Writer Adaptation for Online Handwriting Recognition System Using Virtual Examples

Hidetoshi Miyao and Minoru Maruyama
Shinshu University, 4-17-1 Wakasato, Nagano 380-8553, Japan
{miyao, maruyama}@cs.shinshu-u.ac.jp

Abstract

For an online handwriting recognition system equipped with a writer-independent classifier to progressively improve the recognition performance for a specific writer with an increase in his/her handwriting inputs, the following method is proposed: (1) For a handwriting pattern that causes a recognition error, a two-class classifier of the corresponding class is (re)constructed as a part of a writer-dependent classifier, separately from the writer-independent one, where artificially generated examples are used to compensate for lack of training examples. (2) In the recognition stage, the writer-independent classifier is applied first, and then the constructed writer-dependent classifier is used only in cases in which a result obtained by the writer-independent classifier possesses lower reliability.

We examine the effectiveness of the proposed method using 6,000 Japanese Hiragana characters written by 3 users. As a result, an average recognition rate of 98.07% was obtained by the exclusive use of the writer-independent classifier. On the other hand, the rate improved to 99.92% with at most 7 (re)constructions of a writer-dependent classifier.

1. Introduction

For devices with a digital pen, such as that used with a Personal Digital Assistant (PDA) and Tablet PC, an online handwritten recognition technique is important to input characters and symbols naturally. Many methods have been proposed [1], [2]. A generic (writer-independent) classifier can be constructed using them and training examples acquired from many individuals. However, the recognition performance is insufficient, since the shape of handwritten patterns varies widely among users.

A device for personal use is usually used by only a specific user. If recognition errors are corrected properly by the user while writing, input patterns with a class label can be acquired. Therefore, a classifier can be constructed by using them as a user-specific (writer-dependent) one. It is considered that the performance of the user-specific classifier is better than that of the generic one. However, in order to obtain a user-specific classifier with high performance, a large number of training examples must be gathered. This requires that a user spends considerable time and effort collecting samples.

To solve this problem, several methods have been proposed [3], [4]. In these methods, to construct a classifier adapted to the input patterns of a specific user, a prepared generic classifier is updated or modified progressively every time the user writes a pattern. As a result, it has been reported that the constructed classifier can reduce the recognition error rate even when very few samples per class have been entered. However, these methods have the following problems:

- Since all of the input patterns are used for updating or modifying the classifier, the computational cost may be large.
- The systems tend to be complex.
- For a specific class, the user input patterns can vary significantly from those of the training examples used for the construction of a generic classifier. In this case, it is difficult for the system to adapt to large deformation because the user-specific classifier is constructed on the basis of the generic classifier for the corresponding class.

We propose a method with the following features:

- Using only an input pattern that causes a recognition error, a two-class classifier of the corresponding class is (re)constructed as a part of a user-specific classifier, separately from the generic one. As reported earlier, a large number of training examples are needed to construct a new classifier. In previous research [5], we emphasized that patterns can be artificially generated by applying affine transformation to a few real samples. Our system uses this method

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.174

and generates many training examples to construct a user-specific classifier.

- In the recognition stage, the generic classifier is applied first, and then the constructed user-specific classifier is used only in cases in which a result obtained by the generic classifier possesses lower reliability.

In the present experiments, we examine the effectiveness of the proposed method using handwritten Japanese Hiragana characters.

2. Construction of generic classifier

A generic classifier can be constructed making use of a large number of training examples written by many users. In this system, the first 20 Japanese Hiragana characters in the lexical order, shown in Fig.1, were selected as target patterns. In Fig.1, the sharp symbol followed by the number denotes the assigned class number. We used 20,000 characters written by about 100 individuals (10 characters per person for an individual class) as training examples, which were selected from the character database **HANDS-nakayosi_t-98-09**[6].

Figure 1. Target characters and assigned class #

In the online database, we assume that each character is divided into strokes, each of which is a connected component from pen-down to pen-up. We also assume that each stroke is represented as a sequence of 2D coordinates of pen positions. Most Japanese Hiragana characters consist of several strokes. The bounding box that surrounds a sequence of points of each character is normalized to 64×64 pixels. An image of each character is generated by connecting the sequence of points for each stroke. The character image is then partitioned into 8×8 blocks. Four patterns emphasizing four directions (vertical, horizontal, left slant, and right slant) at every block are detected. As a result, 256 features ($8 \times 8 \times 4$) are extracted.

To classify the characters, we use SVM (a two-class classifier) [7]. For an individual class, an SVM is trained using the features extracted from all the test examples and the corresponding target values (1 for positive examples and -1 for negative). As a result, 20 SVMs are constructed. In practice, we adopted the Gaussian kernel as a kernel function and used SVM^{light} [8] to train the SVMs. To apply SVM to a multiclass character recognition problem, we used the one-versus-the-rest (1vr) approach [7]. In this way, a generic classifier was built.

We examined the recognition performance of the constructed generic classifier using handwritten characters written by 3 users (different from those who wrote the training examples). The test samples consisted of 2,000 characters for each user (100 characters for each class). The recognition result is shown in Table 1. For the classes not shown in this table, the recognition rates of all users were 100%. These results show that the generic classifier has high classification ability for almost all class patterns; however, a specific character of a specific user is often misclassified. This means that the generic classifier cannot respond to a user's unique deformation for a specific character class. Therefore, if a user-specific classifier were constructed only for the misclassified patterns and if it could be combined with the generic classifier properly, the entire recognition rate would be improved.

Table1. Performance of the generic classifier

class#	recognition rate [%]		
	user#1	user#2	user#3
1	100	97	100
6	100	89	100
7	99	99	99
11	52	100	100
16	100	100	99
17	98	100	100
19	100	92	66
20	100	100	99

3. Writer adaptation system

Since the generic classifier can recognize almost all input patterns correctly, as we mentioned before, and the construction of a new classifier is a time-consuming process, only if an input pattern causes a recognition error, a user-specific two-class classifier (SVM) of the corresponding class is (re)constructed separately from the generic one. Consequently, a user-specific classifier consists of one or more SVMs. Figure 2 shows the flowchart of the system.

To build a user-specific SVM, a large number of training examples are needed. The negative examples consist of 50 patterns for each class, written by 50 users (1 character per user). They are selected from the training examples used for the generic classifier construction. That is, 1,000 fixed patterns are used as negative examples. On the other hand, it is

recommended that about 1,000 positive examples be also prepared. However, the number of input patterns is limited. We, therefore, use our previous method [5] to generate many artificial patterns (virtual examples) from the real input pattern.

A character consists of one or more strokes, and each stroke is represented as a sequence of 2D coordinates of pen positions. In our previous work, we simply applied the following affine transformation to each point of a stroke.

$$\mathbf{x}' = \bar{\mathbf{x}} + \mathrm{A}(\mathbf{x} - \bar{\mathbf{x}}) + \mathbf{t}, \tag{1}$$

where $\mathbf{t} = (\mathrm{t_x}, \mathrm{t_y})^{\mathrm{T}}$ is the translation and $\bar{\mathbf{x}}$ is the center of the bounding box of the stroke. A 2×2 matrix A is given as the product of a shear matrix S and a rotation matrix R

$$\mathrm{A} = \mathrm{A}(\theta, \varepsilon_\mathrm{x}, \varepsilon_\mathrm{y}) = \mathrm{R}(\theta)\mathrm{S}(\varepsilon_\mathrm{x}, \varepsilon_\mathrm{y}), \tag{2}$$

where $\mathrm{R}(\theta)$ and $\mathrm{S}(\varepsilon_\mathrm{x}, \varepsilon_\mathrm{y})$ are given by

$$\mathrm{S}(\varepsilon_\mathrm{x}, \varepsilon_\mathrm{y}) = \begin{pmatrix} 1 & \varepsilon_\mathrm{x} \\ \varepsilon_\mathrm{y} & 1 \end{pmatrix}, \mathrm{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{3}$$

The transformation in (1) is specified by 5 parameters $(\mathrm{t_x}, \mathrm{t_y}, \theta, \varepsilon_\mathrm{x}, \varepsilon_\mathrm{y})$. They are given uniformly at random for each stroke.

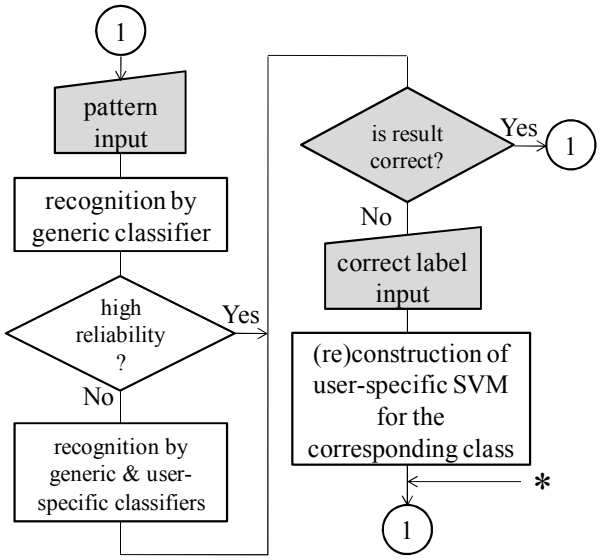

Figure 2. Flowchart of the system (shaded boxes refer to user's confirmation or user inputs)

For a specific class, if an input pattern of the class is misclassified for the first time, the positive examples consist of the real input pattern itself and 999 virtual examples generated from the input pattern using the above method. For the class, if another input pattern is misclassified, the positive examples are re-generated, and the user-specific SVM for the corresponding class is also reconstructed. In such cases, the system generates as few patterns as possible by using the previously generated patterns. For the i^{th} (i is a 2 or higher integer value) misclassified pattern of a specific class, positive examples consist of the following patterns:

1. misclassified patterns $\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_{\mathrm{i}-1}$,
2. misclassified pattern P_i,
3. $\lfloor(\mathrm{N} - \mathrm{i})/\mathrm{i}\rfloor$ patterns extracted from virtual examples that are generated from each of the patterns $\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_{\mathrm{i}-1}$, that is, $(\mathrm{i} - 1)\lfloor(\mathrm{N} - \mathrm{i})/\mathrm{i}\rfloor$ patterns in total,
4. $\mathrm{N} - (\mathrm{i} + (\mathrm{i} - 1)\lfloor(\mathrm{N} - \mathrm{i})/\mathrm{i}\rfloor)$ virtual examples are newly generated based on the pattern P_i,

where N denotes the total number of positive examples (in this system, N is 1,000). The patterns in item 3 are previously generated ones. (In practice, the directional features extracted from the patterns in items 1 and 3 were preserved at the previous pattern generation.)

The user-specific SVM is (re)constructed by using directional features extracted from the positive and negative examples in the same way as for the construction of the generic classifier. In the experiment, the SVMs were trained by using $\mathrm{SVM}^{\mathrm{light}}$ as well.

If the constructed user-specific classifier is always used to recognize an input pattern, it may worsen the entire recognition performance. Specifically, the performance of the user-specific classifier should not be high enough in a situation in which not enough patterns are entered. Consequently, as shown in Fig.2, the generic classifier is applied first, and then the user-specific classifier is used only in cases in which a result obtained by the generic classifier possesses lower reliability.

To evaluate the reliability, we used the following two values: the largest output value (V_1) of SVMs constructed for the generic classifier and the difference value (V_2) between the value V_1 and the second largest one; this was because both values tend to be small when the recognition fails in a preliminary experiment. In practice, if $\mathrm{V}_1 \leq \mathrm{Th}_1$ and $\mathrm{V}_2 \leq \mathrm{Th}_2$, where Th_1 and Th_2 are threshold values, the reliability is judged to be low. In such a case, both classifiers (generic and user-specific ones) are applied to the input pattern, and the pattern is recognized by applying the 1vr approach to all SVMs of generic and user-specific classifiers.

4. Experimental results

In the experiment, we used test patterns written by 3 users who were the same as those shown in Table 1. All character patterns were written within a 200×200 pixel window using a Tablet PC. In total, 4,000 patterns (200 patterns for each class) were acquired from each user. With regard to the patterns of each user, we call half of them dataset-A (100 patterns for each class and 2,000 patterns in total) and the other half, dataset-B. For dataset-A, the data for the j^{th} pattern ($j = 1,2,\cdots,100$) of i^{th} class ($i = 1,2,\cdots,20$) is denoted by A[i][j].

We prepared the same generic classifier used in Section 2. The parameters of the affine transformation $(t_x, t_y, \theta, \varepsilon_x, \varepsilon_y)$ used in the experiment were $|t_x|, |t_y| \leq 7(\text{pixel}), |\theta| \leq 2(\text{degree}), |\varepsilon_x|, |\varepsilon_y| \leq 0.2$. The thresholds Th_1 and Th_2 were 0.8 and 1.5, respectively. According to the procedure shown in Fig.2, patterns A[1][1], A[2][1], $\cdots$, A[20][1], A[1][2], A[2][2], $\cdots$, and A[20][100] were processed one by one in this order. In the process, in order to assess the improvement of the system performance using the newly constructed system, all patterns of dataset-B were recognized every time a user-specific SVM was (re)constructed (this timing is denoted by an asterisk in Fig.2). Note that the system does not (re)construct user-specific SVMs but it does recognize dataset-B. Table 2 shows the results of this process for the patterns of user #1.

Table2. Results for the patterns of user #1

(re) built #	class # of (re)built SVM	misclassified pattern class and number of errors for dataset-B	total error
0		#11(→#7)/50, #11(→#10)/7, #11(→#17)/1, #17(→#3)/1	59
1	#11(→#7)	#7(→#11)/31, #17(→#3)/1	32
2	#17(→#3)	#7(→#11)/31	31
3	#7(→#11)	#7(→#11)/1, #11(→#7)/1	2
4	#11(→#7)	#7(→#11)/23	23
5	#7(→#11)	#7(→#11)/2	2
6	#7(→#11)		0

In the table, the first column (denoted by "(re) built #") represents how many times user-specific SVMs are (re)constructed in the process of feeding patterns of dataset-A into the system one by one. Therefore, in the second row, denoted by "(re) built # is 0," the patterns of dataset-B were recognized using only the generic classifier, and the corresponding results are represented (the results for dataset-A, recognized using only the generic classifier, are shown in Table 1). The second column represents the class number of the (re)constructed user-specific SVM and the class number of the pattern as which the system misclassified an input pattern. In this case, three user-specific SVMs (for classes #11, #17, and #7) were finally built. The third and fourth columns represent the recognition results for dataset-B. We take the fourth row as an example. #17(→#3) in the second column denotes that a pattern of class #17 was misclassified as a pattern of class #3 in the process of recognizing dataset-A and the user-specific SVM for the class (#17) was constructed. Using the system at that time, the patterns of dataset-B were recognized. The results are shown in the third and fourth columns. #7(→#11)/31 denotes that the system misclassified 31 patterns of class #7 as patterns of class #11. Consequently, the number of total errors was 31, as shown in the fourth column.

The results show that the total errors are reduced progressively every time that the user-specific SVM is (re)constructed. However, a few fluctuations can be seen. For user #1, it is difficult for the system to distinguish between the patterns of classes #7 and #11. If a user-specific SVM for one of the classes is (re)constructed, the recognition error for the other class tends to increase. This is a reason for the occurrence of the fluctuation. However, after the user-specific SVMs for both classes are reconstructed several times, the errors suddenly decrease.

Figure 3 shows the relationship between the recognition rates for dataset-B and the number of times user-specific SVMs are (re)constructed. For 3 users' patterns, the average recognition rate with the initial condition was 98.07%, and the rate improved to 99.92%, with at most 7 (re)constructions of user-specific SVMs.

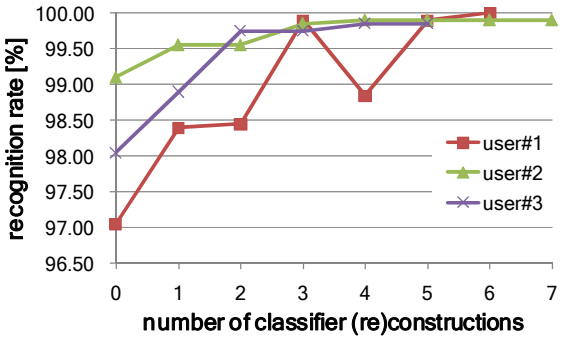

Figure 3. Relationship between recognition rates and number of times user-specific SVMs are (re)constructed

It is considered that the longest computing time is required to construct a new user-specific SVM for a class because the system has to generate the largest

number of virtual examples (999 samples). Using a few patterns, we measured the time, and a time range between 0.3s and 0.4s (CPU: Xeon Quad-Core 3.16GHz, Memory: 4GB) was obtained. Therefore, very little waiting time is required for a user to enter patterns.

5. Conclusion

For an online handwriting recognition system to progressively improve the recognition performance for a specific user with an increase in his/her handwriting inputs, the following method was proposed:

- For a handwriting pattern that causes a recognition error, a user-specific SVM of the corresponding class is (re)constructed separately from the generic classifier, in which artificially generated examples are used.
- In the recognition stage, the generic classifier is applied first, and only if the reliability of the result is not high, the constructed user-specific classifier is also applied.

We examined the effectiveness of the proposed method used with 6,000 Japanese Hiragana characters (20 classes) written by 3 users. An average recognition rate of 98.07% was obtained with the generic classifier, and the rate then improved to 99.92% with at most 7 (re)constructions of user-specific SVMs. The results show that the method can minimize the number of constructions of user-specific SVMs and reduce the computational cost while the user is writing; furthermore, it adapts easily to the writer. As to the computing time for the construction of an SVM, it is less than approximately 0.4s. The system is concluded to be sufficiently practical.

In the system, we adopted the directional features of the character image and SVM classifiers. However, our method could be applied to any system using other features and classifiers; however, the type required is a two-class classifier. Therefore, our method has wide applications.

In the current system, the negative examples used for the construction of the user-specific SVM are extracted equally from the training examples of an individual class, in which the training examples are used for the construction of the generic classifier. The system tends to misclassify many patterns of a specific class (denoted as class-A) as those of another specific class (denoted as class-B) depending on the user. If negative examples are extracted more frequently from the class-B patterns, the recognition rate could be improved more rapidly.

References

[1] C.C. Tappert, C.Y. Suen, and T. Wakahara, "The State of the Art in On-Line Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 8, pp.787-808, 1990.

[2] C.L. Liu, S. Jaeger, and M. Nakagawa, "Online Recognition of Chinese Characters: The State-of-the-Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 2, pp.198-213, 2004.

[3] W. Kienzle and K. Chellapilla, "Personalized Handwriting Recognition via Biased Regularization," *Proc. of International Conference on Machine Learning*, pp.457-464, 2006.

[4] P. Haluptzok, M. Revow, and A. Abdulkader, "Personalization of an Online Handwriting Recognition System," *International Workshop on Frontiers in Handwriting Recognition*, 2006.

[5] H. Miyao, M. Maruyama, Y. Nakano, and T. Hananoi, "Off-Line Handwritten Character Recognition by SVM based on the Virtual Examples Synthesized from On-Line Characters," *Prof. of International Conference on Document Analysis and Recognition*, Vol. 1, pp. 494-498, 2005.

[6] http://www.tuat.ac.jp/~nakagawa/ipdb/

[7] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[8] T. Joachims, "Making large-scale SVM learning practical," *In Advances in Kernel Methods*, Chapter 11, MIT Press, 1999.