Finding Minimal Multiple Generalization over Regular Patterns with Alphabet Indexing

Michiyo Yamaguchi [1]
michiyo@donald.ai.kyutech.ac.jp

Shinichi Shimozono [2]
sin@ces.kyutech.ac.jp

Takeshi Shinohara [1]
shino@donald.ai.kyutech.ac.jp

[1] Department of Artificial Intelligence
[2] Department of Control Engineering and Science
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820, Japan

Abstract

We propose a learning algorithm that discovers a motif represented by patterns and an alphabet indexing from biosequences. From only positive examples with the help of an alphabet indexing, the algorithm finds k regular patterns as a k-minimal multiple generalization (k-mmg for short). The computational results for transmembrane domains indicate that the combination of k-mmg and alphabet indexing works quite successful. We also introduce a partial alphabet indexing that transforms symbols dependently on the position in sequences.

1 Introduction

Extracting a consensus motif from proteins that have common features is one of the best ways to understand proteins. To find likeness of arrangements of amino acid residues on computers, the multiple alignment technique is frequently applied to amino acid sequences [10]. This technique is quite useful when a few sequences have to be regarded. However, the time and the space needed to compute multiple alignments rapidly blow up with the growth of the number of sequences. Therefore, to deal with a large amount of sequences, it is reasonable to consider "finding the motif from sequences" as "learning a rule from examples" by assuming the motif can be found in a specific class of rules that has efficient learning algorithms.

Along with the development of databases of sequences on computers in Molecular Biology, the problems of learning motifs from sequences have attracted a lot of interest from Computer Science. Especially, the importance of algorithms that would find rules which can be easily understood by human is increasing. In this research direction, learning algorithms that use negative examples as well as positive examples for specifying rules have been studied, and are known to be very effective to find accurate rules (e.g.[2]). They also serve new viewpoints such as the discovery of a negative motif from negative examples rather than the rules explaining positive examples. However, it is more often required to find a motif directly from positive examples. Furthermore, negative examples may not be available in some features since experiments for obtaining positive examples are rarely intended to identify or specify negative examples; In other words, it is usually not true that non-examples are negative examples.

Algorithms for learning from only positive examples are developed to cope with these situations. In the criteria of learning from positive examples, the main difficulty is in the requirement that the ideal rule must explain all the positive examples and at the same time should reject most of unknown negative examples. This requirement appears as a kind of "tightness" of rules, which eliminates the suspicion that the rule explains negative examples as well as positive examples.

As applications in bioinformatical knowledge acquisitions, Arimura et al. [3] dealt with the problem of learning from only positive examples by k-minimal multiple generalization (k-mmg, for short), which expresses a motif in a disjunction of regular patterns and is guaranteed to explain a minimal set of strings. Brāzma et al. [5] adopted the minimum description length principle as the tightness of rules, and developed the algorithm that finds a PROSITE pattern from positive examples. They proved that their algorithm finds a pattern whose description length is reasonably small and thus is considered having less redundancy.

In this paper, we propose a method to find rules and exceptions from sequences as regular patterns and an alphabet indexing of amino acid residues. This combines the learning algorithm that finds a k-mmg, and the local search algorithm to find an alphabet indexing, which is basically analogous to that developed in [13]. The algorithm for k-mmg adopts the several heuristics, one of which makes the algorithm to find not only a rule explaining examples but also exceptions improving the rule. Furthermore, we introduce a partial alphabet indexing whose translation depends on the position of symbols. We apply this new technique to the problem of identifying signal regions of signal peptides.

2 Minimal Multiple Generalization

In this section, we introduce pattern languages [14] and minimal multiple generalizations [3].

For a set A, we denote by $\sharp A$ the number of elements in A. Let $\Sigma = \{\mathtt{A}, \mathtt{B}, \ldots\}$ be a finite set of *constant symbols*. Then, Σ^* denotes the set of all the finite strings over Σ including the *empty string* ε, and Σ^+ denotes the set of all the nonempty strings.

A *pattern* is a string in $(\Sigma \cup \{*\})^*$, where $*$ is the *gap symbol*. A *substitution* θ for a pattern p is a set of replacements for gap symbols in p with patterns. The *language* $L(p)$ of a pattern p is the subset of Σ^* obtained by replacing gap symbols in p with strings in Σ^*. Note that patterns and their languages in this paper refer to regular patterns and extended regular pattern languages in literature (e.g.[14]). For a finite set P of patterns, $L(P)$ denotes the *union*

$\bigcup_{p \in P} L(p)$ of languages.

A pattern p is *canonical* if p contains no consecutive occurrences of gap symbols. Without loss of generality, we assume that every pattern is canonical. For different patterns p and q, we say that p is *more specific than* q if there exists a substitution θ such that $p = \theta(q)$. Also, for sets P and Q of patterns, we say that P is more specific than Q, if and only if for any $p \in P$ there is $q \in Q$ such that p is more specific than q. A set P of patterns is *reduced* if no pattern $p \in P$ is more specific than other pattern $q \in P$. Let D^k be the class of reduced sets of at most k patterns. Then a set $P \in D^k$ is a *k-minimal multiple generalization* (k-mmg for short) of $S \subseteq \Sigma^*$ if $S \subseteq L(P)$ and no $Q \in D^k$ with $S \subseteq L(Q)$ is strictly more specific than P.

Let p be a pattern. A substitution θ is *basic* if θ is either (i) $\theta = \{* := *a*\}$, where $a \in \Sigma$, or (ii) $\theta = \{* := \varepsilon\}$. We define the set of *refinements* of p by $\{\theta(p) \mid \theta \text{ is basic}\}$ and denote $\rho(p)$. Note that, for any $p' \in \rho(p)$, p' is more specific than p. A pattern p is *refined with respect to* $S \subseteq \Sigma^*$ if there is no refinement $r \in \rho(p)$ such that $S \subseteq L(r)$. Let p be a pattern satisfying $S \subseteq L(p)$, and let $k > 1$ be a positive integer. A *k-division of p with respect to* S is a set $P \subseteq \rho(p)$ of at most k patterns such that P is reduced and $S \subseteq L(P)$. Also, we say that p is *k-divisible with respect to* S if there exists a k-division of p with respect to S. Note that k-divisibility implies k'-divisibility for $k \leq k'$, but not for $k \geq k'$ in general.

Now we describe the algorithm $\mathbf{MMG}(k, S)$ (Fig. 1) that finds a k-mmg for given $k > 1$ and $S \subseteq \Sigma^*$. Given a set S of positive examples, the algorithm searches for a k-mmg in D^k: It starts from the most general generalization $\{*\}$, then iterates refining and dividing a pattern in the current mmg, while the number of patterns does not exceed k. In the algorithm, the operator ρ is repeatedly applied to a pattern to produce a set of more specific patterns.

For example, let S be the set consisting of the following sequences:

$e_1 =$	IATGMVGALLLLLVVALGIGLFI	$e_5 =$	LVIGTIAVLIGIVNLGL
$e_2 =$	FIIATVEGVLLFLILVVVVGILI	$e_6 =$	LYLGVVLSAVVIITGCF
$e_3 =$	QSYMIVLMVTCCITPLSIIVLCYL	$e_7 =$	YHLTSVWMIFVVTASVFTNGLVLA
$e_4 =$	IAILLLTVVTLATSVASLVYSMGASTPS		

Then the algorithm $\mathbf{MMG}(k, S)$ for $k = 3$ finds, for example, the followng 3-mmg:

$$P = \left\{ \begin{array}{ll} p_1 = & *\mathtt{I}*\mathtt{T}*\mathtt{V}*\mathtt{L}*\mathtt{V}*\mathtt{L}* \\ p_2 = & \mathtt{LYLGVVLSAVVIITGCF} \\ nnnp_3 = & \mathtt{QSYMIVLMVTCCITPLSIIVLCYL} \end{array} \right\}$$

Here the languages $L(p_1)$, $L(p_2)$, and $L(p_3)$ cover subsets $\{e_1, e_2, e_4, e_5, e_7\}$, $\{e_6\}$, and $\{e_3\}$ of S, respectively. Two patterns p_2 and p_3 in P contain no gap symbols. Such a pattern explains only oneself, and is called *exception*.

Although the algorithm $\mathbf{MMG}(k, S)$ is guaranteed to find a k-mmg for any input [4], there is unspecified choice in the following points:

(*1) A k-divisible pattern p in P.

(*2) A k-division ΔP from $\rho(p)$.

(*3) A pattern p in the current multiple generalization which should be refined.

(*4) A refinement r of p from possible refinements in $\rho(p)$.

In order to find an appropriate k-mmg for representing a motif of sequences, we try the following three heuristic strategies introduced in [3] and [15]:

• *Randomized* (Rand): Choose patterns at random.

• *Maximal covering* (Max): Choose a pattern covering the maximum number of positive examples to obtain a k-mmg with less exceptions.

• *Minimal covering* (Min): Choose a pattern covering the minimum number of positive examples to obtain a k-mmg containing many exceptions.

In Max and Min strategies, the choices of patterns in $(*1)$, $(*3)$ and $(*4)$ are clear. However, in $(*2)$, the algorithm must choose an appropriate subset of $\rho(p)$ for p. Let R be the set of subsets of $\rho(p)$ that are reduced with respect to a set $S' \subseteq L(p)$ and consist of at most k patterns. In our implementation, we choose the smallest set from R. If there are two or more smallest subsets, then we first compute the sequences of positive integers for subsets in R as follows. Let $P' = \{p_1, \ldots, p_m\}$ be a set in R, and let $n(p_i)$ be the number of examples in S' covered by $p_i \in P'$. In Max strategy, the sequence $seq(P')$ is defined as $\langle n(p_{i(1)}), n(p_{i(2)}), \ldots, n(p_{i(m)})\rangle$, where $i(1), \ldots, i(m)$ are the indices of patterns in P' sorted as giving the decreasing sequence of integers $n(p_{i(1)}) \geq n(p_{i(2)}) \geq \cdots \geq n(p_{i(m)})$. In Min strategy, the sequence is similary defined by the indices giving an increasing order. Then, we choose a set of patterns that gives the lexicographically largest sequence in Max strategy, and choose a set that gives the lexicographically smallest sequence in Min strategy. For example, if there are two smallest sets $P_1 = \{p_1, p_2\}$ and $P_2 = \{p_1, p_3\}$ in R with $n(p_1) = 4$, $n(p_2) = 5$ and $n(p_3) = 1$, then in Min strategy, we choose P_2 since $seq(P_2) = \langle 1, 5\rangle$ is smaller than $seq(P_1) = \langle 4, 5\rangle$.

MMG(k, S)
  $P :=$ **Refinement**$(\{*\}, S)$;
  $\Delta k := k$;
  while $\Delta k \geq 2$ **do:**
    if P contains Δk-divisible patterns **then**
      Choose such a pattern$^{(*1)}$ $p \in P$, and let
      $\Delta P :=$ **Divide**$(p, k, S - L(P - \{p\}))$;
    else "there is no further Δk-divisions"
      quit while;
    $\Delta P :=$ **Refinement**$(\Delta P, S - L(P - \{p\}))$;
    $P := (P - \{p\}) \cup \Delta P$;
    $\Delta k := \Delta k - \sharp \Delta P + 1$;
  end while;
  Output P.

Divide(p, k, S')
  Choose a set$^{(*2)}$ $P' \subseteq \rho(p)$ such that
  $\sharp P' \leq k$, $S' \subseteq L(P')$ and P' is reduced;
  Return P';

Refinement(P', S')
  while some patterns in P' can be refined
  with S' **do:**
    Choose such a pattern$^{(*3)}$ $p \in P'$;
    Select a refinement$^{(*4)}$ $r \in \rho(p)$ such that
    $S' - L(P' - \{r\})$;
    $P' := (P' - \{p\}) \cup \{r\}$;
  end while;
  Return P';

Figure 1: Algorithm for finding k-mmg.

3 Local Search for Alphabet Indexing

This section gives a notions of alphabet indexing which will be combined with k-mmg together, and describes a local search algorithm to find an approximate alphabet indexing.

Let I be a finite alphabet, and let f be a mapping form Σ to I. By $\tilde{f}$ we denote the homomorphism $\tilde{f}(s) = f(s_1) \cdots f(s_n)$ for $s \in \Sigma^*$ and $\tilde{f}(S) = \{\tilde{f}(s) | s \in S\}$ for $S \subseteq \Sigma^*$.

Given disjoint sets P and N of strings over Σ and an alphabet I with $\sharp I < \sharp\Sigma$, an *alphabet indexing* f of Σ by I with respect to P and N is a mapping $f : \Sigma \to I$ that maximizes the product $\sharp P[\tilde{f}] \times \sharp N[\tilde{f}]$ of the sizes of the subsets $P[\tilde{f}] = \{p \in P | \tilde{f}(p) \notin \tilde{f}(N)\}$ and $N[\tilde{f}] = \{q \in N | \tilde{f}(q) \notin \tilde{f}(P)\}$.

The definition of an alphabet indexing intends to retain most of positive examples and negative examples to be consistent after the translation by the indexing. However, it is known in [12] that the problem of finding an indexing that perfectly separates the transformed sets $\tilde{f}(P)$ and $\tilde{f}(N)$ is NP-complete. Furthermore, this maximization problem has turned out to be hard to approximate in polynomial-time [11]. Even though, in some actual applications, it is known that heuristic algorithms such as a local search algorithm in [13] can find an appropriate alphabet indexing in a reasonable amount of time. In this paper, as in the system presented in [13], we employ a local search algorithm for finding a near-optimal indexing together with an accurate k-mmg.

A local search algorithm **Find_Indexing** takes small sets `pos` and `neg` of positive and negative examples. A neighbor of an indexing f for Σ is any indexing f' for Σ in which for only one symbol $a \in \Sigma$ the index $f'(a)$ differs from $f(a)$. The goodness measure of an indexing f with the k-mmg P, obtained as the output of $\mathbf{MMG}(k, \tilde{f}(\mathtt{pos}))$, is $\sharp\{q \in \mathtt{neg} | \tilde{f}(q) \notin L(P)\}$. We consider $I^{|\Sigma|}$ as the set of all indexing of Σ by I, by regarding a string $f(a_1) \cdots f(a_n)$ for $\Sigma = \{a_1, \cdots, a_n\}$ as an indexing $f : \Sigma \to I$. The algorithm **Find_Indexing** begins with an initial indexing $f \in I^{|\Sigma|}$. Then, the algorithm iterates the following operations, while a value of $Score(f)$ can be improved. For the current indexing f, the algorithm computes the measure of every neighbor indexing f'. If there is a neighbor indexing f' such that $Score(f') > Score(f)$, then replace f with f'.

Find_Indexing(`pos`, `neg`)
  Select an indexing f randomly from $I^{|\Sigma|}$;
  repeat
    Find a neighboring indexing f' whose measure is strictly better than that of f ;
    if there is no better neighbor indexing, **then return** f;
    Let $f := f'$;
  forever;

Figure 2: Algorithm **Find_Indexing**

4 Computational Results

We collected the examples as in Table 1, for the following two identification problems.

- Transmembrane domains: From PIR database [9], we collected subsequences of proteins that are specified as transmembrane domains as positive examples: as negative examples, we randomly took subsequences of length 30 that have no overlaps with transmembrane domains.
- Signal peptides: From GenBank database [6], we collected the signal peptides according to the indications as positive examples, and the initial segments of proteins of length 30 that have no signal peptides as negative examples.

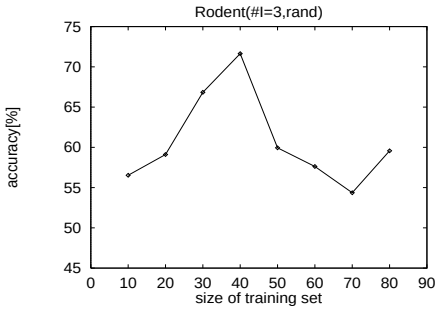

Figure 3: Signal peptides: The result of test runs.

In each run of the algorithm, sets `pos` and `neg` are chosen randomly from `POS` and `NEG`, respectively. All positive examples in `POS` except examples in `pos`, and all negative examples in `NEG` are used in the final evaluation of found hypotheses. The size of `POS`, from which a k-mmg is computed, and the number k of patterns are important unspecified parameters. In the experiments, we supposed that the rules for these two problems can be represented by a few patterns, of length at most 6, with the help of alphabet indexing. We tried various sizes for `pos` for the identification of transmembrane domains, from 10 to 80. We chose $\sharp$`pos`$= 40$ for signal peptide, since we obtained the best result with that value in test runs.

Table 1: The number of provided examples

Data	Sequences	`POS`	`NEG`	`pos`	`neg`
Transmembrane domains		689	1000	10 ~ 80	10 ~ 80
Signal Peptides	Bacterial	495	7330	40	7000
	Plant	370	3074	40	3000
	Primate	1032	3612	40	3000
	Rodent	1018	3158	40	3000

4.1 Transmembrane Domains

In Fig. 4, the best outputs for the normal execution (1) and for finding a negative motif (2) are presented.

As we can see, the obtained alphabet indexing is approximating the hydropathy index. It is known in [7] that transmembrane domains of proteins are well identified by the hydropathy index. It must be noted that the accuracy of the obtained motif is better than or competitive to the result directly obtained by the hydropathy index in [15].

Table 2: Hydropathy index

A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
1.8	2.5	-3.5	-3.5	2.8	-0.4	-3.2	4.5	-3.9	3.8	1.9	-3.5	-1.6	-3.5	-4.5	-0.8	-0.7	4.2	-0.9	-1.3

(1) Use positive examples as POS
$\sharp pos = \sharp neg = 50, k = 5$
Strategy: Max.

indexing	symbols
0	ACFILMSVWY
1	DHKR
2	EGNPQT

patterns
*10000*1000*
*000000*000*
0*000*00000*0
0000*0*01000*
$accuracy$: 81.1%(89.9%, 77.9%)

(2) Use negative examples as POS
$\sharp pos = \sharp neg = 70, k = 5$
Strategy : Min.

indexing	symbols
0	AFGIMQSTVW
1	DEHKNPRY
2	CL

patterns
120112011111100102222011121102
011221121120222222220222222220
2*21*021*021*
1*21*021*21*
*21*221*21*
$accuracy$: 86.7%(89.9%, 83.7%)

Figure 4: Transmembrane Domains: Accuracy $m\%$ ($p\%$, $n\%$) is measured by all examples POS and NEG, where m is the geometric mean p and n, for positive and negative examples, respectively.

4.2 Signal Peptides

Assume that sequences showing a common feature are divided at some fixed place into the left part and the right part that are governed by different properties of amino acid residues. We introduce the following method to improve accuracy for coping with this situation. Firstly, sequences are divided at some fixed division spot, and then the left and the right part are transformed by two independent indexings. For example, KLFIFTCLLAVALA will be divided into KLFIF and TCLLAVALA if the division spot is 5. The learning algorithm is modified to find a k-mmg for amino acid sequences whose the left and the right parts are differently transformed. We call this method *partial alphabet indexing*.

The experimental results obtained for signal peptides using partial alphabet indexing is shown in Fig. 5 and 6. The division spot = 0 is the case that sequences are not divided. We can observe that the accuracy of obtained motif is slightly better than that for division spot = 0. However, it is not so good as those obtained in [1] and [4].

5 Discussion

In this paper, we presented a learning algorithm that combines a k-mmg and an alphabet indexing as a hypothesis, and we confirmed that this approach is effective for extracting motifs from positive examples.

For transmembrane domains, our technique is powerful enough to find an appropriate alphabet indexing and an accurate k-mmg. The result is competitive even with the learning algorithm in [13] that extracts patterns from both positive and negative examples.

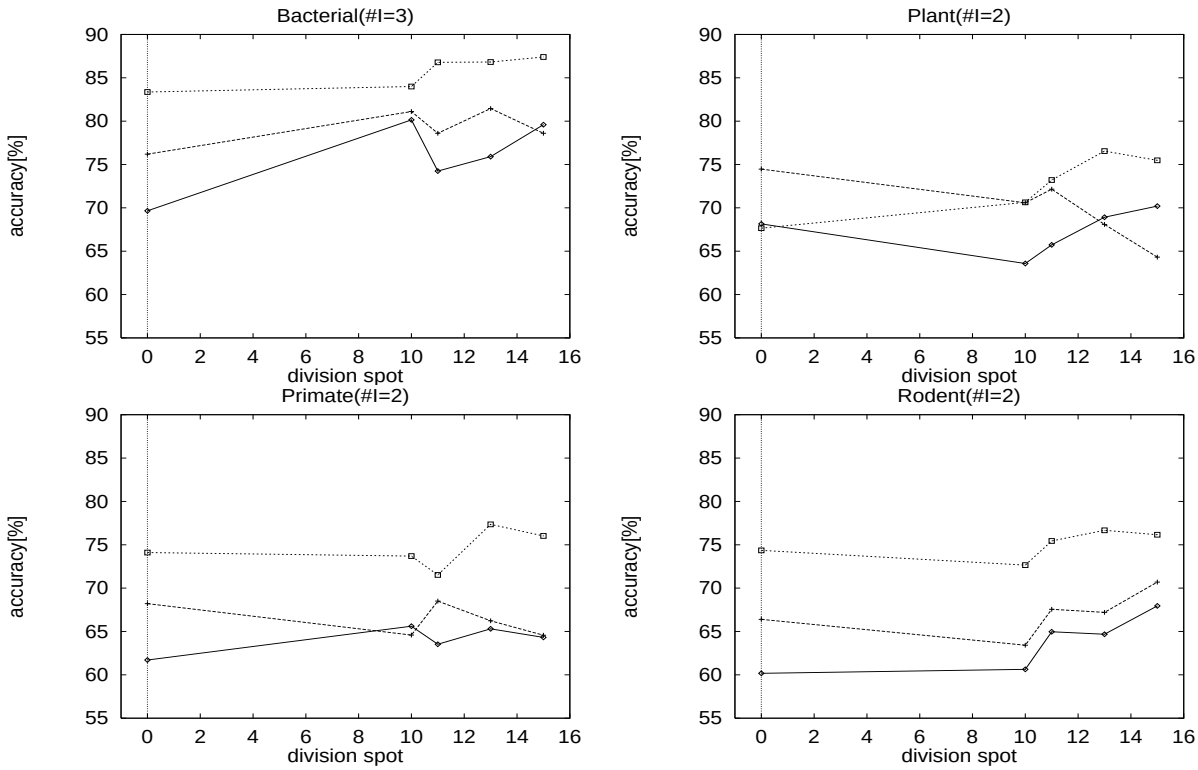

Figure 5: Signal peptides: Accuracies of the best outputs in a few trials vs places of the division spot for four categories of signal peptides. Almost always Min strategy, plotted by squares with dotted lines, achieved the best accuracy: Rand and Max strategy are shown by diamonds with solid lines and crosses with dashed lines, respectively.

For signal peptides, we assumed that the left and the right parts of amino acid sequences are governed by different properties of amino acid residues. In the result, we observed that the obtained motif achieves slightly higher accuracy than that of motif found for examples transformed uniformly. Also, it is observed that obtained partial alphabet indexings for the left parts are not similar to that of the right parts. However, the improvement of the accuracy of obtained motif is less than we expected. Off course, there remains possibility that this technique improves the accuracy of motif if it is combined with other method, such as decision trees over patterns in [13].

The k-mmg finds a motif from only positive examples. Even though, negative examples are indispensable for measuring the goodness of alphabet indexing. If we are not allowed to use negative examples, then we need another definition of the measure of alphabet indexings in our algorithm. On the other hand, if we can use negative examples to find a k-mmg, we may have some other strategies to deal with the choice in the algorithm, such as that in [3]. These issues should be considered in future works.

(1) *division spot* = 0

Indexing	Bacterial	Plant	Primate	Rodent
0	ACFILMV	ACFGILMRSTVY	ACFGLMSVWY	ACFGILMRSTVWY
1	GPSTWY	DEHKNPQW	DEHIKNPQRT	DEHKNPQ
2	DEHKNR			
Strategy	Min	Max	Min	Min

(2) *division spot* = 10 ~ 15

(a) Bacterial
division spot = 15.
Strategy: Min.

Left	Right
DEHMQTWY	CDEFHIKMNPSW
ACFILSV	AGQ
GKNPR	LRTVY

(b) Plant
division spot = 13.
Strategy: Min.

Left	Right
ACFGHILMRSTV	DEFHIKLMNPQTVW
DEKNPQWY	ACGRSY

(c) Primate
division spot = 15.
Strategy: Min.

Left	Right
ACFGHIKLPQVW	CDEFHIKLMNPQRVWY
DEMNRSTY	AGST

(d) Rodent
division spot = 13.
Strategy: Min.

Left	Right
ACFGHILMNRSVW	DEFHIKLMNPQRT
DEKPQTY	ACGSVWY

Figure 6: Signal Peptides: The partial alphabet indexing achieved the highest accuracy in the case of (1) division spot = 0, and (2) division spot = 10 ~ 15. The amino acid symbols categorized in the same index are shown in the same row.

References

[1] S. Arikawa, S. Kuhara, S. Miyano, A. Shinohara and T. Shinohara, A learning algorithm for elementary formal systems and its experiments on identification of transmembrane domains, In *Proc. the 25th Hawaii International Conference on System Sciences*, pp. 675–684, 1992.

[2] S. Arikawa, S. Miyano, A. Shinohara, S. Kuhara, Y. Mukouchi and T. Shinohara, A machine discovery from amino acid sequences by decision trees over regular patterns, *New Generation Computing*, 11, pp. 361–375, 1993.

[3] H. Arimura, R. Fujino, T. Shinohara and S. Arikawa, Protein motif discovery from positive examples by minimal multiple generalization over Regular Patterns, In *Proc. Genome Informatics Workshop*, pp. 39–48, 1994.

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

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

[6] GenBank, GenBank Release Notes, IntelliGenetics Inc., 1991.

[7] J. Kyte and R.F. Doolittle, A simple method for displaying the hydropathic character of protein, *J. Mol. Biol.*, 157, pp. 105–132, 1982.

[8] R. D. King, A. Srinivasan, S. Muggleton, C. Feng, R. A. Lewis and M. J. E. Sternberg, Drug design using inductive logic programming, In *Proc. the 26th Hawaii International Conference on System Sciences*, pp. 646–655, 1993.

[9] PIR, Protein identification resource, National Biomedical Research Foundation, 1991.

[10] D. Sankoff, *Time Warp, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, 1983.

[11] S. Shimozono. Unpublished manuscript. *Pressed in LA'96 summer workshop*, 1996.

[12] S. Shimozono and S. Miyano, Complexity of finding alphabet indexing, *IEICE Trans. Inf. Sys.*, E78-D, pp. 13–18, 1995.

[13] S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara and S. Arikawa, Knowledge acquisition from amino acid sequences by machine learning system BONSAI, *Trans. Inf. Proc. Soc. Japan*, vol. 35, pp. 2009–2018, 1994.

[14] T. Shinohara, Polynomial time inference of extended regular pattern languages, In *RIMS Symposia on Software Science and Engineering*, LNCS 147, pp. 115–127, Springer-Verlag, 1982.

[15] M. Yamaguchi, T. Shinohara, R. Fujino, H. Arimura, S. Arikawa, Protein motif discovery from amino acid sequence by set of regular patterns, *Technical Report of Information Processing Society of Japan 95-FI-38* pp. 33–40, 1995.