M., X, Chen, Li, X., Ma, B., Vitányi, P.M.B.: The similarity metric. *IEEE Trans. Inf. Theory* **50**(12), 3250–3264 (2004)
12. Durkota, K.: Implementation of a discrete Firefly algorithm for the QAP problem within the sage framework. Bachelor thesis, Czech Technical University (2011)



An AI Planning-Based Approach to the Multi-Agent Plan Recognition Problem

Maayan Shvo¹(✉), Shirin Sohrabi², and Sheila A. McIlraith¹

¹ Department of Computer Science, University of Toronto, Toronto, Canada
{maayanshvo,sheila}@cs.toronto.edu

² IBM Research, Yorktown Heights, NY, USA
ssohrab@us.ibm.com

Abstract. Multi-Agent Plan Recognition (MAPR) is the problem of inferring the goals and plans of multiple agents given a set of observations. While previous MAPR approaches have largely focused on recognizing team structures and behaviors, given perfect and complete observations, in this paper, we address potentially unreliable observations and temporal actions. We propose a multi-step compilation technique that enables the use of AI planning for the computation of the probability distributions of plans and goals, given observations. We present results of an experimental evaluation on a novel set of benchmarks, using several temporal and diverse planners.

1 Introduction

Plan recognition (PR) – the ability to recognize the plans and goals of agents from observations – is useful in a myriad of applications including intelligent user interfaces, conversational agents, intrusion detection, video surveillance, and now increasingly in support of human-machine and machine-machine interactions. Originally conceived in the context of single agent plan recognition (e.g., [1]), recent work has turned to the more complex task of Multi-Agent Plan Recognition (MAPR). In MAPR, the goals and/or plans of multiple agents are hypothesized, based upon observations of the agents, providing a richer paradigm for addressing many of the applications noted above. Early work in this area (e.g., [2]) limited observations to activity-sequences, and focused the recognition task on the identification of dynamic team structures and team behaviors, relative to a predefined plan library. While this formulation is effective for certain classes of problems, it does not capture important nuances that are evident in many real-world MAPR tasks. To this end, we provide in this paper an enriched characterization of MAPR that provides support for a richer representation of the capabilities of agents and the nature of observations. In particular, we support (1) differing skills and capabilities of individual agents; (2) agent skills and actions that are durative or temporal in nature (e.g., washing dishes or other durative

M. Shvo—The work was performed during an internship at IBM.

processes (cf. [3]); (3) observations with respect to the state of the system; such observations range over fluents rather than over actions as actions may not be directly observable but rather inferred via the changes they manifest; (4) observations that are missing or unexplainable (i.e. cannot be accounted for by agents' actions).

Our approach to addressing this problem is to conceive the computational core of MAPR as a planning task, following in the spirit of the single-agent characterization of *plan recognition as planning* proposed by Ramírez and Geffner [4]. This contrasts with much of the previous work on MAPR which requires explicit plan libraries. To realize MAPR as planning, we propose a two-step compilation process that takes a MAPR problem as input. We first compile away the multi-agent aspect of the problem and then we compile away the observations. The resulting planning problem is temporal, has temporal actions and temporal constraints; hence, temporal or makespan-sensitive planners can be applied to generate plans that are then post-processed to yield a solution to the original MAPR problem. We propose three different approaches to generating high-quality MAPR results, evaluating them experimentally. Using these approaches, we are able to compute the probability distributions of plans and goals, given observations. The main contributions of this paper are: (i) a formalization of the MAPR problem with unreliable observations over fluents, and actions that are temporal or durative in nature; (ii) characterization of MAPR as planning via a two-step compilation technique that enables the use of temporal AI planning on the transformed planning problem; (iii) three approaches to computing the probability distributions of goals and plans given the observations; (iv) a set of novel benchmarks that will allow for a standard evaluation of solutions to the MAPR problem; (v) experimental evaluation and comparison of our proposed techniques on this set of benchmarks.

2 Problem Definition

In this section, we review basic definitions, and introduce the multi-agent plan recognition problem with temporal actions and its solution.

Definition 1 (MAPP with Temporal Actions). *A Multi-Agent Planning Problem (MAPP) with temporal actions is a tuple $P^m = (F, \{A_i\}_{i=1}^N, I, G)$, where F is a finite set of fluent symbols, $I \subseteq F$ defines the initial state, and G is the goal of the multi-agent problem, achieved by N agents, each with their own set of temporal action descriptions, A_i .*

Each temporal action $a \in A_i$, as defined in [3], is associated with a duration, $d(a)$, precondition at start, $\text{pre}_s(a)$, precondition over all, $\text{pre}_o(a)$, precondition at end, $\text{pre}_e(a)$, add effects at start, $\text{add}_s(a)$, add effects at end, $\text{add}_e(a)$, delete effects at start, $\text{del}_s(a)$, and delete effects at end, $\text{del}_e(a)$. The semantics of a temporal action is often given using two non-temporal actions “start” and “end”. Additionally, the overall precondition, $\text{pre}_o(a)$ must hold in every state in between. The solution to P^m is a set of action-time pairs, allowing actions to

occur concurrently, where each action is executable, and the goal G holds in the final state. The makespan of the solution is the total time that elapses between the beginning of the first action and the end of the final action.

Next, we define the plan recognition problem with temporal actions, as well as unexplainable and missing observations, adapting the definitions of Sohrabi et al. [5], where quality as measured by cost is used instead of action durations.

Definition 2 (PR Problem with Temporal Actions). *A plan recognition problem with temporal actions is a tuple $P^r = (F, A, I, O, \mathcal{G}, \text{PROB})$, where F, I , are defined as before, A is a set of temporal actions as defined earlier, $O = [o_1, \dots, o_m]$ is the sequence of observations, where $o_k = (f_k, t_k)$, $1 \leq k \leq m$, $f_k \in F$ is the observed fluent, t_k is the time f_k was observed, and $\forall o_i, o_j$, if $i < j$ then $t_i < t_j$. \mathcal{G} is the set of possible goals G , $G \subseteq F$, and PROB is the probability of a goal, $P(G)$, or the goal priors.*

Definition 3 (Unexplainable/Missing Observations). *Given an observation sequence O and a plan π for a particular goal G , an observation $o = (f, t)$ in O is said to be unexplainable (aka noisy), if f is a fluent that does not arise as the consequence of any of the actions a_i from the plan π for G . In contrast, an observation $o' = (f', t')$ is said to be missing from O , if o' is not in the sequence O and f' is added by at least one of the executed actions $a_i \in \pi$.*

In this paper, we consider sequences of observations where each observation $o_i \in O$ is an observable fluent, with a timestamp that indicates when that fluent was observed. To address the unexplainable observations, Sohrabi et al. [5] modifies the definition of satisfaction of an observation sequence by an action sequence introduced in [4] to allow for observations to be left unexplained. Given an execution trace and an action sequence, an observation sequence is said to be satisfied by an action sequence and its execution trace if there is a non-decreasing function that maps the observation indices into the state indices as either explained or discarded. Hence, observations are all considered, while some can be left unexplained. Next, we define the problem we address in this paper.

Definition 4 (MAPR Problem with Temporal Actions). *The Multi-Agent Plan Recognition (MAPR) problem with temporal actions is described as a tuple $P = (F, \{A_i\}_{i=1}^N, I, O, Z, \mathcal{G}, \text{PROB})$, where F is a finite set of fluents, A_i is a set of temporal actions for agent i , $1 \leq i \leq N$, $I \subseteq F$ defines the initial state, $O = [o_1, \dots, o_m]$ is the sequence of observations, where $o_k = (f_k, t_k)$, $1 \leq k \leq m$, $f_k \in F$ is the observed fluent, t_k is the time f_k was observed, Z is a set of agents (each element in Z corresponds to an index between 1 and N), $1 \leq |Z| \leq N$, \mathcal{G} is the set of possible goals, $G \in \mathcal{G}$, pertaining to the set of agents Z , $G \subseteq F$, PROB is the prior probability of a goal, $P(G)$.*

Given a MAPR problem with temporal actions, P , a solution to P is in the form of two probability distributions. The first is the probability of plans given the observations, $P(\pi|O)$, where each π is a plan that achieves a goal $G \in \mathcal{G}$, satisfies the observation sequence, O , and involves at least one action performed

by an agent in Z . The second distribution is the probability of goals given the observations, $P(G|O)$, where each G assigned a non-zero probability is a goal achieved by a plan in the first distribution.

3 Transformation

In this section, we briefly describe our multi-step compilation technique compilation technique that allows the use of temporal planning on the MAPR problem. The pipeline consists of transforming a MAPR problem as defined in Definition 4 into a single agent plan recognition problem with temporal actions, and a transformation step that compiles away the observations, allowing the use of temporal planning to compute the posterior probabilities of goals.

To transform the original MAPR problem with temporal actions to a single agent PR problem with temporal actions, we compile away the multi-agent information by using special predicates that keep track of an agent’s access to fluents and objects; every object o and agent i in the domain are assigned a corresponding fluent. For an agent i to be allowed to execute an action on object o , a precondition must be met, in which the corresponding fluent holds.

To incorporate a temporal aspect into the compilation process, our work replaces the notion of cost with that of duration, and compiles the observations into temporal actions that are part of the transformed temporal planning domain. The transformation compiles away observations, using special predicates for each fluent in the observation sequence O , while ensuring that their order is preserved. We also add extra actions, “explain” and “discard” for each observation with a penalty to the “discard” action to encourage the planner to explain as many observations as possible. We also update the duration of the original actions, by adding a constant duration to each action; this is the penalty for the possible missing observations, which encourages the planner to use as few unobserved actions as possible. The transformation ensures that observation o_1 with timestamp t_1 will be considered (explained or discarded) before observation o_2 with timestamp t_2 , where $t_1 < t_2$. Finally, in order to allow the use of diverse planning, the goal of the transformed planning problem is set such that all observations are considered and at least one of the goals $G \in \mathcal{G}$ is achieved.

Theorem 1. *Given a MAPR problem with temporal actions, $P = (F, \{A_i\}_{i=1}^N, I, O, Z, \mathcal{G}, \text{PROB})$, as defined in Definition 4, where $|Z| = N$, and the corresponding transformed temporal planning problem $P' = (F', A', I', G')$ as described above, for all $G \in \mathcal{G}$, if π is a plan for the planning problem $(F, \{A_i\}_{i=1}^N, I, G)$, then there exists a plan π' for the corresponding planning problem, P' , such that the plan π can be constructed straightforwardly from π' .*

Proof is based on the fact that the extra actions only preserve the ordering amongst the observations and do not change the state of the world. The makespans of plans in the transformed planning problem map to $V(\pi)$, which is used to approximate $P(O|\pi)P(\pi|G)$; thus, the probability distributions, $P(G|O)$ and $P(\pi|O)$, can be computed using these makespans.

4 Computation

In this section, we briefly describe our approaches to computing a solution to the MAPR problem, as described in Definition 4, namely the probability distributions of plans and goals, given observations. For further details, we refer the reader to [6].

Delta - This approach is based on finding, for each of the goals, the delta between the costs of two plans, one that explains the observations and one that does not; this method is a modification of the goal recognition approach proposed in [4]. The delta is found by running the planner twice for each goal.

Diverse - This approach computes the probability distribution of goals by finding a set of diverse plans, that serves as a representative approximation of the distribution of plans that satisfy the observations and achieve one of the possible goals ($P(\pi|O)$); it is a modification of the proposed approach in [5]. The set of diverse plans, which serves as an approximation to the probability distribution of plans and goals, given O , is found by running a diverse temporal planner on the transformed planning problem.

Hybrid - This approach is a combination of the two previous approaches, in that it computes a set of plans for each goal. In order to take advantage of both previous approaches, we propose an approach in which we use a temporal planner to compute a smaller set of plans for each of the goals. After merging the sets of plans, we are able to compute the probability distribution of goals, just as we did in the Diverse approach.

5 Experimental Evaluation

In this section, we briefly present the results of our experimental evaluation. For further details, we refer the reader to [6]. To evaluate our MAPR approach, we used a temporal planner, LPG-TD [7], for the delta approach and the hybrid approach, and a diverse planner, LPG-d [8], for the diverse approach. We have created, for evaluation purposes, a set of novel benchmarks, based on International Planning Competition (IPC) domains, namely Rovers, Depots, Satellites, and ZenoTravel. We modified the domains to create benchmark problems for the MAPR problem with temporal actions. In addition, we have introduced missing observations, by creating several variations of each problem that did not include the full observation sequence. Lastly, we have introduced noise by adding extra random observations, with two different levels of noise.

To evaluate the coverage and accuracy of our approaches on the goal recognition task, we compute the average percentage of instances in which the ground truth goal was deemed most and less likely, i.e., whether or not the ground truth goal was assigned the highest posterior probability given the observations. Our results (shown in [6]) show that the Delta approach, on average, does best (i.e., deems the ground truth goal most likely) across all domains when observations are reliable and no noise is introduced. Further, on average, over all our problems, when unreliable observations were not introduced, **Delta** deemed the

ground truth goal most likely in 77% of cases, **Diverse** - 75% of cases, and **Hybrid** - 49% of cases.

The total number of problems solved by each approach is as follows: **Delta** - 458/912; **Diverse** - 611/912; **Hybrid** - 790/912. Note that approach 1 manages to solve the least amount of problems, on average, compared to the other approaches. We plan to conduct further experimentation, testing both the robustness and scalability of our approach.

6 Discussion

The merit of this paper is that it provides a way to solve an important class of Multi-Agent Plan Recognition problems that could not previously be addressed. It does so by leveraging and augmenting a combination of ideas from single agent plan recognition and multi-agent planning. Solving this class of problems, with unreliable observations and temporal actions, is paramount to the applicability of a MAPR approach to many real-world instantiations of the problem. Furthermore, our formalization allows for much needed expressivity, while also providing the foundation for incorporation of various important and interesting aspects of the problem, including, for example, agents with varying and limited knowledge of the state of the world and with differing physical and even cognitive capabilities. Finally, our work enables the application of a MAPR approach to previously unaddressed problems, by modeling them in planning domains. By enabling the use of existing temporal planners, one can choose the planner that works best for their domain and compute a solution to their MAPR problem.

Acknowledgments. The authors gratefully acknowledge funding from IBM and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Schmidt, C.F., Sridharan, N., Goodson, J.L.: The plan recognition problem: an intersection of psychology and artificial intelligence. *AIJ* **11**(1–2), 45–83 (1978)
2. Banerjee, B., Lyle, J., Kraemer, L.: Multi-agent plan recognition: formalization and algorithms. In: *AAAI* (2010)
3. Fox, M., Long, D.: PDDL2.1: an extension to PDDL for expressing temporal planning domains. *JAIR* **20**, 61–124 (2003)
4. Ramírez, M., Geffner, H.: Probabilistic plan recognition using off-the-shelf classical planners. In: *AAAI* (2010)
5. Sohrabi, S., Riabov, A., Udrea, O.: Plan recognition as planning revisited. In: *IJCAI* (2016)
6. Shvo, M., Sohrabi, S., McIlraith, S.A.: An AI planning-based approach to the multi-agent plan recognition problem (extended version). Technical report CSRG-636, Department of Computer Science, University of Toronto, February 2018
7. Gerevini, A., Saetti, A., Serina, I.: LPG-TD: a fully automated planner for PDDL 2.2 domains. In: *ICAPS* (2004)
8. Nguyen, T.A., Do, M.B., Gerevini, A., Serina, I., Srivastava, B., Kambhampati, S.: Generating diverse plans to handle unknown and partially known user preferences. *AIJ* **190**, 1–31 (2012)



Predicting Transportation Modes of GPS Trajectories Using Feature Engineering and Noise Removal

Mohammad Etemad¹, Amílcar Soares Júnior¹, and Stan Matwin^{1,2}

¹ Institute for Big Data Analytics, Dalhousie University, Halifax, Canada
etemad@dal.ca

² Institute for Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract. Understanding transportation mode from GPS (Global Positioning System) traces is an essential topic in the data mobility domain. In this paper, a framework is proposed to predict transportation modes. This framework follows a sequence of five steps: (i) data preparation, where GPS points are grouped in trajectory samples; (ii) point features generation; (iii) trajectory features extraction; (iv) noise removal; (v) normalization. We show that the extraction of the new point features: bearing rate, the rate of rate of change of the bearing rate and the global and local trajectory features, like medians and percentiles enables many classifiers to achieve high accuracy (96.5%) and f1 (96.3%) scores. We also show that the noise removal task affects the performance of all the models tested. Finally, the empirical tests where we compare this work against state-of-art transportation mode prediction strategies show that our framework is competitive and outperforms most of them.

Keywords: Feature engineering · Noise removal
Trajectory classification

1 Introduction

Research on trajectory analysis is a mature area since positioning devices are now used to track people, vehicles, vessels, and animals. In the case of trajectory data, the object's movement is represented as a discrete collection of spatiotemporal points.

A domain where trajectories are frequently analyzed is the prediction of transportation modes from users, which is essential for cities and people to reduce travel time and traffic congestion. Transportation mode estimation involves two steps [11]: (i) extraction of segments of the same transportation modes; and (ii) classification of transportation modes for each segment. For the first step, several segmentation algorithms have been proposed in the past years and include temporal-based [8], cost function-based [5] and semantic-based methods [7]. For the second step, which is the focus of this work, the classification (or prediction) of the transportation modes is performed by creating domain expert features for

supervised classification (e.g., the distance between consecutive points, velocities, acceleration, and bearing).

We classify the research in transportation modes prediction regarding the type of features in two branches: (i) domain expert features; and (ii) learned features. From raw GPS data points (e.g., latitude, longitude and time) it is possible to calculate many attributes regarding the moving object’s movement. Examples include distance traveled between points, estimated speed, bearing, acceleration, etc. For segments of trajectories, it is possible to extract mean, median, minimum, maximum, standard deviations, etc., of point-wise features. These are examples of domain expert features employed to predict transportation modes. Examples of works that apply domain expert features include [6, 11].

In this work, we also explore the effects of noise removal in the prediction of transportation modes. Dealing with noise in trajectories is essential because GPS recorder devices are not accurate in the moving object’s positioning due to many reasons like satellite geometry, signal blockage, atmospheric conditions, and receiver design features/quality. By removing GPS noise, it is expected that the derived features from the trajectories are more likely to represent the standard pattern of a transportation mode.

Noise-perturbed GPS data influences the quality of the domain expert features, e.g. distance traveled, speed or acceleration are susceptible to errors. It is important to point out that these errors may impact the distributions of values, where statistics like the mean, in trajectory segments of transportation modes. This uncertainty of data can lead a classifier to create models that are not able to accurately predict a transportation mode from a trajectory. Thus, the works in transportation mode prediction are classified regarding the (i) presence or (ii) absence of noise removal strategies. An example of work in the transportation mode prediction that does not deal with noise removal is [11]. In others, like [1, 2, 4, 9, 10], noise is removed. This paper applies domain expert features and noise removal to predict transportation are as follows: (i) we introduce new point and trajectory features; (ii) we propose a framework composed of 5 steps for transportation mode prediction; (iii) we compare the proposed approach with state-of-art strategies and show that our results are competitive.

2 A Framework for Transportation Mode Prediction

In this section, we present the sequence of steps used in this work to predict transportation modes (Fig. 1). This framework has five steps and is described in detail below.

In this work, we define a trajectory as a sequence of GPS points that belongs to the same transportation mode. In the first (step 1), we group the raw GPS points by *userid*, *day* and transportation mode to create trajectory samples. We discard trajectory samples with less than 10 GPS points because these examples may affect our model since trajectories with low quality may be created.

In this work, we calculate some point features (step 2) that were used previously in literature [11]: distance, speed, acceleration, jerk [1], and bearing.

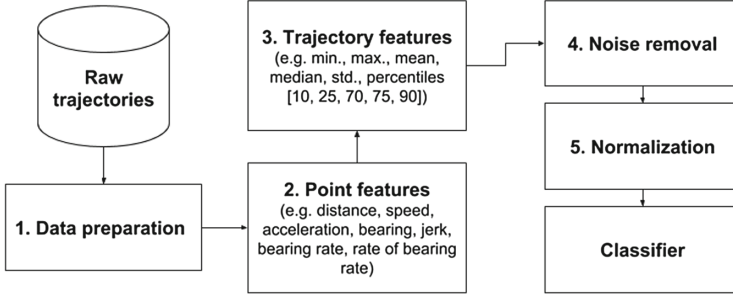


Fig. 1. The steps of the proposed framework to predict transportation modes

Two new features are introduced in this work, named bearing rate, and the rate of bearing rate. They are detailed as follows. The bearing rate was computed using Eq. 1, where B_i and B_{i+1} are the bearing values in points i and $i + 1$, and Δt is the time difference.

$$B_{rate(i+1)} = (B_{i+1} - B_i) / \Delta t \quad (1)$$

Some moving objects tend to change the bearing more often because they commute in a straightforward route. This behavior can be captured by using the rate of the bearing rate. This feature is calculated using Eq. 2.

$$Br_{rate(i+1)} = (B_{rate(i+1)} - B_{rate(i)}) / \Delta t \quad (2)$$

After calculating all the point features for each trajectory, we extract some statistical attributes referred to as trajectory features (step 3). Trajectory features are divided into two different types: (i) global trajectory features, which summarize information regarding the whole trajectory in a single value; and (ii) local trajectory features, which describe a local part of the trajectory. In this work, we extracted global features like the Minimum, Maximum, Mean, Median, and Standard Deviation values of each trajectory point feature to feed our classifier. The local trajectory features extracted in this work was the percentiles of every point feature. Five different percentiles were extracted (10, 25, 50, 75, and 90) and were used in the models tested in this work. In summary, we compute 70 trajectory features (10 statistical measures including five global and five local features calculated for 7 point features) for each transportation mode example.

In step 4, the framework deals with noise in the data. In this work, we used a simple method called median filter to create a mask. The method is described in Algorithm 1 ($threshold = 3$) and it removes the noise based on $speed_{mean}$ (i.e. the average speed of a trajectory) attribute since a human can classify the transportation mode mostly by knowing the mean speed of a trajectory.

Finally, we normalized the features (step 5) using the Min-Max normalization method, since this method preserves the relationship between the values to transform features to the same range and improve the quality of classification process [3].

Data: Speed mean of trajectories

Result: mask vector to remove the noisy trajectories

$difference \leftarrow |speedmean_{Trajectory} - median(speedmean)|;$

$median_difference \leftarrow median(difference);$

if $median_difference == 0$ **then**

 | $indicator \leftarrow 0;$

else

 | $indicator \leftarrow difference/median_difference;$

end

return $indicator > threshold;$

Algorithm 1. Mask the noisy samples to remove from dataset using median

3 Experiments

In this section, we detail the experiments performed in this work to validate our framework. The data used in this work is the GeoLife GPS dataset, that was collected by Microsoft Research Asia from April 2007 to October 2011 [11]. The dataset has a 5,504,363 number of records labeled by eleven transportation modes: taxi (4.41%); car (9.40%); train (10.19%); subway (5.68%); walk (29.35%); airplane (0.16%); boat (0.06%); bike (17.34%); run (0.03%); motorcycle (0.006%); and bus (23.33%).

In the literature, we observed different sub-selections of these classes for evaluating transportation mode prediction strategies; therefore, we decided to select different target subsets for comparing our result with other papers.

To evaluate the performance of classifiers in this work we used the Accuracy and the F1 measure. In all our experiments, we used a 10-fold cross-validation strategy and computed a paired t-test to verify if the difference in the means were statistically different. We executed our framework with different classifiers such as Decision Tree (DT) (with *maxdepth* equals five), Random Forest (RF) (with 50 trees estimators), Neural Network (NN), Naive Bayes (NB), and Quadratic Discriminant Analysis (QDA). In all cases, the random forest surpasses all the other classifiers in both accuracy and f1.

Subsequently, we compared the RF using all the steps of our framework against the results of five papers. It is important to point out that all these papers reported their accuracy values on the Geolife dataset. Table 1 shows a side-by-side comparison between some related works and the results of our framework. Our work does not surpass Jiang’s et al. accuracy [4] but outperforms all the others. It is important to highlight that the complexity and high training time of the RNN model used in his work may not be worth the 1.42% difference in accuracy.

Finally, we evaluated the effects of noise removal performed by our framework. We established as a baseline the performance of our framework using the data to train classifiers with noise and without noise (clean). Table 2 shows the mean of the f1 values obtained by 10-fold cross-validation for the different group of classes. We can observe in Table 2 that for all classifiers and different

Table 1. Comparison of accuracy and f1 measure of proposed model against related works

Related work	Proposed model		
Reference: classes used in the experiments	acc	acc	f1
Dabiri et al. [1]: walk, bike, bus, driving, and train	84.8%	93.35%	93.22%
Jiang et al. [4]: bike, car, walk, and bus	97.9%	96.45%	96.31%
Xiao et al. [9]: walk, bus & taxi, bike, car, subway, and train	90.77%	93.19%	92.81%
Zheng et al. [11]: walk, driving, bus, and bike	76.2%	93.61%	93.51%
Endo et al. [2]: walk, car, taxi, bike, subway, bus, and train	83.2%	90.20%	89.95%

Table 2. F1 measures to classifiers for different class groups.

Reference	DT		RF		NN		NB		QDA	
	With noise	Clean	With noise	Clean	With noise	Clean	With noise	Clean	With noise	Clean
Dabiri et al. [1]	85.56	92.31	88.07	93.22	85.18	89.87	63.30	82.91	54.76	79.83
Jiang et al. [4]	88.26	95.47	91.56	96.31	88.63	94.11	65.68	85.19	54.70	82.55
Xiao et al. [9]	84.38	89.79	88.75	92.81	82.93	89.01	51.40	70.03	47.81	71.45
Zheng et al. [11]	85.62	91.92	88.72	93.51	85.76	91.33	64.61	84.22	51.33	79.48
Endo et al. [2]	79.53	82.09	85.57	89.95	79.33	85.70	57.31	72.68	49.13	72.30

Table 3. Accuracy to classifiers for different class groups.

Class group	DT		RF		NN		NB		QDA	
	With noise	Clean	With noise	Clean	With noise	Clean	With noise	Clean	With noise	Clean
Dabiri et al. [1]	85.54	92.36	88.47	93.35	85.54	90.13	63.56	83.28	53.65	79.76
Jiang et al. [4]	88.41	95.54	91.91	96.45	88.80	94.21	63.70	84.31	53.03	82.07
Xiao et al. [9]	85.01	89.96	89.33	93.19	83.61	89.43	51.96	69.90	46.59	70.99
Zheng et al. [11]	85.77	92.13	89.09	93.61	86.10	91.45	64.36	84.53	50.85	79.50
Endo et al. [2]	80.25	83.61	86.36	90.20	80.27	86.28	56.66	73.27	47.92	71.60

subgroups of classes, performance gains ranging from 2.56 (Decision Tree, using classes of [2]) to 28.15 (QDA, using classes of [11]) in f1.

Finally, Table 3 shows the mean of the accuracy values obtained by 10-fold cross-validation. For all classifiers and different subgroups of classes and classifiers, performance gains ranging from 3.36 (Decision Tree, using classes of [2]) to 29.04 (QDA, using classes of [4]) in accuracy were observed. The results presented in this section indicate that dealing with noise in transportation mode prediction is an important topic, and the lack of this step in the classification task decreases the performance of the classifiers.

4 Conclusions and Future Works

In this work, we propose a framework for transportation mode prediction using feature engineering and noise removal. The results showed that the newly engineered features (e.g., bearing rate, and rate of bearing rate) and the application of a noise removal technique improve the performance of all tested classifiers. We intend to extend this work in two directions: (i) test and evaluate different noise removal techniques like wavelet-based, MCMC and fast Fourier based denoising methods, and (ii) investigate the performance of trajectory segmentation algorithms and include this step in our framework.

Acknowledgments. The authors would like to thank NSERC (Natural Sciences and Engineering Research Council of Canada) for financial support.

References

1. Dabiri, S., Heaslip, K.: Inferring transportation modes from GPS trajectories using a convolutional neural network. *Transp. Res. Part C Emerg. Technol.* **86**, 360–371 (2018)
2. Endo, Y., Toda, H., Nishida, K., Kawanobe, A.: Deep feature extraction from trajectories for transportation mode estimation. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) PAKDD 2016. LNCS (LNAI), vol. 9652, pp. 54–66. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31750-2_5
3. Han, J., Pei, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Elsevier, Amsterdam (2011)
4. Jiang, X., Souza, E.N., Pesaranghader, A., Hu, B., Silver, D.L., Matwin, S.: trajectorynet: an embedded GPS trajectory representation for point-based classification using recurrent neural networks. arXiv preprint [arXiv:1705.02636](https://arxiv.org/abs/1705.02636) (2017)
5. Soares Júnior, A., Moreno, B.N., Times, V.C., Matwin, S., dos Anjos Formiga Cabral, L.: GRASP-UTS: an algorithm for unsupervised trajectory segmentation. *Int. J. Geogr. Inf. Sci.* **29**(1), 46–68 (2015)
6. Lin, M., Hsu, W.-J.: Mining GPS data for mobility patterns: a survey. *Pervasive Mob. Comput.* **12**, 1–16 (2014)
7. Spaccapietra, S., Parent, C., Damiani, M.L., Macedo, J.A., Porto, F., Vangenot, C.: A conceptual view on trajectories. *Data Knowl. Eng.* **65**(1), 126–146 (2008)
8. Stenneth, L., Wolfson, O., Yu, P.S., Xu, B.: Transportation mode detection using mobile phones and GIS information. In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2011*, pp. 54–63. ACM, New York (2011)
9. Xiao, Z., Wang, Y., Fu, K., Fan, W.: Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *ISPRS* **6**(2), 57 (2017)
10. Yanyun, G., Fang, Z., Shaomeng, C., Haiyong, L.: A convolutional neural networks based transportation mode identification algorithm. In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7, September 2017
11. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.-Y.: Understanding mobility based on GPS data. In: *Proceedings of the 10th International Conference on Ubiquitous Computing*, pp. 312–321. ACM (2008)



Prediction of Container Damage Insurance Claims for Optimized Maritime Port Operations

Ashwin Panchapakesan¹, Rami Abielmona^{1,2}, Rafael Falcon^{1,2}, and Emil Petriu¹

¹ School of Electrical Engineering and Computer Science,
University of Ottawa, Ottawa, Canada
{apanc006,petriu}@uottawa.ca

² Research and Engineering Division, Larus Technologies Corporation,
Ottawa, Canada
{rami.abielmona,rafael.falcon}@larus.com

Abstract. A company operating in a commercial maritime port often experiences clients filing insurance claims on damaged shipping containers. In this work, multiple classifiers have been trained on synthesized data, to predict such insurance claims. The results show that Random Forests outperform other classifiers on typical machine learning metrics. Further, insights into the importance of various features in this prediction are discussed, and their deviation from expert opinions. This information facilitates selective information collation to predict container claims, and to rank data sources by relevance. To our knowledge, this is the first publication to investigate the factors associated with container damage and claims, as opposed to ship damage or other related problems.

1 Introduction

Commercial maritime ports require high operational throughput, highlighting the importance of optimizing intensive workflows such as those induced by filing a claim on a damaged shipping container. To handle such a claim, port officials collate and analyze multimodal data including visual inspection logs cargo manifests environmental data, which is a time-consuming process [1,2]. Operational time aside, human predictive accuracy reduces with human cognitive biases [3], data volume and complexity, and prediction specificity [4] and volume [5].

Fast and accurate automated data processing is computationally expensive, due to noisy and incomplete data. Predicting container claims affords restricting data collation to smaller subsets, reducing computational requirements.

To our knowledge, this is the first publication to investigate container damage causes using Machine Learning (ML) techniques, which constrains the scope of the current knowledge against which to compare the presented methodologies. Accurate container damage predictions enable discussing the dynamic selection between data sources and algorithms used to process the data therefrom. This may be used in a decision support system integrated into a terminal OS

to realize container damage causes and alleviate data collation and analysis bottlenecks.

The remainder of the paper is structured as follows: Sect. 2 reports a survey of related work, while our methodologies are outlined in Sect. 3, the results of which are presented and discussed in Sect. 4. Finally, Sect. 5 concludes this work with some directions in which it can be expanded.

2 Related Work

Related publications incorporating ML solutions and features involving ports, ships, personnel, weather, and geography are used to guide this study.

Surveys of Taiwanese domain experts [6] reveal a taxonomy of risks posed to refrigerated containers in their port-to-port travel, while Canadian experts (discussed in Sect. 3) maintain that quay crane operator errors are most relevant human error sources. A study of the determining factors of consignors' port choice shows positive correlation with port proximity due to decreased en route damage probability [7]. Operational conditions such as weather and human factors are the primary travel risk factors [8,9]. Further, terminal-side accident reports are confidential, hence unavailable. Yet, probability distributions modeling operational capabilities of personnel and equipment were used in this study.

A case study of a Seattle-bound ship [10] discusses mathematical constructs used in simulation software [11], whose features guide this work. Yet, no ML methodologies were used to compute feature importance. Ship characteristics extracted from a source like [12] may be correlated with maritime accidents using an ML approach [13]. Mined features and probability densities of environmental have been used in a custom discrete event simulator, to determine oil tanker loading times and port storage capacity [14]. Few publications use ML techniques to discover the causes of shipping container damage. One study compared the efficacy of decision trees in predicting the total loss and damage to a ship [15], supporting the use of a binary classification tree.

3 Methodology

3.1 Data

To predict container damage, container voyage data is required. A trained prediction model may be applied to real-world scenarios in order to learn which data sources and feature/information extraction techniques are best suited for a given scenario. As robust, accurate algorithms typically require more computational power, it is favorable to use them only when absolutely necessary. Determining which data sources to use at a given time, and when to switch between algorithms is out of the scope of this paper and left as future work.

The synthesis of the container damage prediction data set (of 1.6M records) is discussed in this section. Each attribute in this data set was captured in a survey

completed by industry experts in Canada.¹ The survey results revealed how strongly each attribute and correlated with container damage claims. Attribute values were accordingly weighted on a (0, 3] scale to capture their correlation with container damage. The value of each attribute was modeled to contribute some amount of damage to be sustained by the container, which increased the probability of it being claimed. The generation of attribute values and their contributed damage to the container are discussed below.

Data Synthesis. Each shipping container was assigned to one of 46 known tracks in Jan. - Mar. 2014 and Douglas Sea Scale (DSS) [16] measures were computed from environmental data [17]. The average damage contribution of sea state (modeled as the modified sigmoid function $(2(1 + e^{7-DSS}))^{-1}$) weighted at 1 is the damage sustained by the container.

The containers were probabilistically assigned to shipping lines (*SLs*) and trucking companies using roulette wheel selection [18], weighted by annual cargo throughput [19,20]. The error probability per shipping line (*L*) was computed based on fleet size ($FS(L)$), annual throughput ($T(L)$) and market share ($MS(L)$), weighted at 3, as described by Eq. (1), capturing the notion that shipping lines with higher *FS*, *T*, and *MS* are less likely to cause shipping container damage.

$$P(Error|L) = \begin{cases} 0 & c(L) > \frac{2}{3}C, & C = \max(\{c(L)|L \in SLs\}) \\ 0.5 & c(L) < \frac{1}{3}C, & c(L) = 0.3 \times FS(L) + 0.35 \times T(L) \\ \frac{3 \times c(L) - 1}{2} & \text{otherwise} & + 0.35 \times MS(L) \end{cases} \tag{1}$$

Finally, the container’s recipient correlates positively with damage claims, which were modeled as having a claim probability and a claim amount, respectively distributed in $\mathcal{U}([0, 0.5])$ and $\mathcal{U}([0, 10^8])$, weighted at 1. The remaining features (cargo value, fragility, sensitivity, mass distribution, packing and loading seasons, etc) and their correlated error probabilities are listed in Table 1.

Classifiers were trained on this 15-dimensional data describing the above features, to learn which features accurately predict container claims. This allows for the proactive gathering and collation of the terminal-side data, to present to port officials upon the incidence of a claim (as mentioned in Sect. 1).

3.2 Experimental Setup

Table 1 lists the CI methods used from [21], along with their 95% Confidence Intervals performance metrics. These were trained by 10×10 fold cross validation, with training data drawn from the synthesized 1.6M records. This was then run 30 times, accounting for 100 data points per run, shown in Fig. 1.

¹ The survey (named Bottlenecks in Port Operations) was distributed by a Google Forms link in July 2017, after receiving the necessary approval from the Research Ethics Board of the University of Ottawa.

Table 1. Container features and error probabilities

Feature name	Feature weight	Feature distribution	Error probability
Cargo value (v)	3	$\mathcal{N}\left(\mu = \frac{10^5}{2}, \sigma^2 = 10^{10}\right)$	$P(Error v) = \begin{cases} 0 & v \leq V/3 \\ 0.5 & v \geq V/3 \\ 1.5 \times \frac{v}{V} - 0.5 & \text{otherwise} \end{cases}$
Cargo sensitivity (s)		$\mathcal{U}(\{0, 1\})$	$P(Error s) = 0.5 \times s$
Weight balance (d)		$\mathcal{U}(0, 20.4)$	$P(Error d) = 0.5 \times \frac{d}{20.4}$
Quay crane operator (g)	1	$\mathcal{U}(0, 0.1)$	$P(Error g) = g^{-1}$
Packing/loading season (s)		$\mathcal{U}(\{\text{fall, winter, spring, summer}\})$	$P(Error s) = \begin{cases} 0.5 & s \in \{\text{fall, spring}\} \\ 0 & \text{otherwise} \end{cases}$
Time in storage yard (t)		$\mathcal{U}(0, 364)$	$P(Error t) = \frac{366}{365-t}$
Cargo fragility (f)	0.01	$\mathcal{U}(\{0, 1\})$	$P(Error f) = 0.01 \times f$
Container weight (w)		$w = \begin{cases} 55 & \text{with probability 0.2} \\ 275x_{0.2 \leq x \leq 0.4} & \text{with probability 0.2} \\ 110 & \text{with probability 0.6} \end{cases}$	$P(Error w) = 2 \times \left(\frac{w}{110} - 0.25\right)^2$

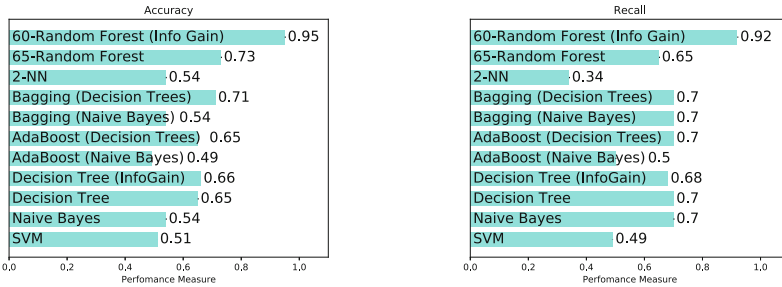


Fig. 1. Performance metrics of various classifiers

4 Results and Discussion

Human reviewers remain the post-data-collation bottleneck. Since investigations are triggered by incoming claims, false negatives are less favorable. It is therefore important to compare the accuracy of classifiers that have equivalent AUC (shown in Fig. 1). The superior performance of Random Forests (significantly better than random chance) suggests that other such nonlinear and ensemble methods may be applicable. Analysis of trained decision trees and Random Forests reveals relative feature importances (see Fig. 2), showing that the amount of time a container spends in the storage yard is the most revealing feature in predicting container damage claims. While a container’s cargo value was expected to be an important feature, the quay crane operator, shipping line, and time spent

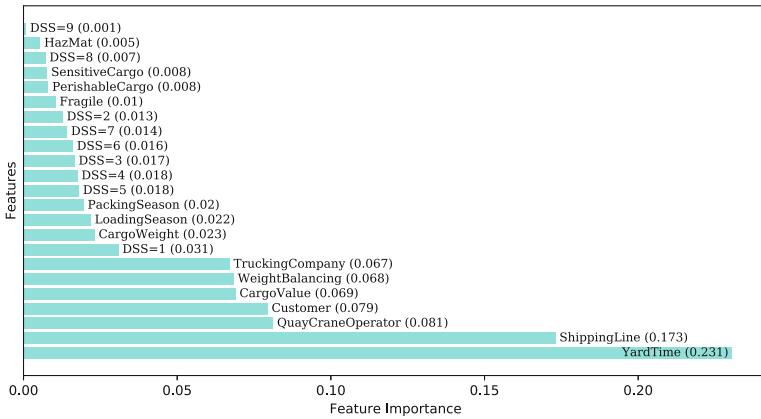


Fig. 2. Relative feature importances

in the storage yard yield the highest container claims predictability, counter to expert opinion.²

Analyzing the most important features shows that cargo value, hazardousness, longevity, sensitivity, mass distribution, storage time, and exposure to rough seas correlate strongly positively with filed claims, while calm sea states have strong negative correlation. Container packing and loading seasons correlate weakly with container claims with 25% support and 13% confidence in the relevant association rules. Logistics companies are also weak features (18% support, 11% confidence).

Cargo Fragility is found to very weakly correlate with container damage claims (Pearson R coefficient of 3.7×10^{-4} and 50% support, 50% confidence for relevant association rules), again counter to expert opinion. This could be due to proper container packing compensates for cargo fragility, while quay crane operator error dominates in human error (with 8% support and 4% confidence).

As expected, rough sea conditions correlate with more insurance claims. Additionally, the amount of time a container spends in the storage yard positively correlates with container damage claims, counter to the survey results. This may be due to increased in-yard storage density decreasing the performance of container moving equipment [2].

5 Conclusions and Future Work

Future directions of this study include identifying and fusing alternative data sources (e.g., sources of weather, commercial cargo values, etc), enabling pertinence based data source ranking, to dynamically switch between them. This would save computational costs without compromising prediction accuracy.

² The survey results showed that cargo value, hazardous and/or sensitive cargo were the most important attributes in predicting insurance claims.

Acknowledgments. We would like to thank SciNet on whose infrastructure our experiments were run (Each GPC node runs Intel’s Xeon E5540 8-core CPU at 2.53 GHz, with 16 GB RAM; each P7 node runs an IBM Power 755 server with four 8-core 3.3 GHz Power7 CPUs and 128 GB RAM. Detailed specifications can be found at <https://www.scinethpc.ca/>). We would also like to thank Montreal Gateway Terminals Partnership and their on-staff domain experts, and professional highway truck driver Alyssa Fred Wai-Yi Wong, for sharing their expertise used in data creation and validation.

References

1. Bichou, K.: Port Operations, Planning and Logistics. CRC Press, Boca Raton (2014)
2. Stahlbock, R., Voß, S.: Operations research at container terminals: a literature update. *OR Spectr.* **30**(1), 1–52 (2008)
3. Caponecchia, C., et al.: It won’t happen to me: an investigation of optimism bias in occupational health and safety. *J. Appl. Soc. Psychol.* **40**(3), 601–617 (2010). <http://doi.wiley.com/10.1111/j.1559-1816.2010.00589.x/abstract>
4. Yoon, S.-O., et al.: The devil is in the specificity : the negative effect of prediction specificity on prediction accuracy. *Psychol. Sci.* **XX**, 1–7 (2010). <http://pss.sagepub.com/>
5. Hart, S.G., Staveland, L.E.: Development of NASA-TLX (Task Load Index): results of empirical and theoretical research. http://stavelandhfe.com/images/NASA-TLX_paper.pdf
6. Tseng, W.-J., et al.: Risk analysis of cargos damages for aquatic products of refrigerated containers: shipping operators’ perspective in Taiwan. *Inf. Manag. Bus. Rev.* **4**(2), 86–94 (2012)
7. Nir, A.-S., Lin, K., Liang, G.-S.: Port choice behaviour-from the perspective of the shipper. *Marit. Policy Manag.* **30**(2), 165–173 (2003)
8. Hsu, W.K., Huang, S.S., Yeh, R.J.: An assessment model of safety factors for product tankers in coastal shipping. *Saf. Sci.* **76**, 74–81 (2015)
9. Hsu, W.-K.K., et al.: Assessing the safety factors of ship berthing operations. *J. Navig.* **68**(03), 576–588 (2015)
10. France, W.N., et al.: An investigation of head-sea parametric rolling and its influence on container lashing systems. *Mar. Technol.* **40**(1), 1–19 (2003)
11. Kawahara, Y., Maekawa, K., Ikeda, Y.: A simple prediction formula of roll damping of conventional cargo ships on the basis of Ikeda’s method and its limitation. In: Almeida Santos Neves, M., Belenky, V., de Kat, J., Spyrou, K., Umeda, N. (eds.) *Contemporary Ideas on Ship Stability and Capsizing in Waves. Fluid Mechanics and Its Applications*, vol. 97, pp. 465–486. Springer, Dordrecht (2011). https://doi.org/10.1007/978-94-007-1482-3_26
12. Lloyd’s Register (2016). <http://www.lr.org/en/>
13. Kelangath, S., et al.: Risk analysis of damaged ships-a data-driven Bayesian approach. *Ships Offshore Struct.* **7**(3), 333–347 (2012)
14. Stanivuk, T., et al.: How to predict cargo handling times at the sea port affected by weather conditions. *Croat. Oper. Res. Rev.* **3**(1), 103–112 (2012)
15. Kokotos, D.X., et al.: A classification tree application to predict total ship loss. *J. Transp. Stat.* **8**(2), 31 (2005)
16. EuroWeather: Douglas Scale (1995). <http://www.eurometeo.com/english/read/doc.douglas>

17. NOA Administration: Wave Watch III Model Data (2014)
18. De Jong, K.: Analysis of the behavior of a class of genetic adaptive systems (1975)
19. List of Shipping Lines (2017). https://en.wikipedia.org/wiki/List_of_container_shipping_companies_by_ship_fleets_and_containers
20. Top 100 (2015). www.todaystrucking.com/top100/2015
21. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)



Drug-Target Interaction Network Predictions for Drug Repurposing Using LASSO-Based Regularized Linear Classification Model

Jiaying You^{1,2}, Md. Mohaiminul Islam^{1,3}, Liam Grenier¹,
Qin Kuang¹, Robert D. McLeod², and Pingzhao Hu^{1,2,3}✉

¹ Department of Biochemistry and Medical Genetics,
University of Manitoba, Winnipeg, Canada
Pingzhao.hu@umanitoba.ca

² Department of Electrical and Computer Engineering,
University of Manitoba, Winnipeg, Canada

³ Department of Computer Science, University of Manitoba, Winnipeg, Canada

Abstract. It has been well-known that biological and experimental methods for drug discovery are time-consuming and expensive. New efforts have been explored to perform drug repurposing through predicting drug-target interaction networks using biological and chemical properties of drugs and targets. However, due to the high-dimensional nature of the data sets extracted from drugs and targets, which have hundreds of thousands of features and relatively small numbers of samples, traditional machine learning approaches, such as logistic regression analysis, cannot analyze these data efficiently. To overcome this issue, we proposed a LASSO-based regularized linear classification model to predict drug-target interactions, which were used for drug repurposing for inflammatory bowel disease. Experiments showed that the model outperformed the traditional logistic regression model.

Keywords: Bioinformatics · Classification · Drug-target interaction
Supervised learning · LASSO · Drug repurposing

1 Introduction and Related Works

Studying drug-target interactions is essential in drug discovery. Biological and experimental methods of drug discovery are time-consuming, expensive and difficult. The shortcomings of the methods have led to increasing interests in applying machine learning methods to analyze the vast amount of complex data available to researchers.

Yamanishi et al. [1] developed a kernel regression-based model to predict a drug-target interaction network that can identify previously unknown drug-target interactions from various types of biological data, such as chemical structures and protein domains. They used these features to generate a kernel similarity matrix, where each descriptor corresponds to a drug-drug similarity or protein-protein similarity. The model used the assumption that similar compounds are likely to interact with similar proteins. Wang et al. [2] used data of 6100 human cell cultures treated by 1309 bioactive compounds contained in Connectivity Map (CMap) to predict drug-target

interactions of unknown drugs. The datasets usually contain more than 10 thousand genes and the number of instances is likely small. Some methods like logistic regression model require a larger number of samples to avoid over-fitting. Feature selection is highly recommended. Ezzat et al. [3] used DrugBank [4] to collect data consisting of 12674 drug-target interactions with 5877 drugs and 3348 protein interaction partners. They used this data to generate features that allowed for drug-target predictions with a chemogenomic approach.

Limitations exist in training machine learning models for the drug-target interaction predictions: (1) there is a much greater number of negative than positive samples, and (2) there are usually thousands of features extracted from the drugs and targets. The first issue makes it difficult to select negative samples so random selection is usually deployed. This can be problematic as it has the possibility of excluding useful data. This study explored a new approach to select the negative samples and applied the regularized method for drug-target predictions.

2 Materials and Methods

2.1 Datasets

Data collection. The drug and target sets were downloaded from Drugbank. We used drug - target interaction data under protein identifier (Version 5.0.10) and external target drug-uniprot link (Version 5.0.10). Biotech drugs were excluded and proteins that are not targets for humans were ignored. Thus, 14,792 known drug-target pairs from drugbank were generated with 5,834 drugs and 3,456 proteins.

Drug and target feature representation. We downloaded drug structure information from Drugbank. To get drug descriptors from the drug structure information, we used an online server named online chemical database with modeling environment [5]. Drugs that are unavailable to get their descriptors from this server were removed. We retrieved 2,216 drug descriptors for each of the remaining 5,134 drugs. To generate target/protein feature representation, we assumed that the complete information of a target protein is encoded in its sequence and domains, thus we extracted protein features from the commonly recognized features-sequence descriptors and protein domains. The sequence and corresponding domain information of target proteins were downloaded from Drugbank and NCBI (National Center for Biotechnology Information Search database) respectively. Protein sequence descriptors contain amino acid composition (ACC), dipeptide composition (DC) and tripeptide composition (TC). Domain information is represented as an adjacency matrix where 1 indicates an interaction and 0 indicates no interaction between a pair of protein and domain. In total, we extracted 11,943 protein features for each target. Therefore, each pair of drug-target/protein (D, P) can be represented as a feature vector $[D_1, \dots, D_{2216}, P_{ACC1}, \dots, P_{ACC20}, P_{DC1}, \dots, P_{DC400}, P_{TC1}, \dots, P_{TC8000}, P_{domain1}, \dots, P_{domain3523}]$.

After all drug and target feature information was generated, the 13,168 drug-target interaction pairs with 5,132 unique drugs and 3,184 unique proteins were collected. All these 13,168 known drug-target interaction pairs are treated as positive samples.

The number of all possible drug-target pairs from the collected drug and target lists is 16,327,120 ($5,132 * 3,184 - 13,168$) of which are considered unknown interactions. These can be potential negative or positive drug-target pairs.

Construction of negative drug-target interaction pairs. Instead of randomly selecting negative samples from the remaining 16,327,120 unknown pairs, we proposed a novel method to select the most likely negative drug-target interaction pairs. The assumption of this method is based on “guilt-by-association”, which indicates similar drugs may share similar targets and vice versa [6]. Based on this intuition, we calculated the drug similarity measured by Tanimoto coefficient for each pair of the drugs using R package Rcp and drugs with Tanimoto coefficients >60% were inferred as similar [6]. It is believed that similar drugs are more likely to be functionally correlated, thus, targets that interact with one drug are also considered to interact with its similar drugs, and pairs of these drug-target interactions were referred as potential interacted pairs as shown in Fig. 1. Protein similarities were also calculated using Protr R package. They were derived by protein sequence alignment, which is a way to identify regions of similarity of proteins based on their functional, structural, or evolutionary relationships. After obtaining the similarities between proteins, the same filtering rule was applied to find potential interacted drug-target pairs based on the protein similarity scores. That is, if the similarity score between two proteins (sequence identity score) is above 40% [6], it will be assumed that these proteins share the same interacted drugs. We followed the similar procedure as Fig. 1 to identify potential

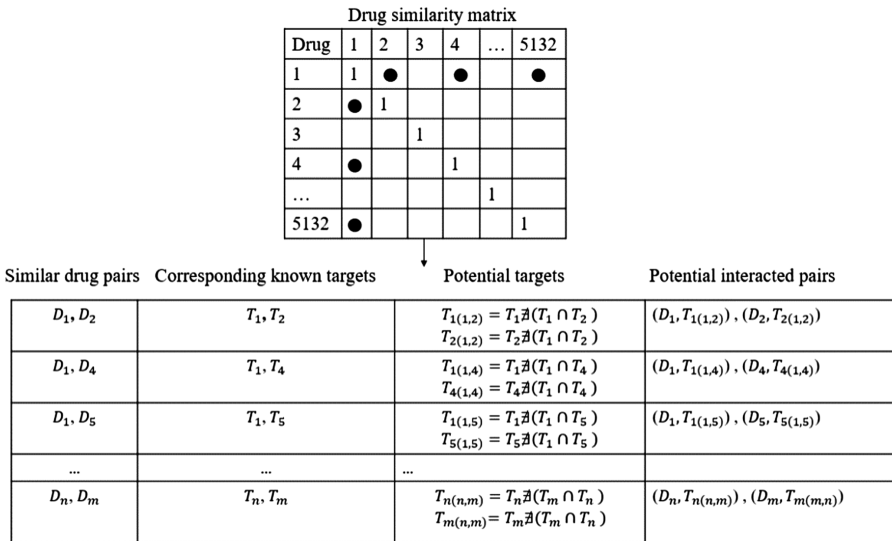


Fig. 1. Flowchart of finding potential interacted drug-target pairs by drug similarities. Bold dots in similarity matrix indicate the Tanimoto coefficients of the drug pairs are larger than 0.6. Thus, the corresponding drug pairs are referred as similar, such as D_1 and D_2 , D_1 and D_4 . D_n denotes drug n, T_n denotes target list of drug n, and $T_{n(n,m)}$ denotes potential target list of drug n emerged from the high similarity with drug m.

drug-target interacted pairs by protein similarity. After we removed all potential positive drug-target interaction pairs, we randomly selected 13,168 negative drug-target pairs from the 16,327,120 unknown drug-target pairs.

2.2 Classification Models

To predict drug-target interactions, we implemented standard logistic regression (SLR) and LASSO-based regularized linear classification (LASSO) [7] models. Overfitting might be noticeable in the SLR model since we had more than 14,000 features representing drugs and proteins while there were a relatively small number of samples (positive and negative drug-target pairs). It is necessary to select informative features for building the SLR models to alleviate the overfitting problem. Since the majority of our features were category variables, feature selection based on the training data set was implemented using Wilcoxon rank sum test. We ranked all features using their p-values of the test. To decrease the effect of those less significant features, the improved logistic regression model LASSO was implemented. It reduces the effect of features which are referred as “not significant” estimated by their p-values. The aim of LASSO is to find the optimized hyperplane $z = \theta^T x + \lambda \sum |\theta|$ that correctly classifies the positive and negative drug-target pairs, where x denotes feature space and the optimized parameter vector θ fits a curve to separate the training data. λ is selected to minimize the output of sample errors by grid search using cross-validation technology. In other words, only features considered “significant” could have contributions on the optimized hyperplane. To simplify the measurement of its output, we considered the sigmoid function to represent the probability of an interacted positive drug-target pair (y) given one sample x : $p(y = 1|z) = g(z) = \frac{1}{1+e^{-z}}$.

In our study, we applied SLR models with the top 100, 200, 500, 1,000, 2,000 of all 14,159 features selected by Wilcoxon rank sum test respectively. We also tried to use more than 2,000 features in the SLR models, but the models could not be fitted due to the large number of features and small number of samples. We implemented 5 LASSO models with all 14,159 features (M1), drug features alone with one single protein feature (ACC/DC/TC/DOMAIN) for M2, M3, M4, M5, respectively.

We built these models using R packages `e1071` for SLR and `glmnet` for LASSO using a training-testing strategy, that is, we split our samples (positive and negative drug-target pairs) into 2/3 and 1/3 of all samples as our training and test datasets respectively. For each of the training and test sets, we ensured there are the same numbers of positive and negative drug-target pairs. For LASSO models, we performed 10-fold cross-validation of the training set to select the optimized parameter λ . The performance of these models was evaluated based on overall accuracy, which is the percentage of drug-target pairs with correctly predicted interactions, and AUC (Area Under the Curve) of the Receiver Operating Characteristics (ROC), which illustrates the pattern of sensitivity and specificity at different probability cutoffs to predict a drug-target pair to be an interacted pair.

2.3 Drug Repurposing Based on the Predicted Drug-Target Interaction Network

We predicted all possible drug-target pairs using the trained LASSO M1. The pairs with predicted probability 0.95 or larger were considered as a predicted drug-target interaction network. Using the predicted network, we repurposed the potential candidate drugs for inflammatory bowel disease (IBD). To do this, we used the 225 unique IBD risk loci identified by genome-wide association study [8]. The 201 IBD risk genes implicated by these IBD risk loci were mapped to proteins through online UniProt mapping server. We extracted the predicted drug-target pairs from the predicted network that contain the IBD risk genes (proteins). We calculated the frequency of each drug interacted with the IBD risk proteins based on these extracted drug-target pairs. We repurposed the top drugs ranked by the frequency for IBD.

3 Result

The accuracies and AUCs of the SLR models using different numbers of top features were shown in Table 1. Generally speaking, models with more selected features had much greater power to predict drug-target interactions. The model with the best performance was based on top 2000 features. The accuracies and AUCs of the LASSO models were shown in Table 2. The best model was M1 with all five feature sets. M2 and M3 with fewer numbers of features seemed weaker than other models. However, the model performances using protein TC (M4) and protein domain (M5) were remarkably improved. Better results were obtained from the testing set than training set as shown in Table 2, which indicated we avoided over-fitting perhaps due to the model lambda selection. Instead of using lambda_min, lambda that achieves the largest AUCs for our training set based on 10-fold cross-validation, we used lambda_1se to fit the models. The Lambda_1se we selected was the largest value of lambda such that AUC was within 1 standard error of the maximum AUC. Although the Lambda_1se did not result in the best performance in training set, it considered model uncertainty to make more reliable predictions for the testing set. Comparison of the results of our proposed LASSO models (Table 2) with our baseline models (Table 1) showed that the proposed LASSO models outperformed the SLR models.

Table 1. Overall accuracy and AUC of the SLR models of the test set

Logistic regression model	Numbers of top features selected by Wilcoxon rank sum tests				
	100	200	500	1000	2000
Accuracy	0.64	0.66	0.70	0.72	0.75
AUC	0.70	0.73	0.77	0.80	0.81

Table 2. Overall accuracy and AUC of the LASSO models

		M1	M2	M3	M4	M5
10-fold cross-validation of training set	Accuracy	0.75	0.63	0.69	0.74	0.74
	AUC	0.82	0.69	0.76	0.81	0.82
Test set	Accuracy	0.75	0.64	0.70	0.75	0.75
	AUC	0.83	0.69	0.77	0.83	0.83

A total of 32,304 drug-target interaction pairs were predicted to have interaction probabilities larger than or equal to 0.95 using the trained LASSO M1. We extracted 815 drug-target pairs that included the 201 IBD risk genes and 463 drugs from the 32,304 predicted drug-target interaction pairs. The top drug Ademetionine interacted with 10 IBD risk proteins in the network. The drug has been used in treating liver disease for alleviating inflammatory activity, which could make the drug a potential candidate for treating inflammatory bowel disease.

4 Conclusion

We showed that the LASSO-based models achieved better results than the SLR models for predicting drug-target interactions. We demonstrated that the LASSO models can work well for data sets with a large number of features, which often results in overfitting in the SLR models. Furthermore, we demonstrated that the predicted drug-target interaction pairs with top-ranked probabilities can be potentially repurposed for inflammatory bowel disease.

Acknowledgement. This work was supported in part by Canadian Breast Cancer Foundation, Natural Sciences and Engineering Research Council of Canada, Manitoba Research Health Council and University of Manitoba.

References

1. Yamanishi, Y., et al.: DINIES: drug-target interaction network inference engine based on supervised analysis. *Nucleic Acids Res.* **42**(W1), W39–W45 (2014)
2. Wang, Q., et al.: A novel framework for the identification of drug target proteins: Combining stacked auto-encoders with a biased support vector machine. *PLoS ONE* **12**(4), e0176486 (2017)
3. Ezzat, A., et al.: Drug-target interaction prediction using ensemble learning and dimensionality reduction. *Methods* **17**(Suppl 19), 509 (2017)
4. Wishart, D.S., et al.: DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Res.* **46**(D1), D1074–D1082 (2017)
5. Sushko, I., et al.: Online chemical modeling environment (OCHEM): web platform for data storage, model development and publishing of chemical information. *J. Comput. Aided Mol. Des.* **25**(6), 533–554 (2011)
6. Luo, Y., et al.: A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *bioRxiv*, 100305 (2017)

7. Friedman, J., Hastie, T., Tibshirani, R.: glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models. R package version 1.1-4 (2009). <http://CRAN.R-project.org/package=glmnet>
8. Liu, J.Z., et al.: Association analyses identify 38 susceptibility loci for inflammatory bowel disease and highlight shared genetic risk across populations. *Nat. Genet.* **47**(9), 979–986 (2015)



Optimal Scheduling for Smart Charging of Electric Vehicles Using Dynamic Programming

Karol Lina López and Christian Gagné^(✉)

Computer Vision and Systems Laboratory/REPARTI,
Dép. génie électrique et génie informatique, Big Data Research Centre,
Université Laval, Québec, QC, Canada
karol-lina.lopez.1@ulaval.ca, christian.gagne@gel.ulaval.ca

Abstract. We are proposing a formulation of the smart charging problem that can be solved by dynamic programming. It allows the optimal charging schedule of EVs to be determined in order to minimize the cost considering the different driving patterns of each car owner as well as the electricity prices varying according to supply and demand. Conclusive experiments are made through simulations, relying upon a database storing the history of the real use of vehicles over several months and an hourly electricity price.

Keywords: Smart charging · Electric Vehicles
Sequential decision-making · Dynamic programming · Smart grid

1 Introduction

Widespread adoption of Electric Vehicles (EVs) may lead to a dramatic increase in electricity demand at peak time, considering that common usage patterns may lead to many vehicles being charged simultaneously (e.g., at supper time). There is thus a particular interest in integrating Demand-Side Management (DSM) into EV charging [1], which would provide financial incentives to EV users using smarter charging approaches that are avoiding demand peaks and as such flattening of the load curve.

Our paper is proposing a method to determine the optimal EVs charging schedule in order to minimize cost owners' charging cost considering driving patterns of each EV [2]. We formulate this as a sequential decisions problem, where an action (charge or standby) should be taken at each time step when the vehicle is connected. Furthermore, when the vehicle is not connected, a penalty is calculated if there is not enough energy available to complete the trip.

Similarly to the method that we propose and describe in the following sections, several other authors have proposed to use Dynamic Programming (DP) to determine the optimal charging schedule of an EV [3–5]. However, these optimization approaches are assuming predefined connection times and charging needs but none have captured the dependence between the charging decisions

and the real energy consumption, which can be impractical and lead to inefficient schedules that are not based on actual EV user requirements. Our approach is evaluated and compared with baseline approaches through computer simulations, having real databases of measured data collected from a fleet of cars, and the data on the price of gasoline as well as the hourly Ontario energy price.

The remainder of the paper is organized as follows. In Sect. 2, the mathematical model of the smart charging problem is provided. The modelling of this optimization problem as a Markov Decision Process (MDP) and its resolution with DP is provided in Sect. 3. Experiments with real-life datasets are presented in Sect. 4, demonstrating the effectiveness of the proposed approach, before concluding the paper in Sect. 5.

2 Optimization Problem

We define the optimal charging schedule of an EV as the best sequence of decisions (i.e., to charge the vehicle or to leave it on standby mode) when the EV is plugged in, in order to minimize the overall cost while satisfying the needs of the vehicle's owner. The charging decisions, which should be made at each time interval (e.g., 15-min period), are based on the electricity price information, the State of the Charge (SoC) of the batteries, and the energy consumed by each vehicle.

The mathematical formulation is divided into two parts corresponding to two states, namely when the car is plugged in (S_p) and when it is not (S_u). When the vehicle is plugged in, the cost function is given as:

$$S_p(t) = C_{el}(t) \frac{E_{ch}(SoC(t))}{\eta}, \quad (1)$$

where:

- $C_{el}(t)$ is the electricity price [\$/kWh] at time t ;
- $E_{ch}(SoC(t))$ is the energy [kWh] supplied to the battery when charging is conducted during a fixed time interval;
- $SoC(t)$ is the charging fraction of the nominal battery capacity at the beginning of time period t ;
- $\eta \in [0, 1]$ is the charger efficiency.

When the vehicle is not plugged in, there is no decision to make, but it is necessary to compute the new SoC at the end of the interval. Moreover, a cost can be induced for driving with a depleted battery, which corresponds to the following cost with a Plug-in Hybrid Electric Vehicle (PHEV) (i.e., driving on the gas engine):

$$S_u(t) = C_{fuel}(t) \cdot \max(F_c(SoC(t)), 0), \quad (2)$$

where $C_{fuel}(t)$ is the gasoline price [\$/l] at time t and $F_c(SoC(t))$ is the fuel consumed in litres by the vehicle in a time interval corresponding to complete

battery depletion, with an $SoC(t)$ energy level at the beginning of the time interval.

Combining Eqs. 1 and 2 yields the following optimization objective:

$$\min_{\{a(t)\}_{t=1}^T} \sum_{t=1}^T [a(t) \cdot z(t) \cdot S_p(t) + (1 - z(t)) \cdot S_u(t)], \quad (3)$$

where $a(t)$ is the decision variable on whether the vehicle is being charged (1) or on standby (0) at time t and $z(t)$ returns a Boolean value indicating whether the vehicle is plugged in (1) or not (0) to a charging station at the time interval t . This objective is subject to technical restrictions of the battery, which indicates that the power used to charge an EV ($E_{ch}(SoC(t))/\eta$) cannot exceed the capacity available in the battery ($E_{nom}(1 - SoC(t))$). The cost $S_u(t)$ is null except when the battery is depleted, in which case the vehicle would incur a penalty. For PHEV, the gas engine acts as a backup when the batteries are depleted. In the case of a Battery Electric Vehicle (BEV), $C_{fuel}(t)$ can be modified by adding a stronger penalty associated with running out of energy.

3 Resolution Through Dynamic Programming

In order to find solutions to the optimization problem given by Eq. 3, we propose to use a recursive formulation of the problem that can be solved through DP [6]. Let us define the $Q(s(t), a)$ function as holding values in a three-dimensional array, with values evaluated recursively backwards (i.e., $t = T, T-1, \dots, 1$), over all the states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. $Q(s(t), a)$ is computed as:

$$Q(s(t), a) = \begin{cases} s(t) \cdot \overline{C_{el}} & \text{if } t = T \\ r(s(t), a) + \max_{a \in \mathcal{A}} Q(s(t+1), a) & \text{otherwise} \end{cases}. \quad (4)$$

The following elements are composing the function:

- $\overline{C_{el}}$ is the average electricity price over $t = 1, \dots, T$;
- $s \in \mathcal{S}$ is a discretized value of the SoC over B bins:

$$s(t) = \frac{\lfloor SoC(t) \cdot B \rfloor + 0.5}{B}; \quad (5)$$

- $a \in \mathcal{A}$ is an action, with $a = 1$ corresponding to charging and $a = 0$ to standby mode;
- $r(s(t), a)$ is the immediate cost function (reward), in our case it is equal to the value of the energy actually transferred to the battery or the penalty cost associated with driving on the gas engine when the batteries are depleted:

$$r(s(t), a) = \begin{cases} 0 & \text{if } z(t) = 1 \text{ and } a = 0 \\ -C_{el}(t) \cdot \frac{E_{ch}(SoC(t))}{\eta} & \text{if } z(t) = 1 \text{ and } a = 1 \\ -C_{fuel}(t) \cdot F_c(SoC(t)) & \text{if } z(t) = 0 \end{cases}. \quad (6)$$

Once the values of the 3D array composing the $Q(s(t), a)$ function are computed by DP, the decision in state $s(t)$ is made according to the best Q -value:

$$a^*(t) = \operatorname{argmax}_{a \in \mathcal{A}} Q(s(t), a). \quad (7)$$

4 Experiments

For our experiments, we assumed that we are using a plug-in hybrid EV similar to the Chevrolet Volt 2013. We completed the specifications with the parameters of Panasonic Lithium-ion NRC18650 batteries given that full specifications of the Volt batteries are not publicly available. It should be stressed that the optimization procedure proposed in this paper is independent from this battery model. Any models, as complex as they can be, can be used as long as they allow the computation of the new SoC resulting from charging for a given duration.

We use a simple method to calculate F_c considering the MPG_e miles per gallon gasoline equivalent of the vehicle ($1 \text{ } MPG_e \approx 0.0470 \text{ [km/kWh]}$) and the distance travelled during a time interval. To calculate E_{ch} we used a simple battery model based on an equivalent electric circuit containing a voltage source in series with a resistor. Given space and relevance concerns, the details of this charging battery model are omitted.

The experiments rely on a database on the use of conventional vehicles collected for a fleet of cars in the city of Winnipeg¹. We also used the hourly Ontario energy price², that is the hourly price that is charged to local distribution companies in Ontario and the price of regular unleaded gasoline³ provided by the Ontario Ministry of Energy, which publishes the weekly average prices paid by the consumers across the province.

4.1 Experiment Parameters

Two experimental parameters need to be defined: the number of bins (B) used to discretize the SoC in DP and the time slot duration for decision-making (Δ_t). Figure 1 shows a direct relation between the length of the time interval and the operational charging cost (Eq. 4), analyzed in two months of summer and two months of winter.

Similarly, the effectiveness of the Δ_t parameter for intervals of 30 and 60 min is significantly hindered in comparison to the effectiveness of intervals of 15 min. This is expected as a small time interval provides greater flexibility to the decision-making process. With respect to B , a discretization of 20 bins appears to be a satisfactory tradeoff, as this number of bins reflects state changes of an order of accuracy of 5%, considered as sufficient given the complexity of the problem. Note that a finer time interval and a high B parameter also imply a substantial increase of the model complexity and the computational burden.

¹ <http://hdl.handle.net/1993/8131>.

² <http://www.ieso.ca/power-data>.

³ <http://www.energy.gov.on.ca/en/fuel-prices>.

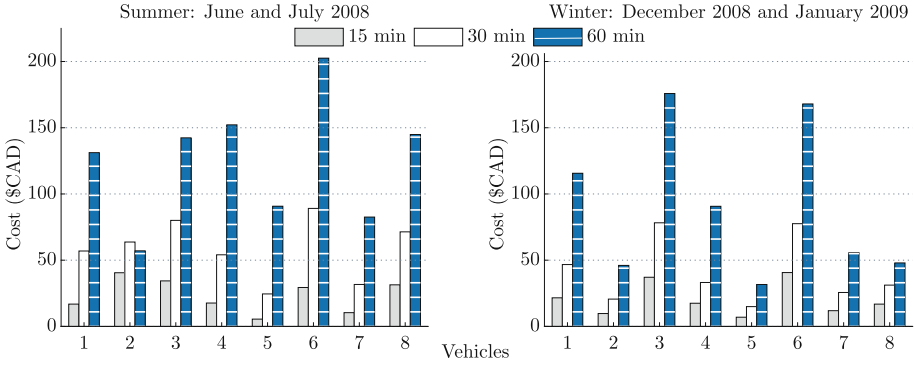


Fig. 1. Average cost of eight vehicles for different numbers of bins by Δt for the summer and winter datasets.

4.2 Results

From this point onwards, a response interval of $\Delta t = 15$ min (i.e. 96 time slots per day) and two different periods, that is August 2008 (Summer) and February 2009 (Winter) are used in the experiments.

Table 1 reports the cost of each vehicle for four situations: (1) when it depends exclusively on gasoline (GAS), (2) by using DP as the optimal charging strategy proposed, (3) by using the Random Decisions (RD) policy which consists in randomly selecting the action (charge or standby) and (4) by using a simple Always Charge (AC) strategy in which the EV users charge their vehicle whenever possible as long as its battery is not full, with no considerations for energy needs or electricity energy price.

Table 1. Costs and gain ($[\$GAS - \$Model]/\$GAS$) regarding gasoline-only vehicles (GAS) for the Dynamic Programming (DP), Random Decisions (RD) and Always Charge (AC) models in summer and winter test datasets (August 2008 and February 2009, respectively), for the 8 vehicles evaluated.

	Summer				Winter			
	GAS \$	DP \$ (%)	RD \$ (%)	AC \$ (%)	GAS \$	DP \$ (%)	RD \$ (%)	AC \$ (%)
1	131.2	34.0 (74)	93.7 (29)	91.0 (31)	65.1	7.9 (88)	30.1 (54)	29.1 (55)
2	93.0	22.1 (76)	57.1 (39)	57.4 (38)	45.5	5.4 (88)	24.8 (45)	22.2 (51)
3	187.4	15.7 (92)	60.4 (68)	60.6 (68)	66.5	7.4 (89)	28.7 (57)	27.2 (59)
4	230.8	66.8 (71)	184.1 (20)	165.6 (28)	58.8	6.3 (89)	25.2 (57)	24.6 (58)
5	81.0	3.2 (96)	20.7 (74)	21.4 (74)	40.5	4.2 (90)	18.8 (53)	18.3 (55)
6	156.8	17.3 (89)	63.0 (60)	56.5 (64)	108.3	14.2 (87)	47.4 (56)	44.4 (59)
7	20.8	0.4 (98)	5.8 (72)	6.2 (70)	61.8	6.4 (90)	27.2 (56)	25.8 (58)
8	32.0	2.3 (93)	8.3 (74)	9.6 (70)	52.8	7.2 (86)	24.7 (53)	23.0 (56)
Mean gain		86%	54%	55%		88%	54%	56%
Median gain		90%	64%	66%		89%	55%	57%

It appears clear that the proposed DP is able to significantly enhance the results in comparison with the simple RD and AC strategies, thereby reaching 90% and 89% median gains in summer and winter, respectively. These gains come from having a charging schedule that is taking into account the electricity price and energy required for making the next trips.

5 Conclusions

This paper proposes a cost-effective solution to the problem of smart charging considering driving patterns. The results show that smart charging can significantly reduce the cost of charging EVs in a context of variable electricity pricing, as is the case in Ontario, Alberta, California, etc. While it is true that the proposed DP approach requires knowing in advance the energy price and car usage for a given period of time, the decisions obtained by DP using a MDP framework can be used as desired output values to infer decision models through some supervised machine learning methods over several states composed by ancillary variables (e.g., electricity price, time of day, weekdays, SoC).

This decision-making problem and associated resolution approach, which considers real activities at non-connection times to evaluate the decisions taken in connection times, can help to personalize the driving patterns of vehicles to make wise decisions based on real requirements of vehicles. Moreover, the method can be used to label a dataset on smart charging, which in turn would allow training of a classifier on a variety of auxiliary variables, and this classifier would run in real time to decide whether a plugged-in vehicle should be charged or not.

Acknowledgments. This work was made possible through funding from NSERC-Canada, FRQNT-Québec, MITACS, and E Machine Learning Inc. The authors are grateful to Annette Schwerdtfeger and Marc-André Gardner for proofreading this manuscript.

References

1. Mou, Y., Xing, H., Lin, Z., Fu, M.: Decentralized optimal demand-side management for PHEV charging in a smart grid. *IEEE Trans. Smart Grid* **6**(2), 726–736 (2015)
2. Jia, L., Tong, L.: Dynamic pricing and distributed energy management for demand response. *IEEE Trans. Smart Grid* **7**(2), 1128–1136 (2016)
3. Li, Z., Khaligh, A., Sabbaghi, N.: Minimum charging-cost tracking based optimization algorithm with dynamic programming technique for plug-in hybrid electric vehicles. In: *Proceedings of the Vehicle Power and Propulsion Conference (VPPC)* (2011)
4. Rotering, N., Ilic, M.: Optimal charge control of plug-in hybrid electric vehicles in deregulated electricity markets. *IEEE Trans. Power Syst.* **26**(3), 1021–1029 (2011)
5. Sarabi, S., Kefsi, L.: Electric vehicle charging strategy based on a dynamic programming algorithm. In: *Proceedings of the International Conference on Intelligent Energy and Power Systems (IEPS)* (2014)
6. Bellman, R.: *Dynamic Programming*. Dover Publications, Mineola (1957)



Combining MCTS and A3C for Prediction of Spatially Spreading Processes in Forest Wildfire Settings

Sriram Ganapathi Subramanian^(✉)  and Mark Crowley

Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, Canada
{s2ganapa,mcrowley}@uwaterloo.ca

Abstract. In recent years, Deep Reinforcement Learning (RL) algorithms have shown super-human performance in a variety of Atari and classic board games like chess and GO. Research into applications of RL in other domains with spatial considerations like environmental planning are still in their nascent stages. In this paper, we introduce a novel combination of Monte-Carlo Tree Search (MCTS) and A3C algorithms on an online simulator of a wildfire, on a pair of forest fires in Northern Alberta (Fort McMurray and Richardson fires) and on historical Saskatchewan fires previously compared by others to a physics-based simulator. We conduct several experiments to predict fire spread for several days before and after the given spatial information of fire spread and ignition points. Our results show that the advancements in Deep RL applications in the gaming world have advantages in spatially spreading real-world problems like forest fires.

Keywords: Monte-Carlo Tree Search · A3C
Reinforcement Learning · Forest wildfire
Computational sustainability

1 Introduction

Recent advances in **Deep Reinforcement Learning (RL)** such as AlphaGo [1] and the more recent Alpha Zero [2] algorithms show that it is possible to combine **Monte Carlo Tree Search (MCTS)** intelligently with Neural Networks to get a sophisticated system that can beat the best Go players in the world. This serves as a motivation to try similar approaches on spatially spreading real world phenomena such as forest wildfires. Forest fires present a more difficult challenge than these board games as the dynamics of the environment in the real world are more prone to uncertainty and randomness. Making learning agents that can give highly accurate solutions in such domains has been attempted before on simpler problems [3] but stopped at the simulation level as a demonstration. This paper introduces a novel algorithmic framework for this problem, **MCTS-A3C**, which merges two existing Deep RL approaches. To validate our approach we use data from real and simulated forest fire events:

Simulated wildfires: For clean images and as a faster testing domain, we are using the forest wildfire simulator by Nova online [4]. Figure 1(d) shows the simulator at work. We used a variety of environment variable settings to create a realistic set of simulations. The fire spread with the given conditions is trained and tested with our RL algorithm.

Alberta wildfires: In an earlier work [5] we introduced the idea of using RL for prediction in spatiotemporally spreading domains such as forest wildfires. For validation we used satellite images from two forest fires in Northern Alberta (Fig. 1(b) and (c)) and compared a number of classical RL algorithms as well as the standard Deep RL algorithm A3C [6] for this domain. We consider this data again here and show that our new MCTS-A3C algorithm outperforms all these.

Saskatchewan wildfires: Physics-based simulations are the current state of the art in the wildfire analysis literature. So, a comparative study will also be made to demonstrate the performance of the MCTS-A3C algorithm against the physics-based simulation in Burn-P3 (BP3) [7] used for analysis of historical fires in central Saskatchewan (52.9399°N, 106.4509°W). For information on the input/output formats and methodology for BP3 please see [7].

2 Problem Formulation

We define a Markov Decision Process (MDP) $\langle S, A, P, R \rangle$ where the set of states S describes any location on the landscape and where the ‘agent’ taking actions is a fire spreading across the landscape. A state $s \in S$ corresponds to the state of a cell in the landscape $(x, y, te, l, w, d, rh, r, i, tm)$ where x, y are the location of the cell, te is the temperature at the particular time and location and l is the land cover type of the cell derived from satellite images (values include: water, vegetation, built up, bare land, other), w is wind speed, d is wind direction, rh is relative humidity, i is the intensity of fire as defined in [8], tm is the time considered in days from the start of fire, and r is the average amount of the rainfall at the spatial coordinates during the time of study. These variables are the largest contributing factors to fire spread in the Canadian Forest fire weather index [9]. The action $a \in A$ indicates the direction the fire at a particular cell ‘chooses’ to move: North, North-West, North-East, South, South-West, South-East, East or West or to stay put (see Fig. 1(a)). The dynamics function for any cell $P(s'|s, a)$ is a mapping from one state s to the next s' given an action a . The reward function R maps a cell state to a continuous value in the range $[-1, 1]$ and is based on the land cover. **The goal is to learn a policy for this agent that recreates the spread of the fire observed in later satellite or simulated images by maximizing discounted rewards designed to encourage high accuracy spread prediction.**

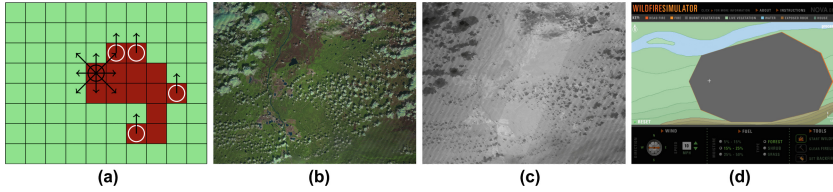


Fig. 1. Forest Wildfire Satellite Data Domain: (a) A schematic of the wildfire motion domain at a particular state and timestep. (b) Raw color satellite image of a target area. (c) Thermal image of the same area at (b) showing hotspots on fire (dark). (d) Nova fire simulator (Color figure online)

3 The MCTS-A3C Algorithm

The high level pseudo-code is given in Algorithm 1. We use an MCTS [10] search tree that contains an average reward value (X) and a visit count (N) for all the nodes. Each node is a cell on fire and has its own state space S . The fire is made to start at the ignition points and each node of the search tree is comprised of a cell burning in the resolution of the source data (30 m in case of Alberta fires and 300 m in case of Saskatchewan fires). The selection step in the algorithm is given by the UCT strategy [10]. To encourage exploration, random selection is also done with probability ϵ . The root node of the tree contains the ignition point and an action choice (selection step) leads to another cell being on fire and this results in child nodes containing the new ignited cell on fire. This process continues until we reach end of the tree containing all the visited nodes. In the next expansion phase an unvisited node is randomly added to the search tree and we simulate forward using the A3C algorithm [6]. The initial state for A3C has 84×84 cells ([11]) surrounding the center ignited cell as its start state. Each worker of A3C is defined as an instance of fire with a distinct environment and related networks. The algorithm spins off a separate worker on a distinct thread every time a new flame is started in the search space. Then each worker propagates fire in its own local condition, obtains rewards and updates the

Algorithm 1. MCTS-A3C

```

1: procedure MCTS (ROOTNODE, TIME)
2:   while time=true do
3:     currentnode  $\leftarrow$  rootnode
4:     while currentnode!=lastnode do
5:       lastnode  $\leftarrow$  currentnode
6:       currentnode  $\leftarrow$  SELECTION(currentnode)
7:     end while
8:     lastnode  $\leftarrow$  EXPANSION(lastnode)
9:     SIMULATE using A3C
10:    while currentnode do
11:      Backpropogate(currentnode)
12:      currentnode  $\leftarrow$  parent(currentnode)
13:    end while
14:  end while
15: end procedure

```

global network. Next, the back propagation step increments the visit count and updates the reward values of each node. The algorithm is configured to stop after the desired number of days in the input state (t) comes to an end for the experiment of consideration. If the intensity of fire at a cell falls below 0.3, it is empirically determined to cease to burn and is stopped from spreading further (in the case of A3C) and hence is removed from the search tree (in MCTS) until it is reignited. The model architecture for A3C follows the scheme in [11].

4 Experimental Setup

The first set of experiments (A)–(F) use the Alberta forest fire datasets. In experiment (A) we consider three consecutive, evenly spaced, timesteps and we want to test the ability of the algorithms to predict the middle time step. The ignition points for training come from the time step 1 and the training happens using the information in time step 3. This is used to determine the fire spread at the intermediate time step 2.

Experiment (B) starts with the initial state of the fire, providing rewards based on time step 2 and the algorithm predicts fire locations at time step 3. This experiment is similar to asking the question: Where will the fire spread in the next 16 days, given its position currently?

In experiments (C)–(F) we apply the learned policy from the Richardson fire to the Fort McMurray fire over four 16-day time steps in the Fort McMurray data after giving the ignition points as the input. This was a fire that happened in Northern Alberta, 5 years after the Richardson fire used for training. As the regions are similar and very near each other, the general properties that the model encapsulates should remain relevant. We provide the start state (satellite image corresponding to the start date of the Richardson Fire, for experiments (A, B) and Fort McMurray fire for experiments(C–F)) to determine the spatial and landscape features of the region under consideration.

In experiment (G), we use the Nova simulator to generate wildfire data and predict forward based on the given situation. Experiment (H) uses the Saskatchewan fire data from 1981 to 1992 for training the fire spread model and analyzed for the years from 1993 to 2002. We also compare to existing results using burn probabilities for 1993 from [7]. In the same way, Experiment (I) uses data from 1981–2002 and tests for 2003–2008. This is done to test the performance on an experiment having more training samples and less predicted spread. For both experiments (H) and (I), the RL algorithms and BP3 were run on 300 m resolution cells as this was the lowest resolution of data source grids for the cell based inputs in the experiments carried out in [7], and due to limitations in computation time.

In our experiment, the results from BP3 for every cell were translated into binary with the last three ranges characterized as a burn and the first four ranges characterized as a no-burn. We have compared the performance of our combined MCTS-A3C algorithm to the A3C, MCTS, DQN algorithms [11] and DQN with prioritized experience relay (PER) [12]. The discount rate γ was 0.99 as the goal

is to make the agent strive for a long term reward and the exploration for the ϵ greedy policy was kept as 0.05, which is along the lines of exploration rates in previous works ([6, 11]). We have rescaled every input to grey scale with 84×84 pixels as followed in [11]. We have 25 input frames per second of simulation in all the experiments and thus, an hour of training yields 90000 observations. For all the experiments, to determine if a cell was on fire, a threshold value beyond which a cell must burn was determined for the value function to balance true positives and false positives. Results are compared against the corresponding satellite images for accuracy.

5 Results

Referring to Table 1 we can notice some general trends in the accuracies for all the experiments from (A) to (I). The MCTS-A3C algorithm beats all the remaining algorithms in terms of the accuracy. The general rule seems to be $MCTS-A3C > A3C > DQN$ with $PER > DQN > MCTS$. The deep learning based approaches perform better than MCTS in most experiments. MCTS has advantages in some experiments such as (B) as it is able to fit the experiments with a limited data set that predicts forward in time better than other algorithms. A3C on the other hand is able to do well in experiments that predict in the middle of a time step (like (A)) and also in experiments that transfer the learned policy to a new test domain ((C)). Thus, as expected, MCTS-A3C, which is a combination of these two algorithms, does well in all test domains. Also, as a general rule the accuracies for all experiments decrease from (C) to (F). This is because as we keep moving into the fire season the fire becomes progressively more intense and erratic, which makes prediction tougher. These results show the generalizability of an RL approach to prediction of a spatially spreading process like fire.

For the simulated fire experiment (G), we see that MCTS-A3C has a clear advantage over all the other algorithms. In the experiments related to the

Table 1. Average Accuracy for each algorithm on the different test scenarios for Alberta fires.

Experiment	MCTS	A3C	DQN	DQN(PER)	MCTS-A3C
A	61.3%	87.3%	73.2%	79.4%	92.4%
B	60.2%	53.2%	49.5%	51.8%	90.8%
C	65.3%	90.1%	49.7%	50.8%	90.3%
D	55.7%	81.8%	72.3%	75.9%	73.4%
E	49.7%	50.8%	45.6%	48.5%	57.8%
F	5.8%	13.4%	9.4%	10.8%	50.6%
G	80.7%	84.2%	81.5%	82.7%	95.8%
H	51.3%	58.5%	52.1%	53.4%	65.8%
I	60.1%	67.2%	60.5%	62.7%	73.8%

Table 2. Accuracy values for the different ecoregions in the area of Saskatchewan for the experiment (H) and (I). Only the MCTS-A3C algorithm is considered for this comparison and is denoted as RL.

EcoRegion	BP3 (H)	MCTS-A3C (H)	BP3 (I)	MCTS-A3C (I)
Boreal Transition	33.7%	68.4%	36.7%	69.3%
Mid-boreal Lowland	53.5%	49.1%	56.5%	69.1%
Mid-boreal Upland	59.2%	75.0%	57.5%	79.2%
Churchill river upland	62.1%	70.5%	66.3%	77.8%

Saskatchewan fires seen in (H) and (I) we can see that all the algorithms have lower average accuracy. This is because there is a lot of inaccuracies in the source data [7] for experiments (H) and (I) and also because we compare several years forward from the training data set for the experiment (H). As more data samples are available for training, performance get better in the experiment (I) as compared to (H). The performance of the MCTS-A3C algorithm in the experiments (G), (H) and (I) demonstrates that it is scalable. The accuracies we obtained for the Saskatchewan fire (Table 2) using MCTS-A3C correspond well with detailed landscape burn proportions using BP3 reported in [7]. Our algorithm has a higher overall average accuracy in comparison to the BP3 outputs and also it does better in most eco regions. The Mid-boreal Lowland region for experiment (H) is the only region in which the BP3 outperforms our RL algorithm. This region is highlighted in [7] as having a much higher uncertainty due to large inaccuracies in the source fuel data. This shows that MCTS-A3C falls short of the physics based simulator only when the source data sets are affected too much by distortions and noise. But even then we can see that in experiment (I) MCTS-A3C outperforms the BP3 simulator in the same region. In general we do better in (I) as compared to (H). This shows that given enough data samples an RL approach can do better than physics based simulators. This is a clear advantage as ever larger datasets and better hyper-spectral sensors and drone images become available during forest wildfires as compared to the last couple of decades.

Concluding Remark: We have introduced and demonstrated the effectiveness of the MCTS-A3C algorithm for spatial prediction on a range of simulated and historical forest fire data. An advantage of this method is that we don't need to use expert rules and it generalizes and scales well. In future, we plan to carry out experiments on other spatial domains and further extend this algorithm to broader classes of RL problems.

References

1. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (2016)
2. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. *Nature* **550**(7676), 354–359 (2017)
3. Forsell, N., Garcia, F., Sabbadin, R.: Reinforcement learning for spatial processes. In: 18th World IMACS/MODSIM Congress, Cairns, Australia, pp. 755–761 (2009)
4. Groleau, R.: Fire wars. <http://www.pbs.org/wgbh/nova/fire/simulation.html>. Accessed 30 Nov 2017
5. Ganapathi Subramanian, S., Crowley, M.: Learning forest wildfire dynamics from satellite images using reinforcement learning. In: Conference on Reinforcement Learning and Decision Making, Ann Arbor, MI, USA (2017)
6. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016)
7. Parisien, M.A., Kafka, V., Hirsch, K., Todd, J., Lavoie, S., Maczek, P., et al.: Mapping Wildfire Susceptibility With the Burn-p3 Simulation Model. Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre Edmonton (AB) (2005)
8. Keeley, J.E.: Fire intensity, fire severity and burn severity: a brief review and suggested usage. *Int. J. Wildland Fire* **18**(1), 116–126 (2009)
9. Cortez, P., Morais, A.: A data mining approach to predict forest fires using meteorological data. In: Proceedings of the 13th Portugese conference on Artificial Intelligence, December, Guimares, Portugal, 512–523 (2007)
10. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006). https://doi.org/10.1007/11871842_29
11. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
12. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint [abs/1511.05952](https://arxiv.org/abs/1511.05952) (2015)



Text-Based Detection of Unauthorized Users of Social Media Accounts

Milton King^(✉), Dima Alhadidi^(b), and Paul Cook

Faculty of Computer Science, University of New Brunswick,
Fredericton, NB E3B 5A3, Canada
milton.king@unb.ca

Abstract. Although social media platforms can assist organizations' progress, they also make them vulnerable to unauthorized users gaining access to their account and posting as the organization. This can have negative effects on the company's public appearance and profit. Once attackers gain access to a social media account, they are able to post any content from that account. In this paper, we propose an author verification task in the realm of blog posts to detect and block unauthorized users based on the textual content of their unauthorized post. We use different methods to represent a document, such as word frequency and word2vec, and we train two different classifiers over these document representations. The experimental results show that regardless of the classifier the word2vec method outperforms other representations.

1 Introduction

Although verification by passwords is implemented to protect social media accounts, this is not completely secure. An unauthorized user can gain access to another person's or organization's social media account and publish content posing as the victim. For example, a post was published from Associated Press' Twitter account containing the phrases *Breaking: Two explosions in the White House and Barack Obama is injured*, which caused nearly 130 billion dollars worth of stocks to temporarily be lost [1]. We propose an author verification framework that predicts if a post from a social media account is written by the legitimate author or an imitator after unauthorized access to the account. This system does not stop the social media account from being accessed by an unauthorized user but rather is a form of damage control to block imitators from posting from social media accounts. We consider a variety of methods to represent documents from the original author, and train two different classifiers over these document representations to predict if a given document belongs to the authorized author or an imitator. The main contributions of our paper are as follows: (1) we propose an author verification framework that considers a variety of different representations of documents and two different classifiers; (2) we extend the dissimilarity approach proposed by Jankowska *et al.* [2] by better representing each document using word embeddings and by adopting the cosine similarity metric; (3) we consider a one-class SVM as an alternative to

the Jankowska; and (4) we show that representing a document via averaging word embeddings captures the style of an author and achieves the highest performance. We further show that a one-class SVM outperforms the classification approach of Jankowska *et al.* [2].

2 Related Work

PAN¹ is an organization that releases shared tasks for researchers to work, and test their models, on. In 2013, PAN released an author verification task.² In this task, models need to predict if a given document belongs to the author of a set of known documents [3]. Eighteen models were submitted, with one using n -grams and another using KNN classifiers with word prefixes and suffixes, parts-of-speech, and character n -grams as features. The model of Jankowska *et al.* [2] performed well on the 2013 task, and was used as the benchmark for the PAN author verification task of 2014.³ Similarly, we use this model [2] as a benchmark in our study.

Schmid *et al.* [4] proposed an approach for the author verification task in the domain of emails that could be used by law enforcement as evidence against a subject. They depend on documents from multiple authors as training data, whereas models in our experiments only see documents written by one authorized author for training. They used multiple associated rules, where each rule uses a feature for classification and a threshold to compare with.

3 Author Verification Framework

The dataset consists of blogs from the corpus of [5]. This corpus is ideal for this task because it contains social media posts from many authors, and also many posts from each author. This allows many training documents for each author as well as many different authors to evaluate on as opposed to the dataset provided by [6], which does not contain more than 10 documents per user. To generate the dataset, we first extract all authors that have more than 300 posts that each contain more than 100 words, to ensure enough material to train on. This resulted in 103 authors. We then selected ten of these authors for the development set (DEV) and 93 authors for the test set (TEST). DEV is used only for tuning parameters of classifiers.

For each author in DEV and TEST, we train a separate classifier on 255–500 randomly sampled “known” documents from this author. The classifier is then tested on a set of “unknown” documents, which consists of an even split of documents from the known author (these documents from the known author are not seen during the classifier training) and documents randomly selected from other authors in the dataset. This gives us a total of 90 and 184 instances

¹ <http://pan.webis.de>.

² <http://pan.webis.de/clef13/pan13-web/author-identification.html>.

³ <http://pan.webis.de/clef14/pan14-web/author-identification.html>.

to test on in DEV and TEST, respectively, for each classifier. The classifier must predict which of these “unknown” documents belong to the author. The sampling of documents for the training and test data for each author is run five times. Accuracy,⁴ precision, recall, and F-score are calculated for each run, and then averaged over the five runs. We then calculate the average of these metrics over all authors in each of DEV and TEST.

4 Document Representations

The different methods used to represent a document in this paper are:

Word Frequency (Bag-of-Words (BOW)). Each document is represented as a vector of word frequencies. The vocabulary is the set of words that are in the training set. We exclude all words that appear less than two times in the training set to decrease the size of the vocabulary.

Word Embeddings (word2vec). Each document is represented as the average of the word embeddings of the words in the document. The word embeddings are generated using word2vec’s skip-gram model [7], trained on a snapshot of Wikipedia with a window size of ± 8 using 300 dimensions. Skip-gram is a feed forward artificial neural network that generates vector representations of words by attempting to predict the words in the context of a given target word.

5 Classifiers

The classifiers used in the proposed framework are described below.

5.1 Jankowska

The method of Jankowska *et al.* [2] measures the dissimilarity between two documents based on the differences in frequencies of words in those documents as follows:

$$diff = \sum_{x \in (D_1 \cup D_2)} \left(\frac{f_{D_1}(x) - f_{D_2}(x)}{\frac{f_{D_1}(x) + f_{D_2}(x)}{2}} \right)^2$$

where D_1, D_2 are two documents, x is a word, and $fD(x)$ is that word’s frequency in a document. Given a set of documents known to be from an author, the *diff* measure is calculated for all document pairs in this set, to determine the maximum document dissimilarity for this author. Then, given an unknown document, the *diff* measure is calculated between this document and each known document

⁴ Accuracy is an appropriate evaluation metric because of the balanced classes in the test data.

from the author. The resulting *diff* values are each divided by the maximum dissimilarity, and then averaged. The resulting value is then compared to a dissimilarity threshold, which must be tuned on validation data. If the value is less than the threshold, then the unknown document is labeled as belonging to the author. We improved this model by representing each document as a bag-of-words, or an average of word embeddings. Moreover, we replace the *diff* measure with a cosine-based distance measure, calculated as $1 - \max(0, \cos(D_1, D_2))$, where $\cos(D_1, D_2)$ is the cosine similarity of two document vectors (D_1 and D_2). We refer to these methods in Sect. 6 as Jan+x, i.e., Jan+BOW uses a bag-of-words document representation and cosine-based dissimilarity.

5.2 One-Class SVM

Here we consider a one-class SVM as an alternative to the method of Jankowska *et al.* [2] for determining whether an unknown document belongs to a given author. For each author, we train scikit-learn’s one-class SVM⁵ on vector representations of their training documents (based on either a bag-of-words or average of word embeddings). Test documents are then represented in the same way, and then classified using the one-class SVM. Stopwords, i.e., very high frequency words, are removed when using the SVM. We do not remove stopwords when using the Jankowska approach to compare with the original Jankowska *et al.* [2] method, which did not remove stopwords.

6 Experimental Results

We tuned the dissimilarity threshold for the Jankowska classifier, and the kernel coefficient for the SVM classifier, using accuracy on DEV, considering the parameter ranges (0.1 to 1.2) for the Jankowska classifier’s threshold, and (0.0001 to 10) increasing by a magnitude of 10 for the SVM’s kernel coefficient. The original Jankowska method, the word2vec representation, and the BOW representation achieve their highest accuracy of 0.539, 0.568 and 0.576, respectively, with a threshold of 0.2, 0.2, and 0.4. In terms of the performance of methods using a one-class SVM classifier, the BOW and word2vec methods achieve their highest accuracies of 0.571 and 0.567, with a kernel coefficient of 0.0001 and 10 respectively. For each method, we use the best parameter setting on DEV in subsequent experiments on TEST.

The results on TEST set are shown in Table 1. First looking at the accuracy, all models outperform the all-in-one baseline. For each classifier (i.e., Jan. or SVM), the word2vec representation achieves higher accuracy than other representations using that classifier. This finding demonstrates that an average-of-word-embeddings document representation can give improvements over count-based document representations for authorship verification. The best accuracy of 59.4% is achieved using SVM+word2vec.

⁵ <http://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html>.

Table 1. Experimental results on TEST. The All-in-one baseline labels all documents as not belonging to the authorized author.

Model	Accuracy	Negative			Positive			Avg. F
		Precision	Recall	F-score	Precision	Recall	F-score	
All-in-one	0.500	0.500	1.00	0.667	0	0	0	0.334
Jan.	0.516	0.534	0.764	0.566	0.268	0.268	0.228	0.397
Jan.+BOW	0.561	0.552	0.912	0.677	0.518	0.211	0.255	0.466
Jan.+word2vec	0.576	0.606	0.542	0.529	0.595	0.610	0.556	0.543
SVM+BOW	0.566	0.557	0.613	0.582	0.582	0.519	0.545	0.564
SVM+word2vec	0.594	0.567	0.704	0.625	0.653	0.484	0.547	0.586

Looking at the negative F-score, which treats a true positive as correctly identifying a document as *not belonging* to the author, Jan.+BOW was the only model to outperform the all-in-one baseline with an F-score of 0.677, making it the best model for detecting documents from unauthorized users; however, this model achieves one of the lowest scores for the positive F-score (in which true positives correspond to correctly identified documents *belonging* to the author). Jan.+word2vec achieves the highest positive F-score of 0.556. To give a better idea of each model’s performance in terms of negative and positive F-score, we further consider average F-score. SVM+word2vec, which achieves the highest accuracy, also achieves the highest average F-score of 0.586. In this case, we again see that, for each classifier, the word2vec representation achieves higher average F-score than other representations using that classifier. Unlike the case of accuracy, here we see that the SVM classifier consistently outperforms the Jan. classifier.

7 Conclusion

Author verification can be used to detect a post from an unauthorized user. We proposed an alteration to the model of Jankowska *et al.* [2], which enables their method to be used for any vector representation of documents. We showed that in terms of both accuracy and average F-score, the SVM+word2vec approach achieves the highest results. We also showed that the word2vec representation can better capture the writing style of the legitimate user and can be leveraged to protect the legitimate user’s social media account by detecting posts from imitators. Future work entails finding better document representations to further improve the accuracy.

References

1. Bradshaw, T., Massaoudi, A., Scannell, K.: Bogus terror tweet sparks shares blip, April 2013
2. Jankowska, M., Milios, E., Keselj, V.: Author verification using common n-gram profiles of text documents. In: Proceedings of COLING 2014, the International Conference on Computational Linguistics: Technical Papers, Dublin, Ireland, pp. 387–397. Dublin City University and Association for Computational Linguistics, August 2014
3. Stamatatos, E., Daelemans, W., Verhoeven, B., Potthast, M., Stein, B., Juola, P., Sanchez-Perez, M.A., Barrn-Cedeo, A.: Overview of the author identification task at PAN 2013. In: CLEF (Working Notes). CEUR Workshop Proceedings, vol. 1179. CEUR-WS.org (2013)
4. Schmid, M.R., Iqbal, F., Fung, B.C.: E-mail authorship attribution using customized associative classification. *Digit. Investig.* **14**(S1), S116–S126 (2015)
5. Schler, J., Koppel, M., Argamon, S., Pennebaker, J.W.: Effects of age and gender on blogging. In: AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs, vol. 6, pp. 199–205 (2006)
6. Stamatatos, E., Daelemans, W., Verhoeven, B., Potthast, M., Stein, B., Juola, P., Sanchez-Perez, M., Barrón-Cedeño, A.: Overview of the author identification task at PAN 2014. In: CLEF (Working Notes). CEUR Workshop Proceedings, vol. 1180, pp. 877–897. CEUR-WS.org (2014)
7. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of Workshop at the International Conference on Learning Representations, Scottsdale, USA (2013)



N-Gram Based Approach for Automatic Prediction of Essay Rubric Marks

Magdalena Jankowska^(✉), Colin Conrad, Jabez Harris, and Vlado Kešelj

Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada
{magdalena.jankowska,colin.conrad,jay.harris,vlado.keselj}@dal.ca

Abstract. Automatic Essay Scoring, applied to the prediction of grades for dimensions of a scoring rubric, can provide automatic detailed feedback on students' written assignments. We apply a character and word n-gram based technique proposed originally for authorship identification—Common N-Gram (CNG) classifier—to this task. We report promising results for the rubric mark prediction for essays by CNG, and perform analysis of suitability of different types of n-grams for the task.

Keywords: Automatic Essay Scoring · Text classification
Character n-grams

1 Introduction

Essay writing is an important component of formal education. In order to improve their writing, students require effective and specific feedback on different writing dimensions. Automated Essay Scoring (AES) is an ongoing area of research that seeks to expedite the feedback process. Human-generated scores are often treated as a gold standard and automated tools are built to learn from and predict these scores [1]. Recently, the subject has received renewed attention, due to the Automated Student Assessment Prize (ASAP) Competition data¹ [2].

Detailed rubrics are often used to give specific qualitative feedback about ways of improving writing, based among others on stylistic elements contained in the text itself. The task of rubric prediction can be treated as a way of generating detailed feedback for students on the different elements of writing, based on the characteristics typical of a student's performance. This is fundamentally similar to the task of the automatic identification of an author of a text. In this paper, we explore the application of the Common N-Gram (CNG) classifier [3], which is frequently used in authorship analysis, to the task of rubric value prediction. Using the ASAP data, we compare performance of the CNG algorithm to two other, popular text classifiers: linear Support Vector Machines (SVM) with stochastic gradient descent (SGD) learning and Multinomial Naïve Bayes algorithm (NB). We also compare the results to the reference of the inter-rater agreement between human raters.

¹ <https://www.kaggle.com/c/asap-aes>.

2 Related Work

Automated Essay Scoring is not a new concept. The earliest AES systems date back to the 1960's. One of the most notable commercial AES systems is e-Rater [4], which is used to evaluate second-language English capabilities on the Test of English as a Foreign Language (TOEFL) and essay rating of the Graduate Record Examination (GRE). Other innovations such as Gradescope [5] are now enabling a transformation in education delivery. Much of the work on AES treats automated grading as a supervised learning problem, using features related to content (e.g., through latent semantic analysis [6]) or style. The e-Rater system, for instance, performs feature extraction related to specific writing domains such as *grammar*, *usage*, *mechanics* and *style*. This paper likewise treats AES as a supervised classification problem.

The Common N-Gram classifier (CNG) technique explored in this paper is conventionally used in the context of authorship attribution, which is the task of detecting who among candidate authors wrote a considered text. CNG is based on a similarity measure between documents relying on differences between frequencies of the character n-grams. The CNG similarity, or its variants, has been successfully applied to tasks related to authorship analysis of texts and related applications [7,8]. It has also been explored in the context of AES [9] but has not been evaluated using a popular dataset.

3 Methodology

Prediction of the rubric grade is performed using supervised classification and is performed separately for each detailed rubric dimension, such as “Style”, “Organization”, etc., with classifiers trained using marks given by human raters. We applied three classification algorithms: the CNG classifier, linear SVM with SGD learning, and NB. The representation of documents is based on n-grams of characters or words. We also tested “stemmed word” n-grams.

A representation of a document used by CNG is a “profile”—a list of the most frequent n-grams of a particular type, coupled with their frequency normalized by the text length. The total number of unique n-grams in a document was set as the length of a profile. Training data for a class is represented by CNG as a single class document, created by concatenating all training documents from the class. For SVM and NB, we used a typical bag-of-n-grams representation (using a particular n-gram type as features), with either raw counts or *tfidf* (term frequency—invert document frequency) scores as weights. To mitigate the effect of unbalanced training data for SVM and NB, we experimented with random upsampling of minority classes. CNG is known to be especially sensitive to unbalanced training data [10], but its nature prevents the use of upsampling of training documents; for CNG we addressed the problem by truncating all class profiles to the same, maximum possible, length.

4 Experiments

4.1 Data

Experiments were performed on essays of three ASAP datasets: set 2, set 7 and set 8. These three sets of essays are chosen for experiments (out of eight sets available in the dataset), because for these sets marks assigned to individual dimensions of the evaluation guideline rubric (such as “Style”, “Organization”, etc.) are available. Table 1 presents information about the essay sets.

Table 1. Information about ASAP datasets used in experiments

Name	set2	set7	set8
Grade level	10th	7th	10th
# of essays	1800	1569	723
Average # of words	381	168	606
Rubric dimensions	“Writing Applications” “Language Conventions”	“Ideas” “Organization” “Style” “Conventions”	“Ideas and Content” “Organization” “Voice” “Word Choice” “Sentence Fluency” “Conventions”

For each rubric dimension, grades from two raters are available. Classification is performed for each dimension and each rater separately, and so there are 24 classification tasks in total. The number of classes (different marks) is 4 for all sets and dimensions except for “Writing Applications” in set 2, in which the number of classes is 6. For set 8, our classification is for 4 classes, but the original scale of marks has 6 marks: from 1 to 6. We combined for this set mark 1 with mark 2, and mark 6 with mark 5 because the extreme marks are very rare.

4.2 Experimental Settings

We performed experiments for 13 feature sets: character n-grams of the length from 2 to 10, and word and stemmed word n-grams of the length of 1 and 2. For each of 13 feature sets, one classification by CNG was performed while for SVM and NB each, four classifications were performed, for the combinations of two types of n-gram weights and two types of processing of unbalanced training data (upsampling and no upsampling). The performance measure used in the experiments is Quadratic Weighted Kappa (QWK). Testing was performed using 5-fold stratified cross-validation. For SVM with SGD learning and NB we used implementations of the classifiers from Scikit-learn Python library [11]. Processing of documents in order to extract “stemmed word” n-grams was performed using Snowball Stemmer and the English stop words corpus from the NLTK platform [12]. The package imbalanced-learn [13] was utilized to perform the upsampling of training data.

4.3 Results and Discussion

Table 2 presents the best result among all trained classifiers for each classification task (a row corresponds to a rubric dimension). For each rubric dimension, the QWK of the inter-rater agreement between rater1 and rater2 is also stated as a reference. “diff” is a relative difference between an inter-rater QWK and a given classifier QWK, as a percentage of the inter-rater QWK. Classifier QWK values denoted by the symbol * (respectively †, ‡) are statistically significantly higher ($p < 0.05$) than the best in the task result of the CNG classifier (respectively SVM with tfidf and upsampling, NB with counts and upsampling).

We can observe that the algorithms which achieved the best overall results for a given task were CNG (11 tasks), SVM with tfidf scores and upsampling of training data (10 tasks) and NB with counts (3 tasks). It can be noted that the best results of classifiers often do not differ in a statistically significant way.² Finally, we can note that the best performance achieved on particular classification tasks varies substantially when compared to the agreement of the human raters. Set 2 demonstrates the highest agreement between the human raters, and demonstrates the highest discrepancy between the raters and the classifiers. On set 8, by contrast, the results of classifiers are relatively close to the inter-raters QWK (especially for rater1, on three dimensions, the QWK of CNG differs from the inter-rater QWK by less than 5%). These results indicate that CNG is a promising method for the problem.

We performed feature analysis for four selected classifiers. For each of the classifiers, we ranked the n-gram types by a number of tasks (out of 24), in which

Table 2. Best QWK results among all trained classifiers for each task

inter-rater QWK	rater1			rater2		
	Best classifier	QWK	diff	Best classifier	QWK	diff
set2						
0.814	SVM tfidf upsam. char6	0.567‡	30%	SVM tfidf upsam. char6	0.571‡	30%
0.802	SVM tfidf upsam. char5	0.570	29%	SVM tfidf upsam. char4	0.567	29%
set7						
0.695	CNG char4	0.657	6%	NB counts char3	0.628	10%
0.577	SVM tfidf upsam. char6	0.508*	12%	SVM tfidf upsam. char6	0.515*	11%
0.544	SVM tfidf upsam. char5	0.480	12%	SVM tfidf upsam. char5	0.493*	9%
0.567	SVM tfidf upsam. char4	0.428‡	25%	SVM tfidf upsam. char5	0.486‡	14%
set8						
0.523	NB counts upsam. char3	0.482†	8%	CNG char5	0.394	25%
0.533	NB counts char3	0.455†	15%	CNG char5	0.377	29%
0.456	CNG char4	0.440	4%	CNG word1	0.377	17%
0.477	CNG char4	0.493	-3%	CNG char5	0.431†	10%
0.498	CNG char4	0.489	2%	CNG char4	0.459†	8%
0.532	CNG char4	0.454†	15%	CNG char4	0.436†	18%

² A correction for multiple hypothesis testing was not applied; it would be valuable if outperforming performance of any classifier was to be established.

Table 3. Six best performing features for selected classifiers

CNG		SVM tfidf upsam.		NB counts no samp.		NB counts upsam.	
feature	"#good"	feature	"#good"	feature	"#good"	feature	"#good"
char 4	24	char 4	20	char 3	22	char 3	23
char 5	23	char 5	18	char 2	19	char 4	19
char 6	19	char 3	18	char 4	7	word 1	18
word 1	12	char 6	17	word 1	3	char 2	17
char 7	11	word 1	13	stem 1	3	stem 1	12
char 8	9	stem 1	12	char 5	2	char 5	10

a given type was not statistically significantly worse than the best performing n-gram type in the task—we called this number “#good”. In Table 3, we report six best feature sets for each classifier, and denote as bold the ones for which “#good” is greater or equal to the half of the total number of tasks.

The analysis indicates that character 4-grams and 5-grams are the best features for CNG and SVM; character 6-grams and word unigrams are also well suited for these two classifiers. Character 3-grams perform well for SVM, and short character n-grams of the length 2, 3 and 4 perform especially well for NB.

5 Conclusion and Future Work

Promising results were obtained for rubric prediction based on character n-grams and word unigram representations, using CNG classifier, SVM with SGD learning with tfidf scores, and Naïve Bayes with raw counts, when compared to the inter-rater agreement between the scores of human raters. CNG algorithm, proposed originally for author identification, performed well compared to the other classifiers and shows promise for future study. Several methods of improving the performance of the prediction could be investigated in future work. They include combining different types of n-grams, either by using them together in document representation, or by an ensemble of classifiers based on different n-grams, as well as combining n-gram-based features with other types of features, such as parts of speech, detected spelling/grammar errors or presence of prescribed words. It would be worthwhile to investigate the relationship between the classifier performance and the type of rubric dimensions, as well as the impact of the level of inter-rater agreement. Future research could focus on investigating the role CNG and its similarity can play in complementing existing processes in AES tasks.

Acknowledgement. The project was supported by the NSERC Engage grant EGP/507291-2016 with industry partner, D2L Corporation. The authors would like to thank D2L members: Brian Cepuran, VP, D2L Labs and Rose Kocher, Director, Grant & Research Programs, for their guidance in the project and the feedback on the paper. The authors would also like to acknowledge a support from Killam Predoctoral Scholarship.

References

1. Shermis, M.D., Burstein, J.: *Handbook of Automated Essay Evaluation: Current Applications and New Directions*. Routledge, New York (2013)
2. Phandi, P., Chai, K.M.A., Ng, H.T.: Flexible domain adaptation for automated essay scoring using correlated linear regression. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 431–439 (2015)
3. Kešelj, V., Peng, F., Cercone, N., Thomas, C.: N-gram-based author profiles for authorship attribution. In: *Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING 2003*, Dalhousie University, Halifax, Nova Scotia, Canada, pp. 255–264, August 2003
4. Attali, Y., Burstein, J.: Automated essay scoring with e-rater® v. 2.0. ETS Res. Rep. Ser. **2004**(2) (2004)
5. Singh, A., Karayev, S., Gutowski, K., Abbeel, P.: Gradescope: a fast, flexible, and fair system for scalable assessment of handwritten work. In: *Proceedings of the Fourth 2017 ACM Conference on Learning@ Scale*, pp. 81–88. ACM (2017)
6. Foltz, P.W., Laham, D., Landauer, T.K.: Automated essay scoring: applications to educational technology. In: *EdMedia: World Conference on Educational Media and Technology*, pp. 939–944. Association for the Advancement of Computing in Education (AACE) (1999)
7. Juola, P.: Authorship attribution. *Found. Trends Inf. Retrieval*. **1**(3), 233–334 (2008)
8. Jankowska, M., Milios, E., Kešelj, V.: Author verification using common n-gram profiles of text documents. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin, Ireland, pp. 387–397. Dublin City University and Association for Computational Linguistics, August 2014
9. Doyle, J.: *Automatic evaluation of student essays using n-gram analysis techniques*. Master’s thesis, Dalhousie University (2007)
10. Stamatatos, E.: Author identification using imbalanced and limited training texts. In: *Proceeding of the 18th International Workshop on Database and Expert Systems Applications, DEXA 2007*, Regensburg, Germany, pp. 237–241, September 2007
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
12. Bird, S., Loper, E., Klein, E.: *Natural Language Processing with Python*. O’Reilly Media Inc., Sebastopol (2009)
13. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: a Python toolbox to tackle the curse of imbalanced datasets in machine learning. *J. Mach. Learn. Res.* **18**(17), 1–5 (2017)



Matching Résumés to Job Descriptions with Stacked Models

Peng Xu and Denilson Barbosa^(✉)

Department of Computing Science, University of Alberta, Edmonton, Canada
{pxu4,denilson}@ualberta.ca

Abstract. We describe a method for matching résumés to job descriptions provided by employers, and evaluate it on real data from a Canadian company specialized in e-recruitment. We model the task as a classifying each résumé as suitable or not for a follow up interview. We evaluate the methods on two datasets with approximately 1,500 real job descriptions and approximately 70,000 résumés, from two important industry sectors, considering several models individually and also stacked. Our stacked model shows high accuracy (often above 0.8) and consistently outperforms standard methods, including neural networks.

1 Introduction

Recruitment is critical to the success of all organizations, regardless of sector and size: not only they seek to attract the best talent but also minimize high costs associated with hiring unsuitable candidates. Web-based e-recruitment systems, commonly used today, date back to the late 1990s [5], and come in two flavors: passive online job posting boards, and interactive systems that collect all necessary documents and mediate the communication between recruiters and candidates. Interestingly, e-recruiting systems influence how job seekers perceive the companies offering the jobs [11]: a poorly designed system can drive a good candidate away from the company posting the job.

We describe the results of joint research with a Canadian e-recruiting company aimed at using AI for screening candidates for further interview by recruiters. This triage step is difficult even for humans, given the continuous diversification of the economy and the resulting specialization of candidates into narrower fields. Departing from previous work, we view the task as a binary classification problem. Based on techniques of natural language processing (NLP) and information retrieval (IR), we extract various useful features from the text information in the résumés and the job descriptions. We validate the effectiveness of these features and different classification models, and show that a two-level stacking scheme of the base models can achieve promising results.

1.1 Previous Work

There is previous work based on recommender systems [10], in which both job descriptions and résumés are represented as vectors (dimensions correspond to

qualifications, experience, and certifications), and matching is based on vector similarity. However, the “cold start” problem, arising from data sparsity, is particularly hard here: new skills and technologies arrive frequently leading to sparse vectors. Traditional Information Retrieval (IR) techniques have also been used, whereby one breaks job postings and résumés into *sections* (skills, experience, certifications, etc.), and uses IR-based scoring to rank candidates for jobs [12], e.g., with probabilistic Language Models [3]. Systems like PROSPECT [9] mine job-seeking related features from textual résumés and use ranking coupled with appropriate graphical user interface to help the recruiter examine the applicant pool in decreasing relevance for a given job posting.

2 Task and Method

Screening candidates for a job consists in predicting whether a résumé fits a job description sufficiently well to warrant a follow-up action on the part of the recruiter. We model the task as a binary classification problem. Our training data \mathcal{D} contains pairs of job descriptions and résumés, with the corresponding binary labels. The output is a prediction of the label for each résumé from the test dataset \mathcal{D}_t .

Preprocessing. Job descriptions and résumés contain both semistructured fields and textual fields. Semistructured fields are often itemized lists, indicating education, certifications, specific training, etc., while the text fields are often used for the cover letter and for expressing experience. We combine all text fields for each résumé into a single one, which we call the *observed* text in this paper. The text fields in the job descriptions are referred to as the *target* text. We normalize all text fields through: HTML tag removal, case folding, *stop-word* elimination, *stemming* and *lemmatization* [6] with the help of WordNet [7].

2.1 Features

The following features are extracted from résumés and job descriptions.

Basic features. These are descriptive features about the observed and target text, including the length of the text, and indicator features such as whether or not the resume has a cover letter, or the count of experiences the applicant has.

N-Gram Word Intersection Based Features. We generate N-grams for both observed and target text and calculate their intersection count and co-occurrence count as features.

N-Gram IR Scoring Features. We score each résumé and job description using the usual *TF*, *TFIDF* and Okapi BM25 scores used for document ranking [1, 8] at the N-gram level. Then, five statistical features are extracted based on the obtained score list: minimum, maximum, mean, median and standard deviation.

N-Gram LSI Based Features. Latent Semantic Indexing (LSI) [6] is an indexing and retrieval method based on singular value decomposition (SVD) to identify patterns in the relationships between terms and concepts contained in the text corpus. Four kinds of features are extracted based on LSI: (1) Apply truncated SVD on the N-gram TFIDF matrices of the whole corpus, then obtain the transformed feature vectors for the observed text; (2) Apply truncated SVD on the combined N-gram TFIDF matrices of the observed text and target text, then obtain the transformed feature vectors for the paired observed and target text; (3) Compute the cosine similarity between each pair of observed and target text with learned LSI feature vectors in (1); (4) Compute the cosine similarity between each pair of observed and target text with TFIDF matrices in (1).

2.2 Models

We try three different base models: gradient boosting trees [2], random forest [2] and densely-connected neural networks [4], which are known to have good performance on classification tasks with numerical features. In addition, all of them are able to give the probability of the predicted class naturally which is helpful for further stacking. The probability predictions of the three base models are stacked together as the features to the second-level model. We use gradient boosting trees as the second-level model and use both stacked features and extracted features. The details of how the stacking works is described next.

3 Experiments

Datasets and Experimental Setting. We evaluate our models on two datasets from two industry sectors: energy and services. Table 1 shows statistics about the training and test datasets used.

In the experiments, we make five different splits of the training set. The five splits consist of a training part $\mathcal{D}_{train}^{(i)}$ and a validation part $\mathcal{D}_{valid}^{(i)}$ each, and are used to tune the hyper-parameters for the base models. Due to the small size of our datasets, our validation part for each split is larger than 20% of the whole training set to help further stack the models. Except for this difference, our validation strategy is exactly the same as 5-fold cross-validation [2].

Table 1. Statistics about the datasets.

		Energy	Services
# of résumés	Training	37,677	31,013
	Testing	4,424	2,189
# of job descriptions	Training	783	860
	Testing	87	86
% of positive labels	Training	42.0	41.9
	Testing	40.8	40.0

The top 10 hyper-parameter settings for each base model are used to generate features for the stacked model. To prevent leaking label information to the model, we further split each validation part $\mathcal{D}_{valid}^{(i)}$ to a training part and a validation part to tune the hyper-parameters for the stacked model. The final stacked model is the average of the five models trained on $\mathcal{D}_{valid}^{(i)}, i = 1, \dots, 5$.

3.1 Results and Analysis

First, we investigate the importance of each feature set. The results are shown in Table 2. The base model is gradient boosting trees and we use the same hyper-parameter setting tuned on all extracted features denoted by **All** for all feature sets. In the table, **Basic** stands for basic features, **F1** for word intersection based features, **F2** for IR scoring features, and **F3** for LSI based features.

From the table, we have the following observation: (1) The first three rows indicate that **F3** has the largest share of contribution to our model and **F1** has the smallest. (2) **All - F1** performs the best on both datasets. (3) **All** tends to over-fit the data due to the feature set **F1** which makes its performance worse. This suggest that the features based on word intersection, which are commonly strong features in other NLP and IR applications, are not applicable in this task. As a result, we only use **All - F1** in the following experiments.

Table 3 shows the validation and testing accuracy on the two datasets with the corresponding tuned hyper-parameter settings on each model. We can observe that: (1) The validation accuracies are consistent with the testing accuracies of the time. In other words, the higher the validation accuracy is, the higher the testing accuracy will be. This observation strengthens the effectiveness of our validation strategy and suggest that our models do not over-fit the relatively small datasets. (2) Neural networks perform the worst for both of the datasets, which may be caused by the small size of our datasets, since neural models need a lot of training data due to their high complexities. (3) Stacked models consistently outperform all base models on both datasets. Overall, the results show that our stacking strategy is effective screening résumés.

Table 2. Performance comparison between models with different feature sets.

Model	Energy		Services	
	Valid Acc.	Test Acc.	Valid Acc.	Test Acc.
Basic + F1	0.767	0.705	0.632	0.655
Basic + F2	0.773	0.711	0.656	0.678
Basic + F3	0.817	0.784	0.722	0.787
All - F1	0.817	0.797	0.722	0.815
All - F2	0.816	0.794	0.720	0.805
All - F3	0.773	0.709	0.654	0.673
All	0.779	0.788	0.738	0.809

Table 3. Performance comparison between different models.

Model	Energy		Services	
	Valid Acc.	Test Acc.	Valid Acc.	Test Acc.
Gradient Boosting Trees	0.815	0.800	0.731	0.829
Random Forest	0.799	0.786	0.697	0.830
Neural Networks	0.773	0.775	0.664	0.695
Stacked Model	0.823	0.803	0.766	0.849

4 Conclusion

Many aspects of modern office activities are being increasingly automated, and this trend has started several decades ago. Recruitment is an aspect of workforce management that is bound to undergo profound changes in the near future, as jobs become ever more specialized and candidates obtain more and more specific skills. Matching job seekers to available positions is a task where AI can greatly help humans make faster, better, and more consistent decisions.

In this paper, we employ a stacked model for the task of screening candidates suitable for further inspection by a recruiter, supporting the business process of a successful Canadian recruitment agency. We extract standard features in natural language processing and information retrieval and compare the stacked model against established baselines. We report on an evaluation with real résumés and job descriptions from two industry sectors. Experimental results show that our stacked model consistently outperform all the base models on the both datasets. The achieved results on these two datasets are promising and can be helpful for job recommendation applications.

We identify several avenues for future work. For example, a large-scale evaluation of the effectiveness of our model is planned by deploying it live. Such an evaluation takes time and must be done carefully, through protocols such as AB testing, so as to reduce the effects of possible confounding factors. Another area for future work would be deploying the algorithm at Web-scale on cloud-based high-performance computing platforms which are popular nowadays. We envision the use of Deep Learning methods to allow for semantic translation of terms appearing in the résumé and the job description. While LSI and SVD can account for synonymy, it would be interesting to see if deep neural methods can fare better, especially on harder cases, e.g., where the job requires experience on a broad technology (say data analytics) and the résumé mentions expertise on a specific tool (say, R). Along the same lines, it would be interesting to train a neural language model on the résumés of the candidates that were hired with the intent of modeling so-called “soft skills” which are desirable by employers but not necessarily expressed in the job description. Our dataset has proven too small for us to achieve that goal.

Acknowledgments. This work was supported in part by an ENGAGE grant by the Natural Science and Engineering Research Council of Canada.

References

1. Büttcher, S., Clarke, C.L., Cormack, G.V.: *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press, Cambridge (2016)
2. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*. SSS. Springer, New York (2001). <https://doi.org/10.1007/978-0-387-21606-5>
3. Lafferty, J., Zhai, C.: Document language models, query models, and risk minimization for information retrieval. In: *ACM SIGIR Forum*, vol. 51, pp. 251–259. ACM (2017)
4. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
5. Lee, I.: An architecture for a next-generation holistic e-recruiting system. *Commun. ACM* **50**(7), 81–85 (2007)
6. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2008)
7. Miller, G.A.: WordNet: a lexical database for English. *Commun. ACM* **38**(11), 39–41 (1995)
8. Robertson, S., Zaragoza, H., et al.: The probabilistic relevance framework: BM25 and beyond. *Found. Trends® Inf. Retrieval* **3**(4), 333–389 (2009)
9. Singh, A., Rose, C., Visweswariah, K., Chenthamarakshan, V., Kambhatla, N.: PROSPECT: A system for screening candidates for recruitment. In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010*, pp. 659–668. ACM, New York (2010). <https://doi.org/10.1145/1871437.1871523>
10. Siting, Z., Wenxing, H., Ning, Z., Fan, Y.: Job recommender systems: a survey. In: *7th International Conference on Computer Science & Education (ICCSE) 2012*, pp. 920–924. IEEE (2012)
11. Thompson, L.F., Braddy, P.W., Wuensch, K.L.: E-recruitment and the benefits of organizational web appeal. *Comput. Hum. Behav.* **24**(5), 2384–2398 (2008)
12. Yi, X., Allan, J., Croft, W.B.: Matching resumes and jobs based on relevance models. In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 809–810. ACM (2007)



Towards a Comprehensive Evaluation of Recommenders: A Cognition-Based Approach

Alaa Alslaity^(✉) and Thomas Tran

University of Ottawa, Ottawa, Canada
Aals1005@uottawa.ca, ttran@eecs.uottawa.ca

Abstract. Evaluating Recommender Systems (RSs) is a challenging issue that is significantly magnified by the multifaceted properties of RSs, which makes it insufficient to use only one metric to evaluate recommenders. This challenge necessitates the need for a unified evaluation model that comprehensively assesses multiple aspects of the recommender. This position paper proposes a cognition-based comprehensive evaluation to evaluate the main activities of RSs. We innovated the proposed model based on the cognitive dimension of Bloom's taxonomy, a widely used model for classifying learning objectives in the teaching area. We created a phase-wise mapping between RSs and Bloom's taxonomy to come up with an overall evaluation for recommenders. Based on these connections, we believe that the proposed evaluation model would have the potential to support the decision of selecting the most appropriate recommender systems by giving a benchmarked score for different aspects of RSs.

Keywords: Recommender systems · Recommendation evaluation
Evaluation model · Cognition-based · Bloom's taxonomy · Learning objectives

1 Introduction

Several evaluation approaches and metrics have been adopted to facilitate the comparison of recommender systems [2, 3]. RSs were usually evaluated using only one or two metrics [4, 5]. However, using only one metric is considered insufficient due to the multifaceted properties of RSs. Also, there are different properties of RSs that may affect its acceptance by users [1]. Therefore, considering multiple metrics becomes an essential need to evaluate the multi-faceted properties of RSs.

To the best of our knowledge, there is a lack of an evaluation model that comprehends multiple metrics to produce an overall evaluation result. To bridge this gap, we propose the **Comprehensive Performance (ComPer)**, a novel cognition-based evaluation model that is built based on the Bloom's taxonomy (a well-known hierarchy for learning objectives). The essence of *ComPer* is evaluating RSs based on their cognitive abilities by inferring mappings between the main activities of an RS and the cognitive dimension of Bloom's taxonomy. Unlike other evaluation approaches, *ComPer* takes into consideration multiple evaluation metrics. With this diversity of

evaluation metrics, we believe that *ComPer* would have the potential to support the decision of selecting the most appropriate recommender systems.

This paper is organized as follows; Sect. 2 describes the proposed model. Section 3 presents the evaluation methodology. Finally, Sect. 4 concludes the paper.

2 Proposed Model

This section introduces Bloom's taxonomy, illustrates the mapping between RSs and this taxonomy, and shows how our model can be used to evaluate a recommender.

Bloom's Taxonomy. Bloom's taxonomy is a hierarchical model that is used to classify the intended learning objectives and skills [7]. It is divided into three domains: cognitive, affective and sensory. The cognitive domain (the one of particular interest of this paper) underlines intellectual outcomes. This domain is further divided into six categories, as follows: (1) *Remember*: the ability to retrieve knowledge from memory. (2) *Understand*: the ability to determine meaning from all kinds of messages. (3) *Apply*: the ability to implement or execute a procedure in a given situation. (4) *Analyze*: the ability to break material into constituent parts and detect how the parts are related to each other. (5) *Evaluate*: the ability of making judgments. (6) *Create*: the ability to put elements together, and recognize elements into a new pattern.

Main Phases of Recommender Systems. According to Isinkaye et al. [8], recommendation process consists of three phases: *Information collection*, "collecting relevant information of users to generate a user profile.", *Learning* "applying a learning algorithm to filter and exploit the user's features from the feedback gathered in information collection phase.", and finally, the prediction/recommendation phase which is "recommending or predicting what kinds of items the user may prefer."

2.1 Mappings Recommender Systems and Bloom's Taxonomy

This section describes the mapping between RSs and the Bloom's taxonomy by giving an example that analogizes RSs with human beings who have cognitive skills, followed by a mapping between Bloom's learning objectives and both of main phases of RS and RSs evaluation metrics.

Recommender Systems and Human Beings (An Analogy). John is a fresh salesman working at ABC clothing store. When he started his job, John was asking each customer about what kind of clothes he/she wants, what is his/her style, which color he/she prefers, etc. After that, John started recognizing customers and became aware of their preferences. Awhile after, he became able to expect their needs, so he started suggesting clothes for his customers. During his work, John was trying to remember his customers and understand their needs to **collect** correct **information** about their purchasing trends. Thereafter, he applied the already collected information about customers in an attempt to analyze their attitudes. Such analysis leads John to **learn** about his customers' preferences. By the end of this learning process, John reached a level

where he was able to link pieces of information together and creates patterns for each customer. Therefore, he was able to **predict** what may attract a customer and to **recommend** the best product suits that customer.

From this example, we notice that human beings can perform the tasks of a recommender system (i.e., collect information, learn, and predict/recommend). Hence, we can say that a human being is analogous to a recommender system. Our vision in this paper is to extend this analogy to make it valid in the other direction as well. That is, we are inspired by the idea that an RS could be analogous to human beings and could have the comprehension skills of humans. The essence of our vision is that such “cognitive” systems not only can perform the conventional recommendation process but also, recommend with consciousness. To conceptualize the vision of cognition-based RSs and correlate it to humans’, we need to create mappings between the two worlds; RSs and humans. To do so, an important research question that needs to be considered is: how can we match the recommendation tasks of RSs with the learning process of a human? The next subsection addresses this question.

Mappings RS Phases with Bloom’s Taxonomy. Based on the above example and the definitions of RSs phases and Bloom’s learning objectives, we match both concepts by the following mappings: to *collect information*, an RS should be able to retrieve information (i.e., *remember*) and construct meanings from different types of messages (i.e., *understand*). To *learn*, an RS should be able to *apply* prior knowledge in different situations and *analyze* the interrelationships between different components of a system. Finally, an RS’s *recommendation/prediction* depends on its ability to make a judgment (i.e., *evaluate*), and reorganize elements into new structures (i.e., *create*). A visualization of this connection is presented in Fig. 1.

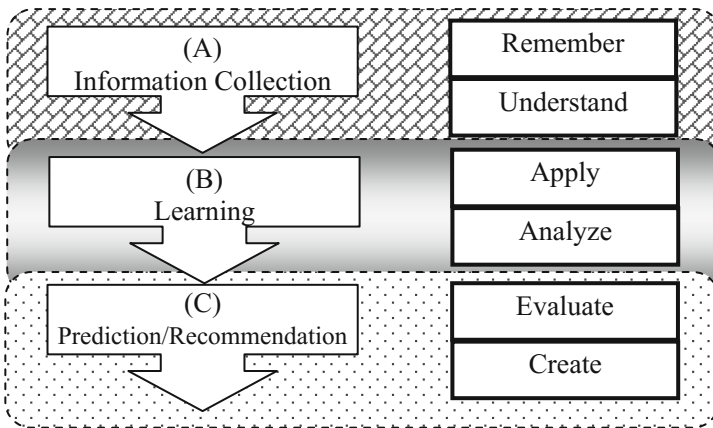


Fig. 1. Matching RS main phases with learning objectives

Since both, RSs and humans, are correlated, we argue that the six learning objectives of humans can assist in evaluating the recognition abilities of RSs; that is, they can be used as evaluation dimensions for RSs. However, these metrics might seem

unfamiliar in the RS context. To mitigate this issue, we inferred correlations between learning objectives and the most popular evaluation metrics, as described next.

Mappings RS Evaluation Metrics with Bloom’s Taxonomy. A vast variety of evaluation metrics have been introduced to the recommendation field [1, 9], not all metrics are of the same importance. To find out the most popular dimensions, we surveyed 135 articles published in the last decade in the recommendation field. The results show that *Correctness* is the most popular property. On the other hand, some properties, like *Confidence* and *Stability*, have not been considered by any of the articles. Based on these results, we will consider the most commonly used dimensions, which are: *Correctness*, *Coverage*, *Diversity*, *Utility*, *Usability*, *Robustness*, and *Scalability*. However, some of these measures, such as *usability* and *utility*, do not give any indication of systems’ cognitive skills. Thus, we omitted them from our model.

Based on the definitions of RS’s evaluation metrics as presented in [1, 9] and the definitions of Bloom’s learning objectives mentioned previously, we correlated both concepts as summarized in Table 1. Rows show metrics while columns show learning objectives. Cells represent the correlation between each learning objective and the corresponding dimension, where **S**, **P**, and **N** indicate a **Strong**, **Partial**, and **No** correlation, respectively. To clarify, consider the first column; as mentioned previously, the objective “*remember*” is the ability to retrieve knowledge from memory, and the evaluation metric “*coverage*” represents the proportion of available information for which recommendations can be made [9]. Therefore, the more the information available, the better the remembering skill is. Given these two pieces of information, we infer that “*coverage*” and “*remember*” are strongly correlated. Following the same rationale, we inferred other correlations.

Table 1. Mapping learning objectives with RS metrics

	Remember	Understand	Apply	Analyze	Evaluate	Create
Correctness	P	P	P	S	S	S
Coverage	S	P	S	P	S	S
Diversity	S	S	P	S	P	S
Robustness	P	P	S	S	S	P
Scalability	S	N	S	P	P	N

3 Evaluation Methodology

The evaluation process depends mainly on the correlation presented in Table 1. The quantitative values of the three qualitative values (S, P, and N) are (1, 0.5, and 0), respectively. The evaluation process starts by evaluating each metric separately; then these values are normalized to get a consistent scale for all metrics. After that, each cell of Table 1 is replaced by a value equal to the corresponding metric multiplied by the correlation value. The total of each objective (i.e., each column) and the total of all objectives are calculated next. We calculate *ComPer* by Eq. (1), where O is the set of all objectives (i.e., the six objectives of Bloom), M is the set of the values of all metrics, and $corr_{om}$ is the correlation value between objective o and metric m .

$$ComPer = \sum_{o \in O} \sum_{m \in M} (m * corr_{om}) \quad (1)$$

It is noteworthy that *ComPer* depends mainly on metrics that already exist in the literature. The rationale behind this is that these metrics have already been validated, and they are widely used. Instead of combining these values arbitrarily or presenting them separately, *ComPer* merges them in an innovative way that gives different weights for different metrics based on their effects on the performance, and it comes up with a single value, which simplifies the comparison between recommenders. That is, *ComPer* evaluates multiple aspects using a single value.

Further work is required to address some issues, including: the differences between metrics in terms of their ranges of values and their judgment (i.e., is higher value better than lower?). Also, different measures may be used to evaluate the same metric based on the recommendation task. In addition, trade-offs exist between metrics (e.g., similarity and diversity) and between measures of the same metric (e.g., precision and Recall) [10, 11]. All these issues should be considered when implementing the model.

Threats to Validity. In our work, there is a threat to internal validity, which is represented by having the authors construct the mapping between Bloom's taxonomy and RSs based on their definitions (i.e., they are not validated).

4 Conclusion

In the last two decades, a variety of recommendation systems have been produced. Accordingly, an evident question is how we can choose the most appropriate RS from this vast variety? One of the recently emerged answers is that comparing RSs requires a common method of evaluation [11]. In this position paper, we proposed the **Comprehensive Performance (*ComPer*)** model, a cognitive-based evaluation model that is based on Bloom's taxonomy. *ComPer* originates mapping between RS and cognitive learning objectives of Bloom's taxonomy. Based on the correlations between RS's properties and learning objectives, we believe that we can provide a comprehensively unified evaluation metric that provides benchmarked scores for RSs. Currently, we are working on validating and proving the applicability of the proposed model.

References

1. Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.) *Recommender Systems Handbook*, pp. 257–297. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_8
2. Pu, P., Chen, L., Hu, R.: A user-centric evaluation framework for recommender systems. In: *Proceedings of the Fifth ACM Conference on Recommender Systems*, pp. 157–164. ACM, October 2011
3. Seminario, C.E., Wilson, D.C.: Robustness and accuracy tradeoffs for recommender systems under attack. In: *FLAIRS Conference* (2012)

4. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **17**(6), 734–749 (2005)
5. Kowald, D., Lex, E.: Evaluating tag recommender algorithms in real-world folksonomies: a comparative study. In: *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 265–268. ACM (2015)
6. Said, A., Tikk, D., Stumpf, K., Shi, Y., Larson, M., Cremonesi, P.: Recommender systems evaluation: a 3D benchmark. In: *RUE@ RecSys*, pp. 21–23 (2012)
7. Karthwohl, D.R., Anderson, W.: *A revision of Bloom’s taxonomy: an overview theory into practice*. The Ohio State University (2002)
8. Isinkaye, F.O., Folajimi, Y.O., Ojokoh, B.A.: Recommendation systems: principles, methods and evaluation. *Egypt. Inform. J.* **16**(3), 261–273 (2015)
9. Avazpour, I., Pitakrat, T., Grunske, L., Grundy, J.: Dimensions and metrics for evaluating recommendation systems. In: Robillard, M., Maalej, W., Walker, R., Zimmermann, T. (eds.) *Recommendation Systems in Software Engineering*, pp. 245–273. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-45135-5_10
10. Salfner, F., Lenk, M., Malek, M.: A survey of online failure prediction methods. *ACM Comput. Surv. (CSUR)* **42**(3), 10 (2010)
11. Smyth, B., McClave, P.: Similarity vs. diversity. In: Aha, D.W., Watson, I. (eds.) *ICCBR 2001. LNCS (LNAI)*, vol. 2080, pp. 347–361. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44593-5_25



A Sentence-Level Sparse Gamma Topic Model for Sentiment Analysis

Tao Chen^(✉) and Jeffrey Parsons

Faculty of Business Administration, Memorial University of Newfoundland,
St. John's, NL, Canada
{tao.chen, jeffreyp}@mun.ca

Abstract. Online consumer reviews have become an essential source of information for understanding markets and customer preferences. This research introduces a novel topic model to identify product attributes and sentiments toward them at the sentence level. The model uses a recursive definition of topic distribution in a sentence to avoid the problem of over-parametrization in topic models. The introduction of the inference network enables the utilization of rich features in the content to drive the identification of sentiments, in contrast with other multi-aspect sentiment analysis models that rely on single words. The sentence topic model has a superior performance in producing coherent topics, and the sentence topic-sentiment model outperforms the existing model on the task of predicting product attribute rating.

1 Introduction

Advances in information technology and the popularity of social networks have led to widespread customer sharing of product and service experiences. Such customer-generated online reviews provide valuable information about markets and customer preferences and become a de facto “sales assistant” to help customers to identify products fit their needs [3]. Many studies have demonstrated the usefulness of online review data in revealing customer preferences towards products [4]. Though these studies have shown online reviews can be valuable, they have simplified the rich text content of online reviews into, for example, overall rating, disregarding various product attributes discussed in the text content and the possibility that certain product attributes are perceived differently from overall sentiment and helpfulness. To address the limitation, we examine the text content of online reviews closely to identify product attributes and the associated sentiments, from which to derive customer preferences towards product attributes.

In the literature, the problem of sentiment analysis on product attributes is usually termed as multi-aspect sentiment analysis [9]. Multi-aspect sentiment analysis inherently involves two tasks: opinion aspect extraction and sentiment analysis on the extracted opinion aspects. The statistical topic model approach that conducts opinion aspect extraction and sentiment analysis at the same time is often used in accomplishing the task. Many studies extend LDA [2] to

accomplish the task. Though such models are useful, they rely on the assumption that the words are independent given the hidden topics and sentiment variables. The presumption may work well in product attributes discovery but may fail in sentiment analysis as complex or idiomatic expressions can convey the sentiment without using any sentiment explicit adjectives. The above models are less likely to capture the sentiment expressed in such expressions, for example, “when pigs fly”. Our model utilizes not only the topic information in the topic model but also the features exhibited in the local context, for example, examining whether “when pigs fly” exists in the sentence, to understand the sentiment.

Meanwhile, such topic models are less efficient when applied to online customer reviews for multi-aspect sentiment analysis as customer reviews are shorter and often have one sentence talking about one product attribute, and the next sentence talking about another product attribute. Multi-aspect sentiment analysis requires understanding the product attribute and sentiment discussed in a sentence rather than the overall product attribute distribution of the whole review as the product attributes change with the sentences and the sentiments change as well. The overall product attribute distribution cannot distinguish the sentiment difference between different product attributes. One approach is to model the topic distributions of sliding windows of sentences and use the topic distribution from one sliding window that covers the sentence to generate the words in the sentence [15]. However, such an approach suffers from over-parametrization and leads to low-quality topics [14]. Our approach also models the topics and sentiments at the sentence level but uses a recursive way of defining the topic distribution of a sentence without significantly increasing the number of variables, and effectively avoid over-parametrization.

Our research introduces a novel topic model to address these problems. We model the topics and sentiments at the sentence level of a review. The model recursively defines a topic distribution of a sentence without significantly increasing the number of variables and produces coherent topics. It aggregates the sentiments from the sentence to predict the overall rating and uses the inference network to integrate language features and allow rich representations of sentiments.

2 Sentence-Level Topic and Sentiment Analysis Model

The model is developed upon the sparse Gamma model [10] and can capture sparse factors suitable for modelling topics. In the model, the topic weight of the current sentence is a weighted sum of the topic weights of the preceding sentences and the overall topic weight of the review. The generative story of a document d with rating r in the model is as follows,

- Generate topic word matrix W such that each entry $W_{w,k} \sim \text{Gamma}(\alpha_w, \beta_w)$
- For each document $d \in D$,
 - Generate a document topic weight θ_d where each entry $\theta_{d,k} \sim \text{Gamma}(\alpha_d, \beta_d)$
 - For each sentence s_t in the document,

1. Generate a context weight η_s where each entry $\eta_s \sim \text{Gamma}(\alpha_c, \beta_c)$
 2. The sentence topic weight $\theta_{s_t} = (\theta_d + \sum_c \eta_{s,c} \theta_{s_t-c}) / (\sum_c \eta_{s,c} + 1)$
 3. Generate a sentence sentiment weight r_{s_t} where $r_{s_t} \sim \text{Gamma}(\alpha_r, \beta_r)$
 4. For each word w in the vocabulary,
 - (a) Generate word occurrence $n_{w,s_t} \sim \text{Poisson}(W_w^T \theta_{s_t})$,
- Draw an overall rating r_d that $r_d \sim \text{Poisson}(\frac{\sum_t r_{s_t} \theta_{s_t}}{\sum_t \sum_k \theta_{s_t,k}})$

The hyper-parameters α_w and α_c of the prior Gamma distribution is set to be less than 1 for sparsity. The formulation of the model is given as follows:

$$\begin{aligned}
 P(D, r | \alpha, \beta) &= \int dW P(W | \alpha_w, \beta_w) \\
 &\times \prod_{d \in D} \int d\theta_d P(\theta_d | \alpha_d, \beta_d) P(r_d | \theta_{s_t \in d}, r_{s_t \in d}) \\
 &\times \prod_{s_t \in d} \int d\eta_{s_t} dr_{s_t} P(\eta_{s_t} | \alpha_c, \beta_c) P(r_{s_t} | \alpha_r, \beta_r) \\
 &\times \prod_{w \in s_t} P(n_{w,s_t} | W, \theta_{s_t})
 \end{aligned}$$

The use of a context weight instead of a full topic weight for each sentence significantly reduces the number of variables and thus avoids over-parametrization. The formulation of the sentence topic weight emphasizes local context while keeping the topic weight of the sentence from drifting too far from the overall topic weight. It is different from the standard topic model in which the topic assignment of the current word is determined by the topic assignment of all words in the document, and all words contribute equally. The sentence topic distributions before the first sentence are randomly sampled from $\text{Gamma}(\alpha_d, \beta_d)$ for each document and are normalized to a vector summed to 1.

The sentiment weight r_s is a regression weight on sentences towards overall rating and captures the sentiment of product attributes conveyed in the sentence. The rating of a product attribute k in the review is defined as follows,

$$r_{d,k} = \mathbb{E} \left[\frac{\sum_{s_t} r_{s_t,k} \theta_{s_t,k}}{\sum_t \theta_{s_t,k}} \right] \quad (1)$$

The underlying assumption is that the most discussed product attribute contributes more to the overall rating and that negative sentiment expressions are more likely exhibited in low rating reviews and positive sentiment expressions are more likely exhibited in high rating reviews. The sentiment weight r_s is associated with an inference network using Long Short-Term Memory (LSTM) network [5]. The inference network is to use a neural network to approximate the posterior distribution of a generative model [11]. The input of the network is a feature vector of a review. A feature function consisting of n-gram features and sentiment lexicons applies to each review to produce the feature vectors. The output is the parameters of the posterior distributions of r_s . With the inference network, the content of the review determines the sentiment weight: more

positive sentiment features are more likely to produce high ratings, and more negative sentiment features are more likely to produce low ratings. The design presents an advantage over the existing multi-aspect sentiment analysis models. In the implementation, dropout [13] and batch normalization (BN) [7] are used in the input and output layer and are essential to enable the model.

3 Model Evaluation

We examine the sentence-level topic model for sentiment analysis from two perspectives: (1) capability of extracting coherent product attributes, and (2) capability of predicting product attribute rating. We use two review datasets for the evaluation: the Yelp review data [18] and the TripAdvisor hotel review data [17]. The Yelp dataset consists of over one million restaurant reviews with overall ratings. The TripAdvisor data consists of 50,000 hotel reviews with overall ratings along with ratings for seven pre-defined product attributes (including “Value”, “Room”, “Location”, “Cleanliness”, “Check in/front desk”, “Service”, and “Business Service”). Both datasets use the 5-star rating scale. For both data sets, we remove the non-English reviews and the reviews with less than four sentences. We exclude stop words and use TF/IDF to select the top 10,000 words as the vocabulary, but we do not exclude stop words in the feature function that produces input to the LSTM inference network. For both data sets, we use 50 topics for modelling product attributes and the number of hidden units in the LSTM inference network is set to 50.

The capability of extracting coherent product attributes is evaluated quantitatively using perplexity and topic coherence measures. We use the Yelp dataset to extract product attributes and the English Wikipedia dataset as the reference corpus in computing topic coherence. The English Wikipedia dataset consists of 9,611,451 documents. The first 100,000 reviews in the Yelp dataset are split into the training set and the testing set with a 50-50 ratio. The perplexity measure has been widely used to evaluate topic models [2, 6]. It measures the average number of word choices at each position in a review and the lower number of choices suggests better modelling capability. The topic coherence, in particular, normalized pointwise mutual information (NPMI), informs the coherence of identified topics and closely matches human judgement in evaluating topic quality [8]. The topic coherence is calculated with the top 20 topic words. We compare our model against the standard LDA, the CTM [1], the NVLDA [12], and the standard Sparse Gamma model [10]. The result is summarized in Table 1. Compared to the closely related Sparse Gamma model, our model produces better perplexity and topic coherence. The use of context captures topic shifting between sentences, is more flexible in modelling the data, and avoids the problem of over-parameterization. Though the CTM produces better perplexity than ours, our model outperforms the CTM significantly in producing coherent topics. In the experiment, we notice that the models with a normal distribution as the topic prior, for example, CTM and NVLDA, does not produce the same quality of topics as the models with a Gamma-based topic prior (a Dirichlet distribution

Table 1. Perplexity and topic coherence

	Perplexity (lower is better)	Topic coherence (higher is better)
LDA	1719	0.26
CTM	1453	0.13
NVLDA	1718	0.11
SGM	1652	0.28
STM	1603	0.29

Table 2. Product attribute rating prediction comparison

	RMSE (lower is better)	Pearson correlation (higher is better)
Wang and Ester [16]	0.384	0.854
Joint sentence model	0.305	0.882

can be constructed from Gamma distributions). It suggests that Gamma-based distributions are more suitable for modelling sparse factors.

The evaluation of predicting product attribute rating uses the TripAdvisor dataset. The overall rating is known to the model to predict a set of pre-defined product attribute rating. Root mean square error (RMSE) and Pearson correlation are used in the evaluation. Pearson correlation measures the correlation between the predicted product attribute ratings and the ground-truth and examines how well the predicted ratings preserve the relative order within a review. In the experiment, seven topics among 50 topics are assigned to the pre-defined product attributes using the keywords provided in [16]. The result is compared against the result in [16]. The result is listed in Table 2. Our model outperforms the other topic-sentiment analysis model in predicting product attribute rating significantly. The use of inference network allows the rich representation of text data and improves the prediction performance.

4 Conclusion

In the research, we introduce a novel natural language processing topic model that provides a fine-grained analysis of sentiments toward product attributes. We model product attributes discussed in a sentence, instead of an overall product attributes distribution. The identification of sentiments is driven by rich features in the content of the review through the inference network, different from other topic-sentiment analysis models that rely on single words. Our evaluation demonstrates the model’s promise in analyzing online reviews to identify interesting product attributes and associated sentiment. Our approach can help business to improve the understanding of products and customers in the market. Further, its capability makes it possible to develop a personalized online review

that can better information relevant to the customer. Such a personalized online review may assist customers in using online reviews for better decision making.

References

1. Blei, D.M., Lafferty, J.D.: A correlated topic model of science. *Ann. Appl. Stat.* **1**, 17–35 (2007)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
3. Chen, Y., Xie, J.: Online consumer review: word-of-mouth as a new element of marketing communication mix. *Manage. Sci.* **54**(3), 477–491 (2008)
4. Chevalier, J.A., Mayzlin, D.: The effect of word of mouth on sales: online book reviews. *J. Mark. Res.* **43**(3), 345–354 (2006)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
6. Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J.: Stochastic variational inference. *J. Mach. Learn. Res.* **14**(1), 1303–1347 (2013)
7. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*, pp. 448–456 (2015)
8. Lau, J.H., Newman, D., Baldwin, T.: Machine reading tea leaves: automatically evaluating topic coherence and topic model quality. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 530–539 (2014)
9. Liu, B.: *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies, vol. 5(1), pp. 1–167. Morgan & Claypool Publishers, San Rafael (2012)
10. Ranganath, R., Tang, L., Charlin, L., Blei, D.: Deep exponential families. In: *Artificial Intelligence and Statistics*, pp. 762–771 (2015)
11. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models (2014). arXiv preprint [arXiv:1401.4082](https://arxiv.org/abs/1401.4082)
12. Srivastava, A., Sutton, C.: Autoencoding variational inference for topic models (2017). arXiv preprint [arXiv:1703.01488](https://arxiv.org/abs/1703.01488)
13. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
14. Tang, J., Meng, Z., Nguyen, X., Mei, Q., Zhang, M.: Understanding the limiting factors of topic modeling via posterior contraction analysis. In: *International Conference on Machine Learning*, pp. 190–198 (2014)
15. Titov, I., McDonald, R.T.: A joint model of text and aspect ratings for sentiment summarization. In: *Proceedings of the Conference 46th Annual Meeting of the Association for Computational Linguistics*, vol. 8, pp. 308–316. Citeseer (2008)
16. Wang, H., Ester, M.: A sentiment-aligned topic model for product aspect rating prediction. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014)
17. Wang, H., Lu, Y., Zhai, C.: Latent aspect rating analysis without aspect keyword supervision. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 618–626 (2011)
18. Yelp: Yelp dataset challenge (2015)



Topic Detection and Document Similarity on Financial News

Saeede Sadat Asadi Kakhki^(✉) , Can Kavaklioglu, and Ayse Bener

Data Science Lab, Department of Mechanical and Industrial Engineering,
Ryerson University, Toronto, Canada
{saeedesadat.asadi, can.kavaklioglu, ayse.bener}@ryerson.ca

Abstract. Traders often rely on financial news to come up with predictions for stock price changes. Dealing with vast amount of news data makes it essential to use an automated methodology to identify the relevant news items for a given criteria. In this study we use Latent Dirichlet Allocation (LDA) to model the correlation of news items with stock price time series data. LDA model is trained with news items from a time window in the past and then the trained model is used to measure the similarity between the current news items and the news items used for training. Calculated similarity measure can be used as a predictor for switching points in the future. We tested our methodology using a collection of about 1,700,000 financial news items published between 2015-01-01 and 2015-12-31, and compared the results with various standard classification techniques. Our results indicate that use of LDA instead of standard classification techniques makes it possible to achieve the same level of performance by using a much smaller feature space.

Keywords: Latent Dirichlet Allocation · Variational bayes inference
Natural language processing · Dow Jones dataset · Classification

1 Introduction

Price movements in stock exchange are important for traders as they have direct effect on investment gain/loss [1]. Considering efficient-market hypothesis, all types of information about a company affect its stock price behaviour. One of the richest sources of information is the textual news data [2]. Although there is some debate in the literature on the value of news information, we believe that studies that show financial news is a good indicator of price changes in the market [3], outweigh the alternative point of view.

News media companies are producing huge number of news daily. Therefore using appropriate source of news and extracting only relevant news among it, can help to overcome the complexity of text mining for switching points detection. In this paper we aim to identify the relevant news to stock price switching points. Furthermore, we use one of the richest financial news sources, Dow Jones dataset.

Application of LDA for stock price movement is used in the literature [4]. In this paper, we like to test the performance of this method in a new setting.

First, only the related news to previous switching points are captured. Second, LDA model with variational bayes inference is trained on the captured news items to detect their topics and word distributions. Essentially the methodology uses the distribution of words of detected topics to identify current news item topics. Third, the similarity value between the current news and trained model is measured. We also built baseline learning models employing various standard classification algorithms using the complete dataset. Their performance is measured based on accuracy, precision and recall. Our results indicate that it is possible to reach similar performance levels by using the LDA model on a fraction of previous news, rather than the entirety of news items.

2 Background

Correlating textual news data with stock market data is a popular method to study price signal behaviours [5]. Text mining techniques such as text classification is implementing machine learning algorithms on a content for the purpose of stock switching point prediction [6]. Another technique which has been under investigation frequently, for the same purpose, is semantic and sentiment analysis [5], which uses the technique to find the context behind the whole content. On the other hand, text corpora that contain numerous documents and words, make it essential to develop another text mining technique to summarize a content. Text summarization techniques, like topic modelling, try to discover abstracts or hidden structure of the text bodies which are called topics. The above mentioned techniques can be combined to come up with better results [7]. In this paper, first text classification technique is implemented and then combination of topic modelling and text similarity technique are applied.

Different topic modelling techniques are present in the literature examples include, latent semantic analysis [8], probabilistic latent semantic analysis [9], and latent dirichlet allocation (LDA). LDA generally works best due to its generative nature. LDA is different in how it considers documents as mixture of topics and topics as distribution over words [10]. To solve the posterior problem, there are two main approaches, sampling methods [12], and optimization methods [11], with comparable performance results. In sampling methods like Markov chain Monte Carlo (MCMC), there is no need to explicitly set the parameters and thus, it is simpler in terms of implementation, but it has higher time complexity [10]. On the other hand, optimization methods, like variational bayes (VB) are guaranteed to converge to the posterior probability. This indicates that MCMC approaches are not efficient for large scale datasets. In this work both windowed and full batch implementations of VB are applied to capture the best possible results. Setting model parameters for VB inference depends on the corpus used the study [9] and how the performance is going to be evaluated. While in topic modelling domain the results are mainly reported based on perplexity [10–12], in our work, the experiment results are reported by performance measures that are derived from the confusion matrix.

3 Methodology

Our analysis starts by validating the existence of the relationship between financial news and stock market price movements by various standard classification techniques, and continues with a more efficient way of stock changing points prediction based on LDA. Figure 1 shows all implementation steps of these two methods. First, Dow Jones dataset in standard XML format (DJNML) is investigated and the related meta data are retrieved. The most important information for each news is display date of the news, which can help to determine concurrency of switching points and news, and name of the companies that the news is relevant to, which makes it possible to easily filter relevant news for a given company. As a result, from each news, four important elements are captured: display date, related companies, headline and text of the news. The dataset used for the experiment is from year 2015 which has 1,769,910 news with a data size of 7 GB. To query for desired news easily, these news are structured in a PostgreSQL database. In this paper, we chose to select Microsoft stock price as our test case. There are 1800 news items in Dow Jones dataset from year 2015 that are classified as relevant to Microsoft company. Source of stock market switching points is Yahoo finance. Switching points are manually labelled based on the local extremums in the time series data of the closing price. A total of 70 switching points were detected in year 2015 for Microsoft company.

Baseline Experiment on the Whole Dataset: We first built prediction models that learn from news data by employing different classification techniques such as random forest, penalized support vectors machine, logistic regression, Gaussian Naive Bayes and K nearest neighbours technique. To label the dataset, we used the comparison of display date of Microsoft news and list of its stock switching points. If a news item appears on the same day as a switching point, it is labelled as class positive and negative otherwise. In the test set, if a classification model predicts the label for a news item as positive, that means the model predicted a switching point at the same date of news display date. After labelling the news, we observed that it is an imbalanced dataset with 75% of the news labelled as negative and the rest as positive. To overcome the sampling bias, we over-sample the minority class (positive class) for training set with making multiple copies of the news labelled as positive to come up with exactly

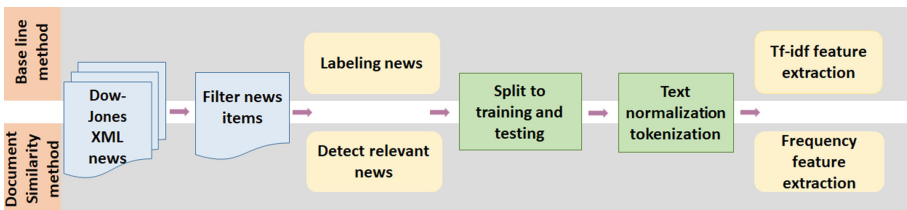


Fig. 1. Step by step implementation of two experiments

a balanced training set. Even though some classification techniques are resilient to imbalances in the dataset. For example, penalized classification techniques do not suffer from this problem by setting some misclassification cost for the minority class. We then split the whole news related to Microsoft stock as 67% for training and 33% for testing.

Document Similarity Based on LDA Method: In this experiment, we would like to test if we can build a model by learning only the most relevant news that caused switching points in the past instead of using the entire news dataset. We define relevant news to switching points by comparing the display date of news and switching points' time of occurrence in two days window size. News appearing on the day with a switching point of the stock price data or the day after it are considered as relevant news, and irrelevant otherwise. We can then use LDA method to capture hidden structure of the text. For implementation of this method through VB inference technique, we use LDA function of scikit-learn package in Python. Once LDA model training is complete, the news items in the test set are transformed to the same features representation (topic representation) generated by the LDA model. So that similarity of the news items in the test set can be compared with the learned model. We use cosine similarity to measure the similarity of testing news and training news by their new feature representation. The highest similarity value in each day can represent the probability of having switching point on the same date of the testing news display date. To simulate a more realistic scenario, we use 100 days of data for training and perform a test with the data from the next day after 100 days. Then we shift the window one day and repeat training and testing phases until the experiment covers the whole year.

The process for both experiments continue by doing text normalization and tokenization steps [13]. These steps include removing special characters, expanding contractions, case conversions, removing stop words, correcting words with misspelling or repeating characters, stemming and lemmatization [14]. In the next step, features of textual content is extracted to create the vector space model for each news. Weights for the features (words) are measured based on tf-idf technique (term frequency-inverse document frequency) for the baseline method and by frequency of words for LDA method due to the nature of VB inference technique [11]. We then measure results of these methods and compare their performance based on the confusion matrix [15]. Confusion matrix compares the predicted labels with actual labels for two different classes; namely: positive and negative. In this study positive class is associated with switching points and negative class is associated with the remaining records. We calculate different metrics such as accuracy, precision and recall based on the confusion matrix. We think among the provided metrics, precision for switching points detection has more importance since traders would like to have a reliable system that can precisely predict the switching points. In the probabilistic setting of the LDA method, different thresholds can be tested and the best performance can be compared to first approach performance.

4 Results

Results of the experiments performed with various classification techniques are presented in the first five lines of Table 1. Last two lines of Table 1 show the results achieved with the tests performed on LDA model with both online and offline VB inference techniques¹. The results of classification techniques clearly show that there is a relation between financial news and stock price time series data. On the other hand, the overall performance of the LDA experiments are comparable to that of standard classification techniques. As it mentioned before, precision for switching points detection has more importance, which is 0.64 for both approaches. Figure 2 illustrates these results with an alternative plot. It can be interpreted that switching points are willing to occur around news similar to previous related news and not willing to occur around non similar news.

Table 1. Performance of both methods on switching points detection

Method	Accuracy	Precision (+)	Recall (+)	Precision (-)	Recall (-)
Random forest	0.76	0.64	0.16	0.76	0.97
Penalized SVM	0.72	0.47	0.48	0.81	0.81
Logistic regression	0.74	0.50	0.39	0.80	0.86
Gaussian-Naive Bayes	0.68	0.39	0.39	0.78	0.79
KNN (K = 3)	0.69	0.43	0.62	0.84	0.71
Online VB LDA	0.61	0.64	0.24	0.60	0.90
Batch VB LDA	0.60	0.60	0.23	0.87	0.86

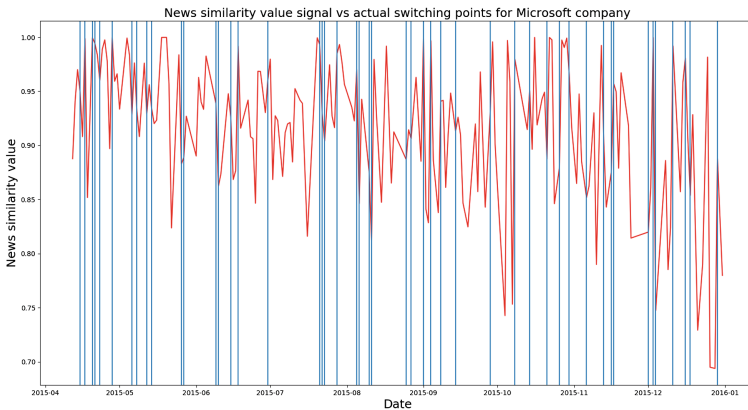


Fig. 2. Red line shows the similarity values, blue vertical lines show ground truth switching points. News with similarity values higher than the threshold are expected to be found in close proximity of switching points. (Color figure online)

¹ With document similarity threshold = 0.99, 0.98, number of topics = 100, 100, doc-topic prior = 0.01, 0.001, topic-words prior = 0.01, 0.005, max-iter = 200, 200, learning-offset = 64, - and learning-decay = 0.5, - for online and offline learning respectively.

5 Conclusion

Prediction of stock switching points is an important asset for stock market traders. In this paper, we suggest new approach of solving this problem by first validating the assumption of having relation between the news items and stock prices with different classification techniques. Then we propose a technique that first detects relevant news to past stock switching points and then train a LDA model using these news items. We implement our proposed method for Microsoft company on Dow Jones news dataset. It causes similar result with implemented classification techniques, but has the advantage of training on only relevant news which are significantly less than the total number of news. This results to reduce the storage cost of news for text mining platform which aim to predict stock market movements. Implementation of this method on different companies, for a larger data, and with an extended LDA model, is what it can be done in the future work.

Acknowledgments. This research is supported by TMX financial services company, NSERC CRDPJ under the grant number 499983-16, and OCE VIPII under the grant number 26280.

References

1. Avellaneda, M., Stoikov, S.: High-frequency trading in a limit order book. *Quant. Financ.* **8**(3), 217–224 (2008)
2. Mittermayer, M.A.: Forecasting intraday stock price trends with text mining techniques. In: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 10-pp. IEEE (2004)
3. Boudoukh, J., Feldman, R., Kogan, S., Richardson, M.: Which news moves stock prices? A textual analysis. National Bureau of Economic Research (2013)
4. Xie, B., Passonneau, R.J., Wu, L., Creamer, G.G.: Semantic frames to predict stock price movement (2013)
5. Li, X., Xie, H., Chen, L., Wang, J., Deng, X.: News impact on stock price return via sentiment analysis. *Knowl.-Based Syst.* **69**, 14–23 (2014)
6. Thomas, J.D., Sycara, K.: GP and the predictive power of internet message traffic. In: Chen, S.H. (ed.) *Genetic Algorithms and Genetic Programming in Computational Finance*, pp. 81–102. Springer, Boston (2002). https://doi.org/10.1007/978-1-4615-0835-9_4
7. Nguyen, T.H., Shirai, K.: Topic modeling based sentiment analysis on social media for stock market prediction. In: *ACL* (1), pp. 1354–1364 (2015)
8. Papadimitriou, C.H., Tamaki, H., Raghavan, P.: Latent semantic indexing: a probabilistic analysis. *J. Comput. Syst. Sci.* **61**, 217–235 (2000)
9. Hofmann, T.: Probabilistic latent semantic indexing. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 289–296 (1999)
10. Hoffman, M., Bach, F.R., Blei, D.M.: Online learning for latent Dirichlet allocation. In: *Advances in Neural Information Processing Systems*, pp. 856–864 (2010)
11. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)

12. Canini, K., Shi, L., Griffiths, T.: Online inference of topics with latent Dirichlet allocation. In: *Artificial Intelligence and Statistics*, pp. 65–72 (2009)
13. Habert, B., Adda, G., Adda-Decker, M., de Marèuil, P.B., Ferrari, S., Ferret, O.: Towards tokenization evaluation. In: *Proceedings of LREC*, pp. 427–431 (1998)
14. Sarkar, D.: *Text analytics with Python* (2016)
15. Alpaydin, E.: *Introduction to Machine Learning*. MIT Press, Cambridge (2014)

Graduate Student Symposium Papers



Software Defect Prediction from Code Quality Measurements via Machine Learning

Ross MacDonald(✉)

Saint Mary's University, Halifax, Canada
ross.e.macd@gmail.com

Abstract. Improvement in software development practices to predict and reduce software defects can lead to major cost savings. The goal of this study is to demonstrate the value of static analysis metrics in predicting software defects at a much larger scale than previous efforts. The study analyses data collected from more than 500 software applications, across 3 multi-year software development programs, and uses over 150 software static analysis measurements. A number of machine learning techniques such as neural network and random forest are used to determine whether seemingly innocuous rule violations can be used as significant predictors of software defect rates.

Keywords: Software defects · Defect prediction · Machine learning

1 Introduction

Lehman's Laws of Software Evolution have demonstrated that software development projects are becoming larger and more complex as the years go by [1]. With this ever-increasing complexity and size and decreasing quality, it is becoming increasingly difficult to accurately predict how a development program will unfold.

Research has demonstrated that the later a software defect is found during development, the more costly it becomes to fix [2]. One reason for this escalation has been attributed to the fact that the longer a development project runs, the higher the overhead cost and the higher the cost required to change the system [2].

Many static analysis tools exist today, each of which provides a particular focus on the quality of software. Research recommends that software development teams should utilize a "meta tool" that would allow them to combine the results from various static analysis tools together, thus achieving a better results [3]. SonarQube is such a "meta-tool" that provides a software development team with the ability to import, customize and automatically execute static analysis on code utilizing a variety of rule definitions.

Unfortunately, with the advent of meta-tools such as SonarQube, a new problem is created: Metrics Galore [4]. Metrics Galore is a situation where a team is paralysed by attempting to monitor too many metrics simultaneously. Many issues can occur such as de-motivation of the team, focusing on the wrong metrics, losing sight of the important goals of the program, etc. In order to address this issue, while still providing metrics for the software development team to track and improve upon, it is possible to employ machine learning algorithms to help. While unsupervised learning can provide a solution to the overwhelming metrics issue [5], regression and classification could serve to solve the prediction issues by providing a predictive model that could be used by the development team and by the software development managers [3].

The goal of the proposed research is to demonstrate the value of static analysis metrics in predicting software defects at a much larger scale than has been proposed previously as is indicated in Table 1:

Table 1. Program measurements

Measure	Program 1	Program 2	Program 3
Duration (Years)	2	1	7
Team size (People)	30	10	80
Applications	290	132	352
LoC (MSLOCs)	4.28	2.96	7.72

2 Methodology

Analyses will be performed on a data set consisting of software metrics data collected from three large-scale software development programs. These programs consisted of software with high-reliability, high-criticality and safety-critical performance requirements, and thus were extensively tested during and after software development activities.

The data are snapshots of a continuum of ever-changing values. In order to limit the scope of the research in this paper to a manageable size, it was determined that a single snapshot in time of software source code and a single snapshot in time of software defect metrics would be analysed.

In order to determine the optimum point in time to perform both of these snapshots, intimate knowledge of the program plans was required. The theory built around these data collections is based on two key points: The source code should be analysed after major software development was completed, but prior to acceptance testing as this source repository will have a maximum number of undiscovered defects; and the software defect measurements should be analysed well after acceptance testing was performed in order to have high confidence that most major defects have been discovered.

Since there are several hundred static analysis metrics and rules that could be counted in the analysis, it will be necessary to narrow down the predictors to a more manageable number in order to aid in analysis as well as to avoid model over-fitting. This feature reduction will be performed in several steps as described below.

- Eliminate zero and near-zero variance predictors
- Eliminate single items of strongly intercorrelated predictor pairs
- Feature clustering using k-means in order to group similar features together
- Recursive feature elimination to remove insignificant features from the model

After feature selection is performed, analysis of each of the models will be compared where appropriate and statistical findings will be provided and analysed upon completion of the research. The research will focus on Regression and Classification prediction models. For regression, the following models will be used: Linear Regression, Neural Network Regression, and Support Vector Machine (SVM) Regression. For classification, the following models will be used: Decision Tree, Random Forest, Neural Network, and SVM.

Regression analysis will target the raw defect count as the outcome, while Binary classification will use a binary definition by answering the following question: “Does the application contain a defect?”. Finally, Multi-class classification will be performed on the software defect count by stratifying it across several classes. These values will be defined as: No defects, Low defect rate, Moderate defect rate, High defect rate, and Extreme defect rate.

3 Results To-Date

Progress to date has followed the plan as laid out in the methodology. The data has been collected, cleaned and prepared for analysis. Feature elimination has been performed by variance elimination and inter-correlation elimination. The feature sets were further narrowed by clustering similar features together using a k-means analysis. Regression and Binary classification has been performed on the reduced feature set and the results can be observed in the following tables.

Table 2. Linear regression performance summary - top three

Name	RMSE (Root Mean Squared Error)
lm-2	11637.48
sl-2	11669.20
sl-1	11844.82

As shown in Table 2, the Regression performance is less than desirable for the dataset, as the RMSE is quite large. In fact, the best RMSE value is nearly three-times that of the mean of the outcome column in the original dataset.

Conversely, Table 3 shows favourable results for binary classification, with a top accuracy of over 70% using Random Forest Classification.

Table 3. Binary classification performance summary - top three

Name	Accuracy
rf-all	70.82%
rf-1	70.31%
rp-all	65.36%

4 Future Work

The thesis has made significant progress in generating the dataset and subsequent data preparation. This dataset is unique in terms of its size and scope - spanning 500 software applications over three years of software development and includes 150 software static analysis measurements. The initial results shown in Table 3 provides strong evidence for our original hypothesis that seemingly innocuous rule violations can be used as strong predictors of software defect rates. The remaining work includes experimentation with a number of machine learning techniques to create a credible model to predict the software defects. In addition to the successful use of Random Forest for predictions, the project will experiment with different neural network architectures. The results of predictions will be compared against multi-class classifications. Finally, the thesis will conclude with a comprehensive analysis of feature importance and make recommendations for best practices in software development process. The remaining work is intended to constitute the majority of the thesis research of the author.

References

1. Lehman, M.M., Ramil, J.F., Wernick, P.D., Perry, D.E., Turski, W.M.: Metrics and laws of software evolution - the nineties view. In: Proceedings of the 4th International Symposium on Software Metrics, METRICS 1997, pp. 20–32. IEEE Computer Society, Washington (1997)
2. Haskins, B., Stecklein, J., Dick, B., Moroney, G., Lovell, R., Dabney, J.: 8.4.2 error cost escalation through the project life cycle. In: INCOSE International Symposium, vol. 14, pp. 1723–1737 (2004)
3. Rutar, N., Almazan, C.B., Foster, J.S.: A comparison of bug finding tools for java. In: 15th International Symposium on Software Reliability Engineering, pp. 245–256, November 2004
4. Bouwers, E., Visser, J., van Deursen, A.: Getting what you measure. *Commun. ACM* **55**(7), 54–59 (2012)
5. Hinton, G., Sejnowski, T.: *Unsupervised Learning: Foundations of Neural Computation*. A Bradford Book, McGraw Hill Book Company (1999)



Automated Scheduling: Reinforcement Learning Approach to Algorithm Policy Learning

Yingcong Tan^(✉)

Mechanical, Industrial and Aerospace Engineering,
Concordia University, Montréal, Canada
t.yingco@encs.concordia.ca

1 Introduction

Automated planning and scheduling continues to be an important part of artificial intelligence research and practice [6, 7, 11]. Many commonly-occurring scheduling settings include multiple stages and alternative resources, resulting in challenging combinatorial problems with high-dimensional solution spaces. The literature for solving such problems is dominated by specialized meta-heuristic algorithms.

Meta-heuristics are high-level algorithms that design or select a heuristic to produce sufficiently good solutions for an optimization problem [4]. In many practical applications, especially solving complex scheduling problems, they are favorable due to their computational efficiency [9]. To ensure good performance, meta-heuristics commonly require a calibration process with extensive experiments for parameter/algorithm tuning. Such a process is computationally expensive.

Meanwhile, there is increasing focus on reinforcement learning (RL) in the machine learning community. RL is well-suited for autonomous decision-making policy learning and has been applied in literature for automatic algorithm tuning [2, 7, 10], but the literature of RL in complex scheduling problems, particularly those that involve multiple stages and alternative resources is scarce. More specifically, the structure of the feature space and search strategy in the solution space are still not well-understood for these complex scheduling problems.

To address the drawback of costly algorithm tuning, I propose a reinforcement learning guided variable neighborhood search (VNS) algorithm for solving combinatorial optimization problems in general and scheduling problems in particular. The rest of this abstract is organized as follows. Details of the proposed VNS algorithm are provided in Sect. 2. Section 3 is a summary of the thesis progress made to date.

2 Methodology: Variable Neighborhood Search

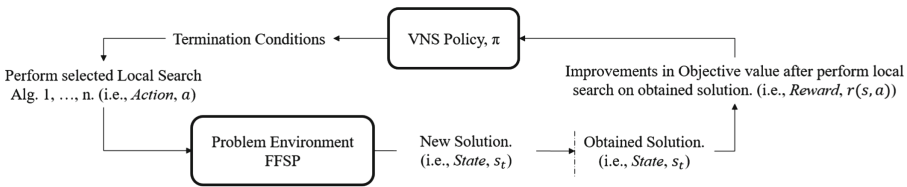
Variable Neighborhood Search (VNS) was first proposed by Mladenović & Hansen in 1997 [8]. VNS algorithms search in predefined neighborhoods of a

given solution and seek for a solution with better “quality” (i.e., objective value). As suggested by the name, VNS normally contains several predefined neighborhoods. By switching from one neighborhood to another, VNS tries to escape from local optima. The key to the success of VNS is to explore the right neighborhood at the right time. In VNS, this policy of neighborhood exploration is controlled by predefined hyper-parameters. To avoid extensive human intervention in parameter tuning, there is an increasing interest in automatic tuning [1, 10]. In [10], a reinforcement-learning (i.e., Q-learning) approach is developed for automatically tuning the hyper-parameters of their proposed VNS algorithm. However, detailed control of algorithm behavior of VNS (e.g., selection of neighborhood exploration during the process of VNS) can not be obtained by tuning the values of hyper-parameters, thereby hampering algorithm flexibility.

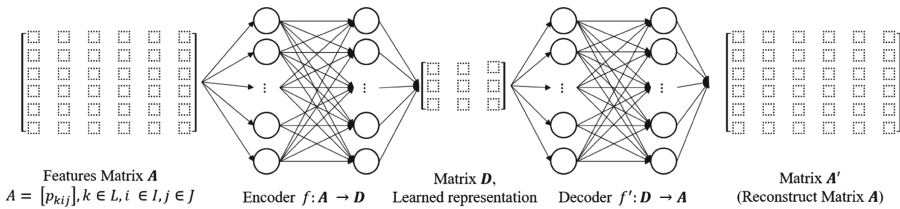
Instead of automatic parameter tuning, I propose a novel approach to directly learn the optimal algorithm policy of meta-heuristics through reinforcement learning. In reinforcement learning, we want to perform the right *action* at a given *state* to maximize the *reward*. Below, I will provide details of two key features of my approach: (1) a Markov decision process (MDP) representation of the proposed VNS, (2) feature representation using artificial neural networks.

2.1 MDP Representation of VNS Algorithm

A Markov decision process (MDP) provides a mathematical framework for modeling environment-dependent decision making process. To formulate a reinforcement learning task as an MDP, it must satisfy the Markov property (i.e., the probability of being in a state depends only upon the immediate past state and the action taken in that state).



(a) MDP representation of VNS.



(b) Architecture of auto-encoder network for feature representation learning of FFSP

Fig. 1. Demonstration of the proposed data-driven VNS algorithm

Starting with an initial solution, VNS performs local search in a selected neighborhood. A new solution and its objective value are then obtained, and this is defined as completion of one iteration. VNS then iterates among different neighborhood search algorithms until the termination condition(s) is (are) met. This iterative flow of VNS is shown in Fig. 1.a. To represent the VNS search process, we view each iteration as a time step and assume that only the most recently obtained solution will be kept and then used in the next iteration. This suggests that the action of neighborhood search and the resulting solution in iteration t depend only on the solution obtained in the immediate past iteration $t - 1$, i.e., $Pr(a, s_t | s_{t-1}, s_{t-2}, \dots, s_1) = Pr(a, s_t | s_{t-1})$, where, *state* s_t corresponds to the solution in iteration t ; *action* a and a *reward* $r(s, a)$ correspond to the selection of neighborhood search and resulting change in solution objective value, respectively. This reward function is expected to make the VNS search more greedy, and would also require us to properly define the discount factor. As shown above, the proposed VNS satisfies the Markov property. However, we must address the feature design problem of the FFSP before implementing the MDP representation of VNS. Details of the proposed feature representation learning are provided below.

2.2 Feature Representation of Scheduling Problem

In combinatorial optimization problems, one way of learning the feature representations is through a collection of a large set of statistical data (e.g., mean job processing time over different stages or machines) [5]. This approach has two drawbacks: (1) no mathematical way to evaluate the accuracy of the feature representation; (2) not all collected statistical data are equally important, and thus an additional process of feature extraction/elimination might be required.

Therefore, I propose to learn a feature representation of FFSP using an auto-encoder network. An auto-encoder is an unsupervised learning technique aiming at learning a data set representation with dimensional reduction [3]. It includes two neural networks, an encoder and a decoder. The encoder reads the input matrix \mathbf{A} (e.g., job processing times, p_{kij}), and transforms it into a matrix \mathbf{D} (i.e., learned representation with reduced size), i.e., $f : \mathbf{A} \rightarrow \mathbf{D}$. In this process, the matrix \mathbf{A} consists of raw data. The cost of data collection is very low; however, preprocessing might be required (i.e., data normalization). Using matrix \mathbf{D} as input, the decoder attempts to reconstruct the original input matrix, i.e., $f' : \mathbf{D} \rightarrow \mathbf{A}'$. An auto-encoder network is trained to minimize a loss function of mean squared error between \mathbf{A} and \mathbf{A}' , i.e., $\{f^*, f'^*\} = \arg \min_{f', f} \|\mathbf{A}' - \mathbf{A}\|^2$. The mean squared error measures how close the reconstructed input \mathbf{A}' is to the original input \mathbf{A} . A schematic of the complete structure is shown in Fig. 1.b. Using the auto-encoder provides a mathematical measurement on how accurate the learned feature matrix \mathbf{D} is. In addition, we can see the encoder network as an intelligent way of screening out irrelevant information.

3 Thesis Progress

To date, I have mainly focused on studying the structure of FFSP. I developed two decomposition-based exact methods, logic-based Benders decomposition and branch-and-check, for solving a two-stage FFSP with unrelated parallel machines. The proposed algorithms obtained substantial improvement over a state-of-the-art mixed-integer programming model. A paper is accepted by the 31st Canadian Conference on Artificial Intelligence. In addition, I independently developed the conceptual idea described in earlier sections, have done a thorough literature review for my thesis, and developed the complete MDP model of VNS.

References

1. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. In: Gent, I.P. (ed.) CP 2009. LNCS, vol. 5732, pp. 142–157. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04244-7_14
2. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv preprint [arXiv:1611.09940](https://arxiv.org/abs/1611.09940) (2016)
3. Bengio, Y., et al.: Learning deep architectures for AI. *Found. Trends® Mach. Learn.* **2**(1), 1–127 (2009)
4. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Comput. Surv. (CSUR)* **35**(3), 268–308 (2003)
5. Gupta, J.N., Sexton, R.S., Tunc, E.A.: Selecting scheduling heuristics using neural networks. *INFORMS J. Comput.* **12**(2), 150–162 (2000)
6. Heching, A., Hooker, J.N.: Scheduling home hospice care with logic-based benders decomposition. In: Quimper, C.-G. (ed.) CPAIOR 2016. LNCS, vol. 9676, pp. 187–197. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33954-2_14
7. Ingimundardottir, H., Runarsson, T.P.: Discovering dispatching rules from data using imitation learning: a case study for the job-shop problem. *J. Sched.* 1–16 (2017)
8. Mladenović, N., Hansen, P.: Variable neighborhood search. *Comput. Oper. Res.* **24**(11), 1097–1100 (1997)
9. Ruiz, R., Vázquez-Rodríguez, J.A.: The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **205**(1), 1–18 (2010)
10. Shahrabi, J., Adibi, M.A., Mahootchi, M.: A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Comput. Ind. Eng.* **110**, 75–82 (2017)
11. Tran, T.T., Vaquero, T., Nejat, G., Beck, J.C.: Robots in retirement homes: applying off-the-shelf planning and scheduling to a team of assistive robots. *J. Artif. Intell. Res.* **58**, 523–590 (2017)



Estimating Vineyard Grape Yield from Images

Tanya Monga^(✉)

Jodrey School of Computer Science, Acadia University,
Wolfville, Nova Scotia, Canada
tanya.mongaa@gmail.com

Abstract. Agricultural yield estimation from natural images is a challenging problem to which machine learning can be applied. Convolutional Neural Networks have advanced the state of the art in many machine learning applications such as computer vision, speech recognition and natural language processing. The proposed research uses convolution neural networks to develop models that can estimate the weight of grapes on a vine using an image. Trained and tested with a dataset of 60 images of grape vines, the system manages to achieve a cross-validation yield estimation accuracy of 87%.

Keywords: Image processing · Machine learning · Yield estimation

1 Introduction

A major challenge faced in agriculture is accurately estimating the amount of produce that a crop will yield while still in the field. Typically, farmers must wait to measure their crop after harvest using manual or mechanical methods. There is value in having methods of yield estimation based on data that can be captured with inexpensive technology in the field, such as the combination of a smartphone and cloud-based computing. This would allow the farmers to better estimate the material and human resources required during and following the harvest.

Estimating the yield of a grape vine in kilograms from a image of that vine prior to harvest is a more challenging problem than counting objects because the predictive model must learn to estimate the volume of grapes when many of them are occluded by other grapes. However, this is possible because viticulturists and experienced grape growers are able to make such estimates based on having seen many harvests¹. The annual variation seen in yield estimation using different methods to calculate weight is 30% [8]. According to industry standard, the acceptable level of accuracy is 5–15% [8]. Our intention is to do the same using a deep learning neural network approach to a level of 90% accuracy on an independent test set of images taken in the field using a smart phone.

¹ This research is a collaboration with Lightfoot and Wolfville Vineyard of Wolfville, Nova Scotia, Canada.

To achieve this level of accuracy, a method is developed and tested that would automatically normalize the images in terms of grape size, brightness, and color balance. Deep learning neural networks are then used that appropriately bias model development based on the relationship between pixels in an image and that use activation functions with improved derivative characteristics which have proven to be very effective for image classification and regression problems [1–6].

2 Approach

Image Processing. There is a need to modify the images so that machine learning software has an easier time estimating the weight of the grapes. Any knowledge that we can bring to bear on the data to create better features for the machine learning software, the more accurate will be the resulting models. To accomplish this, image processing methods are used to automatically normalize the test and training images in terms of mark's size, color, light level and contrast. This results in a more constrained (less variable) set of images for the machine learning system during training and testing.

Scale Normalization: The grapes in the smart phone images can range in size from image to image. If they are approximately the same size, it will be easier for the machine learning system to estimate the yield. A black and white calibration mark is placed near each set of grapes in the image. To normalize the size of grapes, we do the following: (1) Find the calibration mark in the image (Fig. 1); (2) Calculate the height and width of the mark in pixels; (3) Scale the image based on the height and width of the calibration mark using $S = D_L/C_L$, where S is Scaling Factor, D_L is Desired length of square edge and C_L is Calculated length. The height and width of the image is then multiplied by the scaling factor, S . To normalize the size of grapes, OpenCV 3.0 library is used. The automated scaling method is tested by using a Windows Pixel ruler to measure the length of the calibration mark after rescaling the image. The mark got rescaled to within ± 4 pixels.

Brightness Level: Variation in the brightness of the images can distort colors and hamper the machine learning efforts. Having all images at a common brightness level is important. To calculate the image brightness, the image processing software takes the average value of all the pixels and uses these values to calculate the perceived brightness (luminance). The image brightness is then normalized to be out of 100 by multiplying the pixel values of image by $100/L$ where L is the calculated luminance of the image. The image Processing library provided by GNU Octave is used for adjusting the brightness level.

Color and Contrast Enhancement: Contrast is important for distinguishing between objects in an image. Higher contrast in an image will give better edges around objects which should help the machine learning software to learn faster. To adjust the contrast, a technique called histogram matching is used in which histogram of 2-D image is adjusted to match the histogram of the reference image [7]. The images appear clearer and the grapes appear to stand out from



Fig. 1. Original image



Fig. 2. Normalized and cropped image



Fig. 3. The architecture of proposed network.

the background more than the original images. The numpy 1.13.0 package is used for implementing histogram matching.

Convolutional Neural Network. The proposed neural network model, as shown in Fig. 3, is created using Keras and Tensorflow libraries. The normalized image is cropped to a size of $225 \times 225 \times 3$ (RGB pixels) as shown in Fig. 2, to feed into the CNN. The Network model consists of 6 blocks of layers; 5 convolutional blocks and 1 fully connected block. Each convolutional block contains at least one convolution layer having stride 1, followed by max pooling, batch normalization and dropout layers. The final fully-connected block produces the harvest weight value. In order to train the network from scratch, the stochastic gradient descent optimization technique is used with a learning rate of 1×10^{-4} , a batch size of 32 images and a momentum value of 0.9. In addition, weight decay is set to $1e-6$. The network is trained for 300 epochs on NVIDIA GeForce GTX 960M GPU and the best model is selected based on the validation set error to be evaluated against the test set.

3 Empirical Study

Approach. A dataset of 60 Pinot Noir grape images (preprocessed as described in Sect. 2) and associated harvest weights is used to build and test a CNN model (Fig. 3). There were 14 different weights of fruit ranging from 70 g to 450 g. Since the dataset is small, a 6-fold stratified cross validation approach was used to test the performance of the models. The mean absolute error (in grams) and

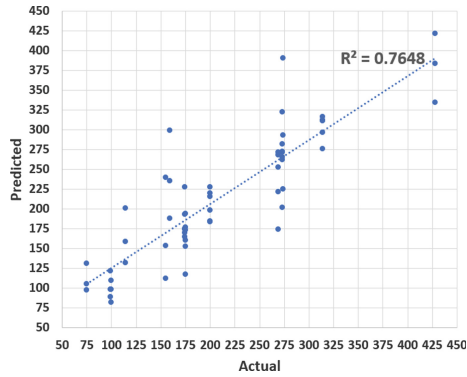


Fig. 4. Scatter plot of actual vs predicted values.

percentage correct (MAE in grams per image/Mean number of grams over all images) are used as performance measures.

Results. The best CNN model described in Sect. 2 produced an MAE = 28.35 g over all test images, with a 95% confidence interval of ± 8.10 g. This is equivalent to a mean accuracy of 87%, given 214.93 as the average number of grams per image. The correlation between the predicted and actual values is shown in Fig. 4.

4 Conclusion

The empirical results show that the proposed approach comes close to reaching our goal of 90% accuracy for predicting harvest grape yield. This supports that convolutional neural network has a great potential to be used for agricultural yield estimation. In future work, we will be using a much large dataset of images and weights captured this fall during the harvest season. We are developing a mobile web app based on the proposed approach which can be used by farmers for harvest yield estimation.

Acknowledgement. I would like to pay special thankfulness to my supervisor Dr. Daniel Silver for his vital support and assistance.

References

1. Cohen, J.P., Boucher, G., Craig, A., Lo, H.Z., Bengio, Y.: Count-ception: counting by fully convolutional redundant counting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 18–26 (2017)
2. French, G., Fisher, M., Mackiewicz, M., Needle, C.: Convolutional neural networks for counting fish in fisheries surveillance video (2015)

3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process Syst.* **25**, 1097–1105 (2012)
4. Xie, W., Noble, J.A., Zisserman, A.: Microscopy cell counting and detection with fully convolutional regression networks. *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.* **2016**, 1–10 (2016)
5. Ding, W., Taylor, G.: Automatic moth detection from trap images for pest management. *Comput. Electron. Agr.* **123**, 17–28 (2016)
6. Cheang, E.K., Cheang, T.K., Tay, Y.H.: Using Convolutional Neural Networks to Count Palm Trees in Satellite Images. arXiv preprint [arXiv:1701.06462](https://arxiv.org/abs/1701.06462) (2017)
7. Maini, R., Aggarwal, H.: A comprehensive review of image enhancement techniques. arXiv preprint [arXiv:1003.4053](https://arxiv.org/abs/1003.4053) (2010)
8. Komm, B., Moyer, M.M.: Vineyard yield estimation. Washington State University Extension Bulletin EM086E (2015)



Real-Time Deep Learning Pedestrians Classification on a Micro-Controller

Zhaoyang Huang^(✉) 

Department of Electrical and Computer Engineering, University of Calgary,
Calgary, AB, Canada
zhaoyang.huang@ucalgary.ca

Abstract. Deep learning neural network is one of the most advanced tools for object classification. However, it is computationally expensive and has performance issues in real time applications. This research's use-case is efficient design and deployment of deep learning neural networks on palm sized computers like Raspberry Pi (RPi) as an in-vehicle-monitoring-system (IVMS) for real-time pedestrian classification. I have developed a system based on a neural network template named Cafenet that runs on an RPi and can classify pedestrians using deep learning. Simultaneously, I have proposed a new classification system based on multiple RPi boards, which offers users two modes of pedestrian detection: one is fast classification, and the other is accurate classification. The experiments results show that the device could classify pedestrians in real-time and the detecting accuracy is acceptable.

Keywords: Real-time · Pedestrian classification · Deep learning
Raspberry Pi

1 Introduction

In modern societies, road accidents caused by motor vehicles are becoming more and more frequent. Fatigue is a main causes of road accidents. According to Canadian Council of Motor Transport Administrator (CCMTA), regarding the causes of road accidents like alcohol impairment, overspeed, unsafe passing, etc., approximately 20% of fatal collisions are caused by fatigue driving [1].

Drivers with the symptom of fatigue like drowsiness, yawning, etc., may result in server traffic accidents. Thus, it is necessary to have an alarm system on vehicles for safer driving. Now, plenty of driving assistant systems have been developed to help drivers observe the environment around the vehicles and continuously send feedback to the driver. However, advance monitoring systems based on deep learning require high computational power to perform in real-time. Hereon, my research problem is developing a lightweight neural network structure and apply it to low-cost hardware system to obtain real-time classification performance.

2 Related Works

Real-time pedestrian detection is the fundamental research content for ADAS. In [8] the Adaboost cascade pedestrian detector was developed which utilized Haar features to characterize pedestrians. Though it achieved a detecting rate of 20 to 25 Hz on a specialized microcontroller, its error detection rate is high. Zhu et al. [10] innovated human detection method by designing detecting blocks with different sizes, positions and scales to increase the space of the eigenvalues calculated by HoG. This method has 88% accuracy and 5 FPS detecting rate, but its false positive rate is high.

Because the hand-designed characteristics of conventional method are not robust for diversity changes of pedestrians in the images. Thus, deep learning algorithm which has better features extraction and classifying ability occurred. Girshick [5] developed R-CNN framework which has 83% mean average precision on VOC 2012. It used selective search algorithm to extract candidate regions and scaled them into fixed size. Then, these candidates are taken as inputs to the convolutional neural network (CNN) and extracted the features. Finally, SVM is used to classify the objects based on the features. But, both its missing rate and processing time are too high for real-time applications.

3 Proposed Method

This section introduces my proposed method from the perspective of feasibility, modified neural network architecture, image dataset and the system framework.

3.1 Feasibility Analysis

There are quite a lot of low-cost microcontrollers with excellent performance, like Arduino, Intel Edison, Raspberry Pi (RPi), UDOO board, etc. From the aspects of performance and price, RPi 3 (1.2 GHz, 1G RAM) was selected.

I ran a simple feasibility test to verify whether a network running on RPi can meet the real-time requirement. I prepared images of four objects for training, which are bus, elephant, flower, and horse, and each object set has 2500 images. The accuracy was about 93% while the processing speed was 0.97s per image which is much more than the acceptable normal reaction time of 0.3s. The maximum size of the Caffe model run on the RPi was 300 MB.

Based on the feasibility analysis, no single Caffe convolutional neural network framework (CNN) with a thorough well-designed structure can be implemented on one RPi 3 board to gain real-time performance and high detecting accuracy at the same time.

3.2 Proposed System Framework

Apparently, improvement could be gained through utilizing more CPU cores for calculation, suitably simplifying the convolutional layers in the neural network

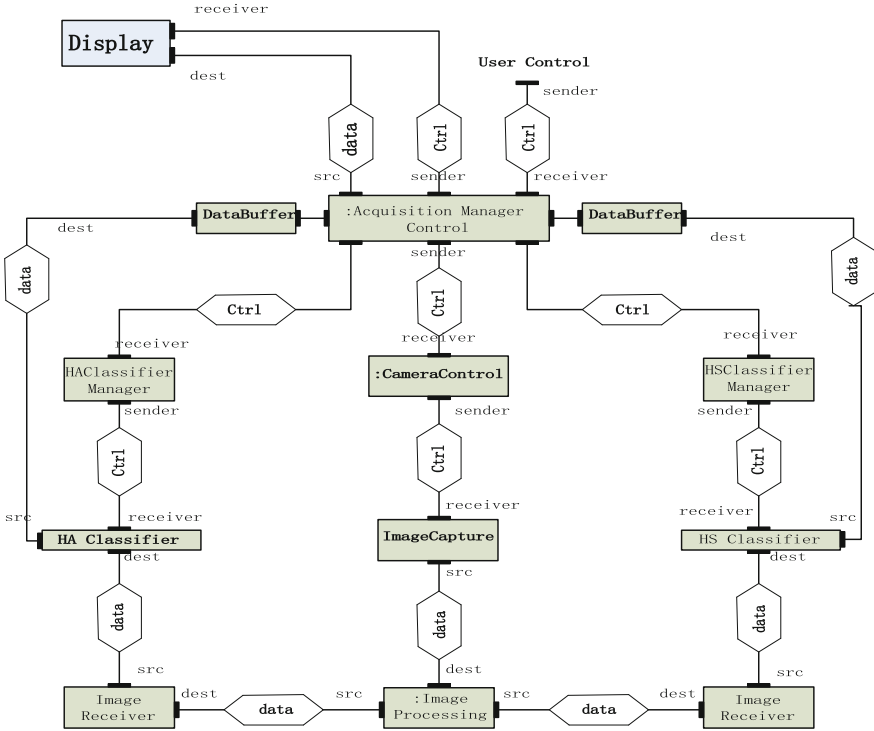


Fig. 1. The overall conceptual view of the proposed system.

structure and using multiple RPi boards. Therefore, I developed a new system that uses three RPi boards, one for main controlling and the other two used for more accurate classifying and fast classifying, respectively, because the proposed system framework intends to separate two properties of real-time pedestrian classification which are fast and accurately classifying respectively and apply these two attributes on two different RPi boards so that it could achieve the effect of real-time classification. The overall conceptual design view for the proposed driving assistance system is shown in Fig. 1.

As shown in Fig. 1, the system consists of five components: System Core Function module, High Accuracy (HA) Classifier module, High Speed (HS) Classifier module, GUI module and Data-transferring module. What includes in the System Core Function module is a RPi 3 and a RPi camera module, where the board is responsible for controlling the two classifier modules and the camera module is for capturing images. For HA classifier, it contains a RPi board that loads a Caffe model which is obtained by fine-tuning a pretrained CaffeNet model with a well-designed structure. For the HS classifier, it also consists of a RPi board that loads another Caffe model that trained with a tiny neural network structure for quicker classifying. The fourth component is the GUI module which is a 7-inch LED screen for the user to interact with the whole system and read the

classification results. And the Date-transferring module are all the data buffers in the design, which could be considered as a virtual model. Because it is defined as a portion of the hard-disk memory of the RPi board of the System Core Function module that shared with other classifying modules under the same local area network, the transferring time for images and results can be neglected, which is preferred for real-time pedestrian classification.

Therefore, combined with the overall design (Fig. 1), the workflow of the proposed system is as follow. First, using the camera module on the control system to capture the images and sending them to HA and HS Classifiers after resizing images respectively. Then, due to the rapid classification of HS, HS first comes up with a preliminary classification result and immediately sends it back to the control system and gives feedback to the driver. Finally, combining the results obtained by the HA classification, the control system will correct the HS classification results and then gives the driver feedback again.

3.3 Data Preparation

For the data used for training and testing my proposed classifier, I planned to utilize different pedestrian databases which are the Caltech Pedestrian Dataset [2], the Penn-Fudan Database for Pedestrian Detection [9], Robust Multi-Person Tracking dataset [3], KAIST Multispectral Pedestrian Dataset [6] and object detection data set of the KITTI Vision Benchmark [4] to build my own training and testing data, because only the dataset with properties reflecting realistic, clear images or videos, huge variety, etc., can affect the performance of the classifier. To integrate all these datasets, image preprocessing is a must.

First is manually separating positive and negative samples from all databases; secondly, to improve efficiency, down sample the image in all databases. The next step is trimming and resizing all the images. Since my pedestrian classifier is to assist drivers detecting pedestrians in front of and near the vehicles, only the midsection of the images is kept. To let these images fit for HA classifier and HS classifier, I resized them to the resolution of 256 by 256 and 128 by 128, respectively. Furthermore, to further increase the size of the training set, I enhanced all the images, including horizontal flip, global histogram processing and three-channel color change. Finally, due to the use of the Caffe library, the input format for training deep learning classifier is lmdb database, all the pedestrian training images need to be saved as lmdb.

In summary, I got 11,458 pedestrians images and 7,548 non-pedestrians images for training and around 1500 pedestrians images and 1900 non-pedestrians images for testing.

3.4 CNN Network Design

Considering the hardware constraints of the RPi, it is unrealistic to use a large CNN structure directly, such as vggNet, ResNet, Inception, etc. For HS and HA classifier, I implement different strategies.

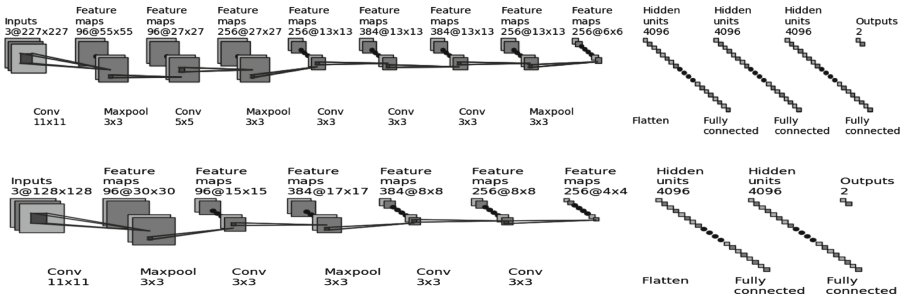


Fig. 2. The CNN architecture of the HA classifier(*above*) and the HS classifier(*bottom*).

Based on the feasibility analysis results, the maximum volume for the Caffe model which can be run smoothly on RPi is around 300 MB. For HA classifier model, to obtain a high classification accuracy with a small number of training dataset, I fine-tuned the pre-trained model (CaffeNet) [7] which is a light-weight CNN network to build my own pedestrian classifier. The size of this Caffe model file is about 200 MB matching the RPi requirement. As for HS classifier, I took the CaffeNet as template and designed a CNN network structure by decreasing the filter size, the number of the convolutional layers as well as the number of output features. The purpose of the modifications is to reduce the computational complexity of deep learning model so that the RPi could have real-time performance. The Fig. 2 describes the CNN neural network structure for accurate classifier and fast classifier respectively.

4 Experiments

4.1 Results

I performed experiments to measure processing time and accuracy, to match the needs indicated in the project’s requirements. For time experiment, I prepared a Python script which could record the processing time for classifying all tested images and calculate the average classification time. The results are in Table 1. Accuracy experiments for the HA and HS classifiers are performed on a desktop computer, since the accuracy should not be affected by hardware configurations. Table 2 shows classification results. The overall accuracy for HA and HS classifier is 79.75% and 70.63%, respectively. And the false negative rate, which indicates the performance of classifier from the aspect of failing to detect pedestrians in the scene, is 8.28% and 8.93% respectively.

Table 1. Results of time experiment for both classifiers.

	HA classifier	HS classifier
Processing time/image(ms)	631	153

Table 2. Results of accuracy experiment for HA and HS classifiers

		Predictive results(HA)		Predictive results (HS)	
		Pedestrians	Not-pedestrian	Pedestrians	Not-pedestrian
Ground truth	Pedestrians	1418	128	1408	138
	Not-pedestrian	574	1347	880	1041

5 Discussions

I proposed a pedestrian classifier system based on CNN and RPi. The results shown above indicate that the overall accuracy and HA classifier processing time are not the best in class, but the HS classifier fulfils the real-time requirement and the False negative rate is at a very low rate, which indicates the proposed method has practical value to some extent from the perspective of safety driving.

Future work will include optimization of neural network architecture to further speed up the classification time in terms of the size of the convolutional filter and the number of output features for each layer and developing a new network architecture based on some other lightweight network templates, like MobileNets, SqueezeNet, ShuffleNets, etc.

References

1. Traffic Injury Research Foundation: Fatigue-Related Fatal Collisions, Canada, 2000–2013 (2016)
2. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: an evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(4), 743–761 (2012)
3. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: A mobile vision system for robust multi-person tracking. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8. IEEE (2008)
4. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The kitti vision benchmark suite. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361. IEEE (2012)
5. Girshick, R.: Fast R-CNN. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448 (2015)
6. Hwang, S., Park, J., Kim, N., Choi, Y., Kweon, I.S.: Multispectral pedestrian detection: benchmark dataset and baseline. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1037–1045 (2015)
7. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 675–678. ACM (2014)
8. Viola, P., Jones, M.J., Snow, D.: Detecting pedestrians using patterns of motion and appearance. *Int. J. Comput. Vision* **63**(2), 153–161 (2005). <https://doi.org/10.1007/s11263-005-6644-8>

9. Wang, L., Shi, J., Song, G., Shen, I.: Object detection combining recognition and segmentation. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007. LNCS, vol. 4843, pp. 189–199. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76386-4_17
10. Zhu, Q., Yeh, M.C., Cheng, K.T., Avidan, S.: Fast human detection using a cascade of histograms of oriented gradients. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 1491–1498. IEEE Computer Society, Washington, DC (2006). <https://doi.org/10.1109/CVPR.2006.119>



A Unified Evaluation Framework for Recommenders

Alaa Alslaity^(✉)

University of Ottawa, Ottawa K1N6N5, Canada
Aalsl1005@uottawa.ca

Abstract. Recommender Systems are usually evaluated by one or two metrics. Due to the multifaceted nature of recommender systems, however, it is insufficient to evaluate them using only one metric. This paper presents my Ph.D. research agenda on evaluating recommenders from different points of view. In particular, I aim to provide a comprehensive evaluation framework that merges different metrics and comes up with an overall result evaluation. The proposed framework is built based on an inferred correlation between the most important metrics and a weight function that assign different weights for different metrics based on the application area of the recommender. This work can be used to evaluate different recommender types that are applied to the most popular application areas such as movies, documents, etc.

Keywords: Recommender systems · Evaluation framework
Unified evaluation models · Metrics

1 Introduction

Researchers have proposed hundreds of recommendation approaches over the last two decades. Such vast variety of algorithms raises the need to introduce evaluation methods to assist designers to decide on the most appropriate algorithm for their applications. Initially, RSs were evaluated based on their correctness (or accuracy) [1]. Recommender Systems, however, are multi-faceted systems; users have different expectations regarding recommenders' results; Examples of such expectations are: discovering new items, preserving privacy, and exploring diverse items. Thus, it is insufficient to evaluate RSs based on a single metric. Therefore, researchers have introduced different metrics to recommendation field [2]. Nevertheless, different properties are of diverse importance for different domains; also they are of variant meaning for different recommendation tasks. Thus, considering multiple metrics to evaluate RSs becomes an indispensable need that has gained increasing attention in the literature. Nonetheless, metrics are evaluated differently (i.e., using different measures), they are of different scales, and there are tradeoffs between some of them. Also, results from different metrics may have opposite meanings (i.e., the bigger value does not necessarily mean better recommender). So, presenting the results of multiple metrics separately is inappropriate and inconvenient. All these issues complicate the comparison process, and they tangle the selection of the most appropriate approach. Hence, a comprehensive evaluation that combines multiple aspects becomes an essential need.

Motivation. Based on the issues above, several researchers highlighted the need for a unified evaluation framework to evaluate the multi-faceted properties of RSs. Alan et al. [3] argued that due to the vast variety of recommendation approaches, it is more feasible to provide a comprehensive benchmark framework. Also, the evaluation of RSs is influenced by the availability of a standard evaluation framework [2]. Therefore, the use of a comprehensive evaluation framework becomes an essential need. To the best of my knowledge, however, there is a lack of such extensive framework. To fill this gap, my Ph.D. thesis aims at proposing a comprehensive evaluation framework that combines the essential recommender's dimensions to come up with an overall evaluation result. The challenge here is to provide one framework that can evaluate the wide range of recommendation approaches.

Scope of the Work. This work focuses on providing an overall evaluation that represents an innovative combination of the existed evaluation metrics. We are not dealing with introducing new evaluation dimensions. However, we are discussing the metrics that have been already introduced to the field in order to find out the most important aspects of each type of recommenders. Also, we will not rely only on the theoretical side of the area; instead, we will focus on what has been presented in practice. The research questions hence are: **(RQ1)** what are the most popular evaluation dimensions that are discussed in practice? **(RQ2)** Which recommendation properties are the most important for the users of different types of RSs? **(RQ3)** How can we provide a unified evaluation method for most types of recommenders such that different dimensions are considered? **(RQ4)** can, and if so, how do, we merge multiple evaluation metrics of RS into a single metric that comprehensively evaluate different dimensions?

2 Proposed Solution

I propose a unified evaluation framework that aims to evaluate different recommendation types comprehensively. The proposed framework is built based on the cognitive domain of Bloom's taxonomy, a well-known hierarchy for learning objectives [4]. We refer to it as the **Cognitive-based Recommender systems Evaluation (CORE)** framework. The essence of the CORE is to analogize the Bloom's taxonomy by studying the cognitive abilities of RSs. To do so, we inferred mappings between the main activities of a recommender system (i.e., information collection, learning, and recommendation) [5] and the cognitive dimension of Bloom's taxonomy, which are: remember, understand, apply, analyze, evaluate and create. Unlike other evaluation approaches that consider only one or two evaluation metrics (such as accuracy and correctness), CORE aims at innovatively correlating many evaluation metrics to provide a single, yet, wide-ranging, indication for different properties of RSs to facilitate the evaluation process. The CORE is based basically on the mapping between humanity (presented by Bloom's taxonomy) and RSs (presented by recommender's main phases and properties); firstly, we inferred a mapping between Bloom's learning objectives and RS's main phases. Then, based on the definition of the learning objectives and RS's evaluation dimensions, we inferred another mapping that shows how strong the correlation between the dimensions and Bloom's learning objectives is. The evaluation process

starts by evaluating each metric separately using the known measures. Then these values are normalized to get a consistent scale for all metrics. After that, the value of each objective is calculated by multiplying each metric by its correlation value. Finally, the overall value is calculated as the total of all objectives.

3 Evaluation and Validation

I will demonstrate the applicability of the proposed framework empirically, based on a large variety of evaluation scenarios. The main goal of the proposed framework is to be a unified framework; so, I will focus on the replicability of the CORE results. To do so, I will extensively evaluate the stability of the results through two levels of experiments; Firstly, I will implement different experimental scenarios that vary in many aspects, including dataset used, splitting method, validation technique, etc. These experiments will be deployed on Librec¹, an open source java library for recommender systems. The reasons behind choosing Librec is because it contains 70 different RSs already implemented and it gives us wide range of evaluation and validation options which do not exist in most of the other libraries. Moreover, Librec has an efficient implementation, and it provides set of interfaces for implementing new recommenders. Secondly, I will assess stability among different frameworks; that is, I will compare the results of CORE for evaluating RSs implemented by different libraries including Librec, Lenskit², and Mahout³.

4 Expected Contribution

The main contribution of this research is a standardization of the evaluation process of recommender systems through a unified evaluation framework. To achieve this contribution, some minor contribution will be presented, these sub contributions are:

1. An innovated analogy between human beings and recommender systems.
2. A quantitative analysis of the evaluation methods presented in practice and to which extent these methods are consent to the theoretical side.
3. A study of users' expectations from recommenders to show the priority level of each evaluation metric for each type of recommender systems.

5 Current Status

As of September 2017, I did a literature review of the recommender's evaluation methodologies and the related topics. Recently, I have submitted a quantitative literature survey that shows the consistency level between theoretical and practical sides.

¹ <https://www.librec.net/>.

² <http://lenskit.org/>.

³ <https://mahout.apache.org/>.

Also, it answers the first research question (see Sect. 1). Another position paper has been submitted recently; it proposes a new comprehensive metric that is built based on the idea of correlating Bloom's taxonomy and RSs. Currently, I am working in parallel on a systematic literature review, a validation of the model, and a user study that aims at answering the second research question. The next step is to merge the results from all the aforementioned studies to build the complete version of the proposed framework.

6 Conclusion

This paper presents a summary of my Ph.D. research, which includes problem statement, motivation, proposed solution, evaluation plan and the current progress. Extensive work is still needed to come up with the full version of the framework.

References

1. Kowald, D., Lex, E.: Evaluating tag recommender algorithms in real-world folksonomies: a comparative study. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 265–268. ACM (2015)
2. Avazpour, I., Pitakrat, T., Grunske, L., Grundy, J.: Dimensions and metrics for evaluating recommendation systems. In: Robillard, M., Maalej, W., Walker, R., Zimmermann, T. (eds.) Recommendation Systems in Software Engineering, pp. 245–273. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-45135-5_10
3. Said, A., et al.: Recommender systems evaluation: a 3D benchmark. In: RUE@ RecSys (2012)
4. Karthwohl, D.R., Anderson, W.: A revision of Bloom's taxonomy: an overview theory into practice. The Ohio State University (2002)
5. Isinkaye, F.O., Folajimi, Y.O., Ojokoh, B.A.: Recommendation systems: principles, methods, and evaluation. Egypt. Inform. J. **16**(3), 261–273 (2015)



Early Detection of Alzheimer's Disease Using Deep Learning

Laura McCrackin^(✉)

Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, Canada
lmccrackin@uwaterloo.ca

Abstract. Using a combination of methods from image processing, signal processing and deep learning, we aim to develop a model to predict whether or not a patient will develop symptomatic Alzheimer's disease using Diffusion MRI (dMRI) imaging data. We first propose a 3D multi-channel convolutional neural network (CNN) architecture to distinguish patients with Alzheimer's from normal controls, then propose an extension of our architecture to incorporate multiple scans from a patient's history to improve classification accuracy and predict future prognosis. Finally, we discuss methods for performing data augmentation to add diversity and robustness to our unique and comparatively small dataset.

Keywords: Convolutional neural networks (CNNs) · Deep learning
Data augmentation · Alzheimer's disease · Computer aided diagnosis

1 Introduction

As Alzheimer's disease progresses, many structural and chemical changes become evident in the brain [2]. Unfortunately, these changes become visible when patients already have mild cognitive impairment (MCI), which is too late in the progression of the disease for any current treatment to be feasible. For this reason, diagnosis of Alzheimer's *before* the early diagnostic criteria are met is crucial to ensuring better patient care, finding ways to delay the onset of later stages, and maximizing chances that an early stage treatment may be found.

In recent years, deep learning architectures have been able to surpass human performance on even complex image identification tasks [5]. With this in mind, we aim to develop a model to predict, based on medical imaging data and other clinical information from a patient, whether or not they will develop symptomatic Alzheimer's disease. We seek first to reproduce the diagnostic capabilities of a human doctor on a single patient scan, ensuring our network can distinguish patients with Alzheimer's from normal controls. We then propose an extension of our architecture to incorporate multiple scans from a patient's history to improve classification performance and predict a future prognosis. Finally, we

Supervised by Dr. Mark Crowley and Dr. Oleg Michailovich, Department of Electrical and Computer Engineering, {mcrowley, olegm}@uwaterloo.ca.

look to address the constraints of performing deep learning on our unique and comparatively small dataset by using data augmentation to add diversity and robustness to our training data.

2 Data and Preprocessing

Our main imaging modality of interest is Diffusion MRI (dMRI), which has been used extensively for MCI and late-stage Alzheimer’s analysis in the medical literature [2]. Because the human brain is neuroplastic – that is, the white matter is capable of rewiring itself to compensate for deficiencies – we believe that dMRI holds the strongest potential for detecting the earliest, most subtle changes due to Alzheimer’s. dMRI non-invasively measures the Brownian motion of water molecules, producing diffusion equations at each voxel location to provide a plethora of information from which to extract features [6]. These include Diffusion Tensor Imaging (DTI) based features such as Fractional Anisotropy (FA) and Mean Diffusivity (MD), which characterize the magnitude and directionality of diffusion at each voxel. In an initial validation study, we showed that using a variety of dMRI-based features, we are able to demonstrate perfect separability of Alzheimer’s patients from normal controls using probabilistically weighted graphs [7]. This demonstrates the strong potential of these features for use with more sophisticated classifiers.

Our primary dataset, the Alzheimer’s Disease Neuroimaging Initiative dataset (ADNI), is a longitudinal, multi-site study designed to track the onset and development of Alzheimer’s Disease. It includes dMRI as one of its modalities [8].

As shown in Fig. 1, we begin by generating our multi-channel 3D feature maps, which will later serve as the inputs to our classifier system. Each of N patient scans P_i , where $i = 1 \dots N$, must be processed using a preprocessing pipeline. This pipeline performs steps such as alignment and eddy current correction, along with fitting the dMRI data to the DTI model, and outputs several 3D feature maps. These feature maps are considered together as being a multi-channel 3D input volume, S_i .

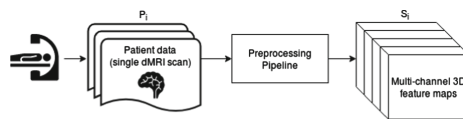


Fig. 1. Preprocessing each patient scan to obtain multi-channel 3D data.

3 Classification Architectures

Next, we consider two different types of classification experiment. In the first, we consider the case where one scan from a single point in time is considered for

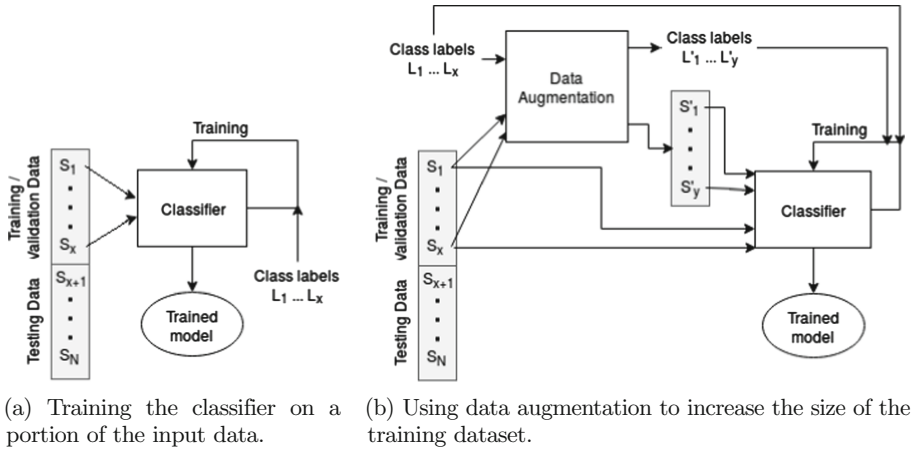


Fig. 2. Training the classifier architecture.

each patient. A portion of the data, $S_1 \dots S_x$, is used as our training data, with a portion of these samples being reserved for validation to monitor the training progress.

As can be seen in Fig. 2a, the classifier is trained in a supervised manner using these training points in combination with their corresponding class labels, $L_1 \dots L_x$. For example, these labels could correspond to whether the patient currently has Alzheimer’s disease (1) or does not (0). The remaining $S_{x+1} \dots S_N$ samples are used as testing points, and are used to evaluate the classifier’s accuracy only after the training process has been completed.

We consider the Voxception-Resnet (VRN) architecture by Brock et al. as our starting point in defining this architecture [1], due to its proven effectiveness compared with 2D slice or projection-based classifiers on 3D image datasets. Our main contributions here will consist of modifications to the VRN’s architecture and refinements of its parameters and tuning for more effective performance on our dataset. In particular, we will vary the network depth, input features selected and activation functions used.

In the second type of classification experiment, we consider each training, testing or validation sample to be a set of multi-channel 3D feature maps, generated from a series of scans from a single patient. For instance, this could be a series of scans taken at 6-month intervals. A single class label accompanies each series of scans – for instance, whether the patient had been diagnosed with Alzheimer’s at the time the final scan was taken (1) or had been cognitively normal (0). As a more complex example, the class labels could be whether the patient was diagnosed with symptomatic Alzheimer’s one year following the final scan (1) or maintained a diagnosis of MCI (0).

This involves integrating the notion of time into our first classifier architecture. To do this, we propose an architecture inspired by the Long-term Recurrent Convolutional Network (LRCN) architecture by Donahue et al. [4], which would

allow for using long short-term memory (LSTM) units to combine a number of VRN classifiers. To our knowledge, this would be the first architecture designed for use with a temporal set of multi-channel 3D images.

4 Data Augmentation

Our next contribution consists of developing methods and techniques to augment our limited dataset, and thus enable our classifier architectures to achieve better classification performance. Most existing methods for data augmentation would not be applicable to our dataset due to its complex nature, and this work would provide valuable insight not only into augmentation for deep learning in our specific domain, but for other complex domains as well.

A high-level overview of how data augmentation may be used in a classification experiment is shown in Fig. 2b. Before the classifier is trained, the training and validation data $S_1 \dots S_x$ and corresponding class labels $L_1 \dots L_x$ are used as inputs for the data augmentation process, which generates a number of synthetic data samples, $S'_1 \dots S'_y$, and corresponding labels $L'_1 \dots L'_y$. These synthetic data points increase the number of training and validation samples from x to $x + y$.

We will begin by applying some commonly-used forms of data augmentation which do make sense on our dMRI data, including additive white Gaussian noise (AWGN) and flipping input images along the sagittal plane (mirroring the left and right sides of the brain). However, we seek to propose additional augmentation methods as well.

The concept of interpolating and extrapolating data points to allow for domain-agnostic augmentation has been shown by Devries and Graham to be effective in several classification tasks [3]. However, their method does not perform well on image data used with CNNs, and thus cannot be directly applied to dMRI data.

One potential approach is *inter-patient interpolation and extrapolation*. Patients sharing similar characteristics and the same diagnostic class can have their feature spaces interpolated to produce a new, plausible mixture of the two, which can be assigned the same classification label. Another method is *temporal interpolation and extrapolation*, between two scans from the same patient taken at different times. Intuitively, this allows for interpolation or extrapolation along a patient's timeline: given scans from 6 and 12 months into the ADNI study, we should be able to approximate scans from 9 months, or from before the study began. Both approaches would require ensuring that the resulting output is meaningful and within reasonable feature values for its given class.

5 Conclusion

Our research seeks first to reproduce the diagnostic capabilities of a human doctor on a single patient scan, and then later to extend them, predicting from a scan history the trajectory of a patient's diagnosis. We then seek to improve our method's performance using novel approaches to data augmentation. It is our

hope that this work not only serves as a step forward in using deep learning for predicting Alzheimer's disease, but also produces methodologies and techniques that prove useful for a variety of spatiotemporal domains.

References

1. Brock, A., Lim, T., Ritchie, J., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint [arXiv:1608.04236](https://arxiv.org/abs/1608.04236) (2016)
2. Chua, T.C., et al.: Diffusion tensor imaging in mild cognitive impairment and Alzheimer's disease: a review. *Curr. Opin. Neurol.* **21**(1), 83–92 (2008)
3. DeVries, T., Taylor, G.: Dataset augmentation in feature space. arXiv preprint [arXiv:1702.05538](https://arxiv.org/abs/1702.05538) (2017)
4. Donahue, J., et al.: Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2625–2634 (2015)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
6. Jones, D.K.: Diffusion MRI. Oxford University Press, New York (2010)
7. Maryam, S., McCrackin, L., Crowley, M., Rathi, Y., Michailovich, O.: Application of probabilistically-weighted graphs to image-based diagnosis of Alzheimer's disease using diffusion MRI. In: SPIE Medical Imaging, pp. 101342–101342–10. International Society for Optics and Photonics (2017)
8. Petersen, R.C., Aisen, P., Beckett, L., et al.: Alzheimer's disease Neuroimaging Initiative (ADNI) clinical characterization. *Neurology* **74**(3), 201–209 (2010)



Learning with Prior Domain Knowledge and Insufficient Annotated Data

Matthew Dirks^(✉) 

University of British Columbia, Vancouver, Canada
mcdirks@cs.ubc.ca

Abstract. Machine learning exploits data to learn, but when not enough data is available (often due to increasingly complex models) or the quality of the data is insufficient, then prior domain knowledge from experts can be incorporated to guide the learner. Prior knowledge typically employed in machine learning tends to be concise, single statements. But for many problems, knowledge is much more messy requiring in-depth discussions with domain experts to extract and often takes many iterations of model development and feedback from experts to collect all the relevant knowledge. In the Bayesian learning paradigm, we learn which hypotheses are most likely given the data as evidence. How can we refine this model when new feedback is given by domain experts? We are working with domain experts on a problem where data is expensive, but we also have prior knowledge. This research has two objectives: (1) automatically refine models using prior knowledge, and (2) handle various forms of prior knowledge elicited from experts in a unified framework.

Keywords: Prior knowledge · Domain knowledge · Bayesian learning
Machine learning · Domain experts · Uncertainty · AI applications

1 Introduction

Learning from data is core to reasoning in AI and Machine Learning; supervised learning, in particular, is standard practice, where a model is trained using a very large dataset consisting of inputs and outputs (also known as annotated or labelled data). For example, in classifying spam messages, the data should contain instances of spam messages and instances of regular messages. Or, in predicting the price of a house, the data should contain many instances of houses' specifications and corresponding prices.

While the exact number of instances needed varies, it is known that more complex models require more training instances [1, 2]. Roughly speaking, it is recommended to have hundreds of thousands or more.

“If you only have 10 examples of something, it’s going to be hard to make deep learning work. If you have 100,000 things you care about, records or whatever, that’s the kind of scale where you should really start thinking about these kinds of techniques.”¹

¹ Jeff Dean, Google Senior Fellow at Google Brain, <https://goo.gl/AWfsrK>.

But what if, for some task, you don't have much data? There are a few options: (1) get more data (e.g. crowd-sourcing or spending more money), (2) change the model to one that needs less data (usually a simpler one), (3) augment with other data (e.g. transfer learning, semi-supervised learning, or synthetic data) and (4) teach the machine in some other way.

When the first three options are not viable—for reasons such as cost of data and uniqueness of the task—then we look to option 4. More specifically, we can better teach the machine by incorporating prior knowledge. Prior knowledge is often used to enhance machine learning performance [3,4,6,8,9] and will better tailor the system to the target domain to allow knowledge discovery and impact to science and society [4].

Often the terms domain knowledge, prior knowledge, and prior domain knowledge have been used interchangeably and refer to any knowledge that is useful to the problem at hand. However, [5] categorizes prior knowledge into two types: solution knowledge, which is specific to the target or learning objective (such as the structure of a Bayes net or architecture of a neural network), and domain knowledge, which describes the world specific to the domain. We will use the term prior knowledge in this paper, as we expect both solution knowledge and domain knowledge to be provided by domain experts.

2 Example: Estimating Elemental Composition of Rocks

To begin solving these research problems, we should first understand what information domain experts can provide and what it might look like to represent and incorporate it. We have been working with domain experts on a problem in mining—involving geology, mineralogy, and physics domains—on the task of estimating elemental composition of rocks from spectra measurements via X-ray Fluorescence (XRF). In lab environments there are established techniques to compute exact composition, but our environment is harsh, noisy, and dirty inside open-pit mines and underground.

In our case, one instance of training data requires sending a 100 metric-tonne mining shovel equipped with X-ray sensors to a mine, digging one scoop of rocks, transporting it to a rock crushing facility, crushing it into fine powder and thoroughly mixing, and finally sending the powder to an assay lab for analysis which determines the geochemical composition. This is an expensive process. We have a couple hundred data points to work with—not much by today's standard.

However, we also have access to domain experts with relevant expertise for solving the desired task. A summary of the prior knowledge provided upfront is as follows: (1) each element in the periodic table produces a set of “peaks” centered at known energies, (2) the set of element peaks occur with known ratios, given by the probability of electrons fluorescing for each transition, (3) peaks are Lorentzian shaped, and (4) spectra from elements and elements' transition peaks are summed in the final spectrum.

We built a simulator that reconstructs a spectrum given an elemental composition based on this prior knowledge and built a probabilistic model around it. Under the Bayesian paradigm, the posterior distribution is defined as

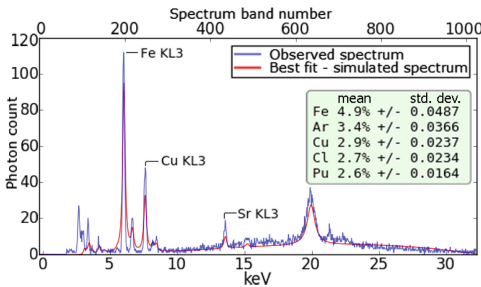
$$P(h|e) = \frac{P(e|h)P(h)}{\sum_{h \in H} P(e|h)P(h)} \tag{1}$$

where h is a hypothesis (elemental composition in our case), e is evidence (i.e. training data), H is the hypothesis space, $P(h)$ is the prior distribution, and $P(e|h)$ is the likelihood distribution.

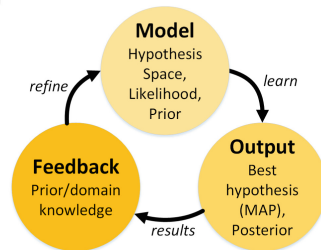
We solved for the posterior using MCMC (implemented in Anglican, a probabilistic programming system [7]). An example of one instance’s elemental composition and the corresponding *maximum a posteriori* estimate as a spectrum is shown in Fig. 1a.

After seeing the output of the model, the domain experts provided additional information, including: (1) matrix effects may cause the amount of one element to increase the photon count of others, (2) 20–21 keV contains Compton peaks, (3) Raleigh peaks may occur at 21–24 keV, (4) peaks between 2–5 keV are due to thermal effects, (5) attenuation and detection efficiency will cause lower and higher keV bands to lose photon counts, (6) peak shape may actually be a combination of Lorentzian and Gaussian, (7) Bragg scattering may occur, and (8) some elements are more likely than others, based on location.

Incorporating this knowledge is not straightforward. Item 8 may fit in nicely as a prior over elements in the hypothesis space, but the rest require changes to the model beyond probabilistic priors. We are investigating methods to allow prior knowledge to alter the hypothesis space, likelihood function, or prior as needed (Fig. 1b).



(a) Example XRF spectrum, reconstruction from MAP estimate, and table of top 5 elements from posterior.



(b) Proposed feedback cycle.

Fig. 1. Example model results and proposed feedback cycle.

3 Discussion

We found that domain experts did not provide all the relevant information for the problem upfront. We also found that prior knowledge is messy and not straightforward to incorporate. Prior knowledge is not always given *a priori*, but instead is given after an initial model is made (or several revisions later). Predicting elemental composition from sensor data with prior knowledge given before and after model development demonstrates the need to iteratively refine models given updated prior knowledge.

How do we elicit this domain-expertise from an expert? Can we gather all the knowledge at the beginning, or should we iteratively acquire it? Can we automatically refine a model given new prior knowledge? How can we exploit all kinds of prior knowledge to better train a model? Can we handle all the various forms of prior knowledge that experts provide? We are working on multiple problems in related domains which we will use to generalize toward a systematic technique to elicit and incorporate prior knowledge—defining a clear path to download relevant domain-expertise into a machine learner.

Acknowledgments. Thanks to my academic supervisor, David Poole; David Turner, David Munoz-Paniagua, Peter How, and others for their domain expertise; and to MineSense Technologies Ltd. for use of their sensors and rocks samples.


References

1. Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C., Popp, J.: Sample size planning for classification models. *Anal. Chim. Acta* **760**(Supplement C), 25–33 (2013)
2. Figueroa, R.L., Zeng-Treitler, Q., Kandula, S., Ngo, L.H.: Predicting sample size required for classification performance. *BMC Med. Inform. Decis. Making* **12**, 8 (2012). <https://doi.org/10.1186/1472-6947-12-8>. [PII]
3. Niyogi, P., Girosi, F., Poggio, T.: Incorporating prior information in machine learning by creating virtual examples. *Proc. IEEE* **86**(11), 2196–2209 (1998)
4. Rudin, C., Wagstaff, K.L.: Machine learning for science and society. *Mach. Learn.* **95**(1), 1–9 (2014). <https://doi.org/10.1007/s10994-013-5425-9>
5. Sun, Q., DeJong, G.: Explanation-augmented SVM: an approach to incorporating domain knowledge into SVM learning. In: *Proceedings of the 22nd International Conference on Machine Learning*, pp. 864–871. ACM (2005)
6. Teng, T.H., Tan, A.H., Zurada, J.M.: Self-organizing neural networks integrating domain knowledge and reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **26**(5), 889–902 (2015)
7. Tolpin, D., van de Meent, J.W., Yang, H., Wood, F.: Design and implementation of probabilistic programming language Anglican (2016). arXiv preprint [arXiv:1608.05263](https://arxiv.org/abs/1608.05263)
8. Yu, T., Simoff, S., Jan, T.: VQSVM: a case study for incorporating prior domain knowledge into inductive machine learning. *Neurocomputing* **73**(13), 2614–2623 (2010)
9. Zhou, Y., Tan, L.: Incorporating prior knowledge into extension neural network and its application to recognition of safety status pattern of coal mines. *Int. J. Signal Process. Image Process. Pattern Recogn.* **84**, 307–324 (2015)

Industry Track



Predicting Crime Using Spatial Features

Fateha Khanam Bappee¹  , Amílcar Soares Júnior¹ ,
and Stan Matwin^{1,2} 

¹ Institute for Big Data Analytics, Dalhousie University, Halifax, Canada
bappeenstu@gmail.com, ft487931@dal.ca

² Institute for Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract. Our study aims to build a machine learning model for crime prediction using geospatial features for different categories of crime. The reverse geocoding technique is applied to retrieve open street map (OSM) spatial data. This study also proposes finding *hotpoints* extracted from crime *hotspots* area found by Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). A spatial distance feature is then computed based on the position of different *hotpoints* for various types of crime and this value is used as a feature for classifiers. We test the engineered features in crime data from Royal Canadian Mounted Police of Halifax, NS. We observed a significant performance improvement in crime prediction using the new generated spatial features.

1 Introduction

In recent years, with the availability of high volume of crime data, scientists have been motivated to pursue research in the field of crime and criminal investigations. Understanding the factors related to different categories of crimes and their consequences is particularly essential. The study shown in [7] applies the procedure of statistical analysis on violent crime, poverty, and income inequality and outlines that homicide and assault has more connection and correlation with poverty or income inequality than other crimes. The research found that crime in the real-world highly correlates with time, place and population which make the researcher's task more complicated [3]. Moreover, this geographical and demographic information contain many discriminatory decision pattern [6, 10]. Leveraging data mining and machine learning techniques with crime research offer the analysts the possibility of better analysis and crime prediction, as well as mining association rules for crime pattern detection.

Our study aims to build a machine learning model to predict the relationship between criminal activity and geographical regions. We choose Nova Scotia (NS) crime data as the target of our study. We focus on four different categories of crime: (i) alcohol-related; (ii) assault; (iii) property crime; and (iv) motor vehicle. In this work, we focus on the creation of two spatial features to predict crime: (i) geocoding; (ii) crime *hotspots*.

The contributions of this work include how geocoding can be used to create features using OSM data and crime *hotspots* are created using a density-based

clustering algorithm. Moreover, *hotpoints* are extracted from the *hotspots*. We show using a real-world scenario that these two new features increase the performance of different classifiers for predicting four different types of crime.

2 Related Work

The existing work on crime prediction can be categorized into three different groups based on the features such as temporal, spatial and demographic aspects.

Bromley and Nelson [1] reveal temporal patterns of crime to predict alcohol-related crime in Worcester city. They also provide valuable insight into the spatial characteristics of the alcohol-related crime. The authors examine the patterns of crime and disorder at street level by identifying *hotspots*. Ratcliffe [11] proposes three types of temporal and spatial *hotspots* for crime pattern detection. The author also shows how the spatial and temporal characteristics combine through his *hotspot* matrix. However, the author did not apply any machine learning strategy to predict crime.

In [2], the authors analyze four categories of crime data which include liquor law violations, assaults and batteries, vandalism, and noise complaints. Different categories of crime show different temporal patterns. Brower and Carroll [2] clarify crime movement through the city of Madison using GIS mapping. The authors investigate the relationships among high-density alcohol outlets and different neighborhoods. Chainey et al. [5] identify crime *hotspots* using Kernel Density Estimation (KDE) to predict spatial crime patterns. Similarly, in another study, Nakaya and Yano [8] create crime *hotspots* with the help of KDE. However, they combine temporal features with crime *hotspots* analysis.

Nath [9] employed a semi-supervised clustering technique for detecting crime patterns. In [13], the authors propose a pattern detection algorithm named Series Finder to detect patterns of a crime automatically. In [12], the authors study crime rate inference problem using Point-Of-Interest data and taxi flow data. Point-Of-Interest data and taxi flow data are used to enhance the demographic information and the geographical proximity correlation respectively.

None of these features reported in the section were used for predicting crime categories alongside crime pattern detection. In our research, we mainly focus on the spatial aspect of crime prediction. We use geocoding technique and crime *hotspots* to generate new features.

3 Engineering Spatial Features

Geocoding is the process of spatial representation of a location by transforming descriptive information such as coordinates, postal address, and place name. The geocoding process relies on GIS and record linkage of address points, street network and boundaries of administrative unit or region. For this work, we used geocoding to extract the spatial information from the crime data. The geocoder library written in Python, was used for geocoding services with the Open Street Map (OSM)

provider. The output of the Geocoder package can be 108 types of location including pubs, bus stops, or hospitals from NS. According to OSM documentation, all of these types are grouped into 12 categories including amenity, shop, office etc. We used both types of location and category as features to predict crime.

The second type of feature used in this work was the creation of *hotspots*. *Hotspot* analysis can emphasize the patterns of data regarding time and location of a geographic area. For a crime analyst, the creation of *hotspots* became very popular to identify high concentrated crime area. In this work, *hotspots* are created and transformed into a feature to predict different crime types. The idea is to cluster crime data into regions with a high rate of occurrence of the same crime type. We decided to use HDBSCAN [4] because of its complexity ($O(n \log n)$) and because it can handle data with variable density and eliminates the ϵ (*eps*) parameter of DBSCAN which determines the distance threshold to cluster data. In this work, we used the Haversine distance in both HDBSCAN and shortest distance to a *hotpoint*. The haversine formula determines the shortest distance between two points on earth located by their latitudes and longitudes.

Figure 1 summarizes the overall process to produce the shortest distance for *hotpoint* feature. Figure 1(a) shows crime examples (gray pins) in downtown Halifax area. Then, a *hotspot* (blue area) found by HDBSCAN is shown in Fig. 1(b).



Fig. 1. An overview of the crime *hotspots*, *hotpoints* and distance to *hotpoint* feature. (Color figure online)

Figure 1(c) shows a *hotpoint* (red pin) extracted from a *hotspot*. Finally, a new crime example (green pin) is evaluated, and the distances to *hotpoints* (yellow line) are calculated. The feature used in this work will select the shortest distance to a *hotpoint* as a feature for classifying a crime type.

4 Experiments

This section outlines the experiments performed in this work and reports the experimental results obtained by the proposed classifiers trained on all raw features and the engineered spatial features.

Crime data from Halifax regional police department are used in this work, and it covers most of the districts in Nova Scotia province in Canada. For our experiments, we explore all of the offenses of 2016 which include 3726 data samples. The crime attributes extracted from the source data include geographic location, `incident_start_time`, `month`, `weekday`, `ucr_descriptions`, and whether the incident happened because of alcohol.

We also group our data using four different classes, named alcohol-related, assault, property damage, and motor vehicle using the `ucr_descriptions` and `alcohol incident` fields. For the alcohol-related crimes, we considered all the cases where alcohol presence was reported in the UCR using the `alcohol incident` field (53% alcohol, 47% no alcohol). For all the remaining classes, the `ucr_description` field was used. The assault group (65% assault, 35% no assault) covers all levels of assault including sexual assault, aggravated assault, bodily harm, threat, etc. Property damage group covers break, theft, robbery, etc (65% property damage, 35% no property damage). Motor vehicle group covers all types of motor vehicle accident, act violation and impair driving (65% motor vehicle, 35% no motor vehicle).

To create the shortest distance to a *hotpoint*, we used UCR form data from the year of 2015. We created *hotspots* for each positive class and the respective shortest distance to a *hotpoint* was used in the experiment.

The classifiers used in this work are Logistic Regression (LR), Support Vector Machine (SVM), and Random Forest (RF) and an Ensemble with all the previous classifiers. We evaluate the classifiers' performance using the accuracy and Area Under the Curve (AUC) of the ROC (Receiving Operator Characteristic) analysis. The baseline used in this work to verify if the newly engineered features help a classifier to improve the crime prediction power was the raw data contained in the UCR form (`incident_start_time`, `month`, and `weekday`). A 10-fold cross-validation was used in all phases to estimate model prediction performance correctly and paired t-tests (significance level of 0.05) were used to test the statistical difference significance of raw and engineered features.

Table 1 shows the classification accuracy for LR, SVM, RF and an ensemble of these methods for all four categories of crime. For each method, the first column displays the accuracy of raw features and the second column for engineered spatial features. The * in Table 1 symbol indicates that the method fails for the statistical hypothesis testing, i.e., the p-value is higher than 0.05.

For the Alcohol-related group, the results show that new spatial features achieve better accuracy in comparison with raw features for all four methods with statistical evidence support, and the Ensemble method performs better than others (75.52% of accuracy) with almost 17% accuracy improvement. The accuracy values of the engineered features for the Assault and Property damage groups shows that all methods, except LR, benefit from their inclusion. For example, adding engineered features with raw features improves nearly 11% (Assault group) and 5% (Property damage group) of accuracy for RF method. Finally, for the Motor vehicle group, all the classifiers showed improvements, except for the Ensemble classifier.

Table 1. Results for accuracy

Crime type	LR		RF		SVM		Ensemble	
	Raw	Eng.	Raw	Eng.	Raw	Eng.	Raw	Eng.
Alcohol-related	59.36	65.27	57.73	73.51	59.28	71.31	58.61	75.52
Assault	65.35	65.03*	47.94	58.89	63.53	65.27	55.96	64.41
Property damage	88.43	88.41*	84.03	88.57	88.19	88.43	88.43	88.44*
Motor vehicle	81.59	82.31	71.82	81.45	81.11	81.45	81.56	81.80*

Table 2 shows the AUC scores for LR, SVM, RF and an ensemble of LR, SVM & RF methods. The * in Table 2 symbol indicates that the method fails for the statistical hypothesis testing. For Alcohol-related and Motor vehicle crimes, the results discovered that spatial features give better AUC scores than raw features for all four methods. For instance, the Ensemble method gives 82.5% and 69.4% AUC score for Alcohol-related and Motor vehicle crimes respectively based on engineered features. Similarly, for Assault and Property damage crime, LR, RF and Ensemble methods perform significantly better with engineered features. Adding engineered features with raw features gives 56.7% and 65.7% AUC score for Assault and Property damage crime respectively with the Ensemble method. Therefore, using spatial features, the Ensemble method performs at least 10% improvement in AUC score for all four categories of crime. However, for SVM method, there is no significant evidence of improvement.

Table 2. Results for AUC

Crime type	LR		RF		SVM		Ensemble	
	Raw	Eng.	Raw	Eng.	Raw	Eng.	Raw	Eng.
Alcohol-related	.575	.723	.649	.818	.635	.747	.661	.825
Assault	.528	.613	.457	.545	.504	.533*	.459	.567
Property damage	.519	.651	.531	.646	.501	.505*	.534	.657
Motor vehicle	.515	.686	.488	.682	.494	.536	.490	.694

5 Conclusions and Future Work

In this work, we explored the creation of spatial features derived from geolocated data. We created two types of spatial features: (i) geocoding; and (ii) shortest distance to a *hotpoint*. The new features were evaluated using four different crime types using only the information provided in the UCR forms as features for a classifier as the baseline. The results show that significant improvements in accuracy and AUC were found when the newly engineered features were added to the tested classifiers.

We intend to extend this work in other directions. As our study focuses on real world datasets, the subject of data discrimination is another important concern. Data discrimination refers to bias that happens because of contradistinction among different data sources. Another research direction we want to explore is the possibility of performing transfer learning from what was learned in NS to other Canadian provinces.

Acknowledgments. The authors would like to thank NSERC, NS Health Authority and Injury Free Nova Scotia for financial and other supports.

References

1. Bromley, R.D., Nelson, A.L.: Alcohol-related crime and disorder across urban space and time: evidence from a British city. *Geoforum* **33**(2), 239–254 (2002)
2. Brower, A.M., Carroll, L.: Spatial and temporal aspects of alcohol-related crime in a college town. *J. Am. Coll. Health* **55**, 267–275 (2007)
3. Buczak, A.L., Gifford, C.M.: Fuzzy association rule mining for community crime pattern discovery. In: ACM SIGKDD Workshop on Intelligence and Security Informatics, ISI-KDD 2010, pp. 2:1–2:10. ACM, New York (2010)
4. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7819, pp. 160–172. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_14
5. Chainey, S., Tompson, L., Uhlig, S.: The utility of hotspot mapping for predicting spatial patterns of crime. *Secur. J.* **21**(1), 4–28 (2008)
6. Executive Office of the President: Big Data: A Report on Algorithmic Systems, Opportunity, and Civil Rights, 2nd edn. CreateSpace Independent Publishing Platform (2016)
7. Hsieh, C.C., Pugh, M.D.: Poverty, income inequality, and violent crime: a meta-analysis of recent aggregate data studies. *Crim. Justice Rev.* **18**(2), 182–202 (1993)
8. Nakaya, T., Yano, K.: Visualising crime clusters in a space-time cube: an exploratory data-analysis approach using space-time kernel density estimation and scan statistics. *Trans. GIS* **14**(3), 223–239 (2010)
9. Nath, S.V.: Crime pattern detection using data mining. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IATW 2006, Washington, DC, USA, pp. 41–44. IEEE Computer Society (2006). <https://doi.org/10.1109/WI-IATW.2006.55>

10. Pedreschi, D., Ruggieri, S., Turini, F.: Measuring discrimination in socially-sensitive decision records. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2009, Sparks, Nevada, USA, 30 April–2 May 2009, pp. 581–592 (2009)
11. Ratcliffe, J.: The hotspot matrix: a framework for the spatio-temporal targeting of crime reduction. *Police Pract. Res.* **5**(1), 5–23 (2004)
12. Wang, H., Kifer, D., Graif, C., Li, Z.: Crime rate inference with big data. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016, pp. 635–644 (2016). <https://doi.org/10.1145/2939672.2939736>
13. Wang, T., Rudin, C., Wagner, D., Sevieri, R.: Learning to detect patterns of crime. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part III. LNCS (LNAI), vol. 8190, pp. 515–530. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_33



A Tool for Defining and Simulating Storage Strategies on the Smart Grid

Dan Russell and Aaron Hunter^(✉)

British Columbia Institute of Technology, Burnaby, Canada
danr94@gmail.com, aaron_hunter@bcit.ca

Abstract. Intelligent distribution of electrical power is a key problem on the Smart Grid. It is known that the introduction of micro-storage devices, such as electric cars, can lead to benefits for consumers both in terms of power cost and emissions output. This process requires consumers to be educated on the importance of power storage, and it also requires the development of intelligent power storage strategies. This paper introduces a simulation tool that can be used to achieve both of these goals. In particular, our software allows users to easily define a Smart Grid topology, and then use a simple scripting language to define intelligent power storage strategies for simulated consumers. The software permits evaluation of different strategies, which can lead to practical improvements for consumers.

1 Introduction

It is well-known that power storage offers many benefits on the Smart Grid, including a reduction in greenhouse emissions and a reduction in power cost for individual consumers [4, 5]. We describe *Flow*, a tool for modelling a power grid and experimenting with intelligent power storage strategies. In order to achieve these benefits, we must be able to develop suitable storage algorithms. We argue that there is value in implementing experimental tools to develop such algorithms. One advantage of this approach is that we are able to gain a better understanding of storage strategies, by manipulating the factors that influence the efficiency of power storage. A second advantage is that an experimental tool allows us to provide a visual simulation to typical consumers, in order to demonstrate the practical advantages. This paper is primarily a system description, in which we introduce new software to address an important problem on the Smart Grid.

We make several contributions to existing work on intelligent power storage. First, we provide a practical tool that can be used to experiment with power storage strategies. Our tool is flexible in that it allows the user to easily specify grid topologies and also to define new sources of power with different characteristics. Second, we use a high-level scripting language to specify strategies in a simple manner. The formalism for specifying strategies is simple enough to be quickly learned by any user with a basic background in programming.

2 Basic Storage Strategies

The Smart Grid promises to allow users to purchase and trade power flexibly at different times of the day. This can solve many economic and environmental problems, because it allows users to avoid demand bottlenecks. For example, on the current grid, most homes consume energy in the morning and evening. If we have renewable sources like solar or hydro power, this can be problematic due to the fact that power production can be exhausted. To make up for this, sources like coal are used to make up the difference. If we could spread power usage around the clock more evenly, we could maximize the use of renewables.

This change can even be made at the level of individual power users, provided that they have access to power storage devices. However, simply having a power storage device is not sufficient; we need to define and evaluate storage strategies that will actually be useful. The goal of this paper is to demonstrate how this can be accomplished.

We define three *basic storage strategies* that can be used with a storage device.

1. **Greedy:** Charge our storage device until it is full and then we use it until it is empty.
2. **Local Average Matching:** Maintain a consistent level of power usage all day. Informally, this means charging a little bit and using a little bit at all times.
3. **Scheduled:** Explicitly specify the times when a battery will charge or discharge.

If all consumers need power at the same time, it seems unlikely the greedy strategy will result in any efficiency gains. The utility of the other strategies depends on the network structure and capacity. Consider the local average matching strategy, for example. If the total power usage of all users is less than the renewable power capacity, then this strategy is actually ideal. Scheduling is particularly useful in the case of solar or wind energy, when we know that certain times of day will have more power available.

Given the complexity of the problem, it is actually hard to evaluate storage strategies in the abstract. But if we can not evaluate strategies, how can we expect users to learn how to use batteries and other storage devices effectively? To address this problem, we suggest that a simulation tool is useful. We describe our simulator in the next section.

3 Simulation

Flow is a Smart Grid simulation that allows us to define and manipulate power storage strategies. The simulation allows the user to create and place different items in the world. Each item has different properties related to power usage and power generation. Unless otherwise noted, power usage and consumption is measured in Kilowatt hours (kWh). The following primitive types of object are defined. For each object type, we give the name followed by the required properties for every instance.

- **Appliances**
 - *Name* - a string identifier.
 - *Standby consumption* - consumption when not in use.
 - *Usage consumption* - consumption when in use.
- **Storage Devices**
 - *Name* - a string identifier.
 - *Transfer capacity* - Kilowatts that can be stored/released per minute.
 - *Storage capacity* - Kilowatts that can be stored.
 - *Storage strategy* - Storage pattern being used.
- **Power Plants**
 - *Name* - a string identifier.
 - *Emission rate* - greenhouse gasses produced in grams per kWh.
 - *Cost* - cost to operate in dollars per kWh.
 - *Capacity* - maximum output at any instant.

In addition to these primitive types objects, the simulation also allows the user to define **buildings**, which are really just containers that aggregate a collection of appliances and storage devices. Internally, one more primitive item type is required: **time spans**. A time span specifies a list of time intervals for each day of the week.

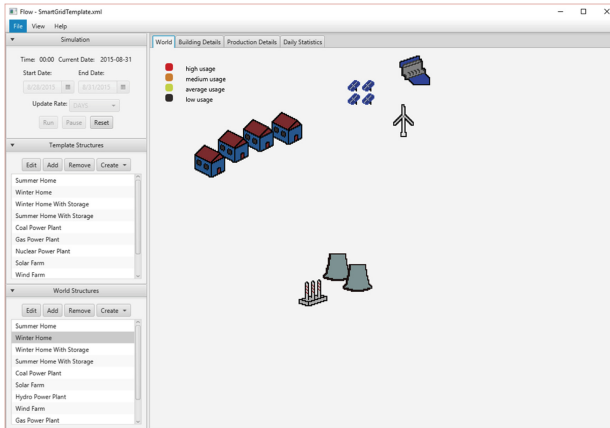


Fig. 1. Flow interface

We briefly walk through the main steps to set up and run a simulation.

1. Open Flow and select File → New from the menu.
2. Create a building.
 - Open the World Structures tab, then click the create building button.
3. Give it an appliance.
 - (a) Select the Building from the World Structures list.
 - (b) Click the Add Appliance button.

- (c) Follow the dialogues to specify its properties.
4. Create a Power Plant (Follows steps and dialogues, as for buildings).
5. Specify simulation details.
 - Specify start time, end time, and frequency of data updates.
6. Run the simulation

This basic process is the same for every simulation. Figure 1 gives an screen capture of the interface used to create and view a simulation.

Since steps 1–4 are typically repeated several times, Flow also allows the user to define re-usable templates for commonly used buildings. For example, the software comes with a pre-defined template for single family homes.

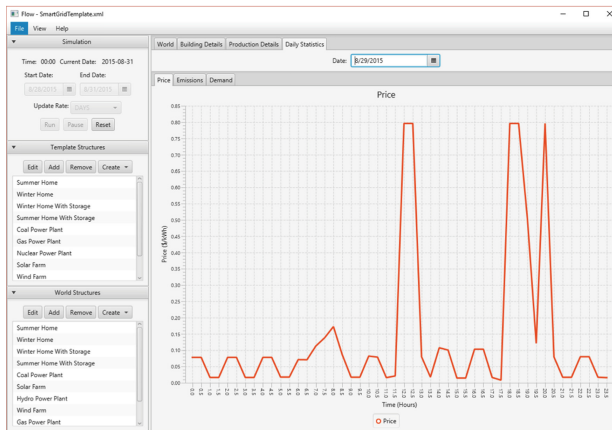


Fig. 2. Price chart

As the simulation runs, there are a variety of visual cues to indicate what is happening. For example, a power plant that is running will display an animation whereas an idle power plant will be static. Buildings can have four potential states: high usage, medium usage, mild usage and low usage. Each of these states is communicated in the world view by the current colour of window on a building; dark gray for low usage, yellow for mild usage, orange for medium usage and red for high usage. Each of these values is relative to the current day in the simulation.

After the simulation completes, Flow produces summary data that can be viewed on the Daily Statistics pane:

- The *Price Chart* displays cost (\$/kWh) versus time, to show power costs over the day. A sample chart is in Fig. 2.
- The *Emissions Chart* displays emissions (g/kWh) versus time, to show the greenhouse gasses per kWh over the day.
- The *Demand Chart* displays demand versus time, to show how needs during the day. A sample chart is in Fig. 3.

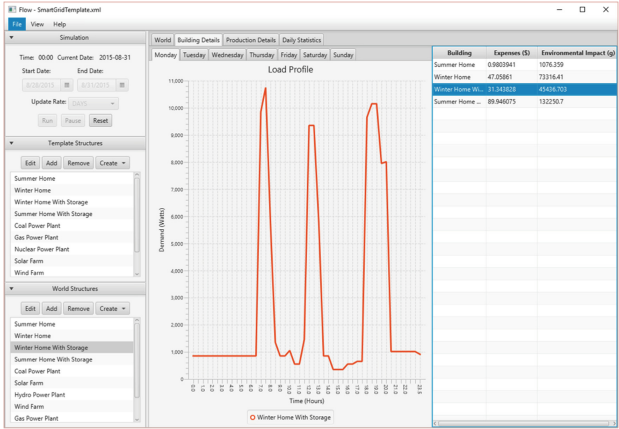


Fig. 3. Demand chart

4 Simulating Power Storage Strategies

The main goal of the Flow simulation is to allow a user to experiment with power storage. In principle, there are two approaches to solving this problem. One approach is to design intelligent storage strategies at the outset, based on known computing algorithms. The second approach is to use Flow as the basis for a learning algorithm that discovers optimal storage strategies. In this paper, we take the former approach as it allows users to experiment and learn about power storage in a hands-on manner. We leave the latter for future work.

The Flow interface allows the user to select a storage device, and assign it a storage strategy. The three basic strategies from Sect. 2 are provided by default, to serve as illustrative examples. All strategies are written in the Lua scripting language; the scripts themselves are included with the application. LuaJ is the interpreter that allows Lua scripts to interact with Flow, which is written in Java.

1. **Greedy:** A greedy device will charge to maximum capacity when it is empty. This strategy does not respond to demand prices or peaks.
2. **Local Average Matching:** Flow calculates the average demand for a given device during the current day. The strategy will charge whenever demand is lower than the average and release energy whenever the demand is higher than the average.
3. **Scheduled:** Uses a series of time spans when it will charge energy and a series of time spans when it will release energy.

In addition to these built-in strategies, Flow allows new strategies to be created through Lua scripts and stored in the Strategies directory. The application

will read through all the files in this directory when the Edit Building dialog is opening. The following is a simple storage script:

```
local minutesOfDay = 1439;
for minute = 0, minutesOfDay do
local transferAmount = 0;
-- (transferAmount Calculated Here)
newStorageProfile:add(transferAmount)
end
```

This script is read as follows. If the variable `minute` is equal to 30 and the `transferAmount` has been calculated as -5000 , then the storage device will release 5000 W of energy at the 30th min of the next day.

5 Discussion

5.1 Related Work

The benefits of power storage have been demonstrated in the work of others [3–5]. The challenge now is how we can get individual consumers to actually use storage devices effectively. Our software is a step in that direction. The most closely related work to our own is the development of the Power Storage Simulator (PSS) [1]. However, our focus here is different. The PSS is focused not only on strategy, but also in demonstrating the utility of formal ontologies like OWL [2]. While PSS provides a flexible model of the Smart Grid, it is very hard to introduce new storage strategies. By contrast, the approach taken here is to start with a simple scripting system for new storage strategies that allows users to flexibly introduce new approaches.

5.2 Conclusion

In this paper, we have introduced Flow, a flexible tool for experimenting with power storage strategies. We have primarily introduced the major features, and we have left practical evaluation for future work. This is a system description, introducing a new tool that gives users a chance to experiment with power storage and learn about the benefits in a hands-on manner.





There are several features that could be added to improve the simulation. One natural addition would be the introduction of new energy sources, such as wind and solar. These sources produce power with few emissions, but suffer from unreliable production levels. In order to accurately model these sources, our simulation would also need to incorporate notions of weather and daylight. We are currently looking at this extension.

References

1. Hunter, A., Young, R.: Power storage on the smart grid: experimentation and education. In: Proceedings of the International Conference on Agents and Artificial Intelligence (2017)
2. Motik, B., Patel-Schneider, P., Parsia, B.: OWL 2 web ontology language: structural specification and functional-style syntax. Technical report, W3C Recommendation (2009)
3. Peña, Y., Garbajosa, J., Ortega, M., Gonzalez, E.: ENERGOS: integral smart grid management. In: Proceedings of the IEEE International Conference on Industrial Informatics (2011)
4. Voice, T., Vytelingum, P., Ramchurn, S., Rogers, A., Jennings, N.: Decentralised control of micro-storage in the smart grid. In: Proceedings of the Conference on Association for the Advancement of Artificial Intelligence (2011)
5. Vytelingum, P., Voice, T., Ramchurn, S., Rogers, A., Jennings, N.: Theoretical and practical foundations of large-scale agent-based micro-storage in the smart grid. *J. Artif. Intell. Res.* **42**, 765–813 (2011)



Decision Assist for Self-driving Cars

Sriram Ganapathi Subramanian^(✉) , Jaspreet Singh Sambee ,
Benyamin Ghogh , and Mark Crowley 

Department of Electrical and Computer Engineering, University of Waterloo,
Waterloo, Canada

{s2ganapa, jssambee, bghogh, mcrowley}@uwaterloo.ca

Abstract. Research into self-driving cars has grown enormously in the last decade primarily due to the advances in the fields of machine intelligence and image processing. An under-appreciated aspect of self-driving cars is actively avoiding high traffic zones, low visibility zones, and routes with rough weather conditions by learning different conditions and making decisions based on trained experiences. This paper addresses this challenge by introducing a novel hierarchical structure for dynamic path planning and experiential learning for vehicles. A multistage system is proposed for detecting and compensating for weather, lighting, and traffic conditions as well as a novel adaptive path planning algorithm named **Checked State A3C**. This algorithm improves upon the existing A3C Reinforcement Learning (RL) algorithm by adding state memory which provides the ability to learn an adaptive model of the best decisions to take from experience.

Keywords: Autonomous cars · Path planning · Obstacle avoidance
Reinforcement learning · Weather estimation · Machine learning

1 Introduction

Building **Autonomous Driving Systems (ADS)** is a major goal of artificial intelligence research. The standard approach breaks this problem into six levels defined by SAE International [2] ranging from no automation (level 0) up to full automation (level 5). Path planning could be used for routing a self-driving car which needs to be undertaken given the coordinates of its starting and ending points. However in practice, this routing needs to be dynamic in a self-driving car with changes necessitated for obstacle, road blockage, bad lighting, bumpy roads, etc. Under harsh weather, darkness, or heavy traffic, the car might face difficulty in obstacle detection or even path traversal. The goal of this work is to fill a gap in the literature for enabling a self-driving car to find the most appropriate path between given starting and ending points not merely in terms of shortest path but in terms of different weather, lighting, and traffic jam conditions and

S. Ganapathi Subramanian, J. S. Sambee and B. Ghogh contributed equally to this work.

considering the abilities and strengths of the driver under those conditions, be it human or automated.

A complete system for a fully autonomous vehicle capable of mapping and localization and low-level navigation planning was proposed in [7]. The key challenges resolved in such complex systems include navigating narrow roads, cross walks, and traffic light recognition and following. However, these systems do not include high-level path planning to dynamically consider the complete alternate routes to the goal points. Our proposed system using Deep Reinforcement Learning (RL) can handle road condition changes dynamically since during training, the assessment of conditions and planning are integrated into a single model learned. This adaptability would be very expensive to obtain in a classical path planning system using shortest path algorithms such as the Dijkstra algorithm.

The proposed system is composed of a (low-level) image condition and obstacle detection module and a (high-level) path planning module. An input image frame is fed to the low-level module which estimates and outputs the amounts (factors) of luminance, haze, rain, and obstacles, namely pedestrians and cars, present in the image. This module also enhances images to normal conditions when necessary. The high-level module applies a Deep RL algorithm to the low-level outputs in order to decide for the best possible path the self-driving car should traverse between starting and ending points. In the rest of this paper, the proposed system is explained and then verified by the experiments.

2 Image Condition and Obstacle Detection

The low-level module estimates four road condition factors directly from images: **luminance**, **haze**, **rain**, and **obstacles** as well as **enhancing** images if necessary (luminance enhancement for night scenes and haze removal for foggy or snowy conditions). These four factors are mapped to integer levels ranging from 1 (for bright, not hazy, not rainy, no obstacles) to 5 (too dark, too hazy, too rainy, too many obstacles) which are then fed to path planning module (see Fig. 1). The obstacle factor ranges are: 1: no obstacle, 2: 1–2 obstacles, 3: 3–5 obstacles, 4: 6–8 obstacles and 5: 9 obstacles or more.

Luminance: The image color model is changed from RGB (Red, Green, Blue) (if gray-scale, taking intensity for all channels) to HSL (Hue, Saturation, Luminance) to extract luminance of each pixel i . The luminance factor is obtained by quantizing the average luminance of N -pixel image. If luminance is less than a threshold τ_1 (is dark), the luminance of all pixels are increased.

Haze: Histograms of intensities for different patches of a hazy image, which were empirically observed to have similar patterns, are found and after thresholding, a 256×1 feature vector is used for feeding to Fisher Linear Discriminant Analysis (LDA) with classes hazy and normal. By voting among the recognized patches of an image, haze factor is the number of hazy patches normalized by the total number of patches. The dark channel prior method for haze removal [3] is used if the haze factor reaches a threshold τ_2 (is hazy).

Rain: We consider both rain and snow streaks similarly as in literature [1]. These are mostly vertical even in strong winds and thus are captured well by a 3×3 vertical Sobel kernel. The Local Binary Pattern (LBP) method is used to extract texture features on patches of image, which then have Principal Component Analysis (PCA) applied for feature extraction. Fisher LDA is then used for classification on the resulting features and voting among recognized patches of an image determines the rain factor.

Obstacles: Obstacle detection is done in images having pedestrians and cars using the Single Shot Detector (SSD) approach [5]. SSD is a Deep Learning object detection approach in which the input image is fed into a set of convolutional layers which yields feature maps of varying scales. SSD consists of default bounding boxes and these bounding boxes are evaluated using a 3×3 convolutional filter which is placed at the end of the main convolution chain. For each of these boxes, a prediction on the bounding box and the class probability is made and the Intersection over Union (IoU) is applied on the ground truth labels and the predicted labels. The box which has an IoU greater than 0.5 is chosen and the rest ignored. The Tensorflow Object Detection API [4] was used to perform the SSD-based object detection and their base checkpoints were trained for further 7000 steps using the images manually labelled by LabelImg tool [11].

3 Path Planning

For the high-level task of deciding which paths to take we now introduce **Checked State A3C (CS-A3C)**, a Deep RL algorithm (see Algorithm 1) which is a modification of **Asynchronous Advantage Actor-Critic (A3C)**, a state-of-the-art policy gradient method [8]. We use a global network of workers, which is composed of convolutional layers and a Long-Short-Term-Memory (LSTM) layer. The function of the convolutional layers is to process spatial dependencies and the work of the LSTM layer is to process the temporal dependencies between the different input images.

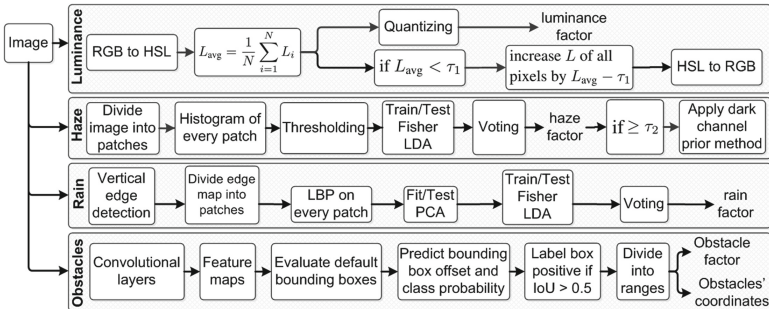


Fig. 1. Low-level processing: image condition and obstacle detection module.

Algorithm 1. Checked state A3C

```

1: Initialize thread step counter  $t \leftarrow 1$ 
2: while  $T > T_{max}$  do
3:   Reset Gradients:  $d\theta \leftarrow 0$  and  $d\theta_v \leftarrow 0$ 
4:   Synchronize thread- specific parameters:  $d\theta' = \theta$  and  $d\theta'_v = \theta_v$ 
5:    $t_{start} = t$ 
6:   Get state  $s_t$  and check if it is a checked state  $s^*$ 
7:   while terminal  $s_t$  or  $t - t_{start} == t_{max}$  do
8:     Perform  $a_t$  according to policy  $\pi(a_t|s_t; \theta')$ 
9:     Receive reward  $r_t$  and new state  $s_t + 1$ .
10:    Discount factor :  $\gamma = 0.9$  for  $s^*$  and 0.1 for  $s \neq s^*$ 
11:    Propagate the rewards until the nearest  $s^*$ .
12:    Apply learning rate  $\alpha_t = 1$  for  $s = s^*$  and  $\alpha_t = 0.0001$  for  $s \neq s^*$ 
13:     $t \leftarrow t+1$ 
14:     $T \leftarrow T+1$ 
15:     $R = 0$  for terminal  $s_t$ 
16:     $R = V(s_t, \theta'_v)$  for non-terminal  $s_t$ 
17:    for  $i \in t - 1, \dots, t_{start}$  do
18:       $R \leftarrow r_i + \gamma R$ 
19:      Accumulate Gradients w.r.t.  $\theta$  and  $\theta_v$ 
20:    Perform asynchronous update of  $\theta$  and of  $\theta_v$ 

```

A Deep Q Network (DQN) network represents the action policy as defined in [9] with 7 worker agents used. For simplicity we assume here the car cannot move in the reverse direction. A discounted reward model is used with the rewards in the range $[-1, 1]$. The reward function assigns +1 for reaching the goal point, for each factor from the low-level module the rewards are normalized to the range $[-0.5, +0.5]$ using its factor, +0.1 for reduction in euclidean distance to the goal point and -0.2 for any increase in distance. We make three important changes to the A3C algorithm from [8]: (I) **Checked states:** Some states are more important to learn about than others. When the car chooses a path at a road intersection, the resulting state (travelling until the next intersection) needs to be updated with rewards from subsequent, more minor states resulting from driving and lane keeping actions. We call this a **checked state** and weight it more strongly in reward propagation. (II) **Selective Learning:** The discount factor (gamma) is different for checked (0.9) and non-checked states (0.1) making learning selective. (III) **Pseudo states:** All states apart from the checked state are pseudo states as the action is completely deterministic.

4 Experimental Results

Results of Low-Level Module: Figure 2 includes several results of the low-level module. Figures 2a and b respectively have luminance factors of 2 and 4 (dark). The result of luminance enhancement for Fig. 2c is shown in Fig. 2e. The dehazing result of Fig. 2d having haze factor of 5 (too hazy) is in Fig. 2e. The rain factor of Fig. 2f is 5 (too rainy) and the obstacle factors of Fig. 2g, h, i, and j are 2, 3, 4, and 5 (too crowded) respectively. The curve of training loss of object detection versus training steps is shown in Fig. 3a.

Overall Results and Comparison: In this work, the Oxford robocar dataset [6] is used for experiments. Linear recurrence random goal points [10] are selected

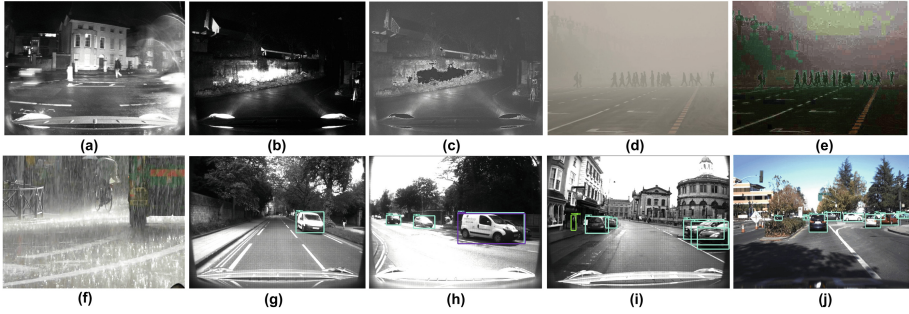


Fig. 2. Several results of image condition and obstacle detection.

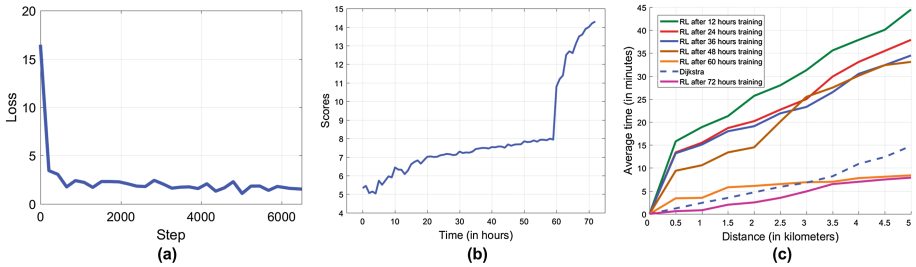


Fig. 3. Overall results: (a) training loss of object detection vs. training steps, (b) accumulation of rewards vs. training time, (c) comparison to the Dijkstra.

on the Oxford area for the training and results are reported for about 72 h of training (Fig. 3b). A car was made to virtually traverse the domain from a given starting point. Camera images are loaded from the dataset based on the spatial location of the vehicle. The CNNs, along with the output from the low-level layer, process the next free location in the image where the car can move to. Once this is done, the corresponding image is loaded from the dataset. This continues until the car reaches the goal point. For training CS-A3C, the rewards are increasing over a period of time which implies that the agent is doing better with time (Fig. 3b).

For comparison, a weighted graph is made by averaging the low-level factors for each image in every edge in the map. The Dijkstra algorithm is run on this weighted graph using 15 different goal points set at equal distance intervals from a fixed starting point. Distance intervals are about 100 m, ranging from 0 to 5000 m. The car is made to virtually traverse the given path in simulation and the average time taken is compared to the path returned by CS-A3C after training times of 12, 24, 36, 48, 60, and 72 h. Figure 3c shows that average time taken by Dijkstra to reach the goal point is less than CS-A3C trained for 12, 24, 36, and 48 h. However, CS-A3C performs better after 60 h. This also corresponds well with the graph seen in Fig. 3b where the accumulation of rewards drastically

increases at about 60 h of training. The training of CS-A3C is stopped within 72 h as at this stage it comfortably beats Dijkstra. The CS-A3C algorithm tunes the weights of the different factors that determine the edge weights of the graph at training time, based on the time to goal point and the learned policy differentiates the edges based on experience. The Dijkstra algorithm, on the other hand, considers only an average edge weight. Thus, CS-A3C ultimately beats the Dijkstra algorithm after sufficient training in terms of time taken to reach the goal point.

5 Conclusion

The new CS-A3C algorithm is a highly efficient hierarchical path planning system with clear advantages over the traditional Dijkstra approach: (I) Our system requires a map only in the training phase and not at test time while Dijkstra needs a new map for every attempt. (II) If the conditions (luminance, weather, and traffic) of the road change during path traversal Dijkstra needs to be run again on a new graph. In contrast, our system needs to be trained only once using a variety of conditions to learn a robust path planning model that applies under many changes in conditions.

References

1. Barnum, P.C., Narasimhan, S., Kanade, T.: Analysis of rain and snow in frequency space. *Int. J. Comput. Vis.* **86**(2), 256–274 (2010)
2. SAE On-Road Automated Vehicle Standards Committee, et al.: Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. SAE Standard J3016, pp. 1–16 (2014)
3. He, K., Sun, J., Tang, X.: Single image haze removal using dark channel prior. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(12), 2341–2353 (2011)
4. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K.: Speed/accuracy trade-offs for modern convolutional object detectors. Honolulu, Hawaii (2016)
5. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) *ECCV 2016*. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
6. Maddern, W., Pascoe, G., Linegar, C., Newman, P.: 1 year, 1000km: the Oxford RobotCar dataset. *Int. J. Robot. Res. (IJRR)* **36**(1), 3–15 (2017)
7. Milford, M.J., Wyeth, G.F.: SeqSLAM: visual route-based navigation for sunny summer days and stormy winter nights. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 1643–1649. IEEE (2012)
8. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning*, pp. 1928–1937 (2016)

9. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
10. Tausworthe, R.C.: Random numbers generated by linear recurrence modulo two. *Math. Comput.* **19**(90), 201–209 (1965)
11. Tzutalin, D.: LabelImg annotation tool. <https://github.com/tzutalin/labelImg>. Accessed 18 Oct 2017



Rule Mining and Prediction Using the Flek Machine – A New Machine Learning Engine

Abbas Taher^(✉)

GoFlek Inc., Montreal, QC, Canada
abbas@goflek.com

Abstract. One of the exciting areas in data science is the development of new machine learning engines to do data mining, analytics and prediction. In this paper, we introduce the “Flek Machine” – an innovative AI engine that learns a Bayes Net model from binary data.

FlekML, the core machine learning engine inside, builds a rich model that can be manipulated by the Toolkit to do rule mining, discover associations and association maps as well as make predictions. The Flek Machine enables binary, multi-class and multi-label classifications all on the fly and over the same built model. This tool has several use cases such as customer behaviour analysis, predicting equipment failure in IoT, or detecting drug combinations that produce side effects.

Keywords: Machine learning · Bayesian Networks · Data mining
Association · Rule mining · Prediction

1 Introduction

Typically when you think of machine learning you think of an algorithm that runs over available data. This works fine when the analytics needs are ad-hoc or narrow in scope. However, organizations today are data-driven and their competitive edge is tied to the insights they gain from collected data. As a result, they require intelligent tools to model, discover, understand, analyze as well as make predictions all in an integrated framework.

To address this challenge, the Flek Machine offers a new approach to data mining. On one hand, Flek utilizes a supervised machine learning engine based on AI techniques to build a Bayes Net model that can generate statistics, association rules and association maps. On the other hand, Flek offers a flexible programming Toolkit based on a concise API that allows data scientists to interact with the model at a higher level of abstraction.

Together, the ML engine and the Toolkit can be thought of as a specialized SDK for learning and then manipulating a Bayes Nets model. To use the SDK for example, data scientists can first build the model in memory then query and retrieve results in the form of “IF THEN” rules that can be sorted by their confidence values. Later, they can make predictions and perform “WHAT IF” analysis using the same learned model.

2 Product Architecture

The Flek Machine is founded on a multi-layered design that integrates more than one module into a uniform architecture. The product is composed of two main components:

1. FlekML – the core supervised machine learning engine and store
2. Toolkit – a library of APIs, algorithms and utilities for interacting with the model

The figure below depicts the various components in the Flek Machine and the central role FlekML plays in the overall architecture (Fig. 1).

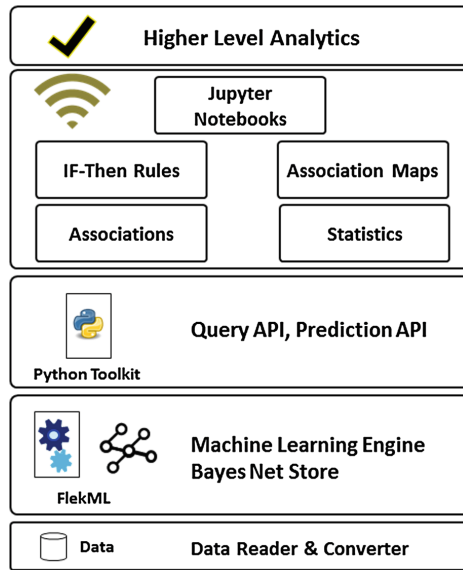


Fig. 1. Flek machine architecture

3 Data Mining Pipeline

Flek provides an integrated framework to perform machine learning and data mining. To achieve its goal, it divides the pipeline into 6 simple stages. The first two stages prepare the raw data and transform it into a binary format that is used by FlekML.

The remaining four stages, as depicted in the diagram below, constitute the core activities performed by the Flek Machine: build the model, query for information, analyze results and make predictions (Fig. 2).

What makes Flek unique is that the entire process combines machine learning, rule mining, and classification activities all into one pipeline. To use Flek, data scientists typically perform two main activities:

1. Build and store the Bayes Net model in memory

2. Write programs that interact with the model at a higher level of abstraction which includes such tasks as:

- Querying the nuggets stored inside the Bayes Net model.
- Searching, sorting, filtering and making computations on the retrieved objects for further mining and analytics.
- Doing predictions, generating scores and testing accuracy using various metrics

Moreover, data scientists can peek into each of the generated prediction scores and examine the rational as well as the rule used during the classification process.

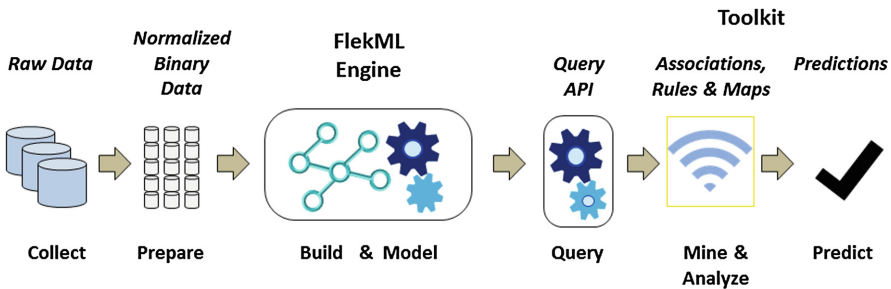


Fig. 2. Flek machine pipeline

4 Concise API and Simple Programming Model

The key programming abstraction in Flek is the Nuggets, which are computed by the ML engine and saved as objects in the store. These objects are later retrieved, mined and manipulated through a Python based API.

We show below two sample programs that only scratch the surface of how Flek can be used. Note that we omitted module imports for brevity. The two code snippets concern an online magazine that collects information about its subscribers such as their profiles and interests and then analyzes their preferences.

4.1 Rule Mining Code Snippet

The following Python code snippet summarizes how data scientists build the Bayes Net model, query for rules and then sort them in descending order.

```

data = [[1,0,1,1], [1,1,1,1], [0,1,1,0], [1,1,0,0], [1,1,0,1], ...]
interest = {"FEATURES": "AUTO, FOOD, SPORTS, TECHNO"}
ds = Dataset(data)
with BuildModel(ds, interest) as model:
    rules = [model.getRule((i,), (2,)) for i in [0,1,3]]
    sRules = sorted(rules, key=lambda r: r.conf, reverse=True)
    print (sRules[0])      #take rule with highest confidence value
    
```

After only 4 lines of code, we have a fully functional model sitting in memory and ready to be queried and discovered. In the above, the programmer retrieves all rules of the form: IF “interest” THEN SPORTS – “interest” being all possible values other than SPORTS. Having the rules in a list, the programmer then sorts the list by confidence and takes the one with highest value (first item).

4.2 Prediction Code Sample

The following code sample illustrates how a prediction application is composed. First the data is read from a file and split into a training and testing datasets. Then the training set is used to build the model and the test set is used to perform the actual prediction. Last the results of the prediction process are checked by printing the confusion matrix and a short summary. All these tasks are done in an easy to understand and concise Python code.

```
if __name__ == "__main__":
    datafile = "/data/SubscribersDataset.csv"
    semfile = "/info/SubscribersSemantics.txt"

    csvdata = CSVReader(datafile, skipHeader= True)
    csvdata.setConverter(CastToBinary())
    main_dset = Dataset(csvdata)
    train_set, test_set = main_dset.split(80,20)

    predict = Predictset(train_set,semfile)
    predict.buildModel()
    predict.doPredict(test_set)

    confusion = predict.confusionMatrix()
    print(confusion)
    summary = predict.scoringSummary()
    print(summary)
    predict.endModel()
```

5 Why Bayes Net and Why Python

Today, analytics is the word du jour and machine learning is the established technique for performing such an activity. However, given the growing volumes of available data and the requirements for intelligent analysis, data scientists are looking for new tools to tackle the challenges they face. Essentially, tools that can go beyond applying algorithms to data and into the space of building models that capture the complex associations found in reality.

Invented by Judea Pearl in the 1980s at UCLA, Bayesian Networks are a mathematical formalism for simultaneously representing a multitude of probabilistic relationships between variables in a system. Whereas traditional statistical models are mathematical

functions in nature, Bayesian Networks do not distinguish between independent and dependent variables. Rather, a Bayesian Network approximates the entire joint probability distribution of the system it models. This unique characteristic makes Bayesian Networks a powerful tool for modeling complex and highly dimensional problems, ranging from bioinformatics to marketing science [1].

With this in mind, we started developing Flek based on an attuned form of Bayesian Networks we call Bayes Nets. Our Bayes Net encapsulates complex association patterns [2] found in binary data and provides a rich model that can be applied in a variety of analytics situations. In addition, we found the works of Ray Solomonoff on algorithmic probability [3] inspiring and the Frequentist approach to probabilistic prediction very useful.

On a different note, the Python programming language [4], invented by Guido Van Rossum and first released in 1991, was a natural choice for developing Flek. Choosing Python for software development was based on a simple fact – Python is the *de facto* standard for machine learning and it is popular amongst data scientists. As an added bonus, the language is both compact and lucid which makes it an effective environment for expressing complex operations and algorithms in an elegant API.

6 Discussion and Further Work

At its core, Flek Machine, is a dedicated machine learning engine that builds and stores in memory a Bayes Net model. It includes a library of APIs, algorithms and utilities for interacting with a learned model.

While Flek can only process binary input, i.e. categorical and discretized data, its paradigm is powerful enough to express several applications that pose challenges for existing machine learning techniques. In particular, those are iterative and interactive applications which manipulate a model or perform computations at a higher level of abstractions. Moreover, we believe that the two main ideas behind Flek, i.e. Associations and Bayes Nets, encapsulates enough information to build a complex model that can be used to do rule mining, generate association maps and make predictions with traceable rational and interpretable results.

To further enhance our product, we plan to focus on four areas:

- Optimize the one pass algorithm that was developed to generate the Bayes Net model which is in and by itself a unique innovation in the field.
- Enhance object store and retrieval.
- Improve error handling.
- Provide new higher-level abstractions for filtering objects retrieved from the store.

References

1. Pearl, J.: Graphical models for probabilistic and causal reasoning. In: Smets, P. (ed.) Quantified Representation of Uncertainty and Imprecision. Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol. 1, pp. 367–389. Springer, Dordrecht (1998). https://doi.org/10.1007/978-94-017-1735-9_12

2. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., pp. 207–16 (1993)
3. Solomonoff, R.: Two kinds of probabilistic induction. *Comput. J.* **42**(4), 256–259 (1999)
4. Python Homepage. <https://www.python.org/>

Author Index

- Abdou, Tamer 84
Abielmona, Rami 265
Alhadidi, Dima 292
Almeida, Hayda 133
Alslaity, Alaa 310, 351
Asadi Kakhki, Saeede Sadat 322
Ataei, Masoud 231
- Baier, Jorge A. 45
Baird, Benjamin 3
Bappee, Fateha Khanam 367
Barbosa, Denilson 304
Bener, Ayse 84, 322
Bian, Chao 181
Bidar, Mahdi 246
Bidar, Mohsen 246
Bouguila, Nizar 211
Briand, Antoine 133
- Camacho, Alberto 45
Chen, Shengyuan 231
Chen, Tao 316
Cohen, Robin 169
Conrad, Colin 298
Cook, Paul 292
Crowley, Mark 285, 381
Custis, Tonya 218
- Dirks, Matthew 360
Dittimi, Tamarafinide V. 156
Drummond, Chris 194
- Etemad, Mohammad 259
- Falcon, Rafael 265
- Gagné, Christian 279
Ganapathi Subramanian, Sriram 285, 381
Ghojogh, Benyamin 381
Gorji Daronkolaie, Aliakbar 218
Greiner, Russell 31
Grenier, Liam 272
- Hajian, Amir 218
Harris, Jabez 298
Hassanpour, Negar 31
Hu, Pingzhao 272
Huang, Zhaoyang 344
Hunter, Aaron 239, 374
- Islam, M. M. Manjurul 144
Islam, Md. Mohaiminul 272
- Jakubina, Laurent 121
Jankowska, Magdalena 298
- Kavaklioglu, Can 322
Kešelj, Vlado 298
Khan, Shehroz S. 181
Khan, Wasif 169
Kim, Jong-Myon 144
King, Milton 292
Klassen, Toryn Q. 72
Kuang, Qin 272
- Langlais, Philippe 121
Loker, Dylan 17
López, Karol Lina 279
- MacDonald, Ross 331
Matwin, Stan 96, 224, 259, 367
McCrackin, Laura 355
McIlraith, Sheila A. 45, 72, 253
McLeod, Robert D. 272
Meurs, Marie-Jean 133
Mihailidis, Alex 181
Monga, Tanya 339
Mouhoub, Malek 246
Muisse, Christian 45
- Panchapakesan, Ashwin 265
Parsons, Jeffrey 316
Pesaranghader, Ahmad 96
Pesaranghader, Ali 96

- Petriu, Emil 265
Peyghami, M. Reza 231
- Russell, Dan 374
- Sadaoui, Samira 246
Sambee, Jaspreet Singh 381
Samuel, Hamman 108
Shvo, Maayan 253
Soares Júnior, Amílcar 259, 367
Sohrabi, Shirin 253
Sokolova, Marina 96
Soleimani, Behrouz Haji 224
Suen, Ching Y. 156
- Taher, Abbas 388
Tan, Yingcong 60, 335
- Terekhov, Daria 60
Toro Icarte, Rodrigo 72
Tran, Thomas 310
- Valenzano, Richard 72
- Xiang, Yang 3, 17
Xu, Peng 304
- Yang, Kevin 231
Yang, Zijiang 231
You, Jiaying 272
Young, Steven 84
- Zaïane, Osmar 108
Zamzami, Nuha 211