Using MDL for grammar induction[*]

Pieter Adriaans[†] and Ceriel Jacobs[‡]

Abstract

In this paper we study the application of the Minimum Description Length principle (or two-part-code optimization) to grammar induction in the light of recent developments in Kolmogorov complexity theory. We focus on issues that are important for construction of effective compression algorithms. We define an independent measure for the quality of a theory given a data set: the *randomness deficiency*. This is a measure of how typical the data set is for the theory. It can not be computed, but it can in many relevant cases be approximated. An optimal theory has minimal randomness deficiency. Using results from Vereshchagin and Vitányi [2004] and Adriaans and Vitányi [2005] we show that:

- Shorter code not necessarily leads to better theories. We prove that, in DFA induction, already as a result of a single deterministic merge of two states, divergence of randomness deficiency and MDL code can occur.
- Contrary to what is suggested by the results of Gold [1967] there is no fundamental difference between positive and negative data from an MDL perspective.
- MDL is extremely sensitive to the correct calculation of code length: model code and data-to-model code.

These results show why the applications of MDL to grammar induction so far have been disappointing. We show how the theoretical results can be deployed to create an effective algorithm for DFA induction. However, we believe that, since MDL is a global optimization criterion, MDL based solutions will in many cases be less effective in problem domains where local optimization criteria can be easily calculated. The algorithms were tested on the Abbadingo problems (Lang et al. [1998]). The code was in Java, using the Satin (van Nieuwpoort et al. [2005a]) divide-and-conquer system that runs on top of the Ibis (van Nieuwpoort et al. [2005b]) grid programming environment.

[*]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).

[†]Address: Department of Computer Science, University of Amsterdam, Kruislaan 419, 1098VA Amsterdam, The Netherlands. Email: `pietera@science.uva.nl`.

[‡]Address: Department of Computer Science, Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands. Email: `ceriel@cs.vu.nl`.

1 Introduction: MDL and grammar 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 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 (Wolff [2003], Wolff [1995]), EMILE (Adriaans and Vervoort [2002] Vervoort [2000]), ABL (van Zaanen [2002]), ADIOS (Solan et al. [2005]) and a number of DFA induction algorithms, specifically evidence driven state merging (EDSM), (Lang et al. [1998], de la Higuera et al. [2003]). In this paper we present a sound theoretical basis to analyze the performance and idiosyncrasies of these algorithms in an MDL context.

2 MDL as two-part code optimization

We give the traditional formulation of MDL:

Definition 2.1 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 (Mitchell [1997]) 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. Ergo:

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

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 Vereshchagin and Vitányi [2004]. 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 Adriaans and Vitányi [2005]. 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 2.2 *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 Li and Vitányi [1997]. 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.

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. 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. The factor $O(1)$ is needed for additional 'syntactic sugar' to reconstruct x from M and the index. Its importance is thus limited. Let $D \subseteq M$ be a subset of a finite model M. We specify $d = |D|$ and $m = |M|$. Now we have:

$$K(D|M,d) \leq \log\binom{m}{d} + O(1) \tag{3}$$

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 3 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:

$$\delta(D|M,d) = \log \binom{m}{d} - K(D|M,d), \tag{4}$$

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

$$\beta_x(\alpha) = \beta_D(\alpha) = \min_M \{\delta(D|M) : M \supseteq D,\ K(M) \leq \alpha\}, \tag{5}$$

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 Vereshchagin and Vitányi [2004].

2.2.1 Learning as incremental compression

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:

$$\Lambda(M_i, d) = \log \binom{m_i}{d} + K(M_i) \geq K(D) - O(1) \tag{6}$$

Adriaans and Vitányi [2005] 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.

3 Using MDL for DFA induction

In the rest of this paper we will study these theoretical results in the practical context of DFA induction. We will follow the presentation in Curnéjols and Miclet [2003]. Below we show that the model improvement between two consecutive compression states during the execution of a DFA learning algorithm in general can not be computed.

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 (Pitt and Warmuth [1993]).

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 the set of infinite regular languages consistent with D^+. MDL identifies such a DFA. Yet, if our results above are relevant, MDL will not help us to *construct* such a DFA in terms of a process of incremental compression or expansion.

Intuitively the MDL code would not give any guidance for compression (or expansion) of a theory if we could show that the randomness deficiency behaves independently of the MDL code: i.e the randomness deficiency could either grow or shrink with a reduction of the length of the MDL code. Below we will show that this is possible for any algorithm that merges or splits states in a DFA. The crucial insight is that when we merge two states (i.e. reduce the complexity of the model) the resulting index set for the data (i.e. the data-to-model code) in general becomes more complex, but also can be more simple. The effects are non-computable. This implies that MDL in the case of DFA is not a good guide when compressing or expanding theories. Before we prove this we give some definitions.

Definition 3.1 *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 3.2 *Let* $A = (Q, \Sigma, \delta, q_0, F)$ *be a DFA. The* quotient automaton $A/\pi = (Q', \Sigma, \delta', B(q_0, \pi), F')$ *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 3.3 *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 the generality of the automaton and the associated language inclusion hierarchies by means of splitting and merging states.

Definition 3.4 *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 3.5 *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.*

Note that there are data sets of low complexity for which the strict inequality $K(I_D|I_{\leq D}) < \sum_{f \in F} \log \binom{|I_{\leq D,f}|}{|I_{D,f}|} < \log \binom{|I_{\leq D}|}{|I_D|}$ holds. Yet for these data sets the individual state entropies might be still high, since any set of low complexity can be split into two sets of high complexity. For such data sets in general the data-to-model code will be optimized when states are merged. On the other hand if $K(I_D|I_{\leq D})$ is 'close' to $\log \binom{|I_{\leq D}|}{|I_D|}$ splitting of states is a better strategy.

According to our definition above M_0 is not worse than M_1 (as an explanation for D), in symbols: $M_0 \leq M_1$, if

- $\delta(D|M_0, d) \leq \delta(D|M_1, d)$; and
- $\Lambda(M_0, d) \leq \Lambda(M_1, d)$.

We will call these conditions the *MDL-step-conditions*. We have to remember that $\delta(D|M, d) = \log \binom{m}{d} - K(D|M, d)$ is the *randomness deficiency* and $\Lambda(M) = \log \binom{m}{d} + K(M) \geq K(D) - O(1)$ is the *total length of two-part code*, or MDL code, of D with help of model M and d. The step from M_1 to M_0 would violate the MDL-step-conditions if $\Lambda(M_0, d) \leq \Lambda(M_1, d)$ and $\delta(D|M_0, d) > \delta(D|M_1, d)$, i.e., if:

$$K(D|M_0, d) - K(D|M_1, d) < \log \binom{m_0}{d} - \log \binom{m_1}{d} \leq K(M_1) - K(M_0) \quad (7)$$

The following theorem states that these violations occur:

Theorem 3.6 *There are combinations of DFA's and data sets D, A/π_0 and A/π_1 such that $D \subseteq L(A/\pi_0) = L(A/\pi_1)$ and $A/\pi_1 = (A/\pi_0)/\pi_{step}$ that violate the MDL-step-conditions. The characteristics of these violations can not be computed. The δ function can fluctuate arbitrarily in a band-width of $\pm K(\pi_{step}) + O(1)$.*

Proof: Take an arbitrary DFA B. Construct two new DFA's A/π_0 and A/π_1 along the following lines: A/π_0 contains two copies of B, B_1 and B_0, the start state q_0 of A/π_0 has two outgoing transitions, one labelled 1 to the start state of B_1 and one labelled 0 to the start state of B_0. The language recognized by A/π_0 is $(0 + L(B)) \cup (1 + L(B))$. A/π_1 contains only one copy of B, the start state q_0 of A/π_1 has two outgoing transitions labelled 1 and 0 to the start state of B. The language recognized by A/π_1 is $\{0,1\} + L(B)$. It is easy to verify that $\{0,1\} + L(B) = (0 + L(B)) \cup (1 + L(B))$, thus $L(A/\pi_0) = L(A/\pi_1)$. Also there is a partition π_{step} such that $A/\pi_1 = (A/\pi_0)/\pi_{step}$: we simply merge B_1 and B_0 in A/π_0 to get A/π_1.

Take the terms of the inequality 7. Independent of any data set $\log\binom{m_0}{d} - \log\binom{m_1}{d} = 0$ since $L(A/\pi_0) = L(A/\pi_1)$. Furthermore $K(A/\pi_0) > K(A/\pi_1)$, since $K(A/\pi_0)$ has strictly more states even if we correct for the redundancy of having two copies of B.

What we have to show is that there are data sets D for which the term $K(D|M_0,d) - K(D|M_1,d)$ can take any value (within a certain bandwidth). Let I_0 and I_1 be index sets associated with respectively A/π_0 and A/π_1. Note that I_0 and I_1 are not independent. Given I_0, A/π_0 and $(A/\pi_0)/\pi_{step}$ we can reconstruct I_1. Therefore $K(I_1|A/\pi_0, I_0) \leq K(\pi_{step}) + O(1)$, i.e. the maximal difference in complexity between I_0 and I_1 given A/π_0 is $K(\pi_{step})$, the complexity of the transformation between the two DFA's. This distance is symmetric within $O(1)$. This limits the bandwidth of the expression $K(D \mid M_0, d) - K(D|M_1, d)$.

Now select an accepting state $f_{1,\{0,1\}}$ in A/π_1 that is the result of merging two corresponding accepting states $f_{0,\{0\}}$ and $f_{0,\{1\}}$ in A/π_0. The related index sets are $I_{0,D,f_{0,\{0\}}}$, $I_{0,D,f_{0,\{1\}}}$ and $I_{1,D,f_{1,\{0,1\}}}$. The maximal state entropy for $f_{0,\{0\}}$ is given by

$$\log\binom{|I_{0,\leq D,f_{0,\{0\}}}|}{|I_{0,D,f_{0,\{0\}}}|}$$

similarly for $f_{0,\{1\}}$ and $f_{1,\{0,1\}}$. If these index sets are random then we have

$$K(I_{0,D,f_{0,\{0\}}}) + K(I_{0,D,f_{0,\{1\}}}) < K(I_{1,D,f_{1,\{0,1\}}})$$

since

$$\log\binom{|I_{0,\leq D,f_{0,\{0\}}}|}{|I_{0,D,f_{0,\{0\}}}|} + \log\binom{|I_{0,\leq D,f_{0,\{1\}}}|}{|I_{0,D,f_{0,\{1\}}}|} < \log\binom{|I_{1,\leq D,f_{1,\{0,1\}}}|}{|I_{1,D,f_{0,\{0,1\}}}|} =$$

$$\log\binom{|I_{0,\leq D,f_{0,\{0\}}}| + |I_{0,\leq D,f_{0,\{1\}}}|}{|I_{0,D,f_{0,\{0\}}}| + |I_{0,D,f_{0,\{1\}}}|}$$

In this case we do not benefit from merging states. Note that we are completely free to select D in such a way that any binary partition of indexes from $I_{1,D,f_{1,\{0,1\}}}$ into corresponding indexes in $I_{0,D,f_{0,\{0\}}}$ and $I_{0,D,f_{0,\{1\}}}$ is realized. In particular we can select a partition that is random in $I_{0,D,f_{0,\{0\}}}$ and $I_{0,D,f_{0,\{1\}}}$ and highly non-random in $I_{1,D,f_{1,\{0,1\}}}$. In this case we have:

$$K(I_{0,D,f_{0,\{0\}}}) + K(I_{0,D,f_{0,\{1\}}}) > K(I_{1,D,f_{1,\{0,1\}}})$$

Here we *do* benefit from merging states. Note that a corresponding argument holds for all triples $f'_{1,\{0,1\}}$, $f'_{0,\{0\}}$ and $f'_{0,\{1\}}$. By selecting appropriate data sets we can create arbitrary fluctuations in the difference of the conditional complexity of the index sets $I_{0,D}$ and $I_{1,D}$ of M_o and M_1. Since $K(D|M_0, d) = K(I_{0,D}|M_0, d) + O(1)$ and $K(D|M_1, d) = K(I_{1,D}|M_1, d) + O(1)$ we can give the expression

$$K(D|M_0, d) - K(D|M_1, d)$$

any value in the bandwidth $\pm K(\pi_{step})$. Note that we can define partitions of any complexity so that the value of $K(\pi_{step})$ can be arbitrary large and that the fluctuations cannot be computed because they are based on the conditional Kolmogorov complexity of the index sets.

End of proof.

Corollary 3.7 *Given a sample D of a regular language and an arbitrary DFA_D consistent with D, the* optimal *DFA_{opt} for D can not be approximated by means of expanding or compressing states in DFA_D. This holds if $D = D^+$, $D = D^-$ and if $D = D^+ \cup D^-$.*

Proof: Case: $D = D^+$, is immediately implied by theorem 3.6. Case: $D = D^-$, is implied by the previous case and the fact that the complement of a regular language is a regular language. The optimal automaton for the negative cases is the same as the one for the positive cases, with the accepting and non-accepting states exchanged, i.e. we try to construct an automaton with exactly the same structure. Case: $D = D^+ \cup D^-$, is implied by the previous two cases and the fact that in the proof of theorem 3.6 the languages accepted by M_0 and M_1 are the same. The positive randomness deficiency can fluctuate independent of the influence of the negative examples. If the randomness deficiency in relation to the positive examples fluctuates, then the randomness deficiency with respect to the negative examples will also fluctuate, i.e. the total randomness deficiency will fluctuate.

End of proof.

3.1 Discussion

It is worth noting that, given the results of Pitt and Warmuth [1993] it was not to be expected that there would exist an algorithm that finds the DFA with minimal generalization error on the basis of D^+ in polynomial time. Under the condition that D^+ is dense enough and given the results above, the *optimal* DFA in terms of MDL will with high probability also be the *minimal* DFA consistent with D^+ and an arbitrary D^-. If we could find the optimal MDL DFA in polynomial time then we could simply focus on D^+, ignoring D^-, in order to find the minimal DFA consistent with D^+ and D^-. This would contradict the result of Pitt and Warmuth [1993]. Note also that the task of finding a *minimal* automaton consistent with $D = D^+ \cup D^-$ is in many cases much simpler than the task of finding the automaton with *optimal randomness deficiency*, e.g. take the random case where there is only one negative example.

Following Gold [1967] it is generally thought that a combination of positive and negative examples gives a better chance of learning a language than positive examples alone. Corollary 3.7 shows that this is only the case in the limit when we can enumerate all solutions. Additional negative examples in general do not help us more than additional positive examples when we want to *construct* better solutions out of bad ones given a finite set D.

4 Implementing MDL for DFA induction

Theorem 3.6 implies that MDL in general will not be a reliable guide for the compression of a DFA. Yet there exist many DFA induction algorithms that use compression. In a lot of empirical cases this approach seems to work. The results presented above suggest that it is possible to use MDL directly as an empirical guidance for the merging of states in DFA induction. They also imply that we can use the same MDL measure both on positive and on complete data sets. The MDL data-to-model code formula below shows that this is indeed the case.

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 Dataset 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, n^+ is the number of strings accepted by A, 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 is:

$$\log(\prod_i \binom{n_i^+}{d_i^+} \times \prod_j \binom{n_j^-}{d_j^-}) = \sum_i (\log \binom{n_i^+}{d_i^+}) + \sum_j (\log \binom{n_j^-}{d_j^-}) \tag{8}$$

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 $\log n!$[1]. 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.

[1]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.

5 Some experiments

We have developed a framework for experimenting with various search heuristics. The framework is built around the blue-fringe algorithm, which works as follows: We start with the prefix tree acceptor, derived from the sample. The root is colored red, its children are colored blue, all other states are white. Then, we apply the following actions:

- As long as there are blue states that cannot be merged with any red node, we promote the shallowest of these states to red, and color its children blue.
- We compute the score for merging all blue/red pairs (see below for an explanation of 'score').
- We merge the best scoring blue/red pair and color all children of red states that are not red themselves blue.

We repeat these steps until no merges are possible.

The score for merging a red/blue pair is the search heuristic. When using evidence driven state merging (EDSM) (Lang et al. [1998], de la Higuera et al. [2003]), the score consists of the number of matching state labels of merged states, the higher the better.

We have used several variants of MDL for the search heuristic. As MDL is the sum of two parts, we describe these parts in detail below.

5.1 MDL scoring

For an approximation of $K(M)$, the model code, we used the following reasoning: suppose a DFA A has n states, the alphabet Σ has $|\Sigma|$ symbols, and A is functionally complete. Then, there are $n \times |\Sigma|$ state transitions, each with n possible destinations. This gives $n^{n\times|\Sigma|}$ possibilities. Furthermore, each state is either accepting or non-accepting. This gives 2^n possibilities. Thus, assuming that state 1 always is the start state, in total there are $2^n \times n^{n\times|\Sigma|}$ possible functionally complete DFA's with n states. However, there is a lot of redundancy here, for each permutation of states $2...n$ there is an equivalent DFA. Thus, the length of an index identifying a specific DFA is $\log(2^n \times n^{n\times|\Sigma|}/(n-1)!) = n + n \times |\Sigma| \times \log(n) - \log((n-1)!)$.

We have experimented with several variations of data-to-model ($K(D|M,d)$) codes:

1. Simple MDL with only positives. This version uses $\log\binom{m^+}{d^+}$.
2. Simple MDL with positives and negatives. This version uses $\log\binom{m^+}{d^+} + \log\binom{m^-}{d^-}$.
3. Simple MDL with positives and complement. This version uses $\log\binom{m^+}{d^+} + \log\binom{(2^{l+1}-1-m^+}{d^-}$.

4. MDL as in equation 8, with only positives.

5. MDL as in equation 8.

Note that there is no difference between MDL as in equation 8 with negatives or complement.

We have tried all of the above variants on the problem set of the Abbadingo DFA inference competition (Lang et al. [1998]). All variants are able to solve problems A, B, C, D, and R. In addition, all variants except for variant 1 can solve problem 1. Variant 4 can also solve problem 2. In comparison, 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.

But EDSM, on its own, fails too on some problems. To solve some of these problems, we have extended our framework allowing it to search a (small) part of the decision tree. The search is to a certain depth, and at each depth only the most promising candidates are maintained in a list. Initially, we place the prefix tree acceptor in the list. Then, for every candidate in the list, for all its red/blue merge candidates, apply the merge and then use the heuristic to compress the DFA until no further merges are possible. Determine the score of the resulting DFA (for EDSM, this is the number of states, for MDL, this is its MDL). We store the DFA from the list, the first merge, and the resulting score in a result list. When all candidates for the list are tried, the result list is sorted according to the score, and only the most promising candidates are kept. For these candidates, we apply its first merge, and store the result in the list for the next depth.

As this process becomes quite compute-intensive, even for moderate values of maximum search depth and list length, we have written this application in Java, using the Satin (van Nieuwpoort et al. [2005a]) divide-and-conquer system that runs on top of the Ibis (van Nieuwpoort et al. [2005b]) grid programming environment. Abbadingo problem 7 was solved using a search depth of 5, a list length of 256, and EDSM as the search heuristic. We did not find a solution when using MDL as the search heuristic. The MDL search heuristic sometimes seems to indicate choices, much deeper in the decision tree, that don't lead to an optimal DFA. In fact, this is the reason why problem 3 is not solved with the MDL heuristic and why problem 2 is only solved by one of the MDL variants: the first 15 or so decisions are decisions that the EDSM heuristic also indicates (albeit in a different order), but then a decision is made that makes a solution impossible. The MDL values of the solutions found with EDSM are lower than the ones found by means of the MDL heuristic, so MDL as a measure is good, but MDL as guidance is not. This suggests that we could use EDSM as the search strategy and MDL as a judge of the result. Indeed, Abbadingo problem 7 was solved with the same depth and list parameters as by using the number of states as a judge of the result. The significance of this is that using the number of states as a judge of the result is not an option when there are only positive examples.

6 Conclusions and further work

We have studied MDL in terms of two-part code optimization and randomness deficiency. In this framework we showed that 1) Shorter code not necessarily leads 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 Gold [1967] 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. We have proved that already as a result of a single merge the divergence of randomness deficiency and MDL code can occur. Using these ideas we have implemented a MDL variant of the EDSM algorithm (Lang et al. [1998]). 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.

Suggestions for further work are: the extension of these ideas to more complex language classes, the implementation of a better estimate of the model code, the development of strategies for hybrid MDL learning (MDL applied to local representations of the problem space to bypass local optima) and a more efficient implementation of the algorithm on the grid.

References

Adriaans, P. and Vervoort, M. (2002). The EMILE 4.1 grammar induction toolbox. In Adriaans, P., Fernau, H., and van Zaanen, M., editors, *Grammatical Inference: Algorithms and Applications; 6th International Colloquium, ICGI 2002*, volume 2484 of *LNCS/LNAI*, pages 293–295. Springer.

Adriaans, P. and Vitányi, P. (2005). The power and perils of MDL. Technical report, Human Computer Studies Lab, Universiteit van Amsterdam.

Curnéjols, A. and Miclet, L. (2003). *Apprentissage artificiel, concepts et algorithmes.* Eyrolles.

de la Higuera, C., Adriaans, P., van Zaanen, M., and Oncina, J., editors (2003). *Proceedings of the Workshop and Tutorial on Learning Context-Free Grammars held at the 14th European Conference on Machine Learning (ECML) and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD); Dubrovnik, Croatia.*

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10(5):447–474.

Lang, K. J., Pearlmutter, B. A., and Price, R. A. (1998). Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In Honavar, V. and Slutzki, G., editors, *Grammatical Inference; 4th International Colloquium, ICGI-98*, volume 1433 of *LNCS/LNAI*, pages 1–12, Ames, Iowa, USA. Springer.

Li, M. and Vitányi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, New York, 2 edition.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill, New York.

Pitt, L. and Warmuth, M. K. (1993). The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, 40(1):95–142.

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

van Nieuwpoort, R. V., Maassen, J., Kielmann, T., and Bal, H. E. (2005a). Satin: Simple and efficient Java-based grid programming. *Scalable Computing: Practice and Experience*, 6(3):19–32.

van Nieuwpoort, R. V., Maassen, J., Wrzesinska, G., Hofman, R., Jacobs, C., Kielmann, T., and Bal, H. E. (2005b). Ibis: a flexible and efficient Java based grid programming environment. *Concurrency and Computation: Practice and Experience*, 17(7-8):1079–1107.

van Zaanen, M. (2002). *Bootstrapping Structure into Language: Alignment-Based Learning.* PhD thesis, University of Leeds, Leeds, UK.

Vereshchagin, N. and Vitányi, P. (2004). Kolmogorov's structure functions and model selection. *IEEE Trans. Information Theory*, 50(12):3265–3290.

Vervoort, M. (2000). *Games, walks and Grammars.* PhD thesis, University of Amsterdam, Amsterdam.

Wolff, J. G. (1995). Computing as compression: An overview of the SP theory and system. *New Generation Comput.*, 13(2):187–214.

Wolff, J. G. (2003). 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.