Controlling On-Line Adaptation of a Prototype-Based Classifier for Handwritten Characters

Vuokko Vuori, Jorma Laaksonen, Erkki Oja
Helsinki University of Technology
Laboratory of Computer and Information Science
P.O. Box 5400, FIN-02015 HUT, Finland
{vuokko.vuori,jorma.laaksonen,erkki.oja}@hut.fi

Jari Kangas
Nokia Research Center
P.O. Box 100, FIN-33721 Tampere, Finland
jari.a.kangas@nokia.com

Abstract

Methods for controlling the adaptation process of an on-line handwritten character recognizer are studied. The classifier is based on the k-nearest neighbor rule and it is adapted to a new writing style by adding new prototypes, inactivating confusing prototypes, and reshaping existing prototypes in a self-supervised fashion. The dissimilarity measure used for the comparison of characters is a nonlinear curve matching method based on dynamic time warping algorithm. Time needed for the evaluation of the dissimilarity measure for a single character depends linearly on the size of the prototype set. The purpose of the control methods is to increase the classifier's tolerance to malformed or mislabeled learning samples and to limit the growth of the prototype set. The control methods either set an upper limit for the number of prototypes per class or switch the adaptation of a particular character class on or off depending on the earlier performance of the classifier.

1. Introduction

On-line recognition of handwriting has been an on-going research problem for four decades and it has been approached in various ways. It has been gaining more interest lately due to the increasing popularity of hand-held computers as it is a good alternative to a keyboard for text input. It is more natural to use handwriting than a miniature keyboard or digit buttons. In addition, pointing operations can be included in the same pen-based interface. The main challenge in handwriting recognition is the vast variety of writing styles. It can be overcome, to some extent, by adapting the recognition system to new writing styles. Another problem is that the contemporary hand-held devices possess fairly limited memory and processing capabilities compared to normal-sized computers. A good description of the state of the art of on-line handwriting recognition methods and the problem field in general is given in a recent thorough survey [4].

In this work, we propose methods suitable for on-line adaptation of a prototype-based classifier. The adaptation is controlled in order to keep the size of the prototype set – and consequently the recognition time and memory consumption – reasonable. In addition, the controls are aimed at decreasing the system's sensitivity to erroneously labeled or malformed learning samples. The control methods either set an upper limit for the number of prototypes per class or switch the adaptation of a particular character class on or off depending on the classifier's performance. The effects of these simple controls are studied with simulations in which incorrectly labeled samples are introduced into the learning process.

2. Description of the classifier

The classifier used in the experiments is based on the k-nearest neighbor (k-NN) rule: an input character is matched against all stored prototypes and the k most similar ones vote for its classification. The prototype set covers several writing styles and it is formed with a clustering algorithm [2] from a database including characters written by several subjects. The number of prototypes per class is always seven, which was the maximum number of different writing styles found for a single character in the manual examination of the data.

The dissimilarity measure [6] used both for the clustering and classification of the characters is based on the dynamic time warping (DTW) algorithm [5] which is a nonlinear curve matching method. The connected parts of a drawn curve in which the writing pressure exceeds a preset value are in our work considered as strokes. The dissimilarity measure is defined on stroke basis so that it is infinite between two characters having different numbers of strokes. The strokes are matched in the same order as they were drawn and the first and last data points of the two curves are strictly matched against each other. The DTW-algorithm finds the point-to-point correspondence between the curves which satisfies these constraints and yields the minimum sum of the costs associated with the matchings of the data points. A cost for matching two data points is their squared Euclidean distance.

The initial writer-independent classifier is adapted to new writing styles by modifying its prototype set. The modifications include: 1) adding new prototypes, 2) reshaping existing protypes, and 3) inactivating confusing prototypes. Adaptation strategy *Add* adds the input character into the prototype set if any of the k most similar prototypes belongs to a wrong class. *AddAndLvq*-strategy reshapes the nearest prototype if any of the k most similar prototypes belongs to the correct class. Otherwise, the input character is included in the prototype set. The reshaping of the prototypes is based on a modified version of the Learning Vector Quantization (LVQ) [1]. Depending on the classes of the characters the data points of the prototype are moved towards, or away from, the data points of the input character along the lines connecting the matched data points. A suitable value for parameter k was found in some previous experiments to be 4 or 3 depending on whether the adaptation was performed according to *Add*- or *AddAndLvq*-strategy, respectively [6].

These basic strategies can be controlled in two ways: an upper limit can be set for the number of prototypes per class, or the adaptation of a particular class can be switched on or off depending on the classifier's performance. Adaptation strategy *PLAdd* (*PL* stands for prototype limit) works as *Add* if the number of prototypes is less than the upper limit N_{U}. Otherwise, the adaptation is turned off. *PLAddAndLvq*- and *PLAddOrLvq*-strategies are similar to *AddAndLvq*- and *Add*-strategies, respectively, but if the upper limit for prototypes is reached, they perform only prototype reshaping. Adaptation strategies *OFAdd* and *OFAddAndLvq* (*OF* stands for on and off) are based on another controlling principle. At the beginning of the adaptation, they are similar to *Add*- and *AddAndLvq*-strategies. However, adaptation of a particular character is switched off if input samples of that class are classified successfully N_{off} times in a row. On the other hand, adaptation is switched back on after N_{on} consecutive unsuccessful recognitions.

IAP-strategy is used for inactivating a prototype which has been the most similar prototype at least N_{I} times and whose class has been incorrect more often than correct. However, the last prototype for a character class cannot be inactivated. *IAP*-strategy can be used together with the other adaptation strategies as a cure for confusing initial prototypes and malformed or incorrectly labeled learning samples. A good value for N_{I} has been found to be 1. Such a low value enables *IAP*-strategy to react immediately when problems arise due to the erroneous prototypes.

The adaptation is presumed to be carried out on-line and in a self-supervised way. This means that the correct labels of the input characters are concluded from the writer's reactions to the classification results. If the writer does not re-enter characters, their classifications are assumed to be correct. Otherwise, they are labeled according to the latest character input in the same position. This kind of labeling method is susceptible to errors of two natures: the writer does not notice all the misclassified characters, or, makes corrections, for example, to the spelling of the words.

3. Experiments

All the experiments were carried out with lower case letters (a-z, and diacriticals å, ä, and ö). Characters were collected with a special tablet the resolution and sampling rate of which are high – 100 lines per millimeter and at maximum 205 data points per second. The datapoints consist of the x- and y-coordinates of the pen point, writing pressure, and time. Characters were written one at a time and the subjects were advised to use their own natural style of writing. All malformed characters were cleaned off manually. The number of data points was reduced by keeping only every third data point. Characters were normalized so that their mass centers were shifted to the origin of the coordinate system. In addition, they were rescaled so that the length of the longer side of the bounding box was set to a constant value while retaining the aspect ratio unchanged [6].

The prototypes of the initial classifier were selected from a database of over 7 500 characters written by 22 subjects by using a clustering algorithm. The database used in the experiments as a test set consists of approximately 12 500 characters. The latter database was written by 24 subjects which were not included in the former database.

In the first experiments, suitable parameter values were selected for the controlled adaptation strategies. Next, the uncontrolled adaptation strategies were compared with the controlled ones. In the last experiments, prototype inactivation strategy was applied together with the most promising adaptation strategies.

Each adaptation strategy was applied in a test run in which the characters of every particular writer were recognized one by one and the classifier was adapted after each classification. In the simulations of type 1, some of the misclassified characters were labeled according to their recognition result with a probability varied from zero to one. This kind of errors can occur if the user does not notice or care to correct all the recognition errors. In the simulations of type 2, the characters were labeled at random with a probability varied from zero to 0.1. Such errors are introduced when text editing, for example, corrections of writing mistakes

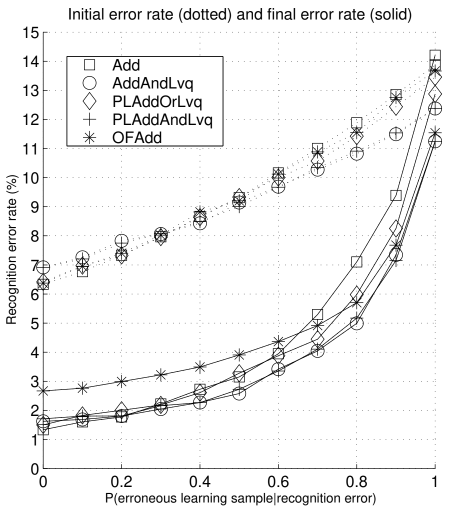

Figure 1. The effect of unnoticed recognition errors on the initial and final error rates.

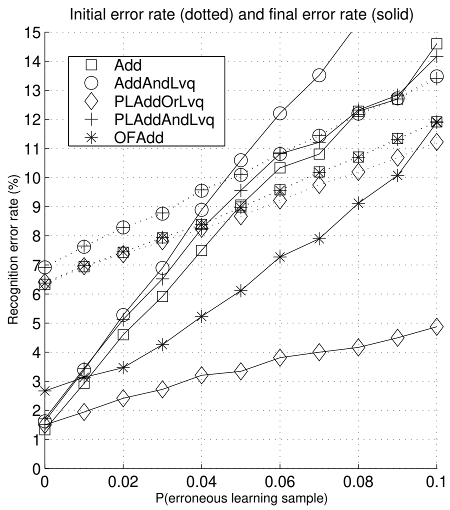

Figure 2. The effect of random errors on the initial and final error rates.

are misinterpreted as corrections of recognition errors. All tests were repeated ten times with different random number sequences to even out statistical variations.

The performance of an adaptation strategy was measured with four figures: intial, final, and total error rates, and the growth percentage of the prototype set. The initial and final error rates were evaluated for the first and last 100 characters, respectively. The figures were averaged over the writers. The selection of the parameter values N_{U}, N_{on}, and N_{off} was performed on the basis of the total error rates averaged over the writers and different error probabilities.

The results of the experiments are summarized in Figures 1-3 and Table 1. Figure 1 illustrates the evolution of the initial and final error rates when the probability for labeling a learning sample according to an erroneous recognition result is increased. In Figure 2, the effects of randomly labeled learning samples are studied in a similar way. Figure 3 shows the effects of the prototype inactivation strategy on the final error rate. Finally, Table 1 gives the average growth percentages of the prototype count for different adaptation strategies. The average is calculated over the error probabilities shown in the figures.

Without adaptation, the error rate is approximately 12% which is far too high for any practical application. The uncontrolled adaptation strategies (*Add* and *AddAndLvq*) work well if there are no erroneous learning samples: the error rate is less than 7% for the first 100 characters and less than 2% for the last 100 characters of the approximately 500 characters from each writer. As could be expected, the adaptation methods are far too sensitive to errors, especially to randomly labeled learning samples as can be seen in Figure 2. The unnoticed recognition mistakes have a similar effect with both strategies but randomly labeled characters are more harmful with *AddAndLvq*-strategy. This can be explained by the fact that *AddAndLvq*-strategy does not only add mislabeled characters into the prototype set but also destroys some of the original prototypes by reshaping them wrongly. *Add*-strategy increases the size of the prototype set significantly. In the worst case, the size of the prototype set is doubled (see Table 1). *AddAndLvq*-strategy acts in a more restrained way and is satisfactory in that sense. In the light of these results, there is a certain need for additional control of the adaptation process.

The experiments with adaptation strategies which set an upper limit for the number of prototypes per class (*PLAdd*, *PLAddAndLvq*, and *PLAddOrLvq*) showed that there is no need to add many new prototypes per class. Best results were obtained if only two new prototypes were allowed per class ($N_{\mathrm{U}} = 9$) and adaptation was continued as pure LVQ-learning after the upper limit for prototype count was reached. *PLAdd*-strategy was outperformed by *PLAddOrLvq*-strategy in all respects. *PLAddAndLvq*-strategy increased the size of the prototype set clearly less than *PLAddOrLvq*-strategy. However, in general, error rates obtained with the latter strategy were better than those obtained with the former strategy. *PLAddOrLvq*-strategy was able to decrease the system's sensitivity to the erroneous learning samples of both types and the system's ability to learn in the error-free situation was preserved.

The idea of controlling the adaptation by switching it on and off did not turn out to be as good an idea as limiting the number of new prototypes. *OFAdd*-strategy did reduce the harmful effects of erroneous learning samples and control the growth of the prototype set. However, the error rate

Table 1. Average growth percentage of the prototype count. The subscript indicates the type of errors: 1) labeling according to recognition result, or 2) random labeling. Superscript *IAP* means that the prototype inactivation strategy has been applied.

Adapt. strategy	ΔP_1	ΔP_1^{IAP}	ΔP_2	ΔP_2^{IAP}
Add	63	59	106	69
AddAndLvq	2	0	15	1
PLAddOrLvq	24	24	26	24
PLAddAndLvq	2	0	12	1
OFAdd	29	26	52	29

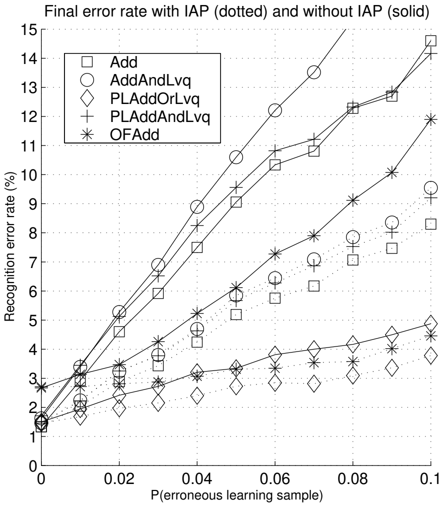

Figure 3. The effect of prototype inactivation strategy on the final error rate. Erroneous learning samples are caused by misinterpreted writing mistake corrections.

for the simulation with perfectly labeled learning samples was increased. Best results were obtained when $N_{\text{on}} = 2$ and $N_{\text{off}} = 3$. *OFAddAndLvq*-strategy did not improve recognition rates in any case. As its uncontrolled version, *AddAndLvq*-strategy, does not increase the number of prototypes too much, it was found useless and left out from the result figures and table.

The results of experiments in which the most promising controlled adaptation strategies and their uncontrolled versions (*Add*, *AddAndLvq*, *PLAddAndLvq*, *PLAddOrLvq*, *OFAdd*) were applied together with the inactivation strategy *IAP* were congruent with the results of some of our previous experiments. *IAP*-strategy is successful in controlling the growth of the prototype set. It increases the initial error rate by approximately one percentage unit. Its effects on the final error rate are insignificant when some of the recognition errors are unnoticed but it is truly beneficial in eliminating prototypes introduced because of randomly labeled learning samples.

4. Conclusions

Prototype-based classifiers are due to their very nature vulnerable to erroneous learning samples. Therefore, we have proposed simple control methods for the adaptation of a prototype-based classifier which are able to limit the growth of the prototype count and are helpful in reducing the unwanted effects of erroneous learning samples on the recognition performance.

Control methods which set an upper limit for the number of prototypes were found to be more succesful than methods which switch adaptation on and off depending on the previous performance of the classifier. The best results were obtained if only two new prototypes were allowed for each class and adaptation was continued as LVQ-learning after that limit had been reached. This way, the growth of the prototype count was at least halved. In addition, the final error rate (2%) was satisfactory, less than the average error rate for humans [3] if at most 6% of the learnining samples were randomly labeled, or, 60% of learning samples were labeled according to the recognition results instead of the correct classes. Tolerance to random errors could be further improved up to 8% by employing a prototype inactivation strategy.

References

[1] J. Laaksonen, J. Hurri, E. Oja, and J. Kangas. Comparison of adaptive strategies for on-line character recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pages 245–250, 1998.

[2] J. Laaksonen, V. Vuori, E. Oja, and J. Kangas. Adaptation of prototype sets in on-line recognition of isolated handwritten latin characters. In S.-W. Lee, editor, *Advances in Handwriting Recognition*, pages 489–497. World Scientific Publishing, 1999.

[3] U. Neisser and P. Weene. A note on human recognition of hand-printed characters. *Information and Control*, (3):191–196, 1960.

[4] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, January 2000.

[5] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley, 1983.

[6] V. Vuori. Adaptation in on-line recognition of handwriting. Master's thesis, Helsinki University of Technology, 1999.