PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT. © ICOT, 1992

A Machine Discovery from Amino Acid Sequences by Decision Trees over Regular Patterns

Setsuo Arikawa† Satoru Kuhara‡ Satoru Miyano† Yasuhito Mukouchi†† Ayumi Shinohara† Takeshi Shinohara‡‡

† Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.
‡ Graduate School of Genetic Resources Technology, Kyushu University 46, Fukuoka 812, Japan.
†† Department of Information Systems, Kyushu University 39, Kasuga 816, Japan.
‡‡ Department of Artificial Intelligence, Kyushu Institute of Technology, Iizuka 820, Japan.

Abstract

This paper describes a machine learning system that discovered a "negative motif", in transmembrane domain identification from amino acid sequences, and reports its experiments on protein data using PIR database. We introduce a decision tree whose nodes are labeled with regular patterns. As a hypothesis, the system produces such a decision tree for a small number of randomly chosen positive and negative examples from PIR. Experiments show that our system finds reasonable hypotheses very successfully. As a theoretical foundation, we show that the class of languages defined by decision trees of depth at most d over k-variable regular patterns is polynomial-time learnable in the sense of probably approximately correct (PAC) learning for any fixed $d, k \geq 0$.

1 Introduction

Hydrophobic transmembrane domains can be identified by a very simple decision tree over regular patterns. This result was discovered by the machine learning system we developed. The system takes some training sequences of positive and negative examples, and produces a hypothesis explaining them. When a small number of positive and negative examples of transmembrane domains were given as input, our system found a small decision tree over regular patterns as a hypothesis. Although the hypothesis is made from just 10 positive and 10 negative examples, it can explain all data in PIR database [PIR] with high accuracy more than 90%. The hypothesis exhibits that "two consecutive polar amino acids" (Arg, Lys, His, Asp, Glu, Gln, Asn) are *not* included in the transmembrane domains. This indicates that significant motifs are not in the inside of the transmembrane domains but in the *outside*. We call such motifs "negative motifs."

Email addresses:
arikawa@rifis.sci.kyushu-u.ac.jp
kuhara@grt.kyushu-u.ac.jp
miyano@rifis.sci.kyushu-u.ac.jp
mukouchi@rifis.sci.kyushu-u.ac.jp
ayumi@rifis.sci.kyushu-u.ac.jp
shino@donald.ai.kyutech.ac.jp

This paper describes a machine learning system together with a background theory that discovered such negative motifs, and reports its experiments on knowledge acquisition from amino acid sequences that reveal the importance of negative data. Traditional approaches to motif-searching are to find subsequences common to functional domains by various alignment techniques. Hence the eyes are focused only on positive examples, and negative examples are mostly ignored. Our approach by decision trees over regular patterns provides new direction and method for discovering motifs.

A regular pattern [Shinohara 1982, Shinohara 1983] is an expression $w_0x_1w_1x_2\cdots x_nw_n$ that defines the sequences containing $w_0, w_1, ..., w_n$ in this order, where each w_i is a sequence of symbols and x_j varies over arbitrary sequences. Regular patterns have been used to describe some features of amino acid sequences in PROSITE database [Bairoch 1991] and DNA sequences [Arikawa *et al.* 1992, Gusev and Chuzhanova 1990]. Our view to these sequences is through such regular patterns. A decision tree over regular patterns is a tree which describes a decision procedure for determining the class of a given sequence. Each node is labeled with either a class name (1 or 0) or a regular pattern. At a node with a regular pattern, the decision tree tests if the sequence matches the pattern or not. Starting from the root toward a leaf, the decision procedure makes a test at each node and goes down by choosing the left or right branch according to the test result. The reached leaf answers the class name of the sequence. Such decision trees are produced as hypotheses by our machine learning system. Since the system searches a decision tree of smaller size, regular patterns on the resulting decision tree exhibit motifs which play a significant role in classification. Hence, compared with neural network approaches [Holly and Karplus 1989, Wu *et al.*], our system shows important motifs in a hypothesis more explicitly.

We employ the idea of ID3 algorithm [Quinlan 1986, Utgoff 1989] for constructing a decision tree since it is sufficiently fast and experiments show that small enough trees are usually obtained. We also devise a new method for constructing a decision tree over regular patterns using another evaluation function. Given two sets of positive and negative examples, our machine learning system finds appropriate regular patterns as node attributes dynamically during the construction of the decision tree. Hence, unlike ID3, we need not assume any concrete knowledge about attributes and can avoid struggles from defining the attributes of a decision tree beforehand. Our system makes a decision tree just from a small number of training sequences, which we also guarantee with the PAC learning theory [Valiant 1984] in some sense. Therefore it may cope with a diversity of classification problems for proteins and DNA sequences.

We made an experiment on raw sequences from twenty symbols of amino acid residues. The system discovered a small decision tree just from 20 sequences with more than 85% accuracy that show if a sequence contains neither E nor D (both are polar amino acids) then it is very likely to be a transmembrane domain.

A hydropathy plot [Engelman *et al.* 1986, Kyte and Doolittle 1982, Rao and Argos 1986] has been used generally to predict transmembrane domains from primary sequences. With this knowledge, we first transform twenty amino acids to three categories (*, +, -) according to the hydropathy index of Kyte and Doolittle [1982]. From randomly chosen 10 positive and 10 negative training examples, our system has successfully produced some small decision trees over regular patterns which are shown to achieve very high accuracy. The regular patterns appearing in these decision trees indicate that two consecutive polar amino acid residues are important negative motifs for transmembrane domains. From the view point of Artificial Intelligence, it is quite interesting that the polar amino acid residues D and E were found by our machine learning system without any knowledge on the hydropathy index.

After knowing the importance of negative motifs, we examined decision trees with a single node with regular patterns x_1-x_2-$\cdots$-x_n for $n \geq 3$. The best is the pattern x_1-x_2-x_3-x_4-x_5-x_6 that gives the sequences containing at least five polar amino acids. The result is very acceptable. The accuracy is 95.4% for positive and 95.0% for negative examples although it has been believed to be difficult to define transmembrane domains as a simple expression when the view point was focussed on positive examples.

2 Decision Trees over Regular Patterns

Let Σ be a finite alphabet and $X = \{x, y, z, x_1, x_2, \cdots\}$ be a set of variables. We assume that Σ and X are disjoint. A *pattern* is an element of $(\Sigma \cup X)^+$, the set of all nonempty strings over $\Sigma \cup X$. For a pattern π, the language $L(\pi)$ is the set of strings obtained by substituting each variable in π for a string in Σ^*. We say that a pattern π is *regular* if each variable occurs at most once in π. For example, $xaybza$ is a regular pattern, but xx is not. Obviously, regular patterns define regular languages, but not vice versa. In this paper we consider only regular patterns. A regular pattern containing at most k variables is called a *k-variable regular pattern.*

A *decision tree over regular patterns* is a binary tree such that the leaves are labeled with 0 or 1 and each internal node is labeled with a regular pattern (see Figure 1). For an internal node v, we denote the left and right children of v by left(v) and right(v), respectively. We denote by $\pi(v)$ the regular pattern assigned to the internal node v. For a leaf u, value(u) denotes the value 0 or 1 assigned to u. The depth of a tree T, denoted by depth(T), is the length of the longest path from the root to a leaf.

For a decision tree T over regular patterns, we define a function $f_T : \Sigma^* \rightarrow \{0, 1\}$ as follows. For a string w in Σ^*, we determine a path from the root to a leaf and define the value $f_T(w)$ by the following algorithm:

```
begin /* Input: $w \in \Sigma^*$ */
    $v \leftarrow$ root;
    while $v$ is not a leaf do
        if $w \in L(\pi(v))$ then $v \leftarrow$right($v$)
                else $v \leftarrow$left($v$);
    $f_T(w) \leftarrow$ value($v$)
end
```

For a decision tree T over regular patterns, we define $L(T) = \{w \in \Sigma^* \mid f_T(w) = 1\}$. It is easy to see that $L(T)$ is also a regular language. But the converse is not true. Let $L = \{a^{2n} \mid n \geq 1\}$. It is straightforward to show that there is no decision tree T over regular patterns with $L = L(T)$. The same holds for the language $\{a^{2n}b \mid n \geq 1\}$.

3 Constructing Decision Trees

This section gives two kinds of algorithms for constructing decision trees over regular patterns that are used in our machine learning system.

The first algorithm employs the idea of ID3 algorithm [Quinlan 1986] in the construction of decision trees. The ID3 algorithm assumes data together with explicit attributes in advance. On the other hand, our approach assumes a space of regular patterns which are simply generated by given positive and negative examples. No

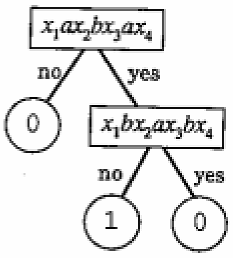

Figure 1: Decision tree over regular patterns defining a language $\{a^m b^n a^l \mid m, n, l \geq 1\}$ over $\Sigma = \{a, b\}$

extra knowledge about data is required. Although the space may be large and contain meaningless attributes, our algorithm finds appropriate regular patterns from this space dynamically during the construction of a decision tree in a feasible amount of time. This is a point which is very suited for our empirical research.

Let P and N be finite sets of strings with $P \cap N = \emptyset$. Using P and N, we deal with regular patterns of the form $w_0x_1w_1x_2 \cdots x_kw_k$ such that $w_0, ..., w_k$ are substrings of some strings in $P \cup N$. Let $\Pi(P, N)$ be some family of such regular patterns made from P and N. The family $\Pi(P, N)$ is appropriately given and used as a space of attributes.

For a regular pattern $\pi \in \Pi(P, N)$, the cost $E(\pi, P, N)$ is the one defined in [Quinlan 1986] by

$$E(\pi, P, N) = \frac{p_1 + n_1}{|P| + |N|} I(p_1, n_1) + \frac{p_0 + n_0}{|P| + |N|} I(p_0, n_0),$$

where p_1 (resp. n_1) is the number of positive examples in P (resp. negative examples in N) that match π, i.e., $p_1 = |P \cap L(\pi)|$, $n_1 = |N \cap L(\pi)|$, and p_0 (resp. n_0) is the number of positive examples in P (resp. negative examples in N) that do not match π, i.e., $p_0 = |P \cap \overline{L(\pi)}|$, $n_0 = |N \cap \overline{L(\pi)}|$, $\overline{L(\pi)} = \Sigma^* - L(\pi)$, and

$$I(x, y) = \begin{cases} 0 & \text{(if } x = 0 \text{ or } y = 0\text{)} \\ -\frac{x}{x+y} \log \frac{x}{x+y} - \frac{y}{x+y} \log \frac{y}{x+y} & \text{(otherwise).} \end{cases}$$

The first algorithm DT1(P, N) (Algorithm 1) sketches our decision tree algorithm for $\Pi(P, N)$, where CREATE(π, T_0, T_1) returns a new tree with a root labeled with π whose left and right subtrees are T_0 and T_1, respectively.

The second algorithm uses a different evaluation function. For a decision tree T over regular patterns, let nodes(T) be the number of nodes in T, and $\mathcal{T}(T)$ be the set of trees constructed by replacing a leaf v of T by the tree of Fig. 2 (a) or Fig. 2 (b) for some pattern π.

The score function $Score(T, P, N)$ balances the information gains in classification and is defined as

$$Score(T, P, N) = \frac{|P \cap L(T)|}{|P|} \cdot \frac{|N \cap \overline{L(T)}|}{|N|}.$$

The second algorithm DT2$(P, N, MaxNode)$ (Algorithm 2) checks all leaves at each phase of a node generation using the evaluation function $Score(T, P, N)$.

Algorithm 2 is slower than Algorithm 1 since all leaves are checked at each phase of a node generation. However,

```
function DT1 ( $P$, $N$ : sets of strings ): node;
 begin
   if $N = \emptyset$ then
     return( CREATE("1", null, null) )
   else if $P = \emptyset$ then
     return( CREATE("0", null, null) )
   else begin
     Find a shortest pattern $\pi$ in $\Pi(P, N)$
     that minimizes $E(\pi, P, N)$;
     $P_1 \leftarrow P \cap L(\pi)$;   $P_0 \leftarrow P - P_1$;
     $N_1 \leftarrow N \cap L(\pi)$;   $N_0 \leftarrow N - N_1$;
     return(CREATE($\pi$,DT1($P_0, N_0$),DT1($P_1, N_1$)))
     end
 end
```

Algorithm 1

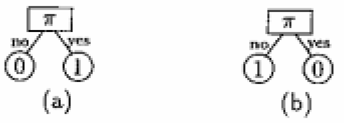

Figure 2: A leaf is replaced by (a) or (b) for some pattern π.

```
function DT2( $P, N$: sets of strings,
              $MaxNode$: int     ) : tree;
 begin
   if $N = \emptyset$ then
     return( CREATE("1", null, null) )
   else if $P = \emptyset$ then
     return( CREATE("0", null, null) )
   else begin
     $T \leftarrow$CREATE("1", null, null);
     while ( nodes($T$) < $MaxNode$
         and $Score(T, P, N) < 1$ ) do
         begin
           find $T_{max} \in \mathcal{T}(T)$
             that maximizes $Score(T_{max}, P, N)$;
           $T \leftarrow T_{max}$
         end
   end
   return ( $T$ )
 end
```

Algorithm 2

Algorithm 2 constructs decision trees which are finely tuned when the size of decision trees is large. Moreover, it is noise-tolerant, i.e., it allows conflicts between positive and negative training examples. If the size of $\Pi(P, N)$ is polynomial with respect to the size of P and N, then these algorithms run in polynomial time.

4 Transmembrane Domain Identification

The problem of transmembrane domain identification is one of the most important protein classification problems and some methods and experiments have been reported. For example, Hartman et al. [1989] proposed a method using the hydropathy index for amino acid residues in [Kyte and Doolittle 1982]. The reported success rate is about 75%. Most approaches deal with positive examples, i.e., sequences corresponding to transmembrane domains, and try to find properties common to them.

The sequence in Figure 3 is an amino acid sequence of a membrane protein. There is a tendency to assume that a membrane protein contains several transmembrane domains each of which consists of 20 $\sim$ 30 amino acid residues. Therefore, if a sequence corresponding to a transmembrane domain is found in an amino acid sequence, it is very likely that the protein is a membrane protein.

Our idea for transmembrane domain identification is to use decision trees over regular patterns for classification. Algorithm 1 and 2 introduced in Section 3 are used to find good decision trees from positive and negative training examples. In order to avoid combinatorial explosion, we restrict the space of attributes to the regular patterns of the form

$$x\alpha y,$$

where x and y are variables and α is a substring taken from given examples.

In our experiments, a *positive example* is a sequence which is already known to be a transmembrane domain. A *negative example* is a sequence of length around 30 cut out from the parts other than transmembrane domains. The length 30 is simply due to the reasonable length of a transmembrane domain. From PIR database our machine learning system chooses randomly two small sets P and N of positive and negative training examples, respectively. Then, at each trial, by using Algorithm 1 or Algorithm 2, the system tries to construct a small decision tree over regular patterns which classifies P and N exactly.

We have evaluated the performance ratio of a produced decision tree in the following way. As the total space of positive examples, we use the set POS of all transmembrane domain sequences (689 sequences) from PIR database. The total space NEG of negative examples consists of 19256 negative examples randomly chosen from *all* proteins from PIR. The success rate of a decision tree for positive examples is the percentage of the positive examples from POS recognized as positive (class 1). The success rate for negative examples is counted as the percentage of the negative examples from NEG recognized as negative (class 0).

Figure 5 (a) is one of the smallest decision trees discovered by our system just from 10 positive and 10 negative raw sequences that achieve good accuracy. The performance ratio is (84.8%, 89.6%) for all data in POS and NEG, respectively. This decision tree suggests that if a sequence of length around 30 contains neither `D` not `E` then it is very likely to be a part of transmembrane domain.

The alphabet of amino acid sequences consists of twenty symbols. It has been shown that the use of the hydropathy index for amino acids is very successful [Arikawa *et al.* 1992, Hartmann *et al.* 1989]. According to the hydropathy index of [Kyte and Doolittle 1982], we transform these twenty symbols to three symbols as shown in Table 1. This transformation reduces the size of a search space drastically small while less information is, fortunately, lost in classification.

Then by this transformation table, the sequence in Figure 3 becomes the sequence in Figure 4.

Figure 5 (b), (c) show two of the best decision trees over regular patterns that our machine learning system found from 10 positive and 10 negative training examples. The decision tree (b) recognizes 91.4% of positive examples and 94.8% of negative examples. Even the decision tree of (c) can recognize 92.6% of the positive examples and 91.6% of the negative examples. The negative motif "--" which indicates consecutive polar amino acid residues plays a key role in classification. This may have a close relation to the signal-anchor structure that consists of two parts, the hydrophobic part of a membrane-spanning sequence and the charged residues around the hydrophobic part [Lipp *et al.* 1989, Von Heijine 1988].

The decision tree (a) also shows the importance of a cluster of polar amino acids in transmembrane domain identification although our machine learning system assumed no knowledge about the hydropathy.

We examined how the performance of our machine learning system changes with respect to the number of training examples. The training examples are chosen randomly ten times in each case and a point of the graph of Figure 6 is the average of these ten results for each case. Figure 6 shows the results. We may observe the following facts:

1. The hydropathy index of Kyte and Doolittle [Kyte and Doolittle 1982] is very useful. When indexed sequences are used, the system can produce from 40 positive and 40 negative examples a decision tree with only several nodes whose accuracy is more

```
MDVVNQLVAGGQFRVVKE(PLGFVKVLQWVFAIFAFATCGSY)TGELRLSVECANKTESALNIEVEFEYPFRLHQVYFDA
PSCVKGGTTKIFLVGDYSSSAE(FFVTVAVFAFLYSMGALATYIFL)QNKYRENNK(GPMMDFLATAVFAFMWLVSSSAW
A)KGLSDVKMATDPENIIKEMPMCRQTGNTCKELRDPVTS(GLNTSVVFGFLNLVLWVGNLWFVF)KETGWAAPFMRAPP
GAPEKQPAPGDAYGDAGYGQGPGGYGPQDSYGPQGGYQPDYGQPASGGGGYGPQGDYGQQGYGQQGAPTSFSNQM
```

Figure 3: An amino acid sequence which contains four transmembrane domains shown by the parenthesized parts.

Amino Acids	Hydropathy Index		New Symbol
A M C F L V I	$1.8 \sim 4.5$	$\rightarrow$	*
P Y W S T G	$-1.6 \sim -0.4$	$\rightarrow$	+
R K D E N Q H	$-4.5 \sim -3.2$	$\rightarrow$	-

Table 1: Transformation rules

```
*-**--***++-*-**--(+*+**-**-+*******+*+++)++-*-*+*-**--+-+**-*-*-*-++*-*--*+*-*
++**-++++-****+-++++*-(***+*******++*+***++***)---+-----(++**-***+******+**+++*+*
)-+*+-*-**+-+--**--*+**--++-+*--*--+*++(+*-++***+**-***+*+-*+***)--+++**+**-*++
+*+---+*++-*++-*+++-++++++++--+++++-+++-+-++-+++-+*++++++++-++-++--+++--+*++++*+--*
```

Figure 4: The sequence obtained by the transformation

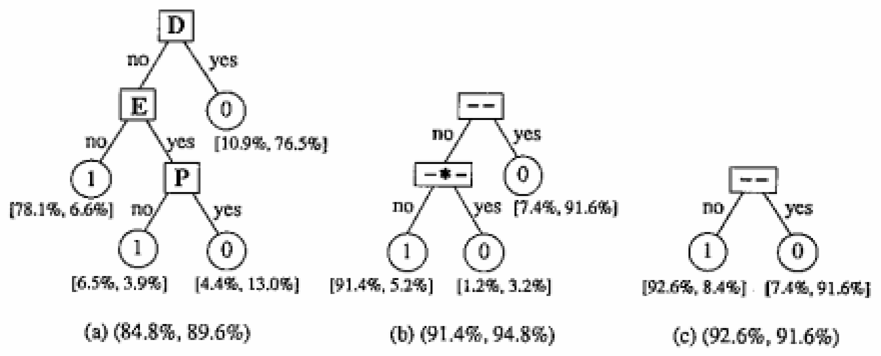

Figure 5: The node label, for example, -- is an abbreviation of x_1--x_2 that tests if a given sequence contains the sequence --. The leaf label 1 (resp. 0) is the class name of transmembrane domains (resp. non-transmembrane domains). The total space consists of 689 positive examples and 19256 negative examples. Each of the decision trees (a)-(c) is constructed from 10 positive and 10 negative training sequences. The pair $[p\%, n\%]$ attached to a leaf shows that $p\%$ of positive examples and $n\%$ of negative examples have reached to the leaf. The pair $(p\%, n\%)$ means that $p\%$ of 689 positive (resp. $n\%$ of 19256 negative) examples are recognized as transmembrane domains (resp. non-transmembrane domains).

than 90% for the total space in average. On the other hand, for raw sequences the accuracy is not so good but both accuracies approach to the same line as the number of training examples increases.

2. The number of nodes of a decision tree is reasonably small. But when the number of training examples is larger, the number of nodes in a decision tree becomes larger while the accuracy is not improved very much. There may arise the problem of overfitting.

A new discovery obtained from these decision trees is that the motif "--" drastically rejects positive examples. After knowing the negative motif "--", we have examined the decision trees with a single node with the patterns of the form

$$x_1\text{-}x_2\text{-}\cdots\text{-}x_n$$

for $n \geq 3$. The best is the pattern containing "-" five times. The result is quite acceptable as shown in Table 2.

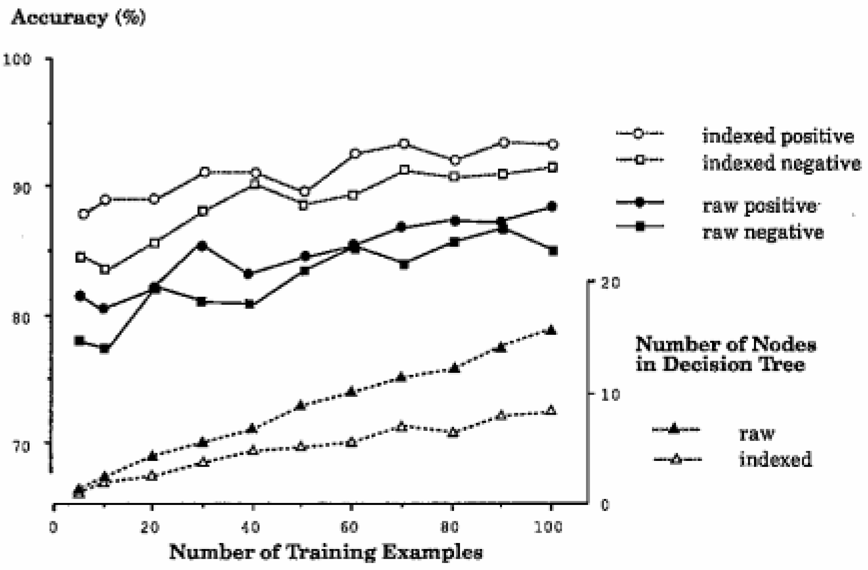

Figure 6: Relations between the number of training examples, accuracy and the number of nodes in a decision tree

Pattern	*POS* (689)	*NEG* (19256)
x_1-x_2-x_3-x_4-x_5-x_6	657 (95.4 %)	18296 (95.0%)

Table 2: Result for x_1-x_2-x_3-x_4-x_5-x_6

With these decision trees over regular patterns, we have developed a transmembrane domain predictor that reads an amino acid sequence of a protein as an input and predicts symbol by symbol whether each location of a symbol is in a transmembrane domain or not. Experiments on all protein sequences in PIR show that the success rate is 85% ~ 90%.

5 PAC-Learnable Class

This section provides a theoretical foundation on the classes of sets classified by decision trees over regular patterns from the point of algorithmic learning theory [Valiant 1984].

For integers $k, d \geq 0$, we consider a decision tree T over k-variable regular patterns whose depth is at most d. We denote by DTRP(d, k) the class of languages defined by decision trees over k-variable regular patterns with depth at most d.

Theorem 1 DTRP(d, k) is polynomial-time learnable for all $d, k \geq 0$.

We need some terminology for the above theorem. When we are concerned with learning, we call a subset of Σ^* a *concept.* A *concept class* $\mathcal{C}$ is a nonempty collection of concepts. For a concept $c \in \mathcal{C}$, a pair $\langle x, c(x)\rangle$ is called an example of c for $x \in \Sigma^*$,where $c(x) = 1$ ($c(x) = 0$) if x is in c (is not in c). For an alphabet Σ and an integer $n \geq 0$, $\Sigma^{\leq n}$ denotes the set $\{x \in \Sigma^* \mid |x| \leq n\}$.

A concept class $\mathcal{C}$ is said to be *polynomial-time learnable* [Blumer *et al.* 1989, Natarajan 1989, Valiant 1984] if there is an algorithm $\mathcal{A}$ which satisfies (1) and (2).

(1) $\mathcal{A}$ takes a sequence of examples as an input and runs in polynomial-time with respect to the length of input.

(2) There exists a polynomial $p(\cdot,\cdot,\cdot)$ such that for any integer $n \geq 0$, any concept $c \in \mathcal{C}$, any real number ε, δ ($0 < \varepsilon, \delta < 1$), and any probability distribution P on $\Sigma^{\leq n}$, if $\mathcal{A}$ takes $p(n, \frac{1}{\varepsilon}, \frac{1}{\delta})$ examples which are generated randomly according to P, then $\mathcal{A}$ outputs, with probability at least $1 - \delta$, a representation of a hypothesis h with $P(c \oplus h) < \varepsilon$.

Theorem 2 [Blumer *et al.* 1989, Natarajan 1989] A concept class $\mathcal{C}$ is polynomial-time learnable if the following conditions hold.

1. $\mathcal{C}$ is of *polynomial dimension*, i.e., there is a polynomial $d(n)$ such that $|\{c \cap \Sigma^{\leq n} \mid c \in \mathcal{C}\}| \leq 2^{d(n)}$ for all $n \geq 0$.

2. There is a polynomial-time algorithm called a *polynomial-time hypothesis finder* for $\mathcal{C}$ which produces a hypothesis from a sequence of examples such that it is consistent with the given examples.

Moreover, the polynomial-time hypothesis finder for $\mathcal{C}$ is a learning algorithm satisfying (1) and (2) if $\mathcal{C}$ satisfies 1.

The following lemma can be easily shown.

Lemma 1 Let T be a decision tree over regular patterns and T_v be a subtree of T at node v. We denote T_v by $\pi(T_0, T_1)$, where π is the label of node v and T_0, T_1 are the left and right subtrees of T_v, respectively. Let S be a set of strings and let T' be the tree obtained from T by replacing T_v with T_0 at node v. If no string in S matches π, then $L(T) \cap S = L(T') \cap S$.

Proof of Theorem 1.

First we show that the concept class $\mathrm{DTRP}(d, k)$ is of polynomial dimension. Let $\mathrm{DTRP}(d,k)_n = \{L \cap \Sigma^{\leq n} \mid L \in \mathrm{DTRP}(d,k)\}$ for $n \geq 0$. We evaluate the cardinality of $\mathrm{DTRP}(d,k)_n$. Let π be a regular pattern with $|\pi| > n + k$, then no string of length at most n matches π. By Lemma 1, we need to consider only regular patterns of length at most $n + k$. The number of such patterns is roughly bounded by $(|\Sigma| + 1)^{n+k}$. Since a tree of depth bounded by d has at most $2^d - 1$ internal nodes and at most 2^d leaves, $|\mathrm{DTRP}(d,k)_n| \leq ((|\Sigma| + 1)^{n+k})^{2^d-1} \cdot 2^{2^d}$ This shows that the dimension of $\mathrm{DTRP}(d,k)_n$ is $O(n)$.

Next we show that there is a polynomial-time hypothesis finder for $\mathrm{DTRP}(d, k)$. Let P and N be the sets of strings which appear in positive and negative examples, respectively. Let $\Pi(k, P, N)$ be the set of regular patterns π up to renaming of variables such that it contains at most k variable occurrences and $\pi\theta$ is a substring of some s in $P \cup N$. By Lemma 1, we need to consider only patterns in $\Pi(k, P, N)$ in order to find a decision tree over regular patterns which is consistent with P and N. Then $|\Pi(k,P,N)| \leq \sum_{s \in P \cup N}((|s|^2)^{k+1})$. Therefore the number of possible trees is bounded by $(|\Pi(k,P,N)|)^{2^d-1} \cdot 2^{2^d}$, which is bounded by a polynomial with respect to the input length $\sum_{s \in P \cup N} |s|$.

It is known that, given a regular pattern π and string w, we can decide in polynomial time whether w matches π or not. Therefore, given a string w and a decision tree T over k-variable regular patterns whose depth is at most d, we can decide whether $w \in L(T)$ or not in polynomial-time.

The required polynomial-time algorithm enumerates decision trees T over regular patterns in $\Pi(k, P, N)$ with depth at most d. Then it checks whether $s \in L(T)$ for each $s \in P$ and $t \notin N$ for each $t \in N$. If such a tree T is found, the algorithm outputs T as a hypothesis. □

We should say that the polynomial-time learning algorithm in the proof of Theorem 1 exhausts an enormous amount of time and is not suited for practical use.

We may understand the relationship of Algorithms 1 and 2 in Section 3 to Theorem 1 in the following way: When we set $\Pi(P, N)$ to be the family of k-variable regular patterns made from P and N, Algorithms 1 and 2 run sufficiently fast in practical use (of course, in polynomial-time) and produce a decision tree over k-variable regular patterns which classifies given positive and negative examples. But the produced decision tree is not guaranteed to be of depth at most d. Hence, these algorithms are not any learning algorithm in the exact sense of (2).

However, experiences tell that these algorithms usually find small enough decision trees over regular patterns in our experiments on transmembrane domains. For the class $\mathrm{DTRP}(d, k)$, Theorem 2 asserts that if a polynomial-time algorithm $\mathcal{A}$ produces a decision tree over k-variable regular patterns with depth at most d which classifies given positive and negative examples then it is a polynomial-time learning algorithm. In this sense, we may say that Algorithms 1 and 2 are polynomial-time algorithms for $\mathrm{DTRP}(d, k)$ which *often* produce reasonable hypotheses although there is no mathematical proof showing how often such small hypotheses are obtained. This aspect is very important and useful when we are concerned with machine discovery.

Ehrenfeucht and Haussler [1989] have considered learning of decision trees of a fixed rank. For learning decision trees over regular patterns, the restriction by rank can be shown to have no sense. Instead, we consider the depth of a decision tree. It is also reasonable to put a restriction on the number of variables in a regular pattern. It has been shown that the class of regular pattern languages is not polynomial-time learnable unless $\mathbf{NP} \neq \mathbf{RP}$ [Miyano *et al.* 1991]. Therefore, unless restrictions such as bound on the number of variables in a regular pattern are given, we may not expect any positive results for polynomial-time learning.

6 Conclusion

We have shown that the idea of combining regular patterns and decision trees works quite well for transmembrane domain identification. The experiments also have shown the importance of negative motifs.

A union of regular patterns is regarded as a special form of a decision tree called a decision list. We have reported in [Arikawa *et al.* 1992] that the union of small number of regular patterns can also recognize transmembrane domains with high accuracy. However, the time exhausted in finding hypotheses in [Arikawa *et al.* 1992] is much larger than that reported in this paper.

Our system constructs a decision tree over regular patterns just from strings called positive and negative examples. We need not take care of which attributes to specify as in ID3. Therefore it can be applied to another classification problems for proteins and DNA sequences. We believe that our approach provides a new application of algorithmic learning to Molecular Biology.

We are now in the process of examining our method for some other related problems such as predicting the secondary structure of proteins.

References

[Arikawa *et al.* 1992] 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. 25th Hawaii Int. Conf. on Sys. Sci*, IEEE, Hawaii, 1992. pp. 675–684.

[Bairoch 1991] A. Bairoch, PROSITE: A Dictionary of Sites and Patterns in Proteins, sl Nucleic Acids Res., Vol. 19 (1991), pp. 2241–2245.

[Blumer *et al.* 1989] A. Blumer, A. Ehrenfeucht, D. Haussler and M.K. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. *JACM*, Vol. 36 (1989), pp. 929–965.

[Ehrenfeucht and Haussler 1989] A. Ehrenfeucht and D. Haussler. Learning Decision Trees from Random Examples. *Inform. Comput.*, Vol. 82 (1989), pp. 231–246.

[Engelman *et al.* 1986] D.M. Engelman, T.A. Steiz and A. Goldman. Identifying Nonpolar Transbilayer Helices in Amino Acid Sequences of Membrane Proteins. *Ann. Rev. Biophys. Biophys. Chem.*, Vol. 15 (1986), pp. 321–353.

[Gusev and Chuzhanova 1990] V. Gusev, N. Chuzhanova. The Algorithms for Recognition of the Functional Sites in Genetic Texts. In *Proc. 1st Workshop on Algorithmic Learning Theory*, Tokyo, 1990. pp. 109–119.

[Hartmann *et al.* 1989] E. Hartmann, T.A. Rapoport and H.F. Lodish. Predicting the Orientation of Eukaryotic Membrane-Spanning Proteins. In *Proc. Natl. Acad. Sci. U.S.A.*, Vol. 86 (1989), pp. 5786–5790.

[Holly and Karplus 1989] L.H. Holly and M. Karplus. Protein Secondary Structure Prediction with a Neural Network, In *Proc. Natl. Acad. Sci. USA*, Vol. 86 (1989), pp. 152–156.

[Kyte and Doolittle 1982] J. Kyte and R.F. Doolittle. A Simple Method for Displaying the Hydropathic Character of Protein. *J. Mol. Biol.*, Vol. 157 (1982), pp. 105–132.

[Lipp *et al.* 1989] J. Lipp, N. Flint, M.T. Haeuptle and B. Dobberstein. Structural Requirements for Membrane Assembly of Proteins Spanning the Membrane Several Times. *J. Cell Biol.*, Vol. 109 (1989), pp. 2013–2022.

[Miyano *et al.* 1991] S. Miyano, A. Shinohara and T. Shinohara. Which Classes of Elementary Formal Systems are Polynomial-Time Learnable? In *Proc. 2nd Algorithmic Learning Theory*, Tokyo, 1991. pp. 139–150.

[Natarajan 1989] B.K. Natarajan, On Learning Sets and Functions. *Machine Learning*, Vol. 4 (1989), pp. 67–97.

[PIR] Protein Identification Resource, National Biomedical Research Foundation.

[Quinlan 1986] J.R. Quinlan, Induction of Decision Trees. *Machine Learning*, Vol. 1 (1986), pp. 81–106.

[Quinlan and Rivest 1989] J.R. Quinlan and R. L. Rivest. Inferring Decision Trees using the Minimum Description Length Principle. *Inform. Comput.*, Vol. 80 (1989), pp. 227–248.

[Rao and Argos 1986] J.K.M. Rao and P. Argos. A Confirmational Preference Parameter to Predict Helices in Integral Membrane Proteins. *Biochim. Biophys. Acta*, Vol. 869 (1986), pp. 197–214.

[Shinohara 1982] T. Shinohara. Polynomial Time Inference of Pattern Languages and its Applications. In *Proc. 7th IBM Symp. Mathematical Foundations of Computer Science*, 1982. pp. 191–209.

[Shinohara 1983] T. Shinohara. Polynomial Time Inference of Regular Pattern Languages. In *Proc. RIMS Symp. Software Science and Engineering* (Lecture Notes in Computer Science, Vol. 147), 1983. pp. 115–127.

[Utgoff 1989] P.E. Utgoff. Incremental Induction of Decision Tree. *Machine Learning*, Vol. 4 (1989), pp. 161–186.

[Valiant 1984] L. Valiant. A Theory of the Learnable. *Commun. ACM*, Vol. 27 (1984), pp. 1134–1142.

[Von Heijine 1988] G. von Heijine. Transcending the Impenetrable: How Proteins Come to Terms with Membranes. *Biochim. Biophys. Acta*, Vol. 947 (1988), pp. 307–333.

[Wu *et al.*] C.H. Wu, G.M. Whiston and G.J. Montllor. PROCANS: A Protein Classification System Using a Neural Network, *IJCNN Int. Joint Conf. Neural Networks*, Vol. 2 (1990), pp. 91-96.