Enhancing Efficiency and Speed of an Off-line Classifier Employed for On-line Handwriting Recognition of a Large Character Set

Ondrej Velek, Masaki Nakagawa
Graduate School of Technology, Tokyo University of Agriculture and Technology
2-24-16 Naka-cho Koganei-shi, Tokyo, 184-8588, Japan
E-mail: velek@hands.ei.tuat.ac.jp, nakagawa@cc.tuat.ac.jp

Abstract

This paper proposes a new approach to accelerating speed and increasing the recognition rate of an off-line recognizer employed for on-line handwriting recognition of Japanese characters. All training patterns are divided according their stroke number to several groups and one single recognizer is dedicated for each group of patterns. Since a number of categories for a single recognizer is smaller, the speed and accuracy improves. First, we make the model of a recognizer and show that our method can theoretically accelerate its recognition speed to 45% of the original time. Then, we employ the method to a practically used off-line recognizer with the result that the recognition rate is increased from 90.73% to 91.60% and the recognition time is reduced to only 49.73% of the original one. Another benefit of our new approach is high scalability so that the recognizer can be optimized for speed and size or for the best accuracy.

1. Introduction

The field of handwritten character recognition can be split into two distinct areas: on-line and off-line recognition. While on-line recognition exploits the dynamic information from writing process, off-line recognition uses only the final character image. Since both on-line and off-line approaches have their advantages and disadvantages, combining the advantages of both approaches is very promising.

On-line recognizers can exploit the dynamic information to recognize cursive handwriting but they are sensitive to incorrect stroke order caused by wrong writing habits, duplicated or added strokes, etc. On the other hand, off-line recognizers are free from these problems but weak to heavily distorted patterns. Moreover, their cumulative recognition rates are often higher but the top recognition rates are lower than on-line methods. Thus, employing off-line methods or combining them with on-line methods for on-line recognition has been sought either during pre-processing; i.e. feature computation [1]-[3], or in post-processing [7]-[6].

We have also proposed a new approach to combining multiple recognizers and applied it to combine on-line/off-line recognizers for on-line handwriting recognition of Japanese characters. We have shown that the combined recognizer can perform much better than either on-line or off-line recognizers by more than 5% from about 90%. In addition, its cumulative recognition rate is even higher than the rate of off-line recognizers. Thus, the combination has taken the advantages and eliminated the disadvantages of their components. Only the problem has remained in larger time and memory consumption by off-line recognizers.

In this work we will speed up the off-line recognizer by utilizing information about a stroke number while considering its wide distributions in real patterns. The uniqueness of our approach and the difference to other methods is the positive effect of our acceleration to the improvement of recognition rates. Our off-line recognizer employed for this research is based on the latest technologies for Japanese or Chinese character recognition so that the method and result described here will have enough generality for other off-line recognizers. Moreover, the method can be adjusted to satisfy different requirements on time and space complexities as well as those on required recognition rates.

Section 2 of this paper defines a theoretical background and Section 3 presents a statistics on the number of strokes to write Japanese characters. In Section 4, we present our idea to utilize the stroke number to speed up the off-line recognition and increase its recognition rates. Section 5 proposes a more concrete model. Section 6 introduces the off-line recognizer used later in experiments described in Section 7. Section 8 summarizes our results and discusses future work.

2. Complexity of a generalized classifier

Let’s consider a generalized classifier based on a nearest neighbor (NN) classification scheme. A classifier evaluates a distance from unknown input patterns \( N \) to all template patterns representing all classes. If a number of all classes is \( c \), each class is represented by \( p \) template patterns and \( d \) is a function for calculating distance, the time for classifying \( N \) patterns is:

\[
t = d(N \times c \times p)
\]

The number \( c \) of the possible classes for western alphabets is about one hundred, but for Chinese and Japanese characters this number is several thousands.
Thus, a classification process is usually speeded up by being divided into two steps: pre-classification selects several class candidates from all possible classes (usually up to 10%) using a simple, but fast distance evaluating function. Precise-final classification employs a more complex and often slower distance function, but only for classes selected in the pre-classification. If a number of classes selected in pre-classification is $c_2$, $c_2 < c$; the time for classifying test patterns is:

$$t = d_{\text{pre}}(N \ast c \ast p) + d_{\text{final}}(N \ast c_2 \ast p) \quad (2)$$

The most common approach to accelerate the recognition process is to develop faster classification methods[9] (faster $d_{\text{pre}}$ and $d_{\text{final}}$ in Equation 2). In this paper, however, we will consider the methods that are independent on the classification functions, methods that speed up recognition by reducing iteration of calculating distance.

The easiest way is to reduce a number of template patterns $p$ representing one class, but lower precision of a class representation leads to lower recognition rates. Identically, reducing a number of candidates for final classification $c_2$ also speed up classification for a penalty of lower recognition rates.

Another approach is to speed up classification by arranging template vectors representing classes during training so that an unknown pattern can be compared only with near template vectors during recognizing it [10]. Also, however, this approach reduces accuracy to some extent.

Generally speaking, we have to understand that each pre-classification itself slightly reduces the recognition rate (if pre-classification is not employed, i.e., final classification is applied to all the classes, a better recognition rate would be realized), but pre-classification allows us to use more efficient (but also much more time consuming) methods for final classification, what’s its great benefit.

In this paper we will introduce a different approach: we will accelerate the speed of classification and in the same time we will increase the rate of recognition. Our time acceleration is also based on the reduction of a number of classes $c$, but we will not degrade accuracy, because we will utilize on-line information about a stroke number in the process of reducing a number of classes $c$.

3. Stroke number of Japanese characters

A number of strokes is an integral feature of Japanese Kanji characters as well as of the original Chinese characters. Each Kanji character has an official writing order and a correct number of strokes. For Japanese Kanji characters, a number of strokes can be from 1 for the simplest characters: (一, 亅) to 30 for the most complex characters: (鸗, 鸱). In the most native case when characters are written by the correct number of strokes, all Kanji characters would be separable to 30 disjunctive classes.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{The statistics for variations on a stroke number of Japanese characters in Nakayosi database.}
\end{figure}

In real life, however, people are very often writing freely, characters are simplified and strokes are connected. In this case, a number of strokes is often smaller than should be. Less frequent cases are characters written by more strokes than is correct, it’s usually done when a character is uncommon and writers are unsure about a writing style. An interesting statistics on character patterns from the Nakayosi database [15] is shown in [12]. In Figure 1 we can see a graph of the correct stroke numbers, variations around this correct stroke numbers and also an average stroke numbers, which lies below a correct stroke number line because strokes are often connected as explained above.

We will utilize this characteristic of Japanese characters and for each group of characters written by the same number of strokes we use a specialized classifier. Since each classifier is devoted for a small number of categories, it will be faster and more accurate.

4. Complexity of modified classifier

If characters are always written by correct numbers of strokes, we can employ 30 specialized classifiers. Template vectors will occupy the same space and computation time will be about 30 times faster¹:

$$t = d(N \ast \frac{c}{30} \ast p) = \frac{1}{30} d(N \ast c \ast p) \quad (3)$$

As we know from Figure 1, however, characters are often written by incorrect numbers of strokes. In Figure 2 we see how many characters are written by $n$ strokes and in Figure 3 how many different categories are written by $n$ strokes, both figures are made for two databases Nakayosi and Kuchibue, which mostly consist of text from a

¹ This equation is correct only if diversification of characters to all groups is the same, but this approximation is good enough for statistically big samples of characters. The exact equation depends also on testing and training database and the exact form for Kuchibue/Nakayosi databases will be introduced later in Section 5-6.
Japanese newspaper. We see that about 20% of all characters are written by one stroke, because simplest characters (Japanese phonetic Kana alphabet) are most often appearing in regular text. We also see from Figure 3 that only a small portion of categories are written by a small or big stroke numbers (about 10-20%), while about 60-70% categories represent characters written by 8-11 strokes.

![Figure 2: A number of characters written by n strokes in Nakayosi and Kuchibue databases.](image1)

![Figure 3: A number of categories written by n strokes in Nakayosi and Kuchibue databases.](image2)

We have investigated an overlapping coefficient, (= in how many stroke number variations is written one Kanji character). This coefficient for Nakayosi database is 7.2. This will decrease an optimal 30-times acceleration (from the equation 3) to about 4 times (the equation 4).

\[ t = d(7.2 \ast N \ast C \ast p) = 0.24 \ast d(N \ast C \ast p) \quad (4) \]

5. Modified classifier after quantization

In this section we will reduce high overlapping of groups by merging groups of a small number of samples (=with a rare stroke numbers). Based on Figure 2 and for the Nakayosi database we will merge groups so that a number of training samples in each group will not be bellow 10% of all the samples. Quantization to seven groups is shown in Table 1, where we see also a number of categories in each group.

**Table 1. Quantization of 1-30 strokes character patterns to 7 groups.**

<table>
<thead>
<tr>
<th>group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4-5</th>
<th>6-8</th>
<th>9-11</th>
<th>&gt;=12</th>
</tr>
</thead>
<tbody>
<tr>
<td>#strokes</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4-5</td>
<td>6-8</td>
<td>9-11</td>
<td>&gt;=12</td>
</tr>
<tr>
<td>#characters</td>
<td>290,804</td>
<td>213,515</td>
<td>208,304</td>
<td>199,217</td>
<td>227,057</td>
<td>192,071</td>
<td>188,009</td>
</tr>
<tr>
<td>#categories</td>
<td>410</td>
<td>802</td>
<td>1,104</td>
<td>1,938</td>
<td>2,592</td>
<td>2,350</td>
<td>1,631</td>
</tr>
</tbody>
</table>

The new overlapping coefficient after quantization is 3.2, the updated complexity equation for this coefficient is bellow (5).

\[ t = d(3.2 \ast N \ast C \ast p) = 0.45 \ast d(N \ast C \ast p) \quad (5) \]

The speed up is to 45% of the original time. It is smaller acceleration than without quantization (24% in section 4), but in our training database we don’t have enough samples for dividing them into all 30 categories. And another benefit of quantization is much lower overlapping coefficient, what leads to smaller size of a training file with template vectors.

We know from Footnote 1 in Section 4 that Equation 3 and also all its derivations Equation 4 and 5 are only approximation. Since a number of categories in each group is different (Table 1) the exact form of Equation 5 for Nakayosi as training and Kuchibue as testing database is Equation 6. The classifier from this section recognizing Kuchibue database needs evaluate 2,063,517,942 times a distance function, while the original recognizer from Section 2 has to do it 4,817,336,640. It’s acceleration to 42.83% of the original time. It also shows that our simplification in Equation 3-5 is sufficiently accurate.

\[ t = d(p \ast \sum_{i=4..7} N_i \ast c_i) = \frac{2063517942}{4817336640} \ast d(N \ast c \ast p) = 0.43 \ast d(N \ast c \ast p) \quad (6) \]

In next sections we apply our approach to a real off-line recognizer.

6. Off-line recognizer

The off-line classifier represents each character as a 256-dimensional feature vector. It scales every input pattern to a 64x64 grid by non-linear normalization [13] and smoothes it by a connectivity-preserving procedure. Then, it decomposes the normalized image into 4 contour sub-patterns, one for each of 4 main orientations. Finally, it extracts a 64-dimensional feature vector for each contour pattern from the convolution with a blurring mask (Gaussian filter).

A pre-classification step precedes the actual final classification. Pre-classification selects the 50 candidates with the shortest Euclidian distances between the categories’ mean vectors and the test pattern.
The final classification employs a modified quadratic discriminant function (MQDF2) developed by Kimura [14] from traditional QDF.

The original classifier is trained with 1,519,077 patterns from Nakayosi database [15]. Categories are the same as in Kuchibue database, 3356 different categories of digits, western characters, symbols, two Japanese phonetic alphabets: katakana and hiragana, and Japanese Kanji characters.

The new modified classifier consists of seven identical classifiers; each instance is trained with patterns from one group (from Table 1).

While off-line databases can be immediately used for training an off-line recognizer, the on-line database Nakayosi must be transformed to off-line format (bitmap images) first. We have employed Constant mode the simplest method of our unique collection of methods for generating realistic Kanji character images from on-line patterns [16]. This method combines on-line patterns with a calligraphic stroke shape library, which contains genuine off-line patterns written with different writing tools. Since the artificially generated off-line images are combinations of on-line and actual off-line patterns they look very natural and realistic.

7. Experiments

In this section we investigate an effect of our method to the recognition rate and also to time acceleration in real application. In section 5 we have presented that our approach accelerates recognizing of a common database to about 45% of the original time and for the benchmark –Kuchibue even another two percent more, to less than 43% of the original time. This is true for an off-line classifier based on any classification scheme. However, if the recognizer classifies in two or more steps as shown in 2-(2) only the first step is speeded up in this ratio.

Our recognizer from section 6 uses two steps and only the pre-classification is affected by this method.

\[ t = d_{pre}(3.2 \ast N \ast C \ast p) + d_{final}(N \ast c_2 \ast p) \]  

(7)

In final classification, the unknown input pattern is compared only with templates vectors selected in pre-classification \(c_2\). Each category is represented by \(p\) template vectors. The optimal number of templates per category was found out by experiments and for Japanese characters it was set to 40. In Table 2 we can compare the recognition rate and computation time for the original and modified classifiers. Numbers of templates vectors per category are 10, 20 and 40. The recognition rate is slightly better for modified classifier; time consumption is about 67% of the original time. An optimal number of template vectors per category depends on a number of training patterns in each category.

<table>
<thead>
<tr>
<th>Table 2: The recognition rate and time for the original and modified recognizers with a constant number of templates per category.</th>
</tr>
</thead>
<tbody>
<tr>
<td>#template</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>original</td>
</tr>
<tr>
<td>modified</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

In Table 1 we see that a number of training patterns per category varies in each group, from 710 samples for the simplest characters to only 80 samples for more complicated characters. Since we have an independent classifier for each group of patterns, it can be trained with a different number of template vectors per category, up to sixty template per category for the first group with the most training patterns and only ten templates for groups with 80 training patterns.

Since the highest number of templates is used only for two groups, we can use up-to 60 templates, what wasn’t possible in the original classifier, because so high number would make a recognizer very slow and huge. In Table 3 we see a contribution of variable number of template vectors. The recognition rate climbs to 91.60% and still recognition time is only 75% of the original time. And this variable number of template vectors is possible only in our modified approach.

<table>
<thead>
<tr>
<th>Table 3: The recognition rate and time for original and modified recognizers with a constant number of templates per category 10, 20, 40 and a flexible number 10-60.</th>
</tr>
</thead>
<tbody>
<tr>
<td># template</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>original</td>
</tr>
<tr>
<td>modified</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>10-60</td>
</tr>
</tbody>
</table>

Our new recognizer is faster and the recognition rate is higher. Only one worth parameter is a size. A template vector file is 2.35 times larger. It’s not a problem for a common computer at all, but for small hand-held computers the dictionary file size can be important. We will reduce a number of templates per category so that the dictionary size will be about the same as the original size. We see it in the last row of Table 4; the size is about the same (101.2% of the original size), time consumption is only 49.73% of the original one and the new recognition rate is 91.44%.

<table>
<thead>
<tr>
<th>Table 4 The original recognizer compared with modified recognizers optimized for the best recognition rate or speed and time.</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimized for:</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>original</td>
</tr>
<tr>
<td>modified</td>
</tr>
<tr>
<td>modified</td>
</tr>
</tbody>
</table>
8. Summary

In this paper we have presented our approach to utilize a stroke number (on-line information) in an off-line recognizer for Japanese characters.

In section 4 we explained our idea and in section 5 we proposed one concrete model. For this model we found out the theoretical acceleration for a general database to 45% of the original time. Later in the same section we proved our theoretical assumption by applying this model for a recognizer trained with Nakayosi database and Kuchibue as the benchmark. Acceleration was 42.84% of the original time.

In experimental section 7 we employed our idea to a practically used recognizer. To the recognizer, which was already speeded up by candidate selection. Even in this case, the time consumption was reduced to 75%. And in the same time the recognition rate increased to 91.60% from the original rate 90.73%. The dictionary size was 2.35 times bigger, but if the number of template vectors was reduced so that the dictionary size was the same like in the original recognizer, the recognition rate was only slightly reduced to 91.44 (but still better than original 90.73%) and time consumption was reduced to only 49.73% of the original time.

This approach is especially useful if the number of categories is high (Japanese, Chinese and so on) and also if we have a large number of training samples. We have used the Nakayosi database with about 1.5 millions patterns and we believe that if the number would be higher (3-10 millions), contribution would be even better.

In the future work we would like to experiment with methods to detect a stroke number from off-line patterns, in order that our approach could be employed also for recognizing off-line patterns.

References

[10] Yiping Yang, Ondrej Velek, Masaki Nakagawa, Accelerating a pre-classification process by employing clustering methods, submitted to 7th ICDAR 2003