Knowledge Discovery in Biosequences Using Sort Regular Patterns

Toru Takae† Toru Kasai†, Hiroki Arimura†, Takeshi Shinohara‡

†Dept. of Informatics
Kyushu University
Fukuoka 812-8581, Japan
{takae,kasai,arim}@i.kyushu-u.ac.jp

‡Dept. of Artificial Intelligence
Kyushu Institute of Technology
Iizuka 820–8502, Japan
shino@ai.kyutech.ac.jp

Abstract

This paper considers knowledge discovery by sort regular patterns, which are strings over sort letters representing finite sets of basic letters. We devise a learning algorithm for the class based on the minimal multiple generalization technique, and evaluate the method by experiments on biosequences from GenBank database. The experiments show that relatively a simple sort pattern can represent a complex motif in biosequences, and the learning algorithm works well in noisy examples.

1 Introduction

Discovery of consensus motifs plays an important role in automatic analysis of biosequences. A *consensus motif* is a description of common syntactic features, in terms of sequences, of a group of biosequences that are believed to be biologically related in structures, functions, or origins. Therefore, discovery of consensus motifs is an important step to understand the meaning of biosequences. We consider the discovery of consensus motifs by a class of simple string patterns.

In bioinformatics, biosequences, such as *nucleotide sequences* for DNA and *amino acid sequences* for proteins, are simply regarded as long strings of basic letters. For example, the alphabet of basic letters is a four letter alphabet $\Sigma = \{\mathtt{A}, \mathtt{T}, \mathtt{C}, \mathtt{G}\}$ for DNA and a 20 letter alphabet $\Sigma = \{\mathtt{D}, \mathtt{E}, \mathtt{K}, \ldots\}$ for proteins. We model a consensus motif by a *regular pattern*, which is a string p of basic letters and the wildcard '$*$' such as $*\mathtt{LF}*\mathtt{KBG}*\mathtt{L}*\mathtt{A}*$. A pattern p *matches* a biosequence if we obtain the sequence from p by replacing wildcards '$*$' with possibly distinct strings. A set of patterns matches a sequence if at least one of the patterns in the set matches the sequence.

In [3], we devised a learning algorithm, called *minimal multiple generalization* (MMG for short) that efficiently finds a set of at most k patterns from *positive* examples alone for protein motif discovery. We designed the algorithm to find a *most specific* hypothesis containing all positive examples to avoid *overgeneralization*, which is a major problem in learning from positive examples. Actually, we showed that the MMG algorithm identifies any set of bounded number of regular patterns from positive examples in the limit under a mild condition on Σ.

The application of MMG algorithm with the hypothesis space of regular patterns is successful for some prediction problems that are well studied and known to be easy, e.g., *transmembrane domain* but not significant for a rather difficult problem such as *signal peptide sequences* [3]. A positive sample set of signal peptide sequences consists of segments taken from many kinds of peptides with different functionalities. Further, some

amino acids at some position may be often replaced with other amino acids of similar property [9]. Therefore, it is less likely that there exists a rigid pattern that explains all signal peptides with high accuracy. This problem is frequently observed in the analysis of biosequences, and also in practical learning problems in noisy environments.

An approach to this problem is to use *alphabet indexing*. An alphabet indexing [8] is a mapping that transforms the letters of an original alphabet Σ into the letters of a smaller alphabet I, where two letters mapped to the same letter are regarded as similar amino acids. Shimozono *et al.* have reported that the use of alphabet indexing greatly improves the accuracy and the simplicity of the hypotheses found by a learning algorithm. Arimura *et al.* uses alphabet indexing same to [8] in an application of MMG algorithm and the algorithm finds moderately good hypotheses. Yamaguchi *et al.* [10] combines MMG algorithm with local search of alphabet indexing.

Another approach is to give a pattern the capability of approximately representing a motif. We extend regular patterns by allowing the use of sort symbols. A *sort symbol* is an expression of the form `[ABFLV]` and represents a finite set $\{\mathtt{A}, \mathtt{B}, \mathtt{F}, \mathtt{L}, \mathtt{V}\}$ of basic letters as the syntax indicates. Let Π be an alphabet of sort symbols. A *sort regular pattern* (a *sort pattern* for short) is a string over $\Pi \cup \{*\}$. Matching of a sort pattern is defined similarly to matching for ordinary regular patterns except that a sort symbol α can be replaced with a basic letter contained in the set represented by α. For example, `*[LF]AL*B[LMS]B*` is a sort pattern and it matches both `MLALRBLBR` and `MFALRBMBR`. Note that the class of sort pattern is exactly a subclass of PROSITE patterns [4] such that the wildcard with the length restriction, such as `*(3,8)`, is prohibited.

In this paper, we propose a learning algorithm for sort regular patterns and apply the algorithm to protein motif discovery from biosequences. First, we investigate a generalization structure of sort patterns and then introduce minimal multiple generalization for sort patterns.

Then, we define an efficient and complete refinement operator for the class. Using this operator, we present a modification of MMG algorithm that computes a minimal multiple generalization consisting of at most k sort regular patterns from positive examples in polynomial time in the total length of positive examples.

We evaluate the effectiveness of the method by the experiments on signal peptide sequences drawn from GenBank database. We compare the proposed algorithm with normal MMG algorithm without indexing described in [3] and the MMG algorithm with local search of partial indexing in [10] on the same dataset. As a result, we observe that MMG algorithm with sort patterns almost constantly produces a hypothesis of high accuracy, while the performance of the other two methods depends heavily on the choice of the size of training examples and frequently poorer than that with sort patterns. The learning curve of these algorithms show that MMG algorithm with sort patterns will be robust for noises.

2 Sort Regular Patterns and MMG

2.1 Sort regular patterns

We introduce sort regular patterns following the formulation in [5]. For a set A, we denote by $\#A$ the number of the elements of A, by A^* the set of all finite strings over A, and by A^+ the set $A^* - \{\varepsilon\}$, where ε is the empty string whose length is zero.

Let Σ be the basic alphabet of *basic letters*. A *sort symbol* is the representation $\alpha = [a_1 \ldots a_m]$ that represents the set $L(\alpha) = \{a_1, \ldots, a_m\} \subseteq \Sigma$ of basic letters. Let Π be the alphabet of sort symbols. In this paper, we assume that Π contains the sort symbols that correspond to all nonempty subsets of Σ, that is, $\Pi = \{[a_1 \ldots a_m] \mid \{a_1, \ldots, a_m\} \subseteq \Sigma, m \geq 1\}$. Thus, Π consists of all basic letters $a \in \Sigma$ as singleton sets $[a]$ and the universal

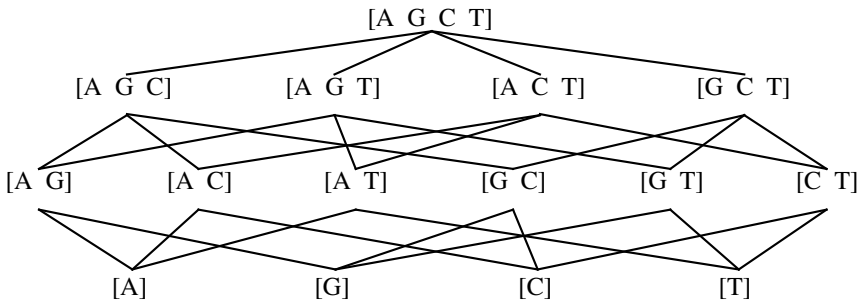

Figure 1: The sort structure of a sort alphabet Π

set Σ as $[\Sigma]$ (Figure 1). The meaning of ordinary symbols are defined as follows. The *wildcard* is denoted by $* \notin \Pi$ and represents the set of all the possibly empty strings $L(*) = \Sigma^*$.

Definition 1 A *sort regular pattern* (sort pattern for short) is a string $p \in (\Pi \cup \{*\})^*$.

We denote the length of p by $|p|$. For a sort pattern $p = \alpha_1\alpha_2 \cdots \alpha_m (\alpha_i \in \Pi \cup \{*\})$, we define the language of p by $L(p) = L(\alpha_1)L(\alpha_2) \cdots L(\alpha_m)$. For a set P of sort patterns, we define the language of P by the union $L(P) = \bigcup_{p \in P} L(p)$. A pattern p *matches* a string $t \in \Sigma^*$ if $t \in L(p)$.

If p has no consecutive occurrences of $*$, that is, $p = p_1 * \cdots * p_n$, where $n \geq 0$ and $p_i \in \Pi^+$ for every $1 \leq i \leq n$, then we say that p is *in canonical form*. We denote by RPS the set of the sort regular patterns in canonical form.

2.2 Minimal multiple generalizations

We define a binary relation $\succeq$ over Π by for every $\alpha, \beta \in \Pi$, $\alpha \succeq \beta$ iff $L(\alpha) \supseteq L(\beta)$. For any pattern $p \in RPS$, we define $* \succeq p$. If $\alpha \succ \beta$ then we say that α is more general than β or β is more specific than α. We extend $\succeq$ to a binary relation over patterns as follows.

Definition 2 For pattern p and q, we define $p \succeq q$ if q is obtained from p by replacing some symbols $\alpha \in \Pi \cup \{*\}$ in p with more specific objects $\beta \in \Pi \cup RPS$. That is, for pattern p and q, we define $p \succeq q$ if $p = \alpha_1 \cdots \alpha_m$, $(\alpha_i \in \Pi \cup \{*\})$ and there is some factorization $q = \beta_1 \cdots \beta_m$ for some $\beta_1, \ldots, \beta_m \in RPS$ such that $\alpha_i \succeq \beta_i$ for every $1 \leq i \leq m$.

If $p \succeq q$ but $q \not\succeq p$ then we write $p \succ q$. If $p \succeq q$ then we say that p is *more general than* q, q is *more specific than* p, or q is a *refinement* of p. If p is a pattern then we know that $L(p) = \{\, t \in \Sigma^* \mid p \succeq t \,\}$.

A set $P \subseteq RPS$ of sort patterns is *reduced* if P contains no p, q such that $p \succ q$. Let k be a positive integer. We denote by RPS^k the set of all the reduced collections of at most k sort patterns in RPS. Now, we define a binary relation $\sqsupseteq$ on RPS^k as follows. For any $P, Q \in RPS^k$, $P \sqsupseteq Q$ iff for any $q \in Q$ there exists some $p \in P$ such that $p \succeq q$. Note that $P \sqsupseteq Q$ implies $L(P) \supseteq L(Q)$, but the converse does not holds in general.

Definition 3 Let $S \subseteq \Sigma^*$ be a finite set of strings. A *k-minimal multiple generalization* (*k-mmg*, for short) of S over RPS^k is a minimal element $P \in RPS^k$ with respect to $\sqsupseteq$ such that $S \subseteq L(P)$, i.e., $L(P) \supseteq S$ and there is no element $Q \in RPS^k$ such that $L(Q) \supseteq S$ and $P \sqsupset Q$.

We say that RPS^k has the *compactness with respect to containment* if $P \sqsupseteq Q$ iff $L(P) \supseteq L(Q)$ for any $P, Q \in RPS^k$. From [2], we know that if RPS^k has the compactness then the polynomial time computability of the k-mmg implies the inductive inferability

Algorithm. $MMG(k, S)$:
Input: An integer $k \geq 0$ and a sample $S \subseteq \Sigma^*$.
Output: A k-mmg P of S over RPS^k.

1 $P := \{*\}$;

2 Repeat the following steps:
  (a) For each $\Delta k := 1$ to $k - \#P + 1$, do:
    — For each $p \in P$, do:
      If there exists a subset $\Delta P = \{p_1, \ldots, p_{\Delta k}\} \subseteq \rho(p)$ such that $L(\Delta P) \supseteq \Delta S$ then goto Step 2.(c), where $\Delta S = S - L(P - \{p\})$.
  (b) Exit the repeat loop and goto Step 3.
  (c) $P := (P - \{p\}) \cup \Delta P$, and then continue the repeat loop.

3 Return the set P.

Figure 2: The algorithm for computing a minimal multiple generalization

of the class RPS^k from positive data with conservative, consistent, and polynomial time update in the framework studied in Angluin [1].

At present, unfortunately, we have not seen what condition make the class have the compactness, and thus do not know if the MMG algorithm is a correct inductive inference algorithm for RPS^k from positive examples. Hence, we use k-mmg as a heuristic method for discovering consensus motifs from biosequences.

2.3 The algorithm

In Fig. 2, we present a polynomial time algorithm for computing a k-mmg of a given finite collection of strings. Given a finite collection S of strings as input, the algorithm starts with the most general hypothesis, say, $\{*\}$ and searches the hypothesis space RPS^k from general to specific by iteratively replacing some pattern $p \in P$ with a set of small number of patterns that are more specific than the original p whenever it is possible.

To make a given pattern p more specific, the algorithm uses a refinement operator ρ. A *refinement operator* ρ for RPS is a mapping ρ that maps a pattern $p \in RPS$ to a finite set $\rho(p)$ of refinements of p. Now, we give the refinement operator for RPS.

Definition 4 For a pattern $p \in RPS$, pattern q is the member of the $\rho(p)$ iff q is obtained from p by applying one of the following operations.

- Select any occurrence of the wildcard $*$ in p, and then replace $*$ with either the patterns "$*[\Sigma]*$", "$*[\Sigma]$", "$[\Sigma]*$", or "ε" with probabilities. We refer to these replacements by those of type 1, type 2, type 3, type 4, respectively.

- Select any occurrence of a sort symbol, say $\alpha \in \Pi$, in p and some symbol $a \in L(\alpha)$. Then replace the occurrence of α with the sort symbol $\beta \in \Pi$ such that $L(\beta) = L(\alpha) - \{a\}$. We refer to this replacement by that of type 5.

We can show that the iterative applications of ρ defined above can exactly generate the proper refinements of any pattern p. Further, $\rho(p)$ is efficiently computable in $|p|$. Thus, from the characterization in [2], we can prove the correctness of the algorithm.

Theorem 1 *For every $k \geq 1$ and $S \subseteq \Sigma^*$, the algorithm MMG in* Fig. 2 *computes a k-mmg of S over sort regular patterns in polynomial time in the total length of the strings in S.*

3 Implementation

We have implemented the proposed algorithm *MMG* in C++. Although the algorithm presented in the previous section correctly produces k-mmg, the empirical studies of mmg algorithms on biosequences [3, 10] have shown that the careful control of in the selections of a pattern $p \in P$ and the generations of the elements of $\rho(p)$ significantly affected the quality of the final hypothesis found by the algorithm. Hence, we have incorporated these control into our implementation. Further, in our implementation, we restrict the patterns to contain a bounded number of wildcards. Although this restriction may violate the completeness in general, our operator ρ defined in Section 2.3 is still complete by this change.

We employ a control strategy called *MIN* from [10]. For the selection of a pattern $p \in P$ to be refined at the second line of Step 2.(a), the pattern p has a higher precedence when $L(p)$ contains less examples in S. For the selections from $\rho(p)$, we employ distinct control strategies separately for $\Delta k = 1$ and $\Delta k \geq 2$.

In the case of $\Delta k = 1$, the pattern q to replace $p \in P$ will be randomly selected from $\rho(p)$ according to the probability in the following table. The table shows the probability to select a replacement of type $t \in \{1, 2, 3, 4, 5\}$, where the second, third, and the fourth column show the probabilities for three biosequence data and the value of the common denominator σ is shown in the last raw:

Type		Plant	Rodent	Bacterial
1	$*[\Sigma]*$	$300/\sigma$	$30/\sigma$	$300/\sigma$
2	$*[\Sigma]$	$600/\sigma$	$60/\sigma$	$600/\sigma$
3	$[\Sigma]*$	$600/\sigma$	$60/\sigma$	$600/\sigma$
4	ε	$1/\sigma$	$1/\sigma$	$1/\sigma$
5	$L(\beta) = L(\alpha) - \{a\}$	$2000/\sigma$	$1200/\sigma$	$2000/\sigma$
σ		3501	1351	3501

Further if the algorithm succeeds to replace some $p \in P$ with a single pattern then the algorithm successively try to select and replace the same pattern p with a single pattern whenever it is possible. In the case of $\Delta k \geq 2$, we try to select a subset ΔP from $\rho(p)$ such that the numbers of examples contained in the members are not balanced, that is, some member cover less examples and other cover more examples.

4 Experimental Results

We have run experiments on biosequences drawn from GenBank [7] to empirically evaluate our method for extracting motifs in biosequences. The problem we examined is the prediction of the signal peptide sequence, which is, given a fragment of an amino acid sequence coding a protein, to predict if the fragment is the coding of a particular kind of prefix called *signal peptide*. We selected this problem as our test case because it is known to be a relatively hard problem and no single motifs with high accuracy has not been found.

4.1 Data

The experiments have used the data on the signal peptide sequences from GenBank database [7]. From three groups of amino acid sequences, namely, *Plant, Rodent, Bacterial*, we obtained large sets `POS` and `NEG` of positive and negative examples as follows. We obtain the positive examples as the set of the initial segments of amino acid sequences according to the indication at the feature field, and the negative examples as the initial segments of 30 amino acids. The number of positive and negative examples in each group is shown in Table 1.

Table 1: The number of examples used in the experiments

Data	Sequences	POS	NEG	pos
Signal Peptides	Plant	370	3074	20∼200
	Rodent	1018	3158	20∼200
	Bacterial	495	7330	20∼200

The leftmost amino acid is almost always with 'M', and thus is removed from the all positive and negative examples. Furthermore, we know from the experience in [10] that the right end of a positive example is likely to consist of a particular set of amino acids, namely, 'A', 'G', or 'S'. To remove the influence of this property, we restrict the candidate patterns to those end with the wildcard $*$ not to capture the appearance of these letters.

4.2 Method

In each run of a learning algorithm, the small training sets pos are chosen randomly from POS. All positive examples in POS, and all negative examples in NEG are used in the final evaluation of hypotheses found by the learning algorithm.

The accuracy of a found hypothesis P is measured by the arithmetic mean $acc = \sqrt{p \cdot n}$ of the positive and the negative accuracies p and n, where p and n are defined by the ratios of the number of positive and negative examples correctly classified by $L(P)$, that is, $p = 100 \times \sharp(\mathtt{POS} \cap L(P))/\sharp\mathtt{POS}$ and $n = 100 \times \sharp(\mathtt{NEG} - L(P))/\sharp\mathtt{NEG}$.

For each value of $n = \#pos$ ranging from 20 to 200 by 20, we iterate 100 runs, and compute the best value, the average value, and the standard deviation of the accuracy acc over these 100 runs. From these measurements, we draw a *learning curve* as a plot of the average or best accuracy to the number of training examples n.

In the experiments, we compare our learning algorithm MMG-RPS to the following algorithms.

- MMG-RAW is the basic MMG algorithm described in [3] that finds a set of at most k regular patterns without sort symbols. The refinement $\rho(p)$ is given as follows; Select any occurrence of the wildcard $*$ in a pattern p, and then replace $*$ with either the patterns "$*a*$", "$*a$", "$a*$", or "ε" for any $a \in \Sigma$ according to a specified control strategy.

- MMG-IDX is a modification of MMG algorithm described in [3] that incorporates local search algorithm to find a partial alphabet indexing using subsets of POS and NEG with size approximately 40 and 3000, respectively. This algorithm divides a sequence into the left and right parts at some fixed division spot, and computes two independent indexings by local search. It uses this pair of indexings to find k-mmg of training examples. We set the size of an index alphabet $\#I = 3$ and the position of a division spot $d = 13$.

Table 2 shows the summary of the values of parameters used in the experiments for three algorithms, where $k = \#P$ is the maximum number of patterns in a hypothesis, m and s are the maximum number of wildcards and sort symbols appearing in a pattern, e is the limit of the number of the exceptional patterns (constant strings) in a hypothesis, and $\mathcal{S}$ is a control strategy. We ran the experiments for three algorithms on computers with Linux (Pentium II 233MH$_z$) or Sun OS 2.5.1(Ultra Sparc II 250 MH$_z$).

Table 2: Learning parameters of three algorithms

Learning Method	k	m	s	e	$\mathcal{S}$
MMG-RPS	5	4	9	2	MIN
MMG-IDX	5	6	—	2	MIN
MMG-RAW	5	4	—	2	MIN

4.3 Results

Accuracy

Fig. 3, Fig. 4, and Fig. 5 show the results of the experiments for the signal peptide data of Plant, Rodent, and Bacterial groups. From the learning curves in these figures, we can observe that MMG-RPS works quite well for signal peptide sequences. MMG-RPS finds hypotheses with moderately high accuracy for most training set sizes n more than $n = 60$ except for very small training set size such as $n = 20$. We can also see that the accuracy does not heavily depend on the training set size n and the variance is quite small. On the other hand, it is observed that MMG-RAW and MMG-IDX produced good hypotheses only for less than $n = 40$, and their accuracies are rapidly decreasing as n grows.

This indicates that the hypothesis spaces for MMG-RAW and MMG-IDX are too poor to approximate signal peptide sequences. Thus, the algorithms might produce overly general hypothesis to explain all the positive training examples.

Hypothesis

Fig. 6 shows best hypotheses found in the experiments on signal peptides of Plant group. In the result for MMG-RPS, a hypothesis typically consists of a single non-constant pattern $*\alpha_1 \ldots \alpha_9 *$, $\pi \in \Pi$, of length around 9 and several constant strings that represents anomalies or noises among examples. It is interesting that most of positive and negative examples are correctly predicted by such a non-constant pattern, and its accuracy is only several percent less than the total accuracy.

In the results for other two algorithms MMG-RAW and MMG-IDX, the hypotheses typically consist of some non-constant patterns and several constant strings to cover all positive examples, but their accuracies are lower than the pattern found by MMG-RPS.

We expect that each sort symbol found by MMG-RPS corresponds to some *substitution group* [9], which is a set of chemically or biologically related amino acid residues. However, we did not confirm this conjecture so far. Also, we did not observe any evident relation between these sort symbols obtained by MMG-RPS and the partial indexing obtained by MMG-IDX.

Running time

Typical running time for performing 100 trials on a workstation with Ultra Sparc II ($250\mathrm{MH_z}$) are about $1.0 \sim 1.3$ hours for MMG-RPS, several minutes for MMG-IDX plus precomputation of a partial index in $1.5 \sim 2.0$ hours, and tens of minutes for MMG-RAW.

Discussion

We have presented a learning algorithm to knowledge discovery in biosequences using sort regular patterns, and confirmed that this approach is effective for extracting motifs from positive examples of signal peptides. The result shows that our algorithm MMG-RPS using sort patterns produces more accurate hypotheses than previous algorithms, MMG-RAW [3] and MMG-IDX [10], for signal peptides. Furthermore, its learning curves show

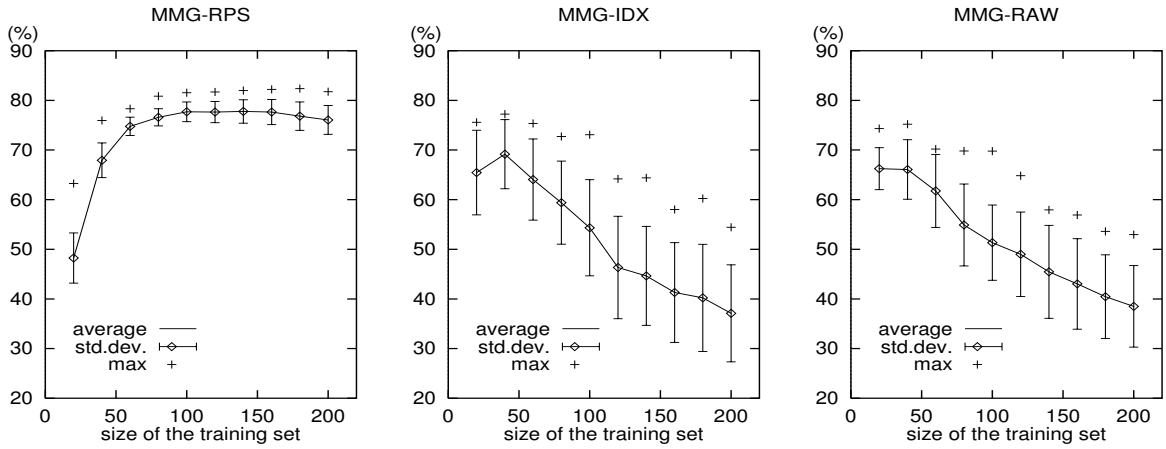

Figure 3: Learning curves for signal peptides (Plant)

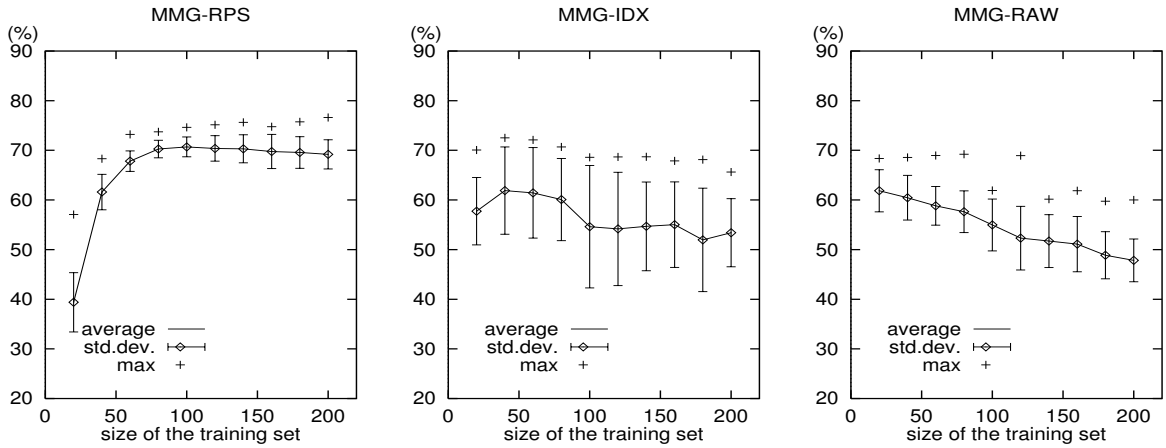

Figure 4: Learning curves for signal peptides (Rodent)

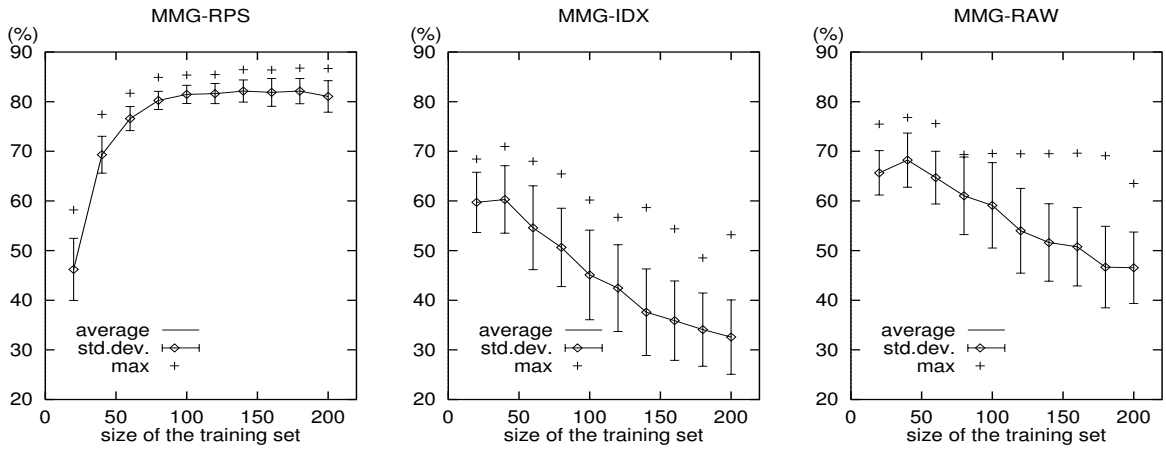

Figure 5: Learning curves for signal peptides (Bacterial)

(a) MMG-RPS: mmg with sort regular patterns

$\sharp pos = 140$

Patterns	$p\%$	$n\%$
LPRSALARSLQLQRGVAARFY	0.3	100
QTLASRPSLRASARVAPRRAPRVAVVTKA	0.3	100
FAFYFLTACISLKGVFG	0.5	100
RFLRGFVFSLAFTLYKVTATA	0.3	100
AMANLARRKGYSLLSSETLRYSFSLRSRAF	0.3	100
ATTIFSRFSIYFCAMLLCQGSMA	0.3	100
* [AFKLP RSTV] [AFIKL MNRSV] [ACFGIL MQSTVY] [AFGKL MSTVW] [ACFI LTV] [ACGIL RSTV] [AFGIL PSTV] [AGI LMS] [AFGI LSTV] *	85.9	76.4
Accuracy	87.8	76.4

(b) MMG-IDX: mmg with regular patterns and partial indexing

$\sharp pos = 40$

	Index	Symbols		Index	Symbols
Left	0	DHPR	Right	3	ADFMNRTVY
	1	EKNQT		4	GHKLSW
	2	ACFGILMSVWY		5	CEIPQ

Patterns (left, right)	$p\%$	$n\%$
1222102112012,3443333434345334	0.3	100
2220122022102,33345443344	0.3	100
2222221222221,4334	0.5	100
1122221222221,35333	0.3	100
0222222221222,4*	0.5	100
*222*222221*2,3*3*	5.4	99.2
*22*2*222,*3*	73.0	75.7
Accuracy	79.5	75.0

(c) MMG-RAW: mmg with regular patterns over raw data

$\sharp pos = 40$

Patterns	$p\%$	$n\%$
WNPILLDTSSFSFQKHVSGVFLQVRN	0.3	100
AASTMAISSTAMAGTPIKVGSFGEGRIT	0.3	100
LRNTFTRAGGLSRITSVR	0.3	100
*L*L*AR*	5.7	96.4
*VL*L*A*	28.4	92.1
*LF*L*A*	26.8	94.2
*L*LA*A*	45.4	89.2
Accuracy	71.4	79.2

Figure 6: Best hypotheses obtained by three learning algorithms for signal peptides (Plant)

that the quality of hypotheses produced by the algorithm does not heavily depends on the size of a training set.

Attribute noise is to randomly replace letters in sequences with other letters. For attribute noise, a sort pattern can absorb the random replacement of letters by including such a wrong letter into a sort symbol. Thus, we expect that sort patterns work well with noisy examples. Theoretical justification of this observation should be required.

Brazma *et al.* [6] proposed the algorithm to find a PROSITE pattern from positive examples of amino acid sequences. Since the class of sort patterns we introduced is a subclass of PROSITE patterns, which is commonly used to describe protein motifs, it is our future work to extend our method for PROSITE patterns.

References

[1] Angluin, D., Finding patterns common to a set of strings. In Proc. the 11th Annual Symp. on Theory of Computing, 130–141 (1979).

[2] Arimura, H., Shinohara, T., Otsuki, S., Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. In Proc. the 11th STACS, LNCS 775, Springer-Verlag, 649–660 (1994).

[3] Arimura, H., Fujino, R, Shinohara, T., Arikawa, S., Protein motif discovery from positive examples by minimal multiple generalization over regular patterns. In Proc. Genome Informatics Workshop 1994, Universal Academy Press, Tokyo, 39–48 (1994).

[4] Bairoch, A., Prosite: a dictionary of sites and patterns in proteins. Nucleic Acids Research, 20:2013–2018 (1992).

[5] Brazma, A., Jonassen, I., Eidhammer, I., Gilbert, D., Approaches to the automatic discovery of patterns in biosequences. Reports in Informatics, University of Bergen, No. 113, Dec. (1995).

[6] Brazma, A., Jonassen, I., Ukkonen, E., Vilo, J., Discovering patterns and subfamilies in biosequences. In Proc. Fourth Int. Computational on Intelligent System for Molecular Biology AAAI Press and MIT Press, 34–43. (1996).

[7] GenBank. GenBank release notes. (1992).

[8] Shimozono, S., Shinohara, A., Shinohara, T., Miyano, S., Kuhara, S., Arikawa, S., Finding alphabet indexing for decision trees over regular patterns: an approach to bioinformatical knowledge acquisition. Proc. 26th Hawaii International Conference on System Science, IEEE Computer Society press, 763–772. (1993).

[9] Wu,TD., Brutlag, DL., Discovering empirically conserved amino acid substitution groups in databases of protein families. In Proc. ISMB-96. (1996).

[10] Yamaguchi, M., Shimozono, S., Shinohara, T., Finding minimal multiple generalization over regular patterns with alphabet indexing. In Proc. Genome Informatics Workshop 1996, Universal Academy Press, Tokyo, 51–60 (1996).