Learning as Data Compression[⋆]

Pieter Adriaans

Department of Computer Science
University of Amsterdam,
Kruislaan 419,
1098VA Amsterdam,
The Netherlands
pietera@science.uva.nl

Abstract. In this paper I describe the general principles of learning as data compression. I introduce two-part code optimization and analyze the theoretical background in terms of Kolmogorov complexity. The good news is that the optimal compression theoretically represents the optimal interpretation of the data, the bad news is that such an optimal compression cannot be computed and that an increase in compression not necessarily implies a better theory. I discuss the application of these insights to DFA induction.

Keywords: learning as compression, MDL, two-part code optimization, randomness deficiency, DFA induction.

1 Learning as Compression

Since the beginning of science in antiquity, the idea that the complexity of the world can be explained in terms of some simple first principles has fascinated researchers. In modern methodology of science this notion is studied under various guises: Occams razor [7], the minimal description length (MDL) principle [8], two-part-code optimization [11], learning as data compression [21] etc. Although there has been some debate about this principle with fierce opponents [7] and strong defenders [21], until recently the view of learning as data compression did not seem to have much practical value. Lots of learning algorithms in fact perform some kind of data compression, but this was not a guiding principle of their design [9; 20]. Two developments in the last five years have changed this perspective quite fundamentally : 1) a better understanding of the mathematics behind compression, specifically Kolmogorovs structure function [11; 10] and 2) the application of existing implementations of compression algorithms to approximate the ideal (and uncomputable) Kolmogorov complexity as pioneered by Cilibrasi and Vitanyi [5; 6]. At this moment we have not only a much better

[⋆] This project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

understanding of the theoretical issues behind data compression, but there is also a wealth of interesting and successful applications. Due to limited space in this paper I will restrict myself to a description of the general principles and a study of the application of MDL to DFA induction [1; 4]. In the tutorial itself I will also describe a number of other applications (e.g. Normalized Compression Distance) [5] and I will touch on some philosophical issues: the relation between data compression, thermodynamics and human cognition [2].

Take a cup of coffee and pour some cream in it (See Figure 1). Take a picture of it with your digital camera. In the beginning the cream will be just an uninteresting blob. Stir slowly and make pictures of various stages that have nice patterns. Continue until the cream has dissolved and your cup has an even brown color. Drink the coffee, then look at the file size of the different pictures. If your camera uses an adequate compression algorithm you will find that the file size has increased up to a certain point and then decreases. The compression algorithm of your camera reflects the complexity of the data set until the moment that the complexity has reached a global equilibrium and is beyond its resolution. In this experiment we have a system that evolves in time, the cup of coffee, and a data set of observations, the pictures. The crux of this experiment is that the size of the individual pictures somehow reflects the 'interestingness' of the system. In the beginning there is a lot of order in the system. This is not very interesting. In the end there is an equilibrium that also has little cognitive appeal.

In general science, in the study of human cognition and even in art we seem to have an interest in systems that have a complexity in the 'sweet spot' between order and chaos, between boredom and noise. The 'interestingness' of these data sets is somehow related to compressibility. It will prove useful to describe these compressions in terms of a so-called two-part-code: a description of a general class of sets, the *model code* and an element or a set of elements of this set, the *data-to-model-code* [11; 10].

Let me give some examples:

– **Symmetry.** This is one of the most fundamental ordering principles in nature. Most living creatures have symmetry: plants, trees, predator, prey. If a data set has symmetry it means that we only have to describe half of it (the data-to-model-code) plus some information about the nature of the symmetry of constant length (the model-code). In the limit such a data set can be compressed to at least half its size. In terms of generating languages symmetry is context free: a symmetric data set can be produced by a simple memoryless central process. Discovering symmetry in a data set can be seen as a very simple learning problem. It can easily be discovered in linear time.
– **Repetition.** In order to describe a repeating pattern I only have to give a description of the generating pattern (the data-to-model-code) and some information about the way the pattern repeats itself (the model-code). Repetition is more complex than symmetry in the sense that it presupposes a generating process with a memory: in terms of languages repetition is context

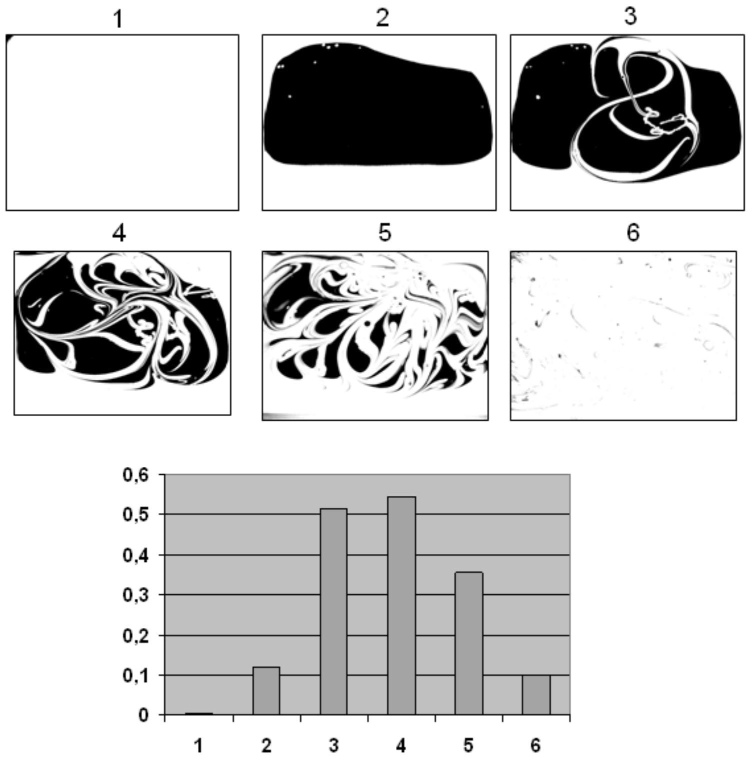

Fig. 1. Facticity scores for mixing black and white paint. The facticity of a data x is the product of the normalized entropy $K(x)/U(x)$ and the normalized randomness deficiency $(U(x) - K(x))/U(x)$. Configuration 4 has the best balance between order and chaos and thus would be the most 'interesting' one. The scores have been calculated using JPEG, followed by RAR compression. Maximal entropy $U(x)$ has been approximated by adding 400 % noise to the images. The standard entropy $K(x)$ is approximated by the file size after compression.

sensitive. Finding repeating patterns in a data set is also a basic learning problem that can be solved in time $n \log n$ [3].

- **Grammar.** A corpus of a language could be described in terms of the grammar G (the model-code) of the language and a set of indexes corresponding to an enumeration of the sentences in the corpus (the data-to-model-code). If the size of the corpus is large enough in relation to the size of the grammar G then this description in terms of two will be shorter than an extensional description of the sentences in the corpus. Finding this description is a well studied learning problem. If the language is regular then the task of approximating the smallest DFA consistent with a set of sentences is NP-hard [14; 4].
- **Program.V** We could ask ourselves, given a certain data set: what would be the shortest program generating this data set in a certain programming

language, or, even more general, we could try to find the shortest combination of a Turing machinenoindent T_i (the model-code) and a program P (the data-to-model-code). In a sense this would be, from a computational point of view, the ultimate compression possible and the Turing machine T_i would be the ultimate 'explanation' of the data set. Needless to say that because of the Halting problem there is no algorithm that will construct this ultimate compression for us. The problem is undecidable. Still, conditional to the programming language we choose, the notion of the shortest program generating a certain data set is well defined. Kolmogorov complexity studies these optimal compressions from the perspective of universal Turing machines [10].

Here I have described four classes of learning problems (varying from very easy, via NP-hard, to undecidable) as compression problems where the task is to find a two-part code compression for a data set. Apparently there is a deep connection between data compression and learning. In this tutorial I will also describe the theory behind these phenomena and explain how they can be used to develop algorithms to analyze data sets and understand the way they work.

2 MDL as Two-Part Code Optimization

It is important to note that two part code optimization is a specific application of MDL. The majority of work on MDL is closer in spirit to the statistical than to the Kolmogorov complexity world. Rather than two-part codes, one uses general universal codes for individual sequences; two-part codes are only a special case. We give the traditional formulation of MDL [9; 8]:

Definition 1. The Minimum Description Length principle*: The best theory to explain a set of data is the one which minimizes the sum of*

– *the length, in bits, of the description of the theory and*
– *the length, in bits, of the data when encoded with the help of the theory*

Let $M \in \mathcal{M}$ be a model in a class of models $\mathcal{M}$, and let D be a data set. The **prior probability** of a hypothesis or model M is $P(M)$. Probability of the data D is $P(D)$. **Posterior probability** of the model given the data is:

$$P(M|D) = \frac{P(M)P(D|M)}{P(D)}$$

The following derivation [9] illustrates the well known equivalence between MDL and the selection of the Maximum A posteriori hypothesis in the context of Shannon's information theory. Selecting the **Maximum A Posteriori hypothesis (MAP)**:

$$M_{MAP} \equiv argmax_{M \in \mathcal{M}}\ P(M|D)$$

$$= argmax_{M \in \mathcal{M}}\ (P(M)P(D|M))/P(D)$$

(since D is constant)

$$\equiv argmax_{M \in \mathcal{M}}\ (P(M)P(D|M))$$

$$\equiv argmax_{M \in \mathcal{M}} \log P(M) + \log P(D|M)$$

$$\equiv argmin_{M \in \mathcal{M}} - \log P(M) - \log P(D|M)$$

where according to Shannon $-\log P(M)$ is the length of the optimal *model-code* in bits and $-\log P(D|M)$ is the length of the optimal *data-to-mode-code* in bits. This implies that the model that is chosen with Bayes' rule is equal to the model that MDL would select:

$$M_{MAP} \equiv M_{MDL}$$

The formula $argmin_{M \in \mathcal{M}} - \log P(M) - \log P(D|M)$ indicates that a model that generates an optimal data compression (i.e. the shortest code) is also the best model. This is true even if $\mathcal{M}$ does not contain the original intended model as was proved by [11]. It also suggests that compression algorithms can be used to approximate an optimal solution in terms of successive steps of incremental compression of the data set D. This is *not* true as was shown by [1]. Yet this illicit use of the principle of MDL is common practice.

In order to understand these results better we must answer two questions 1) What do we mean by the length of optimal or shortest code and 2) what is an independent measure of the quality of a model M given a data set D? The respective answers to these questions are *prefix-free Kolomogorov complexity* and *randomness deficiency.*

2.1 Kolmogorov Complexity

Let $x, y, z \in \mathcal{N}$, where $\mathcal{N}$ denotes the natural numbers and we identify $\mathcal{N}$ and $\{0,1\}^*$ according to the correspondence

$$(0, \epsilon), (1, 0), (2, 1), (3, 00), (4, 01), \ldots$$

Here ϵ denotes the *empty word.* The *length* $|x|$ of x is the number of bits in the binary string x, not to be confused with the *cardinality* $|S|$ of a finite set S. For example, $|010| = 3$ and $|\epsilon| = 0$, while $|\{0,1\}^n| = 2^n$ and $|\emptyset| = 0$. The emphasis is on binary sequences only for convenience; observations in any alphabet can be encoded in a 'theory neutral' way. Below we will use the natural numbers and the binary strings interchangeably. In the rest of the paper we will interpret the set of models $\mathcal{M}$ in the following way:

Definition 2. *Given the correspondence between natural numbers and binary strings, $\mathcal{M}$ consists of an enumeration of all possible self-delimiting programs for a preselected arbitrary universal Turing machine U. Let x be an arbitrary bit* *string. The shortest program that produces x on U is $x^* = argmin_{M \in \mathcal{M}}(U(M) = x)$ and the Kolmogorov complexity of x is $K(x) = |x^*|$. The conditional Kolmogorov complexity of a string x given a string y is $K(x|y)$, this can be interpreted as the length of a program for x given input y. A string is defined to be random if $K(x) \geq |x|$.*

This makes $\mathcal{M}$ one of the most general model classes with a number of very desirable properties: it is universal since all possible programs are enumerated, because the programs are self-delimiting we can concatenate programs at will, in order to create complex objects out of simple ones we can define an a-priori complexity and probability for binary strings. There are also some less desirable properties: $K(x)$ cannot be computed (but it can be approximated) and $K(x)$ is asymptotic, i.e. since it is defined relative to an arbitrary Turing machine U it makes less sense for objects of a size that is close to the size of the definition of U. Details can be checked in [10]. We have:

$$argmin_{M \in \mathcal{M}} - \log P(M) - \log P(D|M) =$$

$$argmin_{M \in \mathcal{M}} K(M) + K(D|M) = M_{MDL} \tag{1}$$

Under this interpretation of $\mathcal{M}$, the length of the optimal code for an object is equivalent to its Kolmogorov complexity.

In this paper I will often use the notions of *typicality* and *incompressibility* of elements of a set, e.g. in those cases where I state that the vast majority of elements of a set have a certain quality. This might at first sight sound a bit inaccurate. To show that this notion actually has an exact definition I give the following theorem due to Li and Vitányi [10] pg. 109):

Theorem 1. *Let c be a positive integer. For each fixed y, every finite set A of cardinality m has at least $m(1 - 2^{-c}) + 1$ elements x with $C(x|y) \geq \log m - c$.*

Proof: The number of programs of length less than $\log m - c$ is

$$\sum_{i=0}^{\log m - c - 1} 2^i = 2^{logm - c} - 1$$

Hence, there are at least $m - m2^{-c} + 1$ elements in A that have no program of length less than $\log m - c$.

This shows that in the limit the number of elements of a set that have low Kolmogorov complexity is a vanishing fraction. In the limit a typical element of a set is a random element. In general the vast majority of elements of a set is not compressible. One of the problems with Kolmogorov complexity is that it specifies the length of a program but tells us nothing about the time complexity of the computation involved. Therefore Kolmogorov complexity can not be used directly to prove lower bounds for the time complexity of problems.

2.2 Randomness Deficiency

It is important to note that objects that are non-random are very rare. To make this more specific: in the limit the density of compressible strings x in the set $\{0,1\}^{\leq k}$ for which we have $K(x) < |x|$ is zero [10]. The overwhelming majority of strings is random. In different words: an element is *typical* for a data set if and only if it is *random* in this data set. In yet different words: if it has maximal entropy in the data set. This insight allows us to formulate a theory independent measure for the quality of models: *randomness deficiency.*

We start by giving some estimates for upper-bounds of conditional complexity. Let $x \in M$ be a string in a finite model M then

$$K(x|M) \leq \log|M| + O(1) \tag{2}$$

i.e. if we know the set M then we only have to specify an index of size $\log|M|$ to identify x in M. Consequently:

$$K(x) \leq K(M) + \log|M| + O(1) \tag{3}$$

The factor $O(1)$ is needed for additional information to reconstruct x from M and the index. Its importance is thus limited for larger data sets. These definitions motivate the famous Kolmogorov structure function:

$$h_x(\alpha) = \min_S\{\log|S| : x \in S, K(S) \leq \alpha\} \tag{4}$$

Here α limits the complexity of the model class S that we construct in order to ‘explain’ an object x that is identified by an index in S. Let $D \subseteq M$ be a subset of a finite model M. We specify $d = |D|$ and $m = |M|$. Now we have:

$$K(D|M,d) \leq \log\binom{m}{d} + O(1) \tag{5}$$

Here the term $\binom{m}{d}$ specifies the size of the class of possible selections of d elements out of a set of m elements. The term $\log\binom{m}{d}$ gives the length of an index for this set. If we know M and d then this index allows us to reconstruct D.

A crucial insight is that the inequalities 2 and 5 become ‘close’ to equalities when respectively x and D are *typical* for M, i.e. when they are random in M. This typicality can be interpreted as a measure for the goodness of fit of the model M. A model M for a data set D is optimal if D is random in M, i.e. the randomness deficiency of D in M is minimal. The following definitions formulate this intuition. The *randomness deficiency* of D in M is defined by:

$$\delta(D|M,d) = \log\binom{m}{d} - K(D|M,d), \tag{6}$$

for $D \subseteq M$, and ∞ otherwise. If the randomness deficiency is close to 0, then there are no simple special properties that single D out from the majority of data samples to be drawn from M.

The *minimal randomness deficiency* function is

$$\beta_x(\alpha) = \beta_D(\alpha) = \min_M \{\delta(D|M) : M \supseteq D,\ K(M) \leq \alpha\}, \tag{7}$$

If the randomness deficiency is minimal then the data set is typical for the theory and with high probability future data sets will share the same characteristics, i.e. minimal randomness deficiency is also a good measure for the future performance of models. For a formal proof of this intuition, see [11].

We now turn our attention to incremental compression. Equation 1 gives the length of the optimal *two-part-code*. The length of the two-part-code of an intermediate model M_i is given by:

$$\Lambda(M_i, d) = \log \binom{m_i}{d} + K(M_i) \geq K(D) - O(1) \tag{8}$$

This equation suggests that the optimal solution for a learning problem can be approximated using an incremental compression approach. This is indeed what a lot of learning algorithms seem to be doing: find a lossy compression of the data set finding regularities. This holds for such diverse approaches as nearest neighbor search, decision tree induction, induction of association rules and neural networks. There is a caveat however; [1] have shown that the randomness deficiency not necessarily decreases with the length of the MDL code, i.e. shorter code does not always give smaller randomness deficiency, e.g. a better theory. This leads to the following observations [1]:

– The optimal compression of a data set in terms of model and a data-to-model code always gives the best model approximation “irrespective of whether the ‘true’ model is in the model class considered or not” [11][1].
– This optimal compression cannot be computed.
– Shorter code does not necessarily mean a better model.

These observations show that the naive use of the MDL principle is quite risky.

3 A Case Study: MDL and DFA Induction

In the domain of machine learning pure applications of MDL are rare, mainly because of the difficulties one encounters trying to define an adequate model code and data-to-model code. The field of grammar induction studies a whole class of algorithms that aims at constructing a grammar by means of incremental compression of the data set represented as a digraph. This digraph can be seen as the maximal theory equivalent with the data set. Every word in the data

[1] This is true only in this specific computational framework of reference. In a probabilistic context, both for Bayesian and MDL inference, the assumption that the true model is in the model class considered can sometimes be crucial - this also explains why in Vapnik-Chervonenkis type approaches, complexity is penalized much more heavily than in MDL [12]).

set is represented as a path in the digraph with the symbols either on the edges or on the nodes. The learning process takes the form of a guided incremental compression of the data set by means of merging or clustering of the nodes in the graph. None of these algorithms explicitly makes an explicit estimate of the MDL code. Instead they use heuristics to guide the model reduction. After a certain time a proposal for a grammar can be constructed from the current state of the compressed graph. Examples of such algorithms are SP [23; 22], EMILE [15; 16], ABL [18], ADIOS [19] and a number of DFA induction algorithms, specifically evidence driven state merging (EDSM), [17; 24]. In this paragraph we present a sound theoretical basis to analyze the performance and idiosyncrasies of DFA induction in an MDL context [4]. We will follow the presentation in [20]. The general methodology for applying two-part-code optimization to a certain learning problem is:

– Design an approximation of the optimal model code. Such a model code should reflect structural changes in the model complexity in an adequate way and should at the same time be computationally feasible. In this case we will use a simple count on the nodes and edges.
– Design an approximation of the optimal data-model-code with the same desiderata, for details see below.
– Select a compression algorithm that is computationally feasible and heuristically adequate. We will use standard evidence driven state merging with MDL as optimization criterion.
– Define a start state for the learning process. This will be the so-called Maximal Canonical Automaton, the graph that exactly generates the data set from one start state.
– Define an adequate stop condition for the compression process. In this case we will simply limit the computation time.

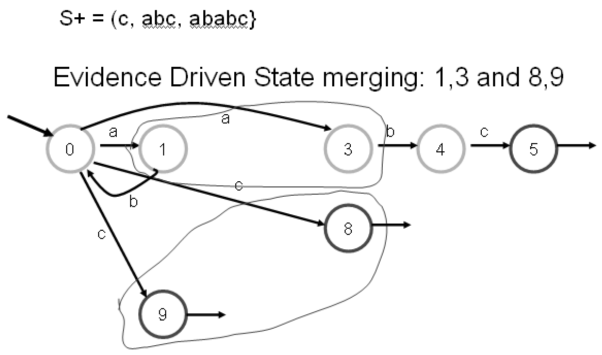

Fig. 2. Compressing a DFA (model) by means of state merging, given some set of positive examples S+

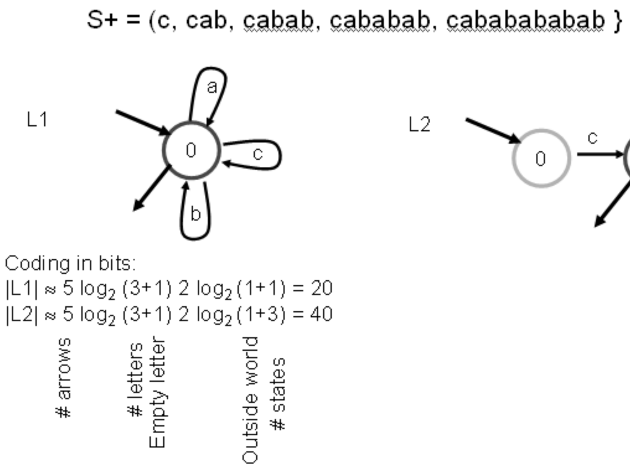
S+ = (c, cab, cabab, cababab, cababababab }
L1
a
c
b
0
L2
0
c
1
a
b
2
Coding in bits:
|L1| ≈ 5 log2 (3+1) 2 log2 (1+1) = 20
|L2| ≈ 5 log2 (3+1) 2 log2 (1+3) = 40
# arrows
# letters
Empty letter
Outside world
# states

Fig. 3. Two DFA generating S+. L1 is the shortest model, but L2 generates the shortest MDL code.

We start with some relevant observations. We will restrict ourselves to languages in $\{0.1\}^*$. The class of DFA is equivalent to the class of regular languages. We call the set of positive examples D^+ and the set of negative examples D^-. The complement of a regular language is a regular language. Consequently the task of finding an optimal model given D^+ is symmetric to the task of finding an optimal model given D^-. The task of finding the minimum DFA consistent with a set of positive and negative examples is *decidable*. We can enumerate all DFA's according to their size and test them on the data set. Yet this minimum DFA cannot be approximated within polynomial time [14].

The task of finding the smallest DFA consistent with a set of positive examples is trivial. This is the universal DFA. Yet the universal DFA will in most cases have a poor generalization error. MDL is a possible candidate for a solution here. Suppose that we have a finite positive data set representing an infinite regular language. The task is then to find a DFA with minimum expected generalization error over the set of infinite regular languages consistent with D^+. MDL in theory identifies such a DFA.

Definition 3. *A* partition π *of a set X is a set of nonempty subsets of X such that every element x in X is in exactly one of these subsets. $B(s, \pi) \subseteq X$ indicates the subset of the partition π of which x is an element.*

Definition 4. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. The* quotient automaton A/π $= (Q', \Sigma, \delta', B(q_0, \pi), F')$ *derived from A on the basis of a partition π of Q is defined as follows:*

– $Q' = Q/\pi = \{B(q, \pi) | q \in Q\}$,
– $F' = \{B \in Q' | B \cap F \neq \emptyset\}$,
– $\delta' : (Q' \times \Sigma) \to 2^{Q'} : \forall B, B' \in Q', \forall a \in \Sigma, B' \in \delta'(B, a)$ *iff* $\exists q, q' \in Q, q \in B, q' \in B'$ *and* $q' \in \delta(q, a)$.

We say that the states in Q that belong to the same block B are merged.

We give without proof:

Lemma 1. *If an automaton A/π is derived from an automaton A by means of a partition π then $L(A) \subseteq L(A/\pi)$.*

The relevance of these definitions for grammar induction lies in the fact that we can increase or decrease the generality of the automaton and the associated language inclusion hierarchies by means of splitting and merging states. We now develop an adequate data to model code based on the idea that a positive data sample has an entropy in each node of the DFA.

Definition 5. *Let A be a DFA. An* index set *for A is a set that associates a unique natural number with each string that is accepted by A. The* index set relative to certain data set *$D \subseteq L(A)$ is $I_D = \{i | i \in \mathbf{N}, L(A)(i) \in D\}$. The* initial segment *associated with an index set D and $L(A)$ is the set $I_{\leq D} = \{i | i \in \mathbf{N}, \exists j \in I_D : j \geq i\}$, i.e. the set of all natural numbers that are smaller than or equal to an index in I_D. The* maximal entropy *of I_D in $I_{\leq D}$ is $\log \binom{|I_{\leq D}|}{|I_D|}$, where $|I_{\leq D}|$ is a measure for the total number of sentences in the language up to the sentence in D with the highest index and $|I_D|$ is the size of D.*

The notion of an initial segment is introduced to make the argument work for infinite languages. We have $K(D|A) \leq K(I_D) + O(1) \leq \log \binom{|I_{\leq D}|}{|I_D|} + O(1)$. Suppose that f is an accepting state of a DFA A, with index set I and that $D \subseteq L(A)$.

Definition 6. *The maximal* state entropy *of f given D is $I_{\leq D,f} = \log \binom{|I_{\leq D,f}|}{|I_{D,f}|}$, where $I_{\leq D,f}$ and $I_{D,f}$ identify those indexes that are associated with strings that are accepted in f.*

These theoretical definitions can be used to define a nearly optimal data-to-model code.

Suppose A is a DFA suggested as an explanation for a data set D. A has i accepting states and j non-accepting states. Since we use both positive and negative examples A must be functionally complete (i.e. have an outgoing arrow for each element of the lexicon from each state). Suppose l is the maximal length of a string in the data set D. D^+ is the set of positive examples, D^- the set of negative examples, d^+ is the number of positive examples, d^- the number of negative examples. There are $2^{l+1} - 1$ binary strings with length $\leq l$. Call this set N, then n^+ is the number of strings accepted by A and n^- is the number of strings not accepted by A. A partitions N in two sets: N^+ and N^-. N^+ is partitioned in i subsets by the i accepting states of A and N^- is partitioned in j subsets by the j non-accepting states of A. The correct data-to-model code has size:

$$\log(\prod_i \binom{n_i^+}{d_i^+} \times \prod_j \binom{n_j^-}{d_j^-}) = \sum_i (\log \binom{n_i^+}{d_i^+}) + \sum_j (\log \binom{n_j^-}{d_j^-}) \qquad (9)$$

One can read this as follows. The formula specifies an index for the data set D given the data set N. There are i pieces of code for the positive states, and j pieces of code for the negative states. If there are states that do not generate elements for D then their contribution to the length of the code is 0. When applying this formula to DFA induction one must estimate the values using the Stirling formula or an integral over $\log n!$[2]. Remember that from an MDL perspective we are only interested in the length of the index, not its specific value. The beautiful thing is that this index can always be used: for positive examples, for complete examples and even for only negative examples.

We have tried this MDL approach on the problem set of the Abbadingo DFA inference competition [17]. We were able to solve problems 1, 2, A, B, C, D, and R. In comparison, standard EDSM can solve all these problems, and also problems 3, 4, 6, and S. So, indeed, it seems that MDL is not a very reliable guide for the compression of a DFA. At least, EDSM is better.

4 Conclusion

I have described the general principles behind two-part code optimization. I have studied MDL in terms of two-part code optimization and randomness deficiency for DFA induction. In this framework we noted that 1) Shorter code does not necessarily lead to better theories, e.g. the randomness deficiency does not decrease monotonically with the MDL code, 2) contrary to what is suggested by the results of [13] there is no fundamental difference between positive and negative data from an MDL perspective, 3) MDL is extremely sensitive to the correct calculation of code length. Using these ideas we have implemented a MDL variant of the EDSM algorithm [17]. The results show that although MDL works well as a global optimization criterion, it falls short of the performance of algorithms that evaluate local features of the problem space. MDL can be described as a global strategy for featureless learning. In the tutorial I will describe recent developments like normalized compression distance (NCD) and also present some philosophical reflection on the data compression and thermodynamics.

[2] The formula $\log \binom{n}{k}$ can be approximated by: $\log \binom{n}{k} \approx \int_{n-k}^{n} logx \; dx - \int_{1}^{k} logx \; dx$, which is easy to compute. Already for $k = 65$ the error is less than 1% and rapidly decreasing.

Bibliography

[1] Adriaans, P., Vitányi, P.: The Power and Perils of MDL, IEEE Trans. Inform. Th. (submitted)

[2] Adriaans, P.W.: The philosophy of learning, Handbook of the philosophy of information. In: Adriaans, P.W., van Benthem, J. (eds.) Handbook of the philosophy of science, Series edited by Gabbay, D. M., Thagard, P., Woods, J. (to appear)

[3] Adriaans, P.W.: Learning Deterministic DEC Grammars Is Learning Rational Numbers. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS (LNAI), vol. 4201, pp. 320–326. Springer, Heidelberg (2006)

[4] Adriaans, P.W.: Using MDL for Grammar Induction, in Grammatical Inference: Algorithms and Applications. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS (LNAI), vol. 4201, pp. 293–306. Springer, Heidelberg (2006)

[5] Cilibrasi, R., Vitányi, P.: Clustering by compression, IEEE Trans. Infomat. Th., Submitted. See `http://arxiv.org/abs/cs.CV/0312044`

[6] Cilibrasi, R., Vitányi, P.M.B.: Automatic Meaning Discovery Using Google (2004), `http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0412098`

[7] Domingos, P.: The Role of Occam's Razor in Knowledge Discovery. Data. Mining and Knowledge Discovery 3(4), 409–425 (1999)

[8] Barron, A., Rissanen, J., Yu, B.: The minimum description length principle in coding and modeling. IEEE Trans. Information Theory 44(6), 2743–2760 (1998)

[9] Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)

[10] Li, M., Vitányi, P.M.B.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn. Springer, New York (1997)

[11] Vereshchagin, N.K., Vitányi, P.M.B.: Kolmogorov's structure functions and model selection. IEEE Trans. Information Theory 50(12), 3265–3290 (2004)

[12] Grünwald, P.D., Langford, J.: Suboptimal behavior of Bayes and MDL in classification under misspecification. Machine Learning (2007)

[13] Gold, E.: Mark, Language Identification in the Limit. Information and Control 10(5), 447–474 (1967)

[14] Pitt, L., Warmuth, M.K.: The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial. Journal of the ACM 40(1), 95–142 (1993)

[15] Adriaans, P., Vervoort, M.: The EMILE 4.1 grammar induction toolbox. In: Adriaans, P., Fernau, H., van Zaanen, M. (eds.) ICGI 2002. LNCS (LNAI), vol. 2484, pp. 293–295. Springer, Heidelberg (2002)

[16] Vervoort, M.: Games, walks and Grammars, Thesis University of Amsterdam (2000)

[17] Lang, K.J., Pearlmutter, B.A., Price, R.A.: Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In: Adriaans, P., Fernau, H., van Zaanen, M. (eds.) ICGI 2002. LNCS (LNAI), vol. 2484, pp. 1–12. Springer, Heidelberg (2002)

[18] van Zaanen, M., Adriaans, P.: Alignment-Based Learning versus EMILE: A Comparison. In: Proceedings of the Belgian-Dutch Conference on Artificial Intelligence (BNAIC), pp. 315–322. Amsterdam, the Netherlands (2001)

[19] Solan, Z., Horn, D., Ruppin, E., Edelman, S.: Unsupervised learning of natural languages. PNAS 102(33), 11629–11634 (2005)

[20] Curnéjols, A., Miclet, L.: Apprentissage artificiel, concepts et algorithmes, Eyrolles (2003)

[21] Gerard Wolff, J.: Unifying Computing And Cognition, The SP Theory and its Applications, CognitionResearch.org.uk (2006)
[22] Wolff, J.G.: Computing As Compression: An Overview of the SP Theory and System. New Generation Comput. 13(2), 187–214 (1995)
[23] Wolff, J.G.: Information Compression by Multiple Alignment, Unification and Search as a Unifying Principle in Computing and Cognition. Journal of Artificial Intelligence Research 19(3), 193–230 (2003)
[24] Proceedings of the Workshop and tutorial de la Higuera, D., Oncina, J., Adriaans, P., van Zaanen, M.: Learning Context-Free Grammars. In: Lavrač, N., Gamberger, D., Todorvski, L., Blockeel, H. (eds.) ECML 2003 and PKDD 2003. LNCS, vols. 2837 and 2838. Springer, Heidelberg (2003)