Regular Paper

Knowledge Acquisition from Amino Acid Sequences by Machine Learning System BONSAI

SHINICHI SHIMOZONO,[†] AYUMI SHINOHARA,[††] TAKESHI SHINOHARA,[†††]
SATORU MIYANO,[††] SATORU KUHARA[††††] and SETSUO ARIKAWA[††]

We present a machine learning system, called BONSAI, for knowledge acquisition from positive and negative examples of strings, and report some experiments on protein data using the PIR and GenBank databases. This learning system is constructed with an algorithmic learning theory for decision trees over regular patterns, which is newly developed for this work. As a hypothesis, the system tries to find a pair of a classification of symbols called an *alphabet indexing* and a *decision tree over regular patterns*, which classifies given examples with high accuracy. Through the experiments, the system discovered very simple hypotheses that exhibit important knowledge about transmembrane domains and signal peptides.

1. Introduction

Algorithmic learning is a process carried out by a computer program that receives examples and guesses the unknown rule that generates the examples. In general words, therefore, it is considered as a computational model of induction. When guessed rules are represented in a way suitable for human understanding, learning processes can also be viewed as a kind of knowledge acquisition.

This paper shows that the research based on algorithmic learning theory that made a great success in practical applications in Molecular Biology. As is well known, most important information such as genes and proteins are coded in sequences of symbols from a finite alphabet. Therefore Molecular Biology should be one of the most important and suitable fields to apply algorithmic learning.

The hydropathy index of amino acid residues has been shown to play an essential role in transmembrane domain identification.[1),2),9),21)] In Ref. 1), 2), we classified 20 symbols of amino acid residues to three categories $\{*, +, -\}$ according to the hydropathy index of Kyte and Doolittle.[9)] Then we transformed amino acid sequences to those consisting of three symbols and used them as examples for our learning algorithms. Interestingly, after this transformation, there are only a few overlaps between transmembrane domain sequences and non-transmembrane domain sequences. By experiments using the learning algorithm in Ref. 1), 10), we have verified that this transformation is very useful for the transmembrane domain identification problem. In Ref. 2), we also developed another learning system that produces a hypothesis from a small set P of strings called *positive examples* and a small set N of strings called *negative examples*. As a hypothesis, the system searches for a decision tree over regular patterns that classfies P and N completely. The system discovered a very small hypothesis that distinguishes all transmembrane domain sequences in the PIR database[12)] from other parts with accuracy more than 91%. With this system, we have seen that the transformation by the hydropathy index is very successful. In this paper we consider a transformation from an alphabet to a smaller alphabet which does not lose any positive and negative information of the original examples. We shall call such a transfor-

[†] Department of Control Engineering and Science, Kyushu Institute of Technology, Iizuka 820, Japan
[††] Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan
[†††] Department of Artificial Intelligence, Kyushu Institute of Technology, Iizuka 820, Japan
[††††] Graduate School of Genetic Resources Technology, Kyushu University 46, Fukuoka 812, Japan

This work is partly supported by Grant-in-Aid for Scientific Research on Priority Areas, "Genome Informatics" from the Ministry of Education, Science and Culture, Japan.

mation an *alphabet indexing* or simply an *indexing*.

Without using the indexing technique, we have seen in Ref. 2) that the learning system has found hypotheses with sufficiently high accuracy. However, the biological knowledge on the hydropathy index greatly increases the accuracy and simplifies hypotheses. This observation inspired us to discover such an indexing itself without any explicit help of biological knowledge just by a learning algorithm with data.

This paper presents a machine learning system BONSAI that has succeeded in discovering such an indexing together with a decision tree that attains very high accuracy. Given positive and negative examples, BONSAI will find an alphabet indexing providing "good" decision trees over regular patterns. The idea behind our method is to combine the local search technique for alphabet indexings and the learning algorithm developed in Ref. 2). The learning algorithm produces a decision tree over regular patterns for positive and negative examples and the other part works to find a good alphabet indexing. This system is developed with an algorithmic learning theory for decision trees over regular patterns and combinatorial optimization schemes for alphabet indexings on symbols of strings.

An alphabet indexing is a transformation of symbols to reduce the size of the alphabet for the positive and negative examples, without missing important information in original data. In the case of amino acid residues, an alphabet indexing can be regarded as a classification of 20 kinds of amino acid residues to a few categories. In the experiment on transmembrane domain identification from the PIR database, this system has found an alphabet indexing that is nearly the same as the hydropathy index of Kyte and Doolittle, without any knowledge on the hydropathy index. It has also discovered hypotheses with high accuracy for recognition of signal regions on coding sequences.[3),5),11),16)−18),20)]

First we present a learning algorithm for decision trees over regular patterns, and then we describe a local search algorithm for optimizing the alphabet indexing to produce good hypotheses. Finally, we report some experiments of our system[15)] on transmembrane domain sequences and signal peptide sequences by using the PIR and GenBank databases.

2. Learning Algorithm and Combinatorial Optimization Scheme for BONSAI System

This section gives algorithms for constructing decision trees over regular patterns and finding indexings that are implemented in our machine learning system.

The whole algorithm consists of two parts: one for constructing a decision tree and the other for finding a better indexing (**Fig. 1**). The first part is almost the same as one which we developed in Ref. 2). The only difference is that our new algorithm can deal with inconsistent training examples. In our previous work,[2)] we assumed that the sets of positive examples P and negative examples N are mutually disjoint, and the algorithm may not terminate if the examples are not consistent. However, in this approach combined with searching for an indexing, the sets of indexed examples may have some overlaps though these overlaps should be small. Thus, our new algorithm is designed to break off the recursion when it recognizes that it is impossible to divide the positive and negative examples into smaller fractions.

2.1 Learning Decision Trees over Regular Patterns

First we briefly review how to construct decision trees over regular patterns from positive and negative examples.

A *regular pattern* π is a string of the form $\pi = w_0 x_1 w_1 x_1 w_2 x_2 \cdots x_n w_n$, where each w_i is a constant string and each x_i is a variable that matches any string. Hence the above pattern defines any string containing substrings w_0, w_1, $\cdots$, w_n in that order. $L(\pi)$ denotes the set of those strings.

A *decision tree over regular patterns* is a procedural definition of rules to determine the class for any given string. Each internal node is labeled with a regular pattern and each leaf is labeled with a name of a class (the class name is 'Positive' or 'Negative'). An internal node has two successors called the left and right children. To classify a given string, we start from the root. The pattern on the current internal node examines whether the string matches the pattern or not. Then according to its answer 'yes' or 'no',

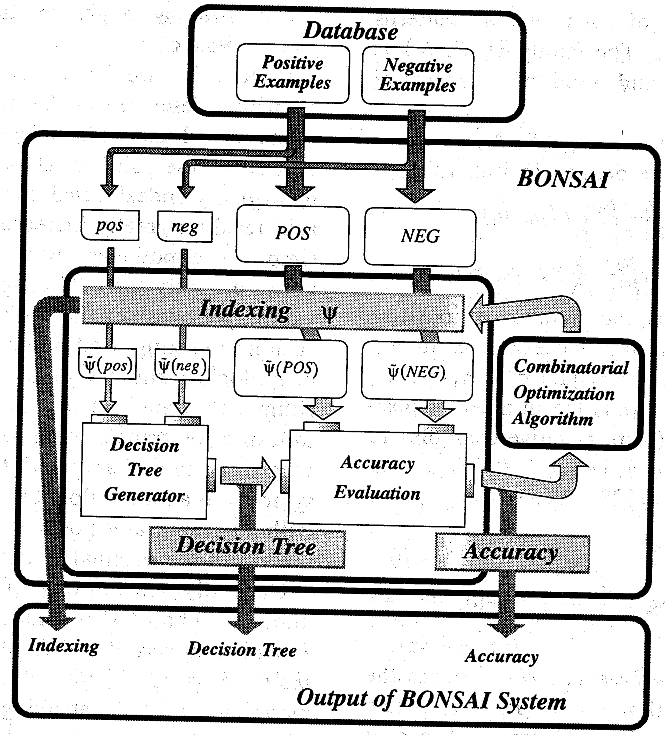


Fig. 1 The BONSAI system. As an input, the system takes a pair of the sets of the positive and the negative examples, and computes an indexing, a decision tree and the accuracy. The first part builds a decision tree and evaluates its accuracy for the examples transformed by the indexing ψ. The second part searches a better indexing by using the accuracy computed in the first part. The interaction of these two parts repeats until the system reaches to a locally optimal indexing and a decision tree.

one of the children is chosen to continue the classification. This continues until the process reaches a leaf. Then we get the classification of the string by the name of that leaf. For a decision tree T over regular patterns, we define $L(T)$ to be the set of all strings that are recognized as 'Positive' by T.

The machine learning system produces decision trees over regular patterns. On each trial, it chooses a sample called positive examples and negative examples randomly, and produces a hypothesis described by a decision tree over regular patterns that classifies those examples perfectly. To find a small decision tree, according to the widely believed principle "a smaller decision tree involves an essential knowledge," we employed and modified ID3 algorithm by Quinlan.[13] The ID3 algorithm assumes examples specified 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. Our algorithm tries to find 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. Notice that P and N may have some intersection. Using P and N, we deal with regular patterns of the form $\alpha_0 x_1 \alpha_1 x_2 \cdots x_k \alpha_k$ such that $\alpha_0, \cdots, \alpha_k$ 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 $\prod(P, N)$ is appropriately given and used as a space of attributes.

For a regular pattern $\pi\in\prod(P, N)$, the cost $E(\pi, P, N)$ is the one defined in Ref. 13) 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

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

Algorithm *DecisionTree* (P, N) sketches the decision tree algorithm for $\prod(P, N)$, where *Tree*(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.

```
function DecisionTree (P, N: sets of strings): node;
  begin
    if $N=\emptyset$ then return Tree("P", null, null) /* leaf labeled with "P"*/
    else if $P=\emptyset$ then return Tree ("N", null, null) /* leaf labeled with "N"*/
    else begin
      Find a shortest pattern $\pi$ in $\prod(P, N)$ that minimizes $E(\pi, P, N)$;
      $P_1\leftarrow P\cap L(\pi)$; $P_0\leftarrow P-P_1$;
      $N_1\leftarrow N\cap L(\pi)$; $N_0\leftarrow N-N_1$;
      if ($P_0=P$ and $N_0=N$)
                    or ($P_1=P$ and $N_1=N$) then
        return Tree ("P", null, null)
      else
        return Tree ($\pi$, DecisionTree ($P_0$, $N_0$), DecisionTree ($P_1$, $N_1$))
    end
  end
```

Algorithm 1: *DecisionTree*

2.2 Finding Alphabet Indexing by Local Search

In Ref. 2), we have seen that the learning algorithm described in the previous section has found hypotheses with sufficiently high accuracy. However, the biological knowledge on the hydropathy index[9] used for classifying amino acid residues greatly increases the accuracy and simplifies hypotheses, even the classified categories are only three.[1),2)] Inspired by this observation, we consider the problem of discovering such an indexing itself without any explicit help of biological knowledge just by a learning algorithm with data. In this section, we define an indexing for transforming sequences as a mapping from an alphabet with the large number of symbols to another alphabet with fewer symbols without losing any positive and negative information of the original examples.

Generally, an *indexing* of an alphabet Σ by another alphabet Γ is a mapping from Σ to Γ. For disjoint sets P and N of strings over Σ, an *indexing* ψ *of* Σ *for* P *and* N *by* Γ is a mapping ψ: $\Sigma\to\Gamma$ satisfying $\tilde{\psi}(P)\cap\tilde{\psi}(N)=\emptyset$, where the homomorphism $\tilde{\psi}(s)$ for a string $s\in\Sigma^*$ is defined by the transformation of each symbols $\tilde{\psi}(c_1\cdots c_n)=\psi(c_1)\cdots\psi(c_n)$.

In the above definition, the indexing ψ for the sets of the positive examples P and the negative examples N must satisfy the condition $\tilde{\psi}(P)\cap\tilde{\psi}(N)=\emptyset$. Although the hydropathy index which we used in Ref. 1), 2) satisfied the above condition, this is a too strong condition for practical applications since the problem of finding an indexing is shown NP-hard.[14)] Thus, in practice, we may relax the condition so that $\tilde{\psi}(P)$ and $\tilde{\psi}(N)$ have a few overlaps.

Now we describe the second part of the system, the algorithm *FindGoodIndex*, that finds better indexings. It is a local search algorithm that uses a function *Score*(ψ) calling *DecisionTree*. Let Ψ be the set of indexings from Σ by Γ. We can consider $\Psi=\Gamma^{|\Sigma|}$ by assuming an indexing ψ as the string $\psi(\sigma_1)\psi(\sigma_2)\cdots\psi(\sigma_n)$ for $\Sigma=\{\sigma_1, \cdots, \sigma_n\}$. Let *POS* be a set of positive examples and *NEG* be a set of negative examples. First, the algorithm randomly chooses two small subsets *pos* of *POS* and *neg* of *NEG*, and begins with an indexing ψ randomly generated. Then it searches an indexing ϕ from its neighbors such that its score is the best among the neighbors of ψ. The neighbors of ψ are the indexings whose distance from ψ is one, where the distance between ψ and ϕ in Ψ is defined by $d(\psi, \phi)=|\{\sigma\in\Sigma|\psi(\sigma)\neq\phi(\sigma)\}|$. To evaluate the $Score(\psi)$ of an indexing ψ, it constructs a tree T by running the procedure *DecisionTree* $(\tilde{\psi}(pos), \tilde{\psi}(neg))$ and then evaluates the success rate that T explains $\tilde{\psi}(POS)$ and $\tilde{\psi}(NEG)$ correctly. Then $Score(\psi)$ is determined by

$$\sqrt{\frac{|L(T)\cap\tilde{\psi}(POS)|}{|\tilde{\psi}(POS)|}\cdot\frac{|\overline{L(T)}\cap\tilde{\psi}(\mathrm{NEG})|}{|\tilde{\psi}(NEG)|}},$$

which represents the geometric mean of the success rates of the decision tree T for positive examples $\tilde{\psi}(Pos)$ and negative examples $\tilde{\psi}(Neg)$. This search process continues until no better indexing is found from its neighbors. The strategy of the algorithm is sketched in Algorithm *FindGoodIndex*.

```
function FindGoodIndex(POS, NEG: sets of strings): indexing;
  begin
    Select small subsets pos of POS and neg of NEG randomly;
    Generate randomly $\psi$ in $\Psi$;
    repeat
      Find $\psi'\in\{\psi\in\Psi|d(\psi, \psi')=1\}$ that maximizes Score($\psi'$);
      if Score($\psi'$) $\leq$ Score($\psi$) then return $\psi$;
      $\psi\leftarrow\psi'$;
    forever;
  end
```

Algorithm 2: *FindGoodIndex*

In order to find good indexings, other methods than local search may be successfully applied. The techniques of simulated annealing and genetic algorithms are candidates for good searching methods.

However, finding a locally optimal indexing is computationally hard. This can be shown by investigating our local search strategy as a polynomial-time local search problem (PLS-problem for short), which has been defined by Johnson et al.[8] to formulate and analyze the local search algorithms (see the Ref. 8) for the details).

For our problem, we have obtained the result that finding locally optimal indexing for the general weighted problem is PLS-complete.[14] Even in the unweighted case, finding an indexing by our algorithm is P-complete. This fact asserts that the algorithm cannot be efficiently parallelizable unless NC=P.[4]

3. Experiments

In this section, we report our experiments on identification problems for transmembrane domains and signal peptides.

3.1 Method of Experiments

The sets *POS* and *NEG* denote the sets of positive and negative examples used as inputs to the system. In applying our machine learning system to these data sets, we also have to specify the size of the small sets *pos* and *neg* which shall be chosen from *POS* and *NEG* at random. Moreover, we have to specify the size of the indexing alphabet. If we specify $|pos|$, $|neg|$ and the size of the indexing alphabet, the system with *POS* and *NEG* will produce a hypothesis (T, ψ) consisting of a decision tree T over regular patterns and an indexing ψ. The accuracy of (T, ψ) for *POS* and *NEG* is represented by a pair $(p\%, n\%)$, where $p\%$ (resp. $n\%$) of *POS* (resp. *NEG*) are recognized as positive (resp. negative). The score of the hypothesis (T, ψ) is defined by

$$\sqrt{\frac{|L(T)\cap\tilde{\psi}(POS)|}{|\tilde{\psi}(POS)|}\cdot\frac{|\overline{L(T)}\cap\tilde{\psi}(NEG)|}{|\tilde{\psi}(NEG)|}}.$$

The system tries to search a hypothesis with a higher score by changing the initial indexing ϕ and the sets *pos* and *neg* randomly. After some amount of time, the system shall give the hypotheses with the highest scores.

In experiments, we assume the size of *pos* and *neg* to be

$$|pos|=|neg|=10$$

and the indexing alphabet Γ has 2~3 symbols. Moreover, in order to avoid combinatorial explosions, we also assume that the regular patterns attached to the nodes of decision trees are of the form

$$xay,$$

where x and y are variables and a is a substring taken from *pos* and *neg*. There is no other special reason why we used only these regular patterns.

3.2 Positive and Negative Examples

First, we describe the positive and negative examples used for our experiments. The data for transmembrane domains[7),19)] are amino acid sequences taken from the PIR database.[12)] For signal peptides,[3),5),11),16)–18),20)] we use the GenBank database.[6)]

Since the PIR database and the GenBank database are not at all complete, the data we shall mention below may contain some noise.

3.2.1 Transmembrane Domains

We use the PIR database, which contains the amino acid sequences with FEATURE field where transmembrane domains are indicated. In this experiment, positive examples are the amino acid sequences of transmembrane domains. As negative examples, we use the amino acid sequences located in other parts than transmembrane domains. Since a transmembrane domain consists of around 30 residues, we selected sequences of length 30 for negative examples.

For example, if there is an amino acid sequence w in the PIR database whose FEATURE field indicates that two transmembrane domains are contained:

10–45 # Domain transmembrane
100–126 # Domain transmembrane

Then the substrings $w[10..45]$ and $w[100..126]$ are taken as positive examples. The sequence $w[46..75]$ is a possible negative example, while $w[40..60]$ is not, because the initial segment of $w[40..60]$ is located in a transmembrane domain $w[10..45]$.

We collect all the positive examples from the PIR database (**Table 1**). The number of positive examples is 689. We use 19256 negative examples randomly chosen.

3.2.2 Signal Peptides

We use the GenBank database which contains

Table1 Examples taken from PIR database.

Sequences	Positive	Negative
Transmembrane	689	19256

Table 2 Examples taken from GenBank database.

Sequences	Positive	Negative
Viral	120	4882
Bacterial	495	7330
Invertebrate	263	1927
Primate	1032	3162
Rodent	1018	3158
Other Mammalian	235	588
Other Vertebrate	207	1056
Plant	370	3074

Indexing

Amino Acid	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
New Symbol	0	0	1	1	0	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0
Hydropathy Index	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

Decision tree

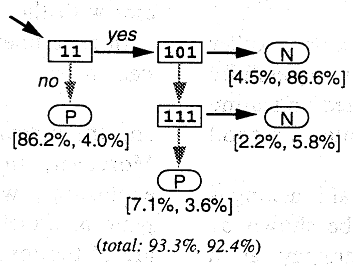


Fig. 2 Transmembrane domains. The indexing alphabet is {0, 1} of size 2. We can see that the indexing is closely related to the hydropathy. The leaf label P (resp. N) is the class name of transmembrane domains (resp. non-transmembrane domains). The pair $[p\%, n\%]$ attached to a leaf means that $p\%$ of 689 positive (resp. $n\%$ of 19256 negative) examples reached the leaf. The pair (*total:* $p\%$, $n\%$) means that $p\%$ of 689 positive (resp. $n\%$ of 19256 negative) examples are recognized as transmembrane domains (resp. non-transmembrane domains).

DNA and RNA sequences with information about their features. Signal peptides are indicated in the FEATURE field.

A signal peptide is located at N-terminal region, that is at the initial segment of an amino acid sequence. We collected the signal peptide sequences beginning with a Methionine (M) and of length at most 32. Such sequences constitute the set of positive examples. Thus, for the negative examples, we take N-terminal regions of length 30 obtained from complete sequences that have no signal peptide and begin with a Methionine.

We made experiments on the following items (**Table 2**). The entries in the second and third columns contain the numbers of positive and

Sequences	Indexing				Tree
	ACDEFGH	IKLMNPQ	RSTVWY	X	
Viral	0022011	0000220	101002	2	(1)
Bacterial	1022000	1000011	111000	2	(2)
Invertebrate	0122022	0210211	011011	2	(3)
Primate	0022012	0201210	010012	2	(4)
Rodent	0122011	1202122	210002	0	(5)
Other Mammalian	0122022	0211211	000010		(6)
Other Vertebrate	0022002	0200012	200100	0	(7)
Plant	0022002	1001220	000102	2	(8)

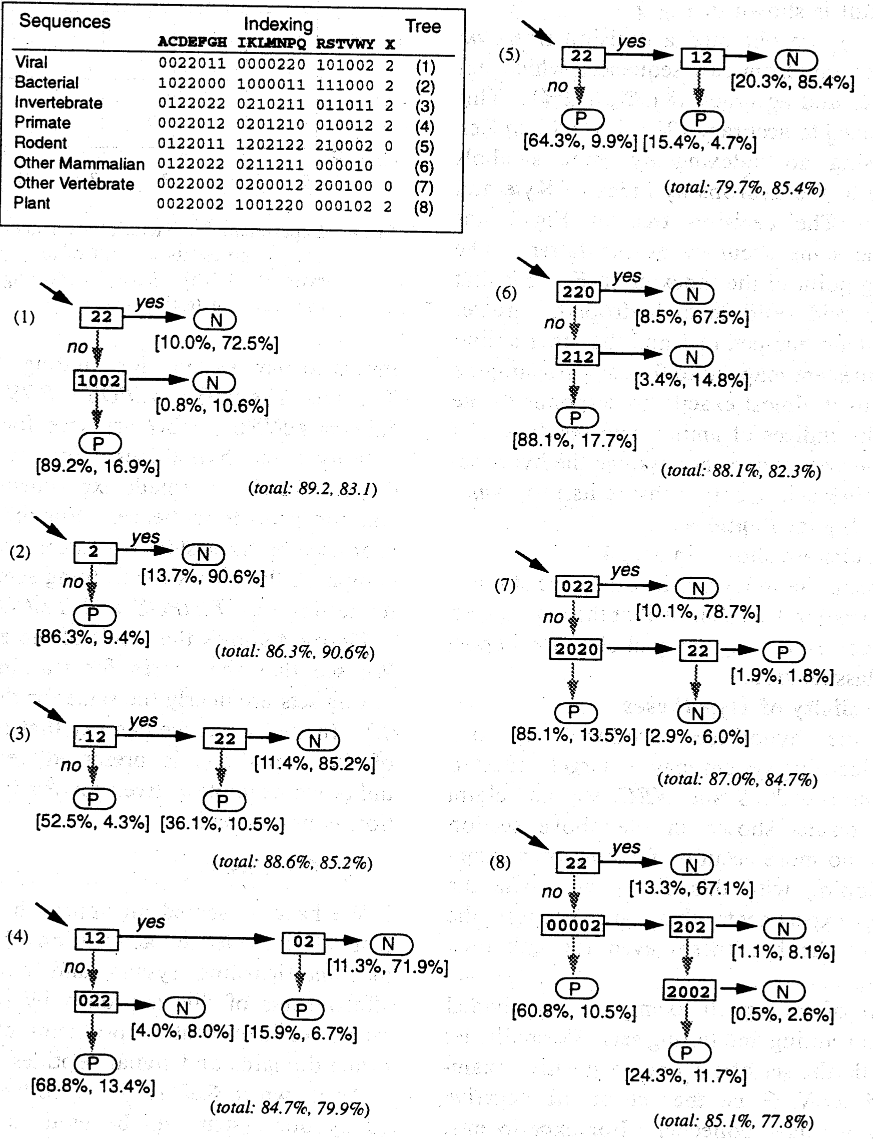


Fig. 3 Signal peptides. Indexings for amino acid residues and decision trees classifying signal peptides obtained by the Bonsai system. Each decision tree numbered from (1) to (8) with indexings in the table describes the rule (see Fig. 2) for various kinds of signal sequences. The amino acid residues corresponding to symbol 2 are important in classification in this result.

negative examples.

3.3 Results

The following figures show good indexings and decision trees that our ML system found from the examples explained in the above section. In the figures, the node label, for example, 11 is an abbreviation of $x11y$ that tests if a given sequence contains the sequence 11.

3.3.1 Transmembrane Domains

The result is shown in **Fig. 2**.

In Ref. 2), we obtained a decision tree over regular patterns from raw sequences which has three nodes and accuracy (84.8%, 86.9%). This was improved to accuracy (91.4%, 94.8%) in Ref. 2) by using an indexing by three symbols according to the hydropathy index of Kyte and Doolittle.[9] The decision tree in Fig. 2 has almost the same accuracy as the latter. The interesting point of the indexing in Fig. 2 is that all amino acid residues of hydropathy greater than -1.0 are mapped to 1 and the other amino acid residues are mapped to 0 except Asparagine (N). Thus it almost exactly corresponds to the hydropathy indices of amino acid residues. Of course, the system does not assume the hydropathy of amino acid residues inside its program.

3.3.2 Signal Peptides

The results are shown in **Fig. 3**.

As is seen, the indexings in Fig. 3 are similar. Decision trees in Fig. 3 show that the amino acid residues corresponding to symbol 2 are important in classification.

3.4 Validity of Hypotheses

Since the hypotheses produced by our machine learning system may be largely affected by training sets *POS* and *NEG*, we may claim that the results shown in the above section would be no more accurate for unknown examples. Coping with this claim, we made the following experiments that will support the validity of the hypotheses even for unknown examples.

Instead of taking all examples, we divided them into training and testing sets. Formally, let *AllPOS* be the set of all known positive examples and *AllNEG* be the set of all negative examples we have collected. For experiments, we choose, at random, subsets *POS* and *NEG* of *AllPOS* and *AllNEG*, respectively. We assume that $|POS|/|AllPOS|=|NEG|/|AllNEG|=R$. These sets *POS* and *NEG* shall be used as inputs to our system for finding hypotheses. The sets $TestPOS=AllPOS-POS$ and $TestNEG=AllNEG-NEG$ are used for testing the validity of the hypotheses. For the ratios $R=$ 0.1, 0.3, 0.5, 0.7, we made experiments for bacterial and primate sequences. For the hypotheses produced by the system with *POS* and *NEG*, we compared the scores for training sets with those for testing sets *TestPOS* and *TestNEG*.

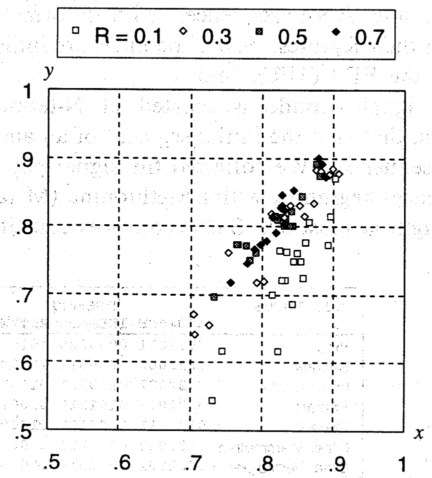


Fig. 4 Experiment for bacterial sequences. Each point (x, y) represents a result of an experiment with ratio R. x is the score for training sets and y is the score for testing sets.

Figure 4 shows the result of the experiments. We see that the scores for training sets and testing sets are nearly the same for the ratio $R \geq 0.3$. In such case, we can say that the accuracy of the hypothesis is preserved very well for unknown examples. Even for $R=0.1$, the situation is not so bad.

4. Conclusion

We have presented an approach to bioinformatical knowledge acquisition by using a machine learning system and confirmed the effectiveness of this approach by some experiments on identification problems of transmembrane domains and signal peptides.

As shown in Ref. 1), 2), hypotheses discovered by our system can be used successfully to predict transmembrane domains. In contrast, for predicting regions of signal peptides, some modifications on the setting will be naturally needed. In our experiments, the positive examples are amino acid sequences of signal peptides and the negative examples are N-terminal regions of length 30. Therefore, a hypothesis produced by our machine learning system can say whether an N-terminal region contains signal peptides, but cannot say how long the signal peptide is. This is a drawback in our system since the view to data is expressed by regular patterns.

The strength of our method is that the system provides a hypothesis in the form of an indexing and a small decision tree over regular patterns, which may be more understandable and suggest key items in classification. It gives a possibility of discovering important knowledge expressed in the indexing and the decision tree over regular patterns. We observed that this is the case in transmembrane domain identification.

We believe that our approach will provide a way to evaluate the studies of biology and can open a new frontiers for biologists to study.

References

1) Arikawa, S., Kuhara, S., Miyano, S., Shinohara, A. and Shinohara, T.: A Learning Algorithm for Elementary Formal Systems and Its Experiments on Identification of Transmembrane Domains, *Proc. 25th Hawaii International Conference on System Sciences*, Vol. I, pp. 675-684 (1992).

2) Arikawa, S., Miyano, S., Shinohara, A., Kuhara, S., Mukouchi, Y. and Shinohara, T.: A Machine Discovery from Amino Acid Sequences by Decision Trees over Regular Patterns, *New Gener. Comput.*, Vol. 11, pp. 361-375 (1993).

3) Austen, B.: Predicted Secondary Structures of Amino-terminal Extension Sequences of Secreted Proteins, *FEBS Letter*, Vol. 103, pp. 308-312 (1979).

4) Cook, S. A.: A Taxonomy of Problems with Fast Parallel Algorithms, *Inf. Control*, Vol. 64, pp. 2-22 (1985).

5) Engelman, D. and Steitz, T.: The Spontaneous Insertion of Proteins into and across Membranes: the Helical Hairpin Hypothesis, *Cell*, Vol. 23, pp. 411-422 (1981).

6) GenBank, Genetic Sequence Data Bank (1991).

7) Hartmann, E., Rapoport, T. A. and Lodish, H. F.: Predicting the Orientation of Eukaryotic Membrane-Spanning Proteins, *Proc. National Academy of Science of the United States of America*, Vol. 86, pp. 5786-5790 (1989).

8) Johnson, D., Papadimitriou, C. and Yannakakis, M.: How Easy Is Local Search?, *J. Comput. Syst. Sci.*, Vol. 37, pp. 79-100 (1988).

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

10) Miyano, S., Shinohara, A. and Shinohara, T.: Which Classes of Elementary Formal Systems are Polynomial-Time Learnable?, *Proc. 2nd Workshop on Algorithmic Learning Theory*, pp. 139-150 (1991).

11) Perlman, D. and Halvorson, H.: A Putative Signal Peptidase Recognition Site and Sequence in Eukaryotic and Prokaryotic Signal Peptides, *J. Mol. Biol.*, Vol. 167, pp. 391-409 (1983).

12) Protein Identification Resource, National Biomedical Research Foundation (1991).

13) Quinlan, J.: Induction of Decision Trees, *Machine Learning*, Vol. 1, pp. 81-106 (1986).

14) Shimozono, S. and Miyano, S.: Complexity of Finding Alphabet Indexing, Technical Report RIFIS-TR-CS-61, Research Institute of Fundamental Information Science, Kyushu University (1992).

15) Shimozono, S., Shinohara, A., Shinohara, T., Miyano, S., Kuhara, S. and Arikawa, S.: Finding Alphabet Indexing for Decision Trees over Regular Patterns: an Approach to Bioinformatical Knowledge Acquisition, *Proc. 26th Hawaii International Conference on System Sciences*, Vol. I, pp. 763-773 (1993).

16) Von Heijine, G.: Patterns of Amino Acids near Signal-Sequence Cleavage Sites, *Eur. J. Biochem.*, Vol. 133, pp. 17-21 (1983).

17) Von Heijine, G.: How Signal Sequences Maintain Cleavage Specificity, *J. Mol. Biol.*, Vol. 173, pp. 243-251 (1984).

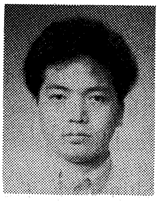
18) Von Heijine, G.: A New Method for Predicting Signal Sequence Cleavage Sites, *Nuc. Acids. Res.*, Vol. 14, pp. 4683-4690 (1986).

19) Von Heijine, G.: Transcending the Impenetrable: How Proteins Come to Terms with Membranes, *Biochimica et Biophysica Acta*, Vol. 947, pp. 307-333 (1988).

20) Watson, M.: Compilation of Published Signal Sequences, *Nuc. Acids. Res.*, Vol. 12, pp. 5145-5164 (1984).

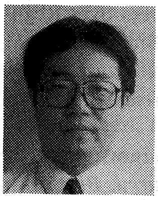
21) Yanagihara, N., Suwa, M. and Mitaku, S.: A Theoretical Method for Distinguishing between Soluble and Membrane Proteins, *Biophysical Chemistry*, Vol. 34, pp. 69-77 (1989).

(Received February 16, 1994)
(Accepted June 20, 1994)

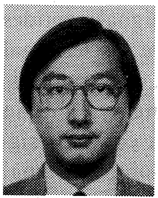


Shinichi Shimozono was born on April 7, 1966. He received the B. S. degree in 1989 in Physics and the M. S. degree in 1991 in Information Systems from Kyushu University. Presently, he is an Assistant of Department of Control Engineering and Science, Kyushu Institute of Technology. His research interests are the computational complexity theory and algorithms for combinatorial optimization problems.

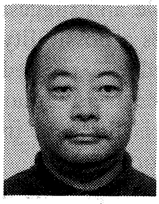
Ayumi Shinohara was born in Fukuoka on July 18, 1965. He received the B. S. degree in 1988 in Mathematics, the M. S. degree in 1990 and the Dr. Sci. in 1994 in Information Systems from Kyushu University. Presently, he is an Assistant of Research Institute of Fundamental Information Science, Kyushu University. His research interests are computational learning theory and algorithms. He is a member of IPSJ.



Takeshi Shinohara is a Professor of Department of Artificial Intelligence, Kyushu Institute of Technology. He received the B. S. degree in 1980 from Kyoto University, the M. S. and the Dr. Sc. degrees from Kyushu University in 1982 and 1986, respectively. His present interests are information retrieval, string pattern matching algorithms and computational learning theory. He is a member of IPSJ.



Satoru Miyano was born in Oita on December 5, 1954. He received the B. S. in 1977, the M. S. degree in 1979 and the Dr. Sci. in 1984 all in Mathematics from Kyushu University. Presently, he is a Professor of Research Institute of Fundamental Information Science, Kyushu University. His present interests include parallel algorithms, computational complexity, computational learning theory and genome informatics. He is a member of IPSJ.



Satoru Kuhara is an Associate Professor of Graduate School of Genetic Resources Technology, Kyushu University. He was born in Fukuoka on April 20, 1950. He received the B. A. degree in 1974, the M. A. degree in 1976 and the Dr. Agr. in 1980 from Kyushu University. His present interests include computer analysis of genetic information and protein structures.

Setsuo Arikawa is a Professor of Research Institute of Fundamental Information Science and the Director of Computer Center, Kyushu University. He was born on April 29, 1941. He received the B. S. degree in 1964, the M. S. degree in 1966 and the Dr. Sc. degree in 1969 all in Mathematics from Kyushu University. He has been working on algorithmic learning theory, logic and inference in AI, and information retrieval systems. He is a member of IPSJ.