Supervised Learning of Hidden Markov Models for Sequence Discrimination

Hiroshi Mamitsuka

C&C Research Laboratories, NEC Corporation
4-1-1 Miyazaki, Miyamae-ku, Kawasaki, Kanagawa 216 Japan.
Tel. 81-44-856-2143, Fax. 81-44-856-2235, E-mail address: mami@sbl.cl.nec.co.jp

Abstract

We present two supervised learning algorithms for hidden Markov models (HMMs) for sequence discrimination. When we model a class of sequences with an HMM, conventional learning algorithms for HMMs have trained the HMM with training examples belonging to the class, i.e. positive examples alone, while both of our methods allow us to use negative examples as well as positive examples. One of our algorithms minimizes a kind of distance between a target likelihood of a given training sequence and an actual likelihood of the sequence, which is obtained by a given HMM, using an additive type of parameter updating based on a gradient-descent learning. The other algorithm maximizes a criterion which represents a kind of ratio of the likelihood of a positive example to the likelihood of the total example, using a multiplicative type of parameter updating which is more efficient in actual computation time than the additive type one. We compare our two methods with two conventional methods on a type of cross-validation of actual motif classification experiments. Experimental results show that in terms of the average number of classification errors, our two methods out-perform the two conventional algorithms.

1 Introduction

In the recent computational molecular biology field, hidden Markov models (hereafter referred to as HMMs) have been the most widely-used approach in a number of basic and crucial tasks such as computing profiles or aligning sequences [8]. HMMs are stochastic models suitable for time-series or sequences including deletion, insertion or even iteration, and this feature makes HMMs suitable for analyzing biological sequences in one-dimensional level.

In training the parameters of HMMs, there is a well-known algorithm called "Baum-Welch" [4], which is the most popular method in a variety of fields to which HMMs are applied. This algorithm, which is based on a dynamic programming approach, is a type of EM (Expectation Maximization) algorithm [7] and is extremely efficient in computation time in training an HMM. When given an HMM representing a certain class and the examples belonging to the class, i.e. positive examples, this algorithm trains its parameters to maximize the sum of log-likelihoods of given positive examples. However, molecular biological sequences such as nucleotide or amino acid sequences are obtained by only time- and money-consuming biochemical experiments, and thus the number of positive examples is usually limited, while Baum-Welch requires a number of positive examples to achieve high discrimination accuracy for unknown sequences.

In light of this disadvantage of the Baum-Welch algorithm, a number of approaches have been proposed to improve sensitivity of HMMs. These approaches include a prior-based approach proposed by a UC Santa Cruz group [6] and the learning methods of Krogh et al.[12] and Eddy et al.[9], both of which introduce new criteria for training HMMs using positive examples, instead of the maximum log-likelihood criterion used in Baum-Welch.

In this paper, we present two supervised learning methods for HMMs to improve sequence discrimination sensitivity, both of which enable us to use not only the sequences belonging to a class represented by an HMM but the sequences which do not, i.e. negative examples. One of the two methods smoothly minimizes a kind of error-distance between a target value of the likelihood of a given sequence and a calculated value of that when given a model, using a gradient-descent algorithm which additively updates parameters and is typically used in training feed-forward type neural networks. Thus, this method allows us to use a dataset in which each training sequence has its own real-valued target likelihood which should be output by the model for the sequence. This gradient-descent algorithm enables us to smoothly change parameters, and Baldi and Chauvin [3] also used this additive smooth parameter change to train HMMs using positive examples alone. On the other hand, the other method maximizes a criterion representing a type of ratio of the likelihood of a given positive example to the likelihood of the given total example, using a multiplicative type of parameter change which corresponds to a modification of the Baum-Welch algorithm. Since a multiplicative type of parameter change is used, this algorithm is more efficient in actual computation time than our smooth algorithm or Baldi's smooth one, both of which let the parameters of HMMs change slowly and require a much greater number of iterations in training HMMs. We evaluate our two methods with a type of cross-validation in an actual motif classification problem and compare the results with those of two other methods, i.e. the Baum-Welch algorithm and Baldi's algorithm.

RECOMB 97, Santa Fe New Mexico USA

In our experiments, we used a consensus pattern of sugar transport proteins, which is noted in the PROSITE database [2], and from Swiss-Prot database [1] Release 29.0, obtained 68 partial sequences corresponding to the consensus pattern, of which 49 are regarded as positive examples and 19 negative. We divided this data into two datasets. In training, we trained HMMs using one dataset and in test, using the trained HMMs, tried to classify the other dataset into two classes, i.e. positive and negative. For each learning algorithm and at each size of HMMs, we repeated this experimental procedure twenty-five times with random initial values and with randomly divided datasets, and evaluated the discrimination accuracy of the learning algorithms by the average number of classification errors over the twenty-five trials. Experimental results showed that for HMMs of all sizes, our two methods greatly out-performed the Baum-Welch and Baldi methods in terms of the number of errors generated in classifying unknown sequences. From these results, we conclude that our two methods using negative examples are effective in this type of classification problem for discriminating unknown biological sequences.

2 Hidden Markov Models

In this paper, we consider a first-order HMM which has been typically used for representing biological sequences [11]. The HMM has *states*, at each of which a symbol is generated, and *arcs*, each of which allows a transition between two states. There are two types of probability parameters, i.e, *state transition probabilities* and *symbol output probabilities*, which are attached to arcs and states, respectively. We specify one or more starting and finishing states in an HMM. Possible transitions are made successively from a starting state to a finishing state, and the relevant transition probability and symbol output probability can be multiplied at each transition to calculate the overall *likelihood* of all the outputted symbols produced in the transition path up to that point. When all transitions are finished, the HMM generates a symbol sequence according to the likelihood of each path along which a sequence could have been formed. In other words, when the sequence is given, there are one or more transition paths that could have produced it, each of which would generate the sequence with a specific likelihood that the sequence is in the path. We regard the sum of the likelihoods obtained for all such transition paths as the *likelihood that the sequence was generated by the HMM.*

According to the definition of [14], let the number of states be N, the number of symbols be M, the state transition probability from state i to state j be a_{ij} $(i = 1, \cdots, N, j = 1, \cdots, N)$, and the symbol output probability at state i be $b_i(c)$ $(c = 1, \cdots, M)$. Let the HMM which is defined by the above notation and is used hereafter in this paper be H. The HMM H has the first state from which any transition starts and the last state at which any transition finishes, and does not generate any symbol at these two states.

The probabilities given above must satisfy the following constraints:

$$\sum_j^N a_{ij} = 1, \quad \sum_c^M b_j(c) = 1.$$
$$a_{ij} \geq 0, \quad b_j(c) \geq 0. \qquad (1)$$

Let the length of the $s-$th training sequence O^s be l_s and the t-th symbol of O^s be O_t^s (i.e. $O^s = O_1^s \cdots O_{l_s}^s$).

We define the forward probability $\alpha_t(j)$, which is the probability that the partial sequence $O_1^s \cdots O_t^s$ is generated, and that the state at time t is j. The forward probability can be iteratively calculated as follows:

$$\begin{aligned}
\alpha_t(j) &= \sum_i^N a_{ij} b_j(O_t^s) \alpha_{t-1}(i) \quad (t = 1, \cdots l_s), \\
\alpha_0(j) &= 1, \quad \textit{if } j \textit{ can be the first state, and} \\
\alpha_0(j) &= 0, \quad \textit{if not.} \\
\alpha_{l_s+1}(j) &= \sum_i^N a_{ij} \alpha_{l_s}(i).
\end{aligned}$$

Similarly, we can define the backward probability $\beta_t(i)$, which is the probability that the partial sequence $O_{t+1}^s \cdots O_{l_s}^s$ is generated, and that the state at time t is i. The backward probability also can be iteratively calculated as follows:

$$\begin{aligned}
\beta_t(i) &= \sum_j^N a_{ij} b_j(O_{t+1}^s) \beta_{t+1}(j) \quad (t = l_s - 1, \cdots, 0), \\
\beta_{l_s}(i) &= \sum_j^N a_{ij} \beta_{l_s+1}(j), \\
\beta_{l_s+1}(i) &= 1, \quad \textit{if } i \textit{ can be the last state, and} \\
\beta_{l_s+1}(i) &= 0, \quad \textit{if not.}
\end{aligned}$$

Note that $\sum_i \alpha_{l_s+1}(i)\beta_{l_s+1}(i)$, as well as $\sum_i \alpha_0(i)\beta_0(i)$, corresponds to the likelihood that the given training sequence O^s is generated by the HMM H, i.e. $P(O^s|H)$.

We further define the two probabilities γ and ξ which are used in the description of the learning algorithms of HMMs, as follows:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O^s|H)},$$

$$\begin{aligned}
\xi_t(i,j) &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}^s) \beta_{t+1}(j)}{P(O^s|H)} \quad (t = 0, ..., l_s - 1), \\
\xi_{l_s}(i,j) &= \frac{\alpha_{l_s}(i) a_{ij} \beta_{l_s+1}(j)}{P(O^s|H)}.
\end{aligned}$$

Here $\gamma_t(i)$ corresponds to the probability that the sequence O^s is generated by the model H, and that the state at time t is i. $\xi_t(i,j)$ is the probability that the sequence O^s is generated by the model H, and that the state at time t and $t+1$ are i and j, respectively.

3 Learning Algorithms

In this section, we first show two existing learning algorithms for HMMs, i.e. the Baum-Welch algorithm [4] (hereafter referred to as the BW) and Baldi's smooth algorithm [3] (hereafter referred to as the BA), and next show our two learning algorithms for HMMs for sequence discriminations, i.e. a smooth algorithm [13] (hereafter referred to as the MA) and an efficient algorithm (hereafter referred to as the SEM).

In both the BW and BA, the parameters of an HMM which represents a certain class of sequences are trained to maximize the sum of the log-likelihood of each training sequence belonging to the class. That is, their goal is the maximum likelihood estimation and they use only positive examples as training data.

On the other hand, the MA is an algorithm which minimizes a kind of distance between a given target value of the

likelihood of each training sequence and the actual likelihood of the sequence output by a given model. Thus, this algorithm allows us to use even a dataset in which each sequence has a target value of the likelihood which should be emitted by the trained HMM. The SEM is also an algorithm which uses the data which does not belong to positive examples, i.e. negative examples, but the data dealt with by the SEM comprise only two classes, i.e. negative and positive. The parameter updating rules of the SEM are similar to those of the BW, but the goal of the SEM is not so simple as the maximum likelihood criterion.

3.1 Baum-Welch Algorithm (BW)

We use the the γ and ξ to briefly describe the BW [14] which is the most popular algorithm for training HMMs and updates a_{ij} and $b_j(O_t^s)$. The goal of the algorithm is to maximize the likelihood of the observed sequence O^s when given the model H, that is, $\prod_s P(O^s|H)$.

The probability $\hat{a}_{ij}$ is reestimated using the γ and ξ, as the expected *proportion* of the transitions made from state i that have state j as their destination among all the transitions made from state i. Similarly, $\hat{b}_i(c)$ is also reestimated using the γ, as the expected *proportion* of the symbols generated at state i that are c (as opposed to any other symbols). The equations of these reestimations are shown as follows:

$$\hat{a}_{ij} = \frac{\sum_s \sum_{t=0}^{l_s} \xi_t(i,j)}{\sum_s \sum_{t=0}^{l_s} \gamma_t(i)},$$

$$\hat{b}_i(c) = \frac{\sum_s \sum_{t=1, O_t^s=c}^{l_s} \gamma_t(i)}{\sum_s \sum_{t=1}^{l_s} \gamma_t(i)}.$$

3.2 Baldi's Smooth Algorithm (BA)

As in the BW, the BA[3] aims at maximizing the log-likelihood of the observed sequences, but this algorithm allows us to implement a smooth parameter change, i.e. an additive type of parameter updating, which is different from a multiplicative type of parameter updating in the BW.

The BA first introduces real-valued parameters ω_{ij} and $v_j(c)$, from which a_{ij} and $b_j(c)$ are calculated respectively using the following Boltzman distribution-like equations used in [5].

$$a_{ij} = \frac{e^{\lambda\omega_{ij}}}{\sum_k e^{\lambda\omega_{ik}}}, \quad b_j(c) = \frac{e^{\lambda v_j(c)}}{\sum_k e^{\lambda v_j(k)}}, \tag{2}$$

where λ is a constant. The equations allow ω_{ij} and $v_j(c)$ to be any real value, satisfying the constraints of Equations (1) for a_{ij} and $b_j(c)$.

The BA gradiently changes these real-valued parameters, ω_{ij} and $v_j(c)$, instead of a_{ij} and $b_j(c)$, based on the gradient-descent algorithm which is typically used in training feed-forward type neural networks. We show the smooth updating rules of the BA as follows:

$$\Delta\omega_{ij} = C_a \sum_s \sum_{t=1}^{l_s} (\xi_t(i,j) - a_{ij}\gamma_t(i)),$$
$$(\omega_{ij}^{new} = \omega_{ij}^{old} + \Delta\omega_{ij})$$

$$\Delta v_j(c) = C_b \sum_s \sum_{t=1}^{l_s} (\gamma_t(j)_{O_t^s=c} - b_j(c)\gamma_t(j)),$$
$$(v_j(c)^{new} = v_j(c)^{old} + \Delta v_j(c))$$

where C_a and C_b are constants.

Online versions of these are shown as follows:

$$\Delta\omega_{ij} = C_a \sum_{t=1}^{l_s} (\xi_t(i,j) - a_{ij}\gamma_t(i)),$$

$$\Delta v_j(c) = C_b \sum_{t=1}^{l_s} (\gamma_t(j)_{O_t^s=c} - b_j(c)\gamma_t(j)).$$

3.3 A Smooth Algorithm for Sequence Discrimination (MA)

As mentioned earlier, the MA [13] allows us to use a dataset including training sequences with a variety of their target likelihoods, and this method tries to minimize a kind of error-distance between the target likelihood for each sequence and the actual likelihood of the sequence, which is obtained by a given model. This algorithm also uses an additive type of parameter updating used in the BA. The MA first replaces the probability parameters a_{ij} and $b_j(c)$ with real-valued parameters ω_{ij} and $v_j(c)$, as is done in the BA.

When given the HMM H and n training sequences, let p_s be the likelihood of the $s-$th sequence, i.e. $P(O^s|H)$, and p_s^* be the likelihood which is set as the target value of the likelihood of the $s-$th sequence.

The MA introduces the following d_s and d_{max} :

$$d_s = \log(\frac{p_s^*}{p_s}),$$

$$d_{max} = \log(\frac{p_{max}^*}{p_{min}^*}),$$

where p_{max}^* is the maximum value of p_s^* $(s = 1, ..., n)$ and p_{min}^* is the minimum value of p_s^* $(s = 1, ..., n)$, or d_{max} is fixed as any real number which satisfies $d_{max} > |d_s|$ $(s = 1, ..., n)$ before the first iteration of this algorithm. The goal of this algorithm is that for each training sequence, d_s should be 0 , and thus we train the real-valued parameters, ω_{ij} and $v_j(c)$, so that the error-distance function $\sum_s -\log(\frac{d_{max}^2 - d_s^2}{d_{max}^2})$ is minimized.

In order to achieve this goal, the MA presents the following smooth learning rules for optimizing real-valued parameters, ω_{ij} and $v_j(c)$.

$$\Delta\omega_{ij} = C_a \sum_s \frac{d_s}{(d_{max}^2 - d_s^2)} \sum_{t=1}^{l_s} (\xi_t(i,j) - a_{ij}\gamma_t(i)),$$
$$(\omega_{ij}^{new} = \omega_{ij}^{old} + \Delta\omega_{ij})$$

$$\Delta v_j(c) = C_b \sum_s \frac{d_s}{(d_{max}^2 - d_s^2)} \sum_{t=1}^{l_s} (\gamma_t(j)_{O_t^s=c} - b_j(c)\gamma_t(j)),$$
$$(v_j(c)^{new} = v_j(c)^{old} + \Delta v_j(c))$$

where C_a and C_b are constants.

Online versions of these are shown as follows:

$$\Delta\omega_{ij} = C_a \frac{d_s}{(d_{max}^2 - d_s^2)} \sum_{t=1}^{l_s} (\xi_t(i,j) - a_{ij}\gamma_t(i)),$$

$$\Delta v_j(c) = C_b \frac{d_s}{(d_{max}^2 - d_s^2)} \sum_{t=1}^{l_s} (\gamma_t(j)_{O_t^s=c} - b_j(c)\gamma_t(j)).$$

3.4 An Efficient Algorithm for Sequence Discrimination (SEM)

We show the SEM for sequence discrimination, but this algorithm deals with only two types of examples, i.e. positive and negative examples, while the MA is able to handle the dataset in which each training sequence has any desired likelihood which should be output after training HMM parameters.

Here we assume that $p_s^* = 1$ if the s-th sequence is a positive one; otherwise $p_s^* = 0$. In the SEM, we maximize $\prod_{\{s|p_s^*=1\}} \frac{p_s}{\sum_s p_s}$, instead of maximizing $\prod_{\{s|p_s^*=1\}} p_s$ as is done in the BW and the BA.

Updating rules to satisfy this criterion are shown as follows:

$$\hat{a}_{ij} = \frac{\sum_{\{s|p_s^*=1\}} \xi^s(i,j) - \frac{1}{P}\sum_s p_s \xi^s(i,j)}{\sum_{\{s|p_s^*=1\}} \gamma^s(i) - \frac{1}{P}\sum_s p_s \gamma^s(i)}, \quad (3)$$

where $\xi^s(i,j) = \sum_{t=1}^{T_s-1} \xi_t(i,j)$, $\gamma^s(i) = \sum_{t=1}^{T_s-1} \gamma_t(i)$ and $P = \sum_s p_s$.

$$\hat{b}_j(c) = \frac{\sum_{\{s|p_s^*=1\}} \gamma_c^s(i) - \frac{1}{P}\sum_s p_s \gamma_c^s(i)}{\sum_{\{s|p_s^*=1\}} \gamma^s(i) - \frac{1}{P}\sum_s p_s \gamma^s(i)}, \quad (4)$$

where $\gamma_c^s(i) = \sum_{t=1,O_t=c}^{T_s-1} \gamma_t(i)$. A brief derivation of Equation (3) is shown in Appendix, and Equation(4) is derived similarly. As shown in Equations (3) and (4), the SEM is a kind of modification of the BW, i.e. the maximum likelihood, and is expected to be efficiently calculated in actual applications.

4 Experimental Results

We evaluated the sequence discrimination ability of our two learning algorithms and two conventional algorithms for HMMs with a protein motif classification problem. The HMMs we used here have between seven and fourteen states and are fully-connected HMMs. That is, any state except for the last (finishing) state has links to any state except for the first (starting) state, and the last state has no link. In training HMMs with the BA and MA, both C_a and C_b are set as 1 in the MA and $10^{-1.8}$ in the BA, and p_s^* is set as $(\frac{1}{20})^{0.01\times l_s}$ and $(\frac{1}{20})^{1.99\times l_s}$ for positive and negative examples, respectively. These values are adjusted to achieve high sequence discrimination ability through simple preliminary experiments.

4.1 Sugar Transport Proteins Consensus

We used the "sugar transport proteins (hereafter referred to as STP)" motif [10] in our experiments. Its consensus pattern noted in the PROSITE database[2] is "[LIVMSTA] - [DE] - x - [LIVMFYWA] - G - R - [RK] - x(4,6) - G " and is twelve to fourteen residues long. In the pattern, the highly conserved "G - R - [RK]" motif is said to be a peculiar feature of transmembrane proteins including STP.

From the Swiss-Prot database [1] Release 29.0, we obtain 68 sequences, each of which comprises the consensus pattern. Of the sequences, we use 49 which are actually in the STP sequences as positive examples and 19 which are not as negative examples. It is worth noting that the negative examples are *false positive* examples in terms of the consensus pattern of the STP motif. Concretely, we cannot recognize positive examples from all the 68 sequences using the consensus pattern only, since both positive and negative sequences have the same pattern. Thus, the problem we now try to solve is rather difficult in that we have to separate positive examples from "false positive " examples.

4.2 Cross-Validation

Using the obtained sequences, we conducted a type of cross-validation to evaluate the sequence discrimination ability of the MA and SEM and to compare them with those of the conventional BW and the BA. The procedure of this cross-validation comprises the following three points.

1. Positive and negative examples are randomly divided into two classes, i.e. training and test.
2. In training HMMs, the BW and BA use only positive training examples while the MA and SEM use both positive and negative examples. In testing, using the trained HMM, we distinguish positive test examples from negative ones, based on the likelihood for each test sequence. All four methods use both positive and negative examples in testing.
3. For each training dataset, we repeat the training five times with different random initial values, and we randomly generate the training and test datasets five times.

Furthermore, in testing, in order to compare test sequences having a variety of lengths with each other, we calculate the modified log-likelihood (hereafter referred to as the MLL) of each sequence, which is obtained by dividing the log-likelihood calculated for the sequence by its length. The discrimination ability of trained HMMs is measured by the minimum number of errors (hereafter referred to as MNE) which can be obtained while changing a cut-off value for the MLLs for test sequences, which classifies test examples into two classes, i.e. positive and negative examples.

For four learning methods in this cross-validation, Figure 1 shows the average MNE over all the repeated trials for all sizes of HMMs. In all HMM sizes, the MA reduces the average MNEs of the BW and BA. The SEM also reduces the average MNE of the BW and BA in all cases except two. Both results indicate that our methods using negative examples reduce the average number of errors made by conventional approaches in training HMMs. In particular, the figure shows that the MA which achieves the minimum number of errors in all HMM sizes except one is the best method for this sequence discrimination problem among the four types of methods.

Figure 2 shows the distribution of MNE for HMMs with no less than ten states, for four types of learning methods. This figure much more clearly shows the advantage of the use of negative examples in this problem. That is, the MNEs of the MA and SEM concentrate on only "one" while those

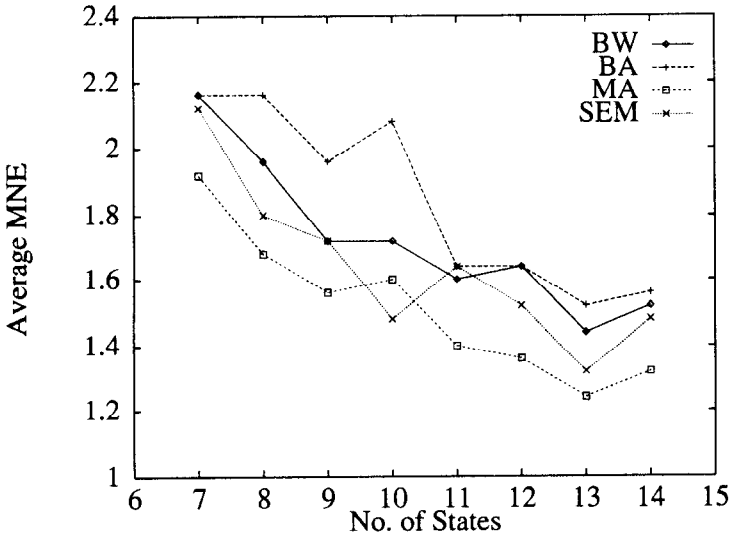

Figure 1: Average MNE for four types of learning methods

	BW	BA	MA	SEM
computation time (sec)	16.8	243.0	85.6	24.2

Table 1: Example of actual computation time for training HMMs

of the BW and BA are distributed over "one to three", and in short, our two methods surpass the BW and BA in this motif classification problem. Totally, from Figures 1 and 2, the MA presents the most favorable result for this problem among the four learning methods. However, in Figure 2, the SEM surpasses the other three methods in the number of HMMs in which the MNE is equal to zero.

For four learning methods in the cross-validation, Table 1 shows the actual computation time in training an HMM on a Silicon Graphics Onyx graphic workstation, as measured under equal conditions. Each is averaged by ten trials using HMMs with seven states. Note that both the MA and SEM deal with negative examples and thus they require longer training time than those of the BA and BW even if the computation time for a single sequence is uniform over the four methods. Further note that the computation time greatly changes in the BA and MA according to the values of the constants in their learning algorithms. As shown in the table, the BW and SEM are faster in training HMMs than the other two adopting smooth algorithms which basically require more iteration than the multiplicative type of iterations done in the BW and SEM.

In this cross-validation test, there are a number of HMMs with only seven states trained by the SEM which actually achieve "zero" error, and we show two examples of them in Figures 3 and 4. In these figures, we draw only the links whose state transition probabilities exceed 0.1 and the symbols whose output probabilities also exceed 0.1.

In Figure 3, the second state from the right corresponds to the two positions fixed at G in the STP consensus pattern, and each of the states except for the first state from the left corresponds to two or more positions in the consensus pattern. In particular, the region "x(4,6)" in STP consensus pattern is modeled by only one state with a self-loop, i.e. the third state from the right. Under this condition, this HMM classifies test examples perfectly. This result indicates that in terms of sequence discrimination sensitivity, the HMM modeling the STP motif can be represented by a rather smaller number of states than the minimum length of the obtained consensus sequences of the STP motif, if appropriate good priors for HMM parameters are given.

In Figure 4, there is also the state outputting G with high probability, and the "G - R - K/R" motif which is said to be peculiar to a transmembrane region is modeled by the two states on the right-hand side of the figure. The region "x(4,6)" in the consensus pattern is modeled by the other three states in the figure, and the result is different from Figure 3. That is, the states in this figure are split into either a particular motif region or another region at which amino acids appear relatively randomly. This result would also support the expectation that with respect to sequence discrimination, an HMM of relatively small size would be sufficient for modeling the STP motif.

5 Conclusion and Discussion

We have presented two learning algorithms, the MA and SEM, for HMMs for sequence discrimination. In experiments, we evaluated these learning methods in comparison with two conventional methods, i.e. the BW and BA. Experimental results show that for any size of HMMs, our two methods reduce the number of discrimination errors made by the two conventional ones. From the experiments, we can conclude that our two methods using negative examples are effective for the motif classification problem shown in this paper.

Each of our two methods has its own merits and demerits, as we described earlier. One advantage of the MA is that it can handle a dataset in which each sequence has a real-valued target likelihood while the SEM deals with only two types of data, i.e. positive and negative. Another advantage is that the MA makes fewer discrimination errors than the SEM (Figures 1 and 2). On the other hand, the SEM is more efficient in terms of computation time for training HMMs, as shown in Table 1. From this comparison, we might say that the MA is the best method for sequence discrimination so long as there is no strong time constraint.

In Figure 1, the difference in the average MNE between

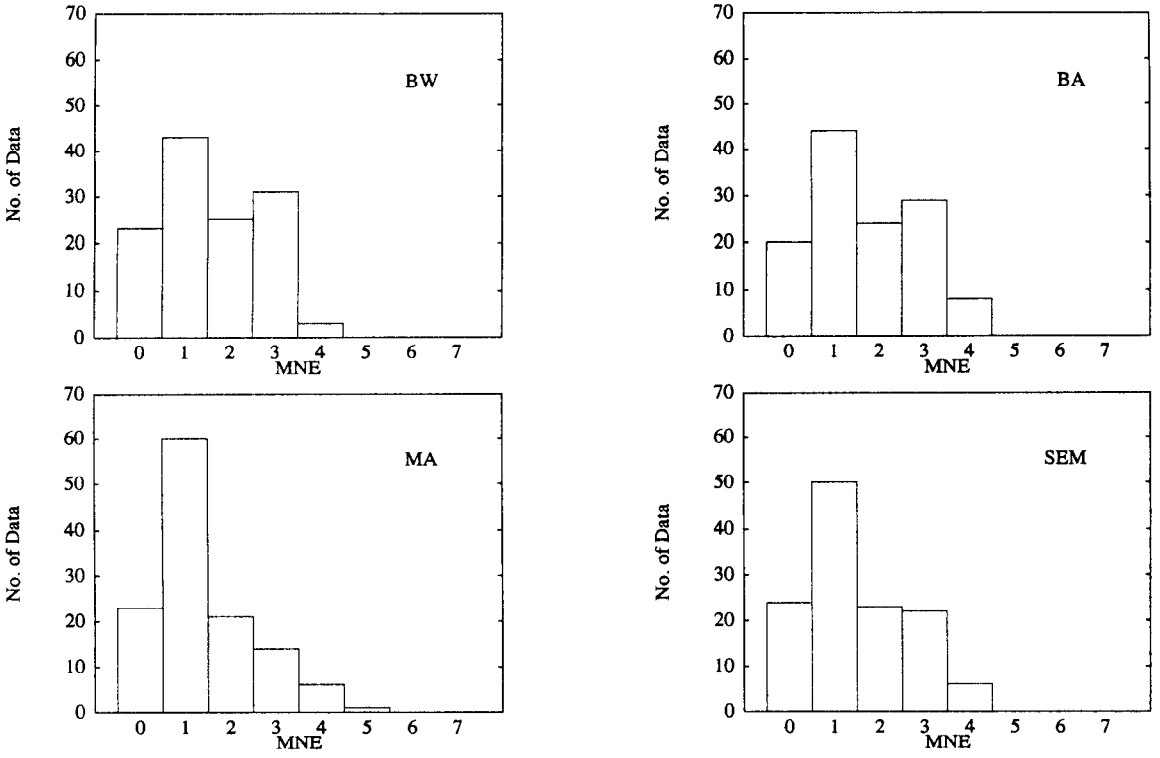

Figure 2: Error distributions of four learning methods

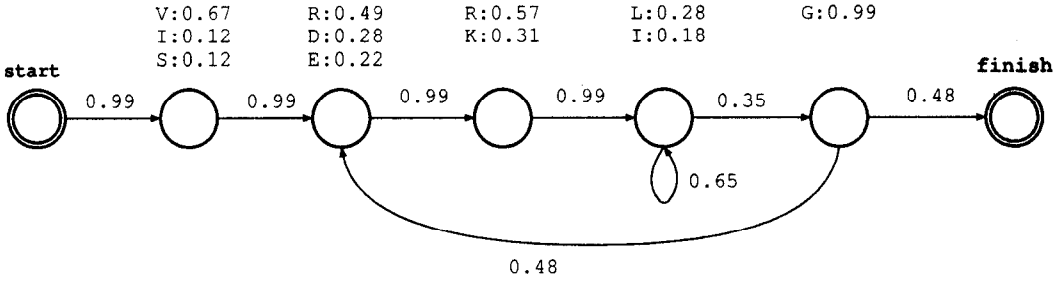

Figure 3: An example of trained HMM

our methods and the others might appear to be small. Actually, even the maximum margin of the average MNE between the best of our two methods and the worst of the other two ones, which is obtained in ten-state HMMs, is no more than 1.0. This result indicates that it is rather easy to discriminate the dataset used here into two classes and that the BW and BA also achieve a relatively small number of errors. Another possible reason for this result lies in the small number of data used here. That is, due to the need for cross-validation in the test, we used only 34 of the 68 sequences obtained. If we were to use a somewhat larger dataset comprising the sequences which cannot easily be separated into two classes, a greater difference in the average MNE between our methods and the others would be seen.

As in our experiments, when we discriminate only two types of examples, i.e. positive and negative, practically we can consider a number of criteria including $\prod_{\{s|p_s^*=1\}} \frac{p_s}{\sum_s p_s}$ used in the SEM, each of which should be maximized and each of which represents a kind of proportion of the likelihood of a positive example to the likelihood of the total example. As in the SEM, we would be able to derive for all the criteria a similar type of updating rule corresponding to a modification of the BW, i.e. a multiplicative type of parameter change. However, using negative examples in this way cannot be applied to a dataset in which each example has its own target likelihood, which dataset can be handled by the MA. To train HMMs with such a type of dataset, we currently have to use an additive type of parameter updating as in the MA, and this requires more computation time than the use of multiplicative updating learning methods. Thus, one possible subject for future work is to build an efficient multiplicative type of algorithm for HMMs, which would allow us to deal with a dataset in which each training sequence has its own target likelihood to be output by a trained model.

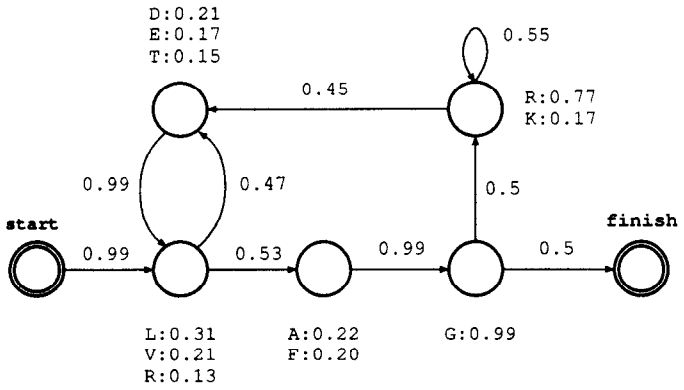

Figure 4: An example of trained HMM

Appendix

We define a log-likelihood L as follows, and the SEM tries to maximize the L.

$$L = \log\left(\frac{\prod_{\{s|p_s^*=1\}} p_s}{\sum_s p_s}\right) = \sum_{\{s|p_s^*=1\}} \log p_s - \log \sum_s p_s$$

In this algorithm, we search for $\hat{a}_{ij}$ which maximizes the L, that is, $\hat{a}_{ij}$ satisfying $L' = 0$. We use Equations (2) to derive the following differentiation :

$$L' = \sum_{\{s|p_s^*=1\}} \frac{p_s^{*\prime}}{p_s^*} - \frac{\sum_s p_s'}{\sum_s p_s} = 0$$

Here, $p_s' = \frac{\partial p_s}{\partial \omega_{ij}}$ and we calculate two terms of the above equation and derive the following two equations :

$$\sum_{\{s|p_s^*=1\}} \frac{p_s^{*\prime}}{p_s^*} = \sum_{\{s|p_s^*=1\}} \sum_{t=1}^{T_s-1} (\xi_t(i,j) - \hat{a}_{ij}\gamma_t(i))$$

$$\frac{\sum_s p_s'}{\sum_s p_s} = = \frac{\sum_s p_s \sum_{t=1}^{T_s-1} (\xi_t(i,j) - \hat{a}_{ij}\gamma_t(i))}{\sum_s p_s}$$

Using both equations, we obtain the $\hat{a}_{ij}$ as in Equation (3).

References

[1] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence data bank and its new supplement TrEMBL. *Nucl. Acids. Res.*, 24:21–25, 1996.

[2] A. Bairoch, P. Bucher, and K. Hofmann. The PROSITE database, its status in 1995. *Nucl. Acids. Res.*, 24:189–196, 1996.

[3] P. Baldi and Y. Chauvin. Smooth on-line learning algorithms for hidden Markov models. *Neural Comp.*, 6:307–318, 1994.

[4] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[5] J. S. Bridle. Alpha-nets: A recurrent 'neural' network architecture with a hidden Markov model interpretation. *Speech Comm.*, 9:83–92, 1990.

[6] M. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using dirichlet mixture priors to derive hidden Markov models for protein families. In *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology (ISMB-93)*, pages 47–55, Menlo Park, CA, 1993. AAAI press.

[7] A. P. Dempster, N.M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm(with discussion). *J. Roy. Statist. Soc.*, Ser B 39:1–38, 1977.

[8] S. R. Eddy. Hidden Markov models. *Curr. Opin. Struct. Biol.*, 6:361–365, 1996.

[9] S. R. Eddy, G. Mitchison, and R. Durbin. Maximum discrimination hidden Markov models of sequence consensus. *J. Comput. Biol.*, 2(1):9–24, 1995.

[10] G. W. Gould and G. I. Bell. Facilitative glucose transporters: an expanding familiy. *Trends Biochem. Sci.*, 15:18–23, 1990.

[11] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology. Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[12] A. Krogh and G. Mitchison. Maximum entopy weighting of aligned sequences of proteins or DNA. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology (ISMB-95)*, pages 215–221, Menlo Park, CA, 1995. AAAI press.

[13] H. Mamitsuka. A learning method of hidden Markov models for sequence discrimination. *J. Comput. Biol.*, 3(3), 1996. To appear.

[14] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.