Fast Discrete HMM Algorithm for On-line Handwriting Recognition

T.Hasegawa, H.Yasuda, and T.Matsumoto
Department of Electrical, Electronics, and Computer Engineering
Waseda University
takashi@mse.waseda.ac.jp

Abstract

A fast Discrete HMM algorithm is proposed for on-line hand written character recognition. After preprocessing input stroke are discretized so that a discrete HMM is used. This particular discretization naturally leads to a simple procedure for assigning initial state and state transition probabilities. In the training phase, complete marginelization with respect to state is not performed.

A criterion based on normalized maximum likelihood ratio is given for deciding when to create a new model for the same character in the learning phase, in order to cope with stroke order variations and large shape variations.

Experiments are done on the Kuchibue database from TUAT. The algorithm appears to be very robust against stroke number variations and have reasonable robustness against stroke order variations and large shape variations.

A drawback of the proposed algorithm is its memory requirement when the number of character classes and their associated models becomes large.

1. Introduction

Recent developments in pen input devices including PDA are calling for good on-line handwriting recognition algorithms.

This is due to the fact that as machines get smaller, keyboards become more difficult to use. Pen input interface is also preferred by those who are not familiar with keyboards.

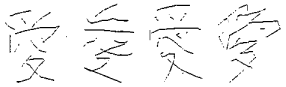
An important distinction between on-line and off-line handwriting recognition problems is the fact that spatio-temporal information is available in the former, while only spatial information is available in the latter. There are two great challenges in on-line hand writing recognition problems;

i) Stroke number variations, stroke connections and shape variations.

A Kanji character is composed of up to 30 strokes. However, for casual writing characters, many writers tend to connect and abbreviate the strokes of a Kanji. The examples in Fig 1 illustrate this problem. Fig.1 (a) shows Kanji "愛" ("love") which consists of 13 strokes when written in a proper manner. The examples from Fig.1 (b)-(d) show the same character written in a casual manner.

ii) Stroke order variations.

Fig.2 shows Kanji character "飛" ("flight") written in three different stroke orders. The left most column data



(a) (b) (c) (d)

Fig.1 Examples of Kanji characters

shows the first stroke, the second column shows the second stroke. Up to this point, the three data sets have the same stroke orders. Significant stroke order variations emerge in the third column on. The final completed "飛" is shown in the right most columns.

Requirements for overcoming i) and ii) are often

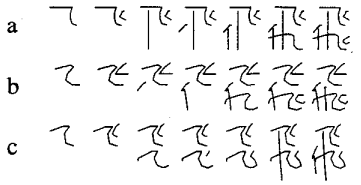


Fig.2 Examples of different stroke orders

contradictory. For instance, if one discards temporal information, then one may be able to overcome stroke order variation problem.

However, such an algorithm often suffers from its inability to cope with stroke number variations. On the other hand, an algorithm, which copes with stroke number variations often, performs unsatisfactory on stroke order variations.

This paper proposes a fast learning / recognition algorithm using a class of Discrete HMM tailored for on-line handwritten character recognition. Then, reports several preliminary recognition experiments against several data sets containing Japanese Kanji characters (JIS first category), Katakana and Hiragana from the Kuchibue database [1]. Many of the characters are written in an extremely casual manner. The average recognition rate

0-7695-0750-6/00 $10.00 © 2000 IEEE

was 87.80% while the top fifth recognition rate was 98.19%.

2. Preprocessing

2.1. Quantization

Typical raw data taken from digitizer, is

$$(x_1(t_i), x_2(t_i), p(t_i)) \in \mathbf{R}^2 \times \{0,1\}, i = 0,1,\ldots,M \quad (1)$$

where $(x_1(t_i), x_2(t_i))$ is the pen position whereas is pen up / down information. The sampling rate is approximately 10 points/sec.

In order to formulate the problem in terms of discrete HMM, we quantize the data. Consider

$$V_1(t) := v_{1k} \in \{1,2\}, k = 1,2 \quad (2)$$

$$V_2(t) := v_{1l} \in \{1,2,\ldots,L\}, l = 1,2,\ldots,L \quad (3)$$

where $V_1(t)$ represents the pen up / pen down information which is already quantized, and $V_2(t)$ represents the quantized angle information which results from the following scheme. Let

$$\theta := \tan^{-1}\frac{x_2(t)-x_2(t-1)}{x_1(t)-x_1(t-1)}, -\frac{\pi}{2} < \theta < \frac{\pi}{2} \quad (4)$$

i) If $x_1(t)-x_1(t-1) \geq 0$, then $V_2(t) = \theta$

ii) If $x_1(t)-x_1(t-1) < 0$, then

$$V_2(t) := \begin{cases} \pi-\theta, x_2(t)-x_2(t-1) \geq 0 \\ \theta-\pi, x_2(t)-x_2(t-1) < 0 \end{cases} \quad (5)$$

Simplified notation t is used instead of t_i. Therefore, at each time t, there are two symbols that $V_1(t)$ can take, whereas there are L symbols that $V_2(t)$ can take.

3. The Discrete HMM

HMM is a general probabilistic structure which is applicable to a broad class of problems where time evolution is important.

Every general discipline must be tailored before being applied to a specific type of problem, which is a basic engineering function, and HMM is no exception. The general framework of HMM must be carefully tuned to the on-line handwritten character recognition problem. This section gives the complete details of the proposed Discrete HMM structure.

3.1. Discrete HMM Structure for Online Handwriting Recognition

In order to make an HMM precise, let us first recall the output symbols $(v_{1k}, v_{2l}), k = 1,2, l = 1,\ldots,L$ defined in the previous section, and let

$$O(t) := (V_1(t), V_2(t)), t = 1,\ldots,T \quad (6)$$

be observed output sequence.

An HMM of a character

$$\mathsf{H} = \mathsf{H}(\{a_{ij}\}, \{b^1_{ik}\}, \{b^2_{il}\}, \pi, N) \quad (7)$$

is defined by the joint distribution of $\{Q(t), O(t)\}_{t=1}^T$ given H;

$$P(\{Q(t), O(t)\}_{t=1}^T \mid \mathsf{H}) = \pi_{Q(1)} \prod_{t=1}^{T-1} a_{Q(t+1)Q(t)} \prod_{t=1}^{T} b^1_{Q(t)V_1(t)} b^2_{Q(t)V_2(t)} \quad (8)$$

Where $\{Q(t)\}$ stands for hidden state at time t, $\{a_{ij}\}$ state transition probability, b^1_{ik}, b^2_{il} output emission probabilities, and π initial state probability.

Learning in HMM amounts to an estimation of parameters $\{a_{ij}\}, \{b^1_{ik}\}, \{b^2_{il}\}, \{\pi_i\}$ and N, whereas *recognition* is deciding from which character $\{O(t)\}_{t=1}^T$ comes.

In order to tune HMM to our current type of problem, we will use **the left to right model** so we put the following constraints on $\{a_{ij}\}$ and π_i:

i) a. $a_{ij} = 0$ unless $i = j$ or $i = j+1, a_{NN} = 1$

b. $a_{NN} = 1$, i.e., the $\{a_{ij}\}$ matrix is restricted to the one in the form

$$\{a_{ij}\} = \begin{bmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & 0 & \\ & \ddots & \ddots & & \\ & & a_{N-1,N-2} & a_{N-1,N-1} & \\ & 0 & & a_{N,N-1} & 0 \end{bmatrix} \quad (9)$$

ii) $\pi = (1,0,\ldots,0)$ (10)

Note that i)-a indicates that the last state, q_N, cannot become any other state and ii) demands that the initial state $Q(1)$ must be always q_1. Observe that constraint i)-a demands that state q_i cannot jump to $q_{j+k}, k \geq 2$, which corresponds to the causality of a trajectory associated with H. The reasons for these constraints demands will become clear when the learning and recognition algorithms are explained.

3.2 Recognition

Learning and *recognition* are closely related in HMM. We will explain our recognition algorithm first. The proposed recognition scheme decides that

$$\arg\max_{\mathsf{H}} P(O(t)_{t=1}^T, Q(T) = q_N \mid \mathsf{H}) \quad (11)$$

is the most probable character. To explain the reason for using the constraint $Q(T)=q_N$ instead of maximizing marginalized likelihood $P\left(O(t)_{t=1}^{T} \mid \mathsf{H}\right)$ let us consider the two Kanji characters "品"("object") and "口"("mouth"). Suppose that a writer meant "口'", from which $O("口")(t)_{t=1}^{T}$ is obtained. Since "口" is a subset of "品", one of

$$P\left(\{O("口")(t)\}_{t=1}^{T}, Q(T)=q_i \mid \mathsf{H}("品")\right) \tag{12}$$

can be large so that the marginalization

$$P\left(O\{("口")(t)\}_{t=1}^{T} \mid \mathsf{H}("品")\right)= \sum_{i=1}^{N} P\left(O\{("口")(t)\}_{t=1}^{T}, Q(T)=q_i \mid \mathsf{H}("品")\right) \tag{13}$$

can be large which can often cause erroneous recognition results.

Equation (11) forces $Q(T)$ to be the final state q_N of $\mathsf{H}("品")$, which significantly reduces erroneous recognition. Similar reasoning applies to other cases including "林" and "森".

3.3 Learning

The well-known Baum-Welch learning algorithm uses a gradient search to compute

$$\arg\max_{\mathsf{H}} P\left(O(t)_{t=1}^{T}, Q(T)=q_N \mid \mathsf{H}\right) \tag{14}$$

with N fixed. The efficiency of this method naturally depends on each problem. In the current on-line character recognition problem,there are two major hurdles -

i) The likelihood (14) is typically non-convex with respect to the parameters, and the serious problems are created by the local minima.

ii) The dimensions of the parameter space are determined by the sum of the non-zero parameters of $\{a_{ij}\},\{b_{ik}^1\},\{b_{il}^2\}$ and, $\{\pi_i\}$ which amounts to $2(N-1)+N(L+2)+N$; the computational effort is significant.

Our learning algorithm given below is fast because it is non-iterative and simple, and, most importantly, works well. Given training data sets $\{O_c(t)\}_{t=1}^{T_c}, c=1,\ldots,C$, our algorithm starts with generating first model from $\{O_1(t)\}_{t=1}^{T_1}$.

3.3.1. Generation of the First Model When the first data set is given for learning, our proposed algorithm first attempts to associate a clear meaning to the states, so that they naturally leads to our learning algorithm. The algorithm, however, is still HMM in that it has nontrivial $\{b_{ik}^1\},\{b_{il}^2\}$.

Let the first data set $\{O_1(t)\}_{t=1}^{T_1}$ be given,

3.3.2. Multiple Training Data Set. Let $\{O_c(t)\}_{t=2}^{T_c}, c=2,\ldots,C$ be the rest of the data sets for training.

Step 1 State Clarification

Given the first data set

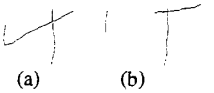
$$\{O_c(t)\}, 1\le c\le C, 1\le t\le T_c \tag{15}$$

make a division between $O_1(t)$ and $O_1(t+1)$ if

1. $V_1(t)\neq V_1(t+1)$, i.e., if the pen up / down information changes; or
2. $|V_2(t)-V_2(t+1)|>\theta_0$, i.e., if the angle variation exceeds threshold, where $\theta_0>0$ is an empirical value;

and then associate the state q_i between the two successive observations

This data also assign the state. Suppose that the first data is as shown in Fig.3-a and the new data is as in Fig.3-b. Because of the stroke connection in Fig.3-b which is not present in Fig.3-a, the two data sets result is different number of states, which gives rise to a difficulty in learning. Our next step in the learning scheme is to use the Viterbi algorithm to make appropriate correspondence between models with different number of state.



(a) (b)

Fig.3 Examples of training data with different stroke numbers.

3.3.3. Model Generation So far, our algorithm has presumed that the number of H coincides with the

Step 2 Most Probable State Transition $\{Q_c(t)\}$

For $\{O_c(t)\}_{t=2}^{T_c}, c=2,\ldots,C$ let

$$Q_c(T_c):=\arg\max_{q_i} P\left(O_c(\tau)_{\tau=1}^{T_c}, q(T_c)=q_i \mid \mathsf{H}\right) \tag{16}$$

$$Q_c(t-1):=\arg\max_{q_i} P(\{O_c(\tau)\}_{\tau=1}^{T}, q(t-1)=q_i, q_t=Q_c(t)\mid \mathsf{H}) \tag{17}$$

and compute $n(O_c,q_i)$, $n(O_c,q_i,v_{ik})$ and $n(O_c,q_i,v_{2l})$, $c=2,\ldots,C$ as in Step 1.2

number of different characters. This must be modified for the following reasons:

i) A character may be written in different stroke orders;

ii. A character may have significantly different shape variations.

This calls for a model generation procedure. It is important that the model generation procedure be automatic. In the following step, we propose a new model generation criterion, normalized likelihood ratio which works well.

Step 3 Model Generation

Let $\mathsf{H}\left(\{a_{ij}\},\{b_{ik}{}^{1}\},\{b_{il}{}^{2}\},\{\pi_i\},N\right)$ be the HMM obtained by the previous steps. Let $\{O_c(t)\}_{t=1}^{T_c}$ be another training set for the same character. If the normalized log likelihood ration exceeds a threshold;

$$\frac{-\log P\left(\{O_1(t)\}_{t=1}^{T_1}, Q(T_1)=q_N(\mathsf{H}_1)\,|\,\mathsf{H}\right)/T_1}{-\log P\left(\{O_c(t)\}_{t=1}^{T_c}, Q(T_c)=q_N(\mathsf{H}_c)\,|\,\mathsf{H}\right)/T_c} > r_{th} \qquad (18)$$

then create a new model H_2 based on $\{O_c(t)\}_{t=1}^{T_c}$, using the previous steps, where $r_{th}>0$ is an empirical value.

4. Experiment

Database Kuchibue [1] contains Kanji, Hiragana, Katakana, Western alphabets, numerals and symbols. In our experiment, we used 2965 Kanji classes (JIS First Level), 46 Hiragana classes, and 46 Katakana classes. Each data set consists of 10502 characters where 5643 are Kanji, 4372 are Hiragana, and 487 are Katakana. Of the 120 data sets, Kuchibue 21 to 120 (100 data sets) were used for learning with the following parameter (see (18)): $r_{th}=0.65$.

Our algorithm described in the previous section created 5224 models (Kanji 5059, Hiragana 72, Katakana 93). Recognition experiment was performed against Kuchibue 1 - Kuchibue 20 (20 data sets). Table (1) shows average recognition rates. Table (2) shows the numbers of the models created. The average recognition time was 0.20 sec/characters (Pentium2 400MHz, WindowsNT4).

Fig.4-a shows several examples of Kanji characters that were correctly recognized and Fig4-b gives examples of erroneously classified characters.

Table (1) Recognition Rate

Data	Recognition rate [%]	Top five recognition rate [%]
Kanji	91.53	97.61
Hiragana	84.04	98.76
Katakana	76.38	98.70
Overall	87.80	98.19

Table (2) Generated Models

Character category	Number of classes	Number of generated models
Kanji	2898	5059
Hiragana	66	72
Katakana	67	96
Overall	3105	5224

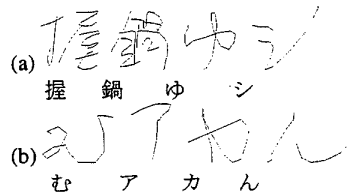


Fig.4 Examples of characters which were correctly / not correctly recognized

5. Discussion

There are significant variations of recognition rates among the individual Kuchibue data sets (Table (1)). This appears to be attributable to the fact that this database contains great varieties of casualness in writing those characters.

When Kanji, Hiragana, Katakana and others are all contained, some of the casually written characters are difficult to classify, e.g., Katakana "エ" and Kanji "工". The proposed algorithm appears to be extremely robust against stroke number variations while maintaining reasonable robustness against stroke order variations.

Since the top fifth recognition rate is 98.19%, a significant improvement recognition performance may be possible by exploiting grammatical structure associated with character sequence instead of individual characters. This is one of the challenging future projects.

Another future project will be adaptation of the algorithm to small portable devices where computational power and memory size are severely limited.

[1] N.Nakagawa (1996). "TUAT Nakagawa Lab. HANDS-kuchibue_d-97-06," Tokyo University of Agriculture and Technology.

[2] L.R.Rabiner. (1996). "A Tutorial on Hidden Markov Models and Selected Application in Speech Recognition,", Proc. IEEE, vol. 77, pp.257-285.

[3] Waseda University. (1996). "A Fast HMM Method for On-Line Handwriting Recognition." Japanese Patent, no.319496

[4] K.Takahashi, H.Yasuda and T.Matsumoto. (1997). "A Fast HMM Algorithm for On-line Handwritten Character Recognition.," 4th ICDAR97, Proceedings Vol.1, pp.369-375.

[5] R.Koehle, T.Matsumoto. (1997). "Pruning Algorithms for HMM On-line Handwriting Recognition," Technical Report of the IEICE, PRMU97-5, pp33-39.