published in the proceedings of ICANN'98, Skovde, Sweden, September 1998, pp. 245-250
available at <URL:http://www.cis.hut.fi/jorma/papers/>

Comparison of Adaptive Strategies for On-Line Character Recognition

Jorma Laaksonen, Jarmo Hurri and Erkki Oja
Laboratory of Computer and Information Science
Helsinki University of Technology
Espoo, Finland
{jorma.laaksonen,jarmo.hurri,erkki.oja}@hut.fi

Jari Kangas
Nokia Research Center
Tampere, Finland
jari.kangas@research.nokia.com

Abstract

Results on a comparison of adaptive recognition techniques for on-line recognition of handwritten Latin alphabets are presented. The classification strategies compared are based on first compressing or distilling a large database of handwritten characters to a small set of character prototypes. Each adaptive classifier then either modifies the original prototypes or conditionally adds new prototypes when they become available from the user of the system. In each case, the classification decision uses the 1-Nearest Neighbor (1-NN) rule for the distances between the input character and the stored prototypes. The distances are calculated using Dynamic Time Warping (DTW). One of the adaptive learning strategies features an extension of the neural Learning Vector Quantization (LVQ) algorithm to the DTW distance metric. All the methods concerned exhibit automatic unsupervised learning from user input simultaneously with the normal mode of operation. The presented experiments show that the assessed methods produce different tradeoffs between the accuracy and complexity of classification. Every version is, however, able to adapt to the user's writing style with only a very few – say some tens of – handwritten characters.

1 Introduction

The aim of the present study has been to evaluate the potential of a totally *unsupervised adaptive handwriting recognizer*. Starting from an initially stored character database, the system should be able to utilize the user input to enhance the classification accuracy simultaneously when it is used. No separate training period should therefore be necessary. The system uses its own recognitions in a *self-supervised* fashion: the recognitions which are not indicated as failures by the user are regarded as correct. Similarly, the user-reported misclassifications are used to revise the system. Such a system is a user-friendly option for text input, e.g., in handheld devices. The key point then is how quickly the system gets tuned to the user's writing style.

We present the results of a series of experiments which assessed the development of neural and other types of adaptive classifiers for on-line recognition of separately written Latin characters. Starting with a classifier which has only a small number of user-independent character prototypes, the recognition systems described in this paper are able to adapt to the user's writing style simultaneously with the normal operation mode of the classifier. The adaptation is either done by modifying the original character prototypes in the neural fashion of the Learning Vector Quantization (LVQ) algorithm or by adding some or all of the user's characters when they become available.

The results of the experiments, when compared against those reported in [5], show that the adaptive classification strategies provide accuracies comparable to those of non-adaptive classifiers with much higher computational complexity.

2 Dynamic Time Warping and LVQ

A key factor in the implementation of on-line character recognition systems is the definition of the distance function between two stroked characters. In our experiments, the distance was set to be infinite if the number of strokes is different in the two characters. Otherwise, the distance between two characters was calculated as the sum of the distances between the corresponding strokes. The strokes were always matched in their original order and direction of drawing.

The distance between two strokes was calculated using the standard Dynamic Time Warping (DTW) technique [6]. The stroke $\mathbf{u}$ consists of U points of the xy-plane, $\mathbf{u} = (\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_U)^T = ((u_{1x}, u_{1y})\ (u_{2x}, u_{2y})\ \cdots\ (u_{Ux}, u_{Uy}))^T$. The indices of the points of $\mathbf{u}$ along the time warping path are denoted with the vector $\mathbf{i} = (\mathbf{i}(1)\ \mathbf{i}(2)\ \cdots\ \mathbf{i}(H))^T$, where H is the length of the warping path. For another stroke $\mathbf{v}$, the notations are similar with $\mathbf{j}$ denoting the indexes of $\mathbf{v}$ in the warping path.

It was required that the endpoints of the strokes were matched precisely, i.e., $\mathbf{i}(1) = \mathbf{j}(1) = 1$, $\mathbf{i}(H) = U$, and $\mathbf{j}(H) = V$. Along the warping path, three continuity constraints were used:

$$(\Delta\mathbf{i}(h),\, \Delta\mathbf{j}(h)) = (\mathbf{i}(h) - \mathbf{i}(h-1), \mathbf{j}(h) - \mathbf{j}(h-1)) = \begin{cases} (1,0) & \text{or} \\ (1,1) & \text{or} \\ (0,1)\ . \end{cases} \tag{1}$$

These three permissible steps were all treated equally and the cost associated with each point pair in the sequences was the squared Euclidean distance between the corresponding points in the xy-plane. The squared time warped distance between the strokes $\mathbf{u}$ and $\mathbf{v}$ can thus be expressed as

$$d^2(\mathbf{u}, \mathbf{v}) = \sum_{h=1}^{H} d^2(\mathbf{u}_{\mathbf{i}(h)}, \mathbf{v}_{\mathbf{j}(h)}) \tag{2}$$

$$= \sum_{h=1}^{H} (u_{\mathbf{i}(h)x} - v_{\mathbf{j}(h)x})^2 + (u_{\mathbf{i}(h)y} - v_{\mathbf{j}(h)y})^2\ . \tag{3}$$

The update rule of the Learning Vector Quantization (LVQ1) [4] algorithm can then be extended to the distance measures obtained with time warping by solving $\partial d^2(\mathbf{u},\mathbf{v})/\partial \mathbf{v}$ from (3). When we denote the Kronecker's delta function by $\delta(k,l)$ we can write

$$\frac{\partial d^2(\mathbf{u},\mathbf{v})}{\partial v_{kx}} = -2\sum_{h=1}^{H} \delta(k,\mathbf{j}(h))\ (u_{\mathbf{i}(h)x} - v_{kx}) \ . \tag{4}$$

After calculating the partial derivatives with respect to all v_{kx} and v_{ky}, the collected result can symbolically be written as

$$\frac{\partial d^2(\mathbf{u},\mathbf{v})}{\partial \mathbf{v}_k} = -2\sum_{h=1}^{H} \delta(k,\mathbf{j}(h))\ (\mathbf{u}_{\mathbf{i}(h)} - \mathbf{v}_k) \ . \tag{5}$$

The adaptation rule of the LVQ1 algorithm can then be written for the input stroke $\mathbf{x}$ and the corresponding stroke $\mathbf{m}$ in the prototype character which is nearest to the input character:

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha(t)\frac{\partial d^2(\mathbf{x},\mathbf{m})}{\partial \mathbf{m}} \ , \text{ if } \Omega(\mathbf{x}) = \Omega(\mathbf{m}) \tag{6}$$

$$\mathbf{m}(t+1) = \mathbf{m}(t) + \alpha(t)\frac{\partial d^2(\mathbf{x},\mathbf{m})}{\partial \mathbf{m}} \ , \text{ if } \Omega(\mathbf{x}) \neq \Omega(\mathbf{m}) \ , \tag{7}$$

where $\Omega(\mathbf{x})$ stands for the class of $\mathbf{x}$ and $\alpha(t)$ is a small positive, possibly decreasing learning rate parameter.

A similar adaptation rule based on the DTW error criterion has been presented for use in speech recognition [3]. The current authors, however, are not aware of such approaches being used in recognition of handwritten characters.

3 Classification and Adaptation

In the classification of the characters, k-Nearest Neighbor (k-NN) classification rule [1] was used. Values one to six for k were tested and it was found out that with every classifier, the value $k = 1$ produced the best classification accuracy. The selection of $k = 1$ is as such in accordance with the LVQ training principle.

The LVQ training of the character prototypes was compared to approaches where new characters were added to the prototype set instead of being used in modifying the existing prototypes. In the first approach, denoted 1-NN in the figure and table below, each character in the training set was added to the set of prototypes if it was erroneously classified with the existing prototypes. This procedure leads to a moderate increase of the number of prototypes used by the classifier.

In the next version, the training character was added to the prototype set also in the situation when the second nearest existing prototype belonged to a class different from that of the training character. This approach was denoted 2-NN. Similarly, in the second modification of the prototype adding approach, denoted 4-NN, it was demanded that all the four existing prototypes nearest to the input character should belong to the correct class in order for the input character not to be included to the prototype set.

These two versions produce faster increase of the number of characters in the prototype set than the original 1-NN addition rule. As a limiting case, we also considered an adaptation policy in which all the training characters are used as prototypes. This approach was named 'all prototypes added', even though it could have been denoted by ∞-NN. These modifications resemble various editing rules used together with k-NN classifiers. One of the earliest descriptions of such algorithms is given in [2]. A noteworthy characteristic of the techniques described here is, however, that each training sample is used only once and discarded from the memory of the system if its class agrees with the classes of the existing prototypes nearest to it.

4 Experiments

The collection and preprocessing of the data used in the experiments is explained in detail in [5]. It will suffice to say that there were 22 subjects each of who drew a sample containing approximately 350 instances of the lowercase letters 'a'–'z' which were thresholded on the basis of pressure information. From this data, we formed a compressed set of 104 character prototypes by using a semi-automatic clustering algorithm. This set was used as a user-independent 'distilled classifier' upon which the adaptive classifiers were built.

We created the five adaptive classifiers described above for each of the subjects with an increasing number of characters included in the training set. The remaining characters were used in testing the resulting classifier. Each classifier was implemented 30 times for each subject by randomly rearranging the data set and averaging the recognition accuracies in order to get representative figures for the performances. The final results, when the number of characters in the training and testing sets were 300 and approximately 50, respectively, are shown in Table 1. The decrease of the classification error percentage when the number of added prototypes was increased is seen in Figure 1.

Classifier	**Prototypes**	**Error %**
distilled	104	8.33
LVQ	104	3.30
1-NN	117	3.20
2-NN	149	2.12
4-NN	200	1.85
all added	404	1.64

Table 1: Final classification accuracies of the adaptive classifiers when 300 characters were used in training and about 50 in testing. The error percentages shown are averages of the results of 30 repetitions with each of the 22 writers involved. The numbers of the used prototypes are corresponding average results.

It can be noted in Figure 1, that the dashed lines connecting the result points with equivalent number of samples in the training set, are nearly horizontal between the LVQ and the 1-NN curves and the 4-NN and the 'all prototypes added' curves, respectively. This is an indication in favor of the conclusion that the performance of the neural LVQ classifier equals that of the 1-NN classifier. Likewise, the 4-NN prototype adding strategy is in accuracy as good as the strategy in which the prototypes are added unconditionally. In both the LVQ and 4-NN cases, the resulting complexity of the final classifier is lower than in the 1-NN and 'all prototypes added' cases, respectively.

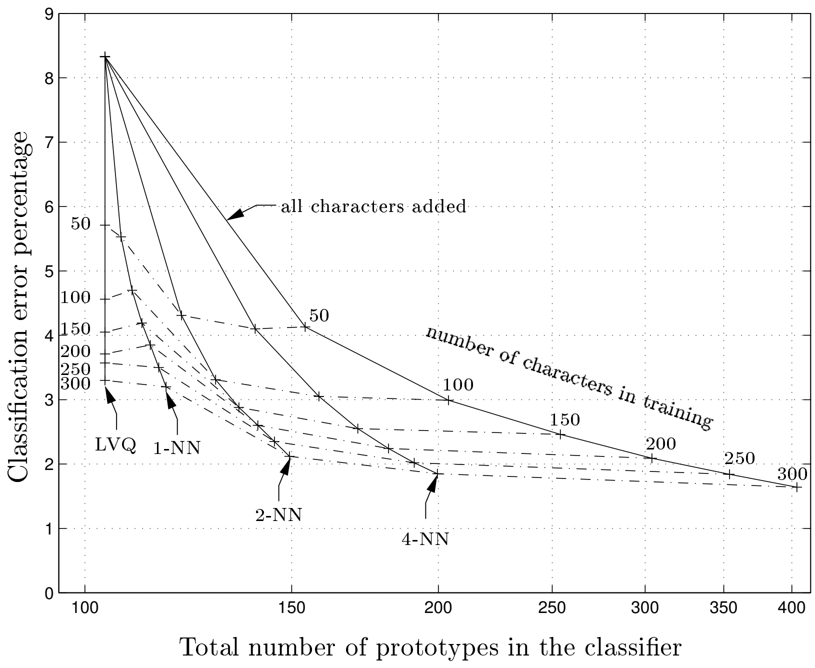

Figure 1: The decrease of the error rate for the classification of the testing set when the total number of character prototypes in the system increases. In the case of LVQ, no characters were added, only the existing prototypes were modified. The dashed lines connect the results with equivalent number of samples in the training set. The numerical values for the final classification accuracies can be seen in Table 1.

In [5], various non-adaptive classifiers were formed from the same data sets as here. Comparing to the accuracies achieved with non-adaptive classifiers with approximately the same number of prototypes, the results of the adaptive classifiers presented here are clearly superior to the non-adaptive results. For example, when each subject's data were classified in a leave-one-out procedure, the average error rate was found to be 2.10%. On the other hand, when only other subject's characters were used as prototypes in classification, the average error rate was 7.51%.

5 Conclusions

One objective of our study was to evaluate whether the number of characters we have in our database is adequate for the creation of the various adaptive classifier types or not. Interpreted in this sense, the results indicate that the number of characters given by each writer – approximately 350 – is not enough for the proposed adaptive classification techniques to reach their final level of accuracy. On the other hand, the average recognition error rate is as such low enough for the proposed system to be used in a real-world application. The collection of data could thus be implemented in an actual prototype system.

The results show that the performance of the neural LVQ learning rule is as good as that of the simplest prototype adding strategy. On the other hand, the more complex prototype adding rules give better classification accuracy while the complexities of the learning and classification tasks increase. Therefore, it would be advisable to try to combine the neural LVQ training with the prototype adding rule to form a hybrid training strategy. Such a classifier would allow a moderate increase of the number of character prototypes stored in the system while concurrently modifying the existing prototypes in order to make full use of the information available in the training samples. The LVQ learning could thus respond to small variations in writing styles, whereas the adding rule would facilitate the adaptation of completely new ways of writing.

References

[1] E. Fix and J. L. Hodges. Discriminatory analysis—nonparametric discrimination: Consistency properties. Technical Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[2] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, May 1968.

[3] S. Katagiri, C.-H. Lee, and B.-H. Juang. New discriminative training algorithms based on the generalized probabilistic descent method. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 298–308, Princeton, New Jersey, USA, October 1991.

[4] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences.* Springer-Verlag, 1997. Second Extended Edition.

[5] J. Laaksonen, J. Hurri, E. Oja, and J. Kangas. Experiments with a self-supervised adaptive classification strategy in on-line recognition of handwritten latin characters. In *Proceedings of the 6th Workshop on Frontiers of Handwriting Recognition*, 1998.

[6] D. Sankoff and J. B. Kruskal, editors. *Time warps, string edits, and macromolecules: The theory and practice of sequency comparison.* Addison-Wesley, 1983.