Influence of Erroneous Learning Samples on Adaptation in On-line Handwriting Recognition

Vuokko Vuori [a] Jorma Laaksonen [a] Jari Kangas [b]

[a] *Helsinki University of Technology*
Laboratory of Computer and Information Science
P.O. Box 5400, FIN-02015 HUT, Finland
{vuokko.vuori,jorma.laaksonen}@hut.fi

[b] *Nokia Research Center*
P.O. Box 100, FIN-33721 Tampere, Finland
jari.a.kangas@nokia.com

Abstract

We have considered problems involved in the self-supervised learning process of an on-line handwriting recognition system. Our system is able to recognize isolated characters by comparing them to prototype characters with a method based on the Dynamic Time Warping algorithm. The recognition system is adapted by adding new prototypes, inactivating confusing or erroneous ones, and reshaping existing prototypes with a method based on the Learning Vector Quantization. We have analyzed the sources of erroneous learning samples and studied the influence of such samples on the performance of the recognizer via simulations. In these simulations, two adaptation strategies combined with four methods for inactivating prototypes were applied. The results of the simulations showed that the adaptation strategies are able to improve the system's recognition rate and the prototype inactivation methods do reduce the harmful effects of erroneous learning samples.

Key words: isolated character recognition; unconstrained writing style; on-line recognition; intelligent user interface; adaptation; erroneous learning samples; k nearest neighbor rule; Learning Vector Quantization (LVQ); Dynamic Time Warping (DTW)

1 Introduction

Recently, hand-held devices such as PDAs, have became popular. Typically, they are so small that a keyboard cannot conveniently be used as an input method. Therefore, interest in new input methods, especially speech and handwriting recognition, has increased. Of these two, handwriting recognition is more attractive as it is not sensitive to noise from the environment. In addition, it provides better privacy. The major problem in handwriting recognition is the vast variation in personal writing styles. This problem can be solved by either constraining the allowed styles of writing or adapting the system to its user's style. The latter alternative is more convenient to users as they do not have to learn and maintain a writing style different from their own.

Adaptation methods can be divided into two categories: supervised and self-supervised methods. In the former methods, learning is carried out with a special program. The learning program, in which the user enters characters prompted by the program, is run prior to or in between the normal use of the device. On the contrary, self-supervised adaptation is performed during the normal use and therefore it remains unnoticed by the user. The classes of the input characters are in self-supervised learning deduced from the user's behavior and reactions to the recognition results. In addition, a dictionary or language model can be used for checking the correctness of the recognized text [1]. However, they pose constraints on what can be written. As a consequence, inputing text which is not included in the language, such as passwords or email addresses, or using different language than what is assumed can become very difficult. For these reasons, we have not applied language restrictions in our work.

In the work presented here, we have studied the effects of erroneous learning samples on the performance of a self-supervised adaptive on-line recognition system for isolated handwritten characters. Even thought cursive or mixed writing styles would be a more convenient input method for the users, we have concentrated on separately written characters because of the limited memory and processing capacities of the current hand-held devices. Segmentation of cursive handwriting into characters is a difficult and computationally demanding task and usually a language model is required in order to achieve satisfactory recognition performance. Recognition techniques based on Hidden Markov Models (HMM) have shown to be well suited for cursive handwriting recognition as they can handle several alternative character segmentations and can easily be combined with probabilistic language models. For example, recognition accuracies reported in a recent article [2] for an HMM-based system are very promising. However, the recognition time for a single word was far too long to be acceptable in an interactive real-world application. If cursive handwriting is to be recognized at the same speed as it is produced, the recognition time should be less than 2 seconds per word [3].

The recognition system used in our work is based on a prototype set. Characters are represented by sequences of xy-coordinates of the pen point and the input is compared to prototypes with a nonlinear curve-matching method. The advantage of the prototype-based methods over statistical HMM-methods and neural networks is the fact that prototype systems can be adapted quickly: a single character sample is sufficient for learning a new writing style entirely different from those already known. Our recognition system is adapted by adding input characters into the prototype set, inactivating confusing or erroneous prototypes, and reshaping existing prototypes.

This work focuses on the problems encountered in the the supervision of the learning process. We have concerned erroneous learning situations of two different natures, writing mistake corrections and unnoticed recognition errors. Both are very likely to occur. Naturally, spelling errors are not completely avoidable, the user may change his mind on what he is writing in the middle of a word, and the character pairs confused by the recognizer often seem very similar to human readers too. The effects of erroneous learning samples and methods for recovering from them have been studied in the present experiments via simulations by introducing errors in the learning process with a fixed probability.

In the following section 2, the supervision scheme of the adaptive character recognizer is explained and the sources of erroneous learning samples are discussed. The character recognizer itself and its adaptation methods are described in sections 3 and 4, respectively. Sections 5 and 6 introduce the data and the performed experiments. The results of the experiments are given in section 7. Finally, conclusions of the results are drawn in section 8.

2 Sources of erroneous learning samples

The adaptation methods used in our recognition system require that the learning samples are labeled. The labeling is performed in a self-supervised fashion according to the user's behavior and reactions to the recognition results. In the experiments, we assume that the device in which a character recognition system has been implemented has an input subprogram with the following properties:

(1) Input characters are written into the desired positions on the display. Or alternatively, the user first selects the input position by pointing it with a pen and then writes the characters into a special writing area. In either case, it is always clear whether the pen is used as a pointing device or for textual input.
(2) Handwritten characters are recognized and replaced by the machine-printed recognition results right after they have been input.
(3) Recognition errors and writing mistakes are corrected by inputing a new character on top of the machine-printed character.
(4) The input text can be edited by drawing special symbols or gestures. For example, the user might like to remove, move, copy, or paste some parts of the text.
(5) All input characters including removed ones are stored.
(6) The recognition result of the latest input character is assumed to be the correct class for all the characters drawn into the same position.
(7) The adaptation is carried out character by character after an entity of text, for example a line, has been accepted by the user.

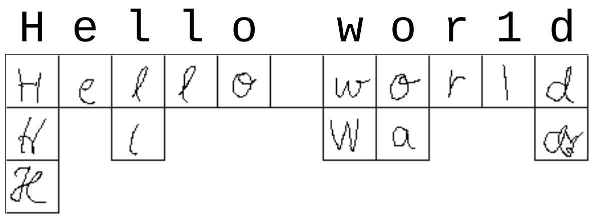


Figure 1. An example of data input. The topmost handwritten characters are the latest inputs into the input locations. The machine printed characters are the corresponding recognition results.

Such an arrangement is illustrated in Figure 1. The machine-printed characters are recognition results, and the handwritten characters beneath them are the corresponding inputs. The topmost handwritten character is the latest input attempt. For example, the first character, 'H', was recognized correctly after the third attempt. All the three input samples are labeled according to the recognition result of the latest one and will be used as learning samples when the user has finished the text sequence. As recognition errors and writing mistakes are corrected by the user in the same way, some of the input characters can be assigned to wrong classes. This could be avoided by using different methods for the two operations but it would be very inconvenient for the user. Therefore, we have studied methods which help the system to recover from such erroneous learning samples.

Erroneous learning situations of three types can take place in this kind of data input. First, the user may not notice all the recognition errors or does not care to correct them. We call this situation the error type *A*. Such a situation occurs in the example shown in Figure 1: the second character from right was intended to be letter 'l' but it was misrecognized and accepted as number '1'. Second, if the user makes writing mistakes and corrects them, some of the learning samples will be incorrectly labeled. This is the error type *B*. In the example, the user has first misspelled the word 'world' as 'warld' and then replaced the correctly-recognized letter 'a' with 'o'. The adaptation system does not now know whether the first character, recognized as 'a' was meant to be 'o' and misrecognized, or not. Third, carelessly written and thus atypical or malformed characters, such as the first input attempt of letter 'd' in the example, give rise to bad learning samples. This is the error type *C*.

There were no malformed learning samples in our experiments as the data had been manually examined and cleaned. Therefore, there are no errors of type *C* left. Erroneous learning samples of type *A* and *B* were artificially generated. With a selected probability, the training samples were labeled according to erroneous recognition results (type *A*) or randomly selected incorrect classes (type *B*). The effects of erroneous learning samples of type *A* and *B* were studied in four experiments described in section 6. First, a very conservative approach to adaptation was taken. Only the input samples corresponding to the latest input attempts were used as learning samples. Next, the risk of erroneous learning samples was increased and all the input characters were used as learning samples. In these two experiments, it was assumed that the user makes no writing mistake corrections and all the erroneous learning samples are of type *A*. The third experiment was otherwise similar to the previous one but the erroneous learning samples were of type *B*. In the last experiment, methods for recovering from erroneous learning samples of type *B* were studied.

3 Prototype-based recognition system

The recognition system used in the experiments is based on prototype matching. It consists of a prototype set covering several writing styles, dissimilarity measure used for comparing the input characters and prototypes, and a decision rule according to which the classifications are carried out. The prototype set was formed by clustering character samples written by 22 subjects and selecting the center most item of each cluster as a prototype. The clustering algorithm [4] was semiautomatic as the number of clusters per each class had to be predefined. The number of clusters, or prototypes, was set to seven for all the character classes as it was the maximum number of different writing styles found for a single class in the manual examination of the data.

The k nearest neighbor (k-NN) [5] rule is used as the classification criterion: the k prototypes most similar to the input character vote for its class. If the dissimilarity measure between the input character and the current prototype exceeds the kth smallest dissimilarity measures evaluated so far, the evaluation procedure can be interrupted. In order to decrease computation time, prototypes can be ordered prior to matching. For this purpose, the prototypes are divided into sixteen categories depending on the rough shape of their first stroke. In particular, the category is defined by the quadrants of coordinate axes drawn on the center of the character. The starting and ending points of the first stroke are studied. A four-bit binary value $b_4b_3b_3b_1$ is constructed for the category in the following way:

$$
\begin{aligned}
b_1 &= (x_1(1) \geq 0) \\
b_2 &= (y_1(1) \geq 0) \\
b_3 &= (x_1(N_1) \geq 0) \\
b_4 &= (y_1(N_1) \geq 0),
\end{aligned}
$$

where N_1 is the number of data points (x_1, y_1) in the first stroke, and true value of an expression corresponds to bit one and false to zero. Thus, sixteen categories are defined. If the value of one of the bits changes, either the starting or ending point moves to a neighboring quadrant. The distance between two categories is the count of bit differences in their binary presentations and its integer value lies within a range from zero to four. Prototypes with a low category distance to the unknown character are, naturally, matched first.

The dissimilarity measure used both in the clustering and classification phases is based on Dynamic Time Warping (DTW) [6] which is suitable for nonlinear curve matching. The dissimilarity measure is defined on stroke basis so that it is infinite between two characters of different number of strokes. Otherwise, it is the sum of the dissimilarity measures between the corresponding strokes. The connected parts of a drawn curve in which the pressure between the pen and writing surface exceeds a given threshold are considered as strokes. The strokes are matched in the same order as they were drawn. DTW-algorithm recursively determines the optimal point-to-point correspondence between the two strokes yielding the minimum sum of the squared Euclidean distances between the matched data points. The matching of the points is restricted by continuity and boundary constraints. The continuity constraints require that the data points are matched in the same order as they have been produced and that all points get matched. According to the boundary constraints, the first and last data points of the input stroke are forced to match the corresponding points in the prototype.

Let us assume that two strokes,

$$S_1 = (p_1(1), \ldots, p_1(N_1)) \text{ and } S_2 = (p_2(1), \ldots, p_2(N_2)), \tag{1}$$

are matched against each other. Stroke S_k consists of N_k data points, $p_k(l)$ is the lth one. Time warping path

$$P(h) = (i(h), j(h)),\ 1 \leq h \leq H, \tag{2}$$

describes how the point-to-point correspondence is established between the strokes: H is the number of required matchings, h is an order index of a matching, and the $i(h)$th and $j(h)$th data points of strokes S_1 and S_2 are matched against each other, respectively. The boundary and continuity constraints are satisfied when

$$\begin{aligned} P(1) &= (1, 1), \\ P(H) &= (N_1, N_2) \end{aligned} \tag{3}$$

and

$$P(h) - P(h-1) = \begin{cases} (1, 0) \text{ or} \\ (0, 1) \text{ or} \\ (1, 1). \end{cases} \tag{4}$$

A cost $d(p_1, p_2)$ for matching two data points is their squared Euclidean distance. The dissimilarity measure $D(S_1, S_2)$ between the two strokes is the minimum sum of the point matching costs:

$$D(S_1, S_2) = \min_{P(h)} \sum_{h=1}^{H} d(p_1(i(h)), p_2(j(h))). \tag{5}$$

The above minimization problem can be solved applying dynamic programming in the following way:

$$D(i, j; h) = d(p_1(i), p_2(j)) + \min \begin{cases} D(i-1, j; h-1) \\ D(i, j-1; h-1) \\ D(i-1, j-i; h-1), \end{cases} \tag{6}$$

when $D(i, j; 0)$ are initialized to zero

$$D(i, j; 0) = 0, \forall i, j \tag{7}$$

and the undefined distances are set to infinity

$$d(p_1(i), p_2(j)) = \infty \text{ , if } i \leq 0, j \leq 0, j > N_1, \text{ or } j > N_2. \tag{8}$$

$D(i, j; h)$ is the dissimilarity of optimally-matched stroke parts consisting of points $(p_1(1), \ldots, p_1(i(h)))$ and points $(p_2(1), \ldots, p_2(j(h)))$, and h is the number of required point-to-point matchings. The total dissimilarity measure between the strokes is thus

$$D(S_1, S_2) = D(N_1, N_2; H). \tag{9}$$

4 Adaptation strategies and inactivation methods

The recognition system is adapted to a new writing style by modifying its prototype set. If an existing prototype is basically similar to the input characters, for example has the same number and order of strokes, it can be reshaped so that it resembles more the user's writing style. Completely new writing styles can be learned by simply adding input characters into the prototype set as such. Some character classes are easily confused, for example 'l' (el) and '1' (one), as their shapes can be very similar to each other. However, if the user writes them differently enough, the confusing prototypes can be inactivated. In addition, the system can recover from erroneous learning samples by inactivating those prototypes which contribute more often to incorrect than correct classifications. In the experiments, we have applied two different adaptation strategies which can be combined with any one of four methods for inactivating prototypes. The adaptation strategies and inactivation methods will be described next.

4.1 Adaptation strategies

Adaptation strategy named *Adding New Prototypes* examines the classes of the k prototypes nearest to the input character. The classification is carried out according to the k-NN rule. The input character is added to the prototype set if any one of these prototypes belongs to a wrong class, even if the classification was correct. If the classes of some input characters are interpreted wrongly from the user's behavior, mislabeled prototypes are in this manner very likely to be added to the prototype set.

Hybrid Prototypes-strategy is based on a modified version of the Learning Vector Quantization (LVQ) [7, 8]. The classes of the k nearest prototypes to the input character are again examined. Also in this case, k-NN rule is used as the classification criterion. If the class of any of the k nearest prototypes is correct, the nearest prototype is reshaped. If the nearest prototype is of the correct class, it is reshaped by moving its data points towards the corresponding points of the input character. Otherwise, the data points of the nearest prototype are moved in the opposite direction. If all of the k nearest prototypes belong to incorrect class or classes, the input character is added to the prototype set. If the learning sample is erroneous, either a good prototype is reshaped in the opposite way it should be or a mislabeled prototype is introduced into the prototype set.

Suitable parameter values for *Adding New Prototypes*- and *Hybrid Prototypes*-strategies were selected in previous experiments [9] the results of which are summarized here in Table 1. The prototype set used in those experiments was the same one as used in the experiments reported in this work.

4.2 Inactivation methods

Inactivating Prototypes 1-4 are used for recovering from confusing or erroneous prototypes. Each of these methods keeps an account for each prototype of the times it has been used, i.e. been the nearest prototype (*Inactivating Prototypes 1* and *3*), or among the k nearest ones (*Inactivating Prototypes 2* and *4*), and of the times its class in these cases has been incorrect and correct. After the user has input to system a character which then has been classified and the correctness of the classification deduced from the user's ensuing activities, potential inactivation of the prototypes involved in the classification decision can take place. A prototype is inactivated if the following three conditions are fulfilled simultaneously: 1) it has been used sufficiently many times, i.e. it has reached its usage limit, 2) its class has been incorrect more often than correct, and 3) it is not the last prototype of its class. *Inactivating Prototypes 1*- and *2*-methods apply the same usage limit for all prototypes whereas *Inactivating Prototypes 3*- and *4*-methods have separate

Table 1
Selection of suitable parameter values for *Adding New Prototypes*- and *Hybrid Prototypes*-strategies. E_{ave} is the average error percentage and E_{best} and E_{worst} are the error percentages for the best and worst results for 16 writers: parameter k is the number of nearest neighbors according to which the classification and adaptation is performed and α is the learning parameter of LVQ. The upper part of the table corresponds to *Adding New Prototypes*-strategy and the lower part to *Hybrid Prototypes*-strategy. The emphasized parameter values were used in the later experiments.

k	α	E_{ave}	E_{best}	E_{worst}
1	–	4.61	1.37	10.50
2	–	3.40	1.03	9.07
3	–	3.09	0.60	9.25
4	–	**2.93**	**0.51**	**8.90**
5	–	3.34	0.51	9.43
6	–	4.00	0.69	10.85
3	0.15	3.52	0.77	8.19
3	0.20	3.36	0.69	7.83
3	**0.30**	**3.06**	**0.69**	**7.65**
3	0.35	3.18	0.60	7.47
3	0.40	3.29	0.60	8.01
4	0.15	3.67	0.77	9.96
4	0.20	3.53	0.69	9.25
4	0.30	3.21	0.69	8.90
4	0.35	3.36	0.60	8.36
4	0.40	3.54	0.60	9.43

limits for the original prototypes and for those which have been included into the prototype set later on due to adaptation. Erroneous learning samples cause errors in the prototype accounts. Therefore, some prototypes will be inactivated sooner, or later, than they would if all the samples were correctly labeled. In the worst case, prototypes are inactivated even though they are performing well for correct data. All of these inactivation methods can be used together with either *Adding New Prototypes*- or *Hybrid Prototypes*-strategies. The properties of the prototype inactivation methods are summarized in Table 2.

Table 2
Summary of the properties of the prototype inactivation methods. A bullet in the second or third column indicates whether the method keeps an account of the number of times a prototype has been the best matching one or among the k best matching ones. A bullet in the fourth or fifth column marks if the usage limit is the same for all prototypes or whether there are separate limits for the original prototypes and for those which have been included into the prototype set due to adaptation.

Inactivation method	1-NN	k-NN	one usage limit	two usage limits
Inactivating Prototypes 1	•		•	
Inactivating Prototypes 2		•	•	
Inactivating Prototypes 3	•			•
Inactivating Prototypes 4		•		•

Table 3
Summary of the databases used in the experiments. The counts of lower case letters and digits are given in the last two columns.

Database	*Subjects*	♮*(a-z,å,ä,ö)*	♮*(0-9)*
Database 1	22	7644	817
Database 2, writers 1-16	16	9035	3137
Database 2, writers 17-24	8	3448	1195

5 Data

All characters were collected with a pressure sensitive Wacom ArtPad II tablet attached to a Silicon Graphics workstation. The resolution of the tablet is 100 lines per millimeter and the sampling rate is at maximum 205 data points per second. The loci of the pen point movements consist of the x- and y-coordinates, pen's pressure against the writing surface, and time stamp. The writing area was a rectangle of size 50 mm $\times$ 20 mm placed at the center of the tablet. The characters were written one at a time. Writers were advised to use their natural handwriting style. All the data were saved in UNIPEN format [10]. The essential details of the databases are summarized in Table 3.

Database 1 consists of characters which were written without any visual feedback. The pressure level thresholding the pen movements into pen up and pen down movements was set afterwards individually for each writer. The a priori probability distribution of the classes was somewhat similar to that of the Finnish language. Database 2 was collected with a program which showed the pen trace on the screen and recognized the characters on-line. The minimum writing pressure for showing the trace of the pen on the screen and detecting pen down movements was same for all writers. The *a priori* probability of the character classes was nearly even. None of the writers of Database 1 appeared in Database 2.

Database 1 was used for forming the initial prototype set. Lower case letters and digits of the first 16 writers of Database 2 were used as a test set for selecting suitable parameter values for *Adding New Prototypes*- and *Hybrid Prototypes*-strategies. Lower case letters of all writers of Database 2 were used as a test set in the experiments with the prototype inactivation methods presented in this work.

Prior to clustering and matching, the characters were preprocessed and normalized. The characters were preprocessed so that the number of data points was reduced by keeping every third data point and abandoning the intermediate ones. The normalization operations included centering and scaling of the characters. The mass centers of the characters were shifted to the origin of the coordinate system. The characters were then 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. These operations were justified by the results of some previous experiments [9].

6 Experiments

We carried out a series of four experiments in which a recognition system with an input setup such as described in section 2 was simulated. Some of the characters were artificially labeled incorrectly in order to simulate erroneous learning samples of type *A* or *B*. There was no meaningful context for the characters as they had been collected in a random order. Therefore, the system was adapted after each sample instead of some longer text entity. All the experiments were repeated ten times with different random labeling errors in order to even out statistical variations.

In Experiment 1, the risk of erroneous learning samples was minimized by using only the topmost, or latest, character of each input position as a learning sample. The classes of these character samples have been accepted by the user and are correct unless the user did not notice the recognition error. This corresponds to error type *A*. The experiment can be characterized with a single parameter which is the probability that a recognition error is not noticed. The system learned after all successful recognitions, and with the given error probability, also after an unsuccessful recognition. In the latter case, the erroneous recognition result was used as the label of the learning sample. The error probability was varied from zero to one. The learning capabilities of two adaptation strategies, namely *Adding New Prototypes* and *Hybrid Prototypes*, were tested for each of the subjects of the test set by recognizing the characters one by one. No prototype inactivation was applied.

In Experiment 2, it was assumed that the user makes no writing mistake corrections. The aim of this experiment was to study whether it is beneficial to use not only the topmost but also the previous unsuccessful input attempts as learning samples. Again, with a given probability, the misrecognized learning samples were labeled according to the recognition results instead of the correct classes. The probability of an unnoticed recognition error, i.e. error type A, was varied from zero to one. The experiment was again performed with *Adding New Prototypes*- and *Hybrid Prototypes*-strategies and without prototype inactivation.

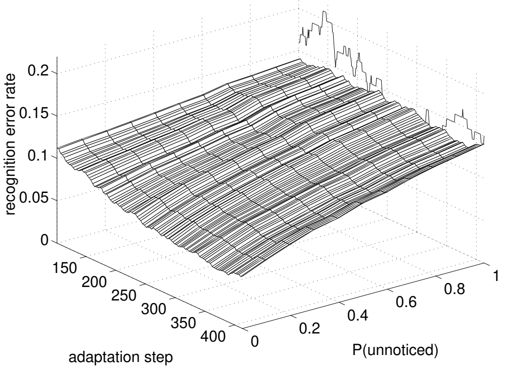
In Experiment 3, the probability of erroneous learning samples was further increased by assuming that the writer makes writing mistake corrections. They were simulated by randomly assigning incorrect labels to the learning samples independently from the recognition results. This corresponds to error type B. The probability of such an error situation was varied from zero to ten percent. There were no error situations of type A. This experiment was carried out in order to study the risks involved in using the same input procedure for correcting both writing mistakes and recognition errors.

In Experiment 4, the prototype inactivation methods *Inactivating Prototypes 1-4* were tested together with *Adding New Prototypes*- or *Hybrid Prototypes*-strategy. As *Adding New Prototypes*-strategy only adds new prototypes into the prototype set, its modifications on the prototype set can easily be reversed. *Hybrid Prototypes*-strategy performs prototype reshaping which is difficult to undo. In order to accomplish that, all the learning samples, the order of their appearances, and information on which prototypes have been reshaped according to them should be stored. Due to the complicatedness of the action, the reversion of reshaping was not implemented in our work. A prototype inactivation method was performed right after each recognition and the application of either *Adding New Prototypes*- or *Hybrid Prototypes*-strategy. The aims of this experiment were to find good usage limits for the prototype inactivation methods and to study how well they can help the system to recover from erroneous learning samples.

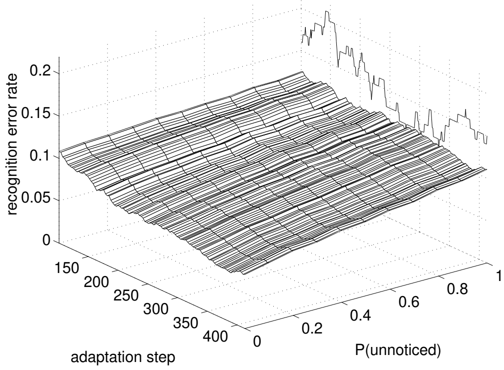
7 Results

Without any adaptation the recognition error rate was on the average approximately 15%. The behavior of a nonadaptive recognizer is naturally independent of the probability of erroneous learning samples. The recognition rate of the nonadaptive system is illustrated with a background plot in the Figures 2-5 so that the benefits of the adaptation can easily be evaluated. All the error rates shown in these figures were calculated by using a sliding back-oriented window of 100 character samples. The initial and final error rates mentioned in the following paragraphs are averages calculated for the first and last 100 characters, respectively, in each writer's input.

The evolution of the recognition error rate in Experiment 1 is illustrated in Figure 2. According to the results, the recognition system is to some extend able to learn from the character samples successfully recognized at the first input attempt. If *Adding New Prototypes*- or *Hybrid Prototypes*-strategy was applied and all the recognition errors were noticed, the initial recognition error rate was about 11.2% or 10.8%



(a) *Adding New Prototypes*-strategy

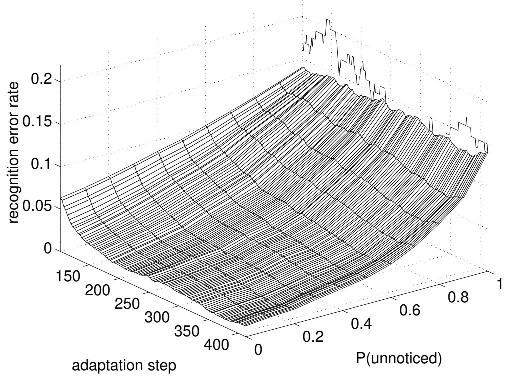


(b) *Hybrid Prototypes*-strategy

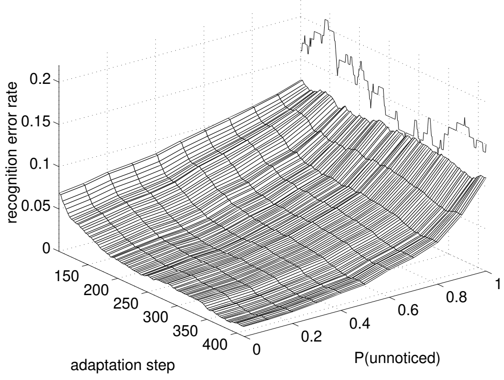
Figure 2. Results of Experiment 1. Evolution of the recognition error rate as a function of the number of input characters used in adaptation (adaptation step) and the probability of unnoticed recognition errors (P(unnoticed)). Performances of nonadaptive and adaptive system are illustrated with a background plot and surface, respectively.

whereas the final error rate was 6.5% or 6.6%. The number of erroneous learning samples had a nearly linear effect on the recognition error rate. If all the recognition errors were unnoticed, the recognition error rate did not change much after 100 input characters and was at the end of the adaptation 14.2% for *Adding New Prototypes*- and 11.3% for *Hybrid Prototypes*-strategy. The size of the prototype set increased on the average by 52% when adaptation strategy *Adding New Prototypes* was applied. Due to the conservative learning strategy which minimized the probability of erroneous learning samples, *Hybrid Prototypes*-strategy was not able to add any new prototypes and the size of the prototype set remained unchanged.

The evolution of the recognition error rates during Experiment 2 are illustrated in Figure 3. The experiment showed that the misrecognized input characters are very valuable. When all recognition errors were noticed, both *Adding New Prototypes*- and *Hybrid Prototypes*-strategy were able to reduce the recognition error rate considerably more than in Experiment 2. For *Adding New Prototypes*-strategy, the initial and final error rates were 6.3% and 1.3%, respectively. For *Hybrid Prototypes*-strategy these figures were slightly worse: 6.9% and 1.6%. The relation between the recognition error rate and the probability of not noticing recognition errors was now not linear as it was in Experiment 1. If the probability was 0.5, the final error rate was approximately 3.2% in case of *Adding New Prototypes*-strategy and 2.6% for *Hybrid Prototypes*-strategy. These result are still significantly better than those attained in Experiment 1. However, if all recognition errors were unnoticed, the results of Experiment 1 and Experiment 2 were similar to each other. *Adding New Prototypes*-strategy provided faster improvement in the recognition accuracy but it increased the size of the prototype set on the average by 63% whereas *Hybrid Prototypes*-strategy increased it only by 2%.



(a) *Adding New Prototypes*-strategy

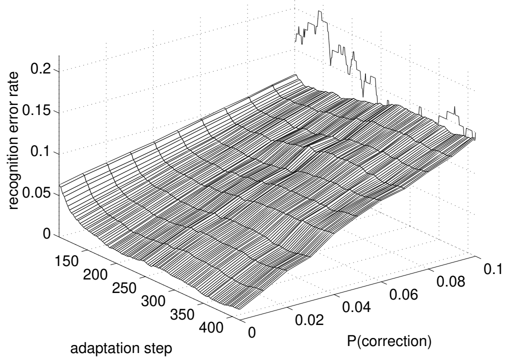


(b) *Hybrid Prototypes*-strategy

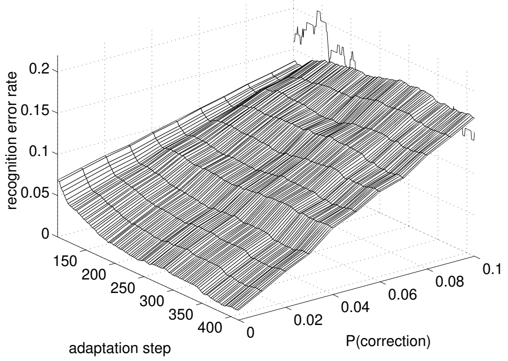
Figure 3. Results of Experiment 2. Evolution of the recognition error rate as a function of the number of input characters used in adaptation (adaptation step) and the probability of unnoticed recognition errors (P(unnoticed)). Performances of nonadaptive and adaptive system are illustrated with a plot and surface, respectively.

The results of Experiment 3 are shown in Figure 4. The results of Experiment 3 were equally good as the results of Experiment 2 when the probability of erroneous learning samples was zero. However, the misleading writing mistake corrections had more severe effect on the recognition accuracy than the unnoticed recognition errors. As the recognition error rate decreases due to adaptation, the unnoticed recognition errors become more rare whereas the frequency of simulated writing mistake corrections remains constant. The relation between the recognition error rate and the probability of writing mistake corrections was almost linear. If ten percent of the input characters were labeled incorrectly, the initial and final error rates were 11.9% and 14.6%, respectively, for *Adding New Prototypes*-strategy. When *Hybrid Prototypes*-strategy was applied instead, these error rates were 13.4% and 18.4%. The results obtained with *Adding New Prototypes*-strategy are better because it does not ruin good prototypes by reshaping them wrongly as *Hybrid Prototypes*-strategy does. With both strategies the recognition error rate decreased at the beginning of the adaptation but started to increase quite soon if the probability of mistake corrections was too high. Such behavior occurred with *Adding New Prototypes*-strategy if the probability was over approximately three percent and with *Hybrid Prototypes*-strategy if it was over four percent. The size of the prototype set increased on the average by 106% and 15% with *Adding New Prototypes* and *Hybrid Prototypes*, respectively.

The evolution of the recognition error rates during Experiment 4 are illustrated in Figure 5. The experiment showed that the proposed prototype inactivation methods do help the system to recover from the erroneous learning samples. There was no significant change in the system's recognition performance whether a separate lower usage limit was used for the prototypes added due to adaption (*Inactivating Prototypes 3* and *4*) or whether the same limit was used for all prototypes (*Inactivat-*



(a) *Adding New Prototypes*-strategy



(b) *Hybrid Prototypes*-strategy

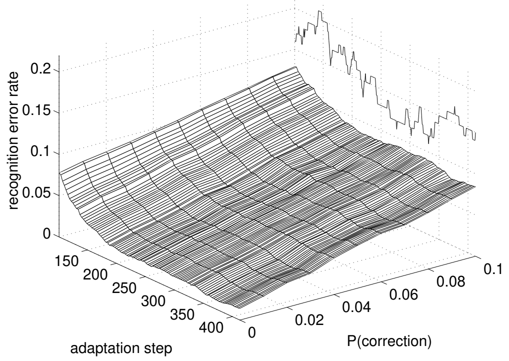
Figure 4. Results of Experiment 3. Evolution of the recognition error rate as a function of the number of input characters used in adaptation (adaptation step) and the probability of misleading writing mistake corrections (P(correction)). Performances of nonadaptive and adaptive system are illustrated with a plot and surface, respectively.

ing Prototypes 1 and *2*). This result showed that it is equally important to inactivate confusing initial prototypes and to recover from the erroneous learning samples.

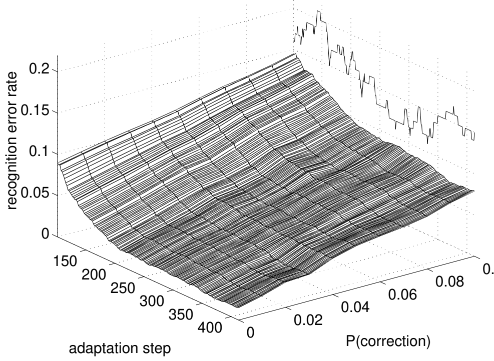
The best results were obtained if a small value was used for the usage limit: one for *Inactivating Prototypes 1* and *3*, two for *Inactivating Prototypes 2* and *4*. The recognition accuracy for the first 100 characters was higher when the recognition results were recorded in the accounts of the nearest prototypes only (*Inactivating Prototypes 1* and *3*) compared to recording them in the accounts of all the k nearest prototypes (*Inactivating Prototypes 2* and *4*). The highest initial accuracy was obtained with no prototype inactivation at all. When the final accuracies were calculated for the last 100 characters, the results were just the opposite. The relation between the probability of erroneous learning samples and the resulting error rate was nearly linear.

The following results were obtained with *Adding New Prototypes*-strategy. As there was no benefit in using two different usage limits in prototype inactivation, the results mentioned for *Inactivating Prototypes 1* and *2* apply also for *Inactivating Prototypes 3* and *4*, respectively. If the probability of writing mistake correction was zero, the initial error rate was 7.6% and 8.9% with prototype inactivation methods *Inactivating Prototypes 1* and *2*, respectively. The final error rate was 1.3% with both inactivation methods. If ten percent of the learning samples were incorrectly labeled, the initial error rates were 12.8% and 14.3%. The final error rate was 8.3% with former and 7.5% with the latter inactivation method. The size of the prototype set increased on the average only by 69% and 61%, respectively.

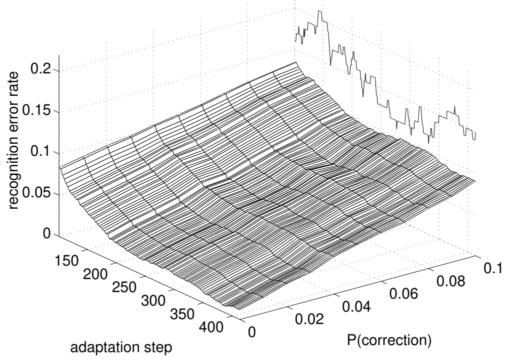
Results obtained with *Hybrid Prototypes*-strategy were only slightly worse than those obtained with *Adding New Prototypes*-strategy even though all modifications on the prototype set could not be completely reversed by the prototype inactiva-



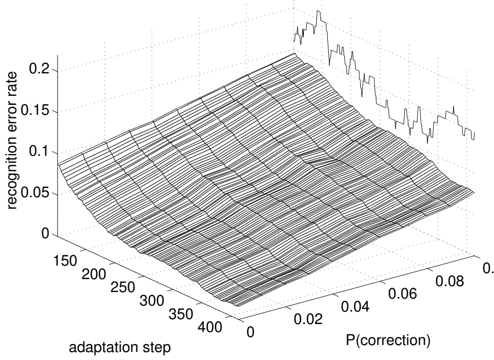
(a) *Adding New Prototypes*-strategy and *Inactivating Prototypes 1*-method



(b) *Adding New Prototypes*-strategy and *Inactivating Prototypes 2*-method



(c) *Hybrid Prototypes*-strategy and *Inactivating Prototypes 1*-method



(d) *Hybrid Prototypes*-strategy and *Inactivating Prototypes 2*-method

Figure 5. Results of Experiment 4. Evolution of the recognition error rate as a function of the number of input characters used in adaptation (adaptation step) and the probability of misleading writing mistake corrections (P(correction)). Performances of nonadaptive and adaptive system are illustrated with a plot and surface, respectively.

tion methods. The size of the prototype set increased on the average by only 5% with *Inactivating Prototypes 1*-method and by 12% with *Inactivating Prototypes 2*-method. If the probability of writing mistake correction was zero, the initial error rate was 8.4% and 8.8% with *Inactivating Prototypes 1*- and *2*-method, respectively. The final error rate was 1.9% with *Inactivating Prototypes 1* and 1.6% with *Inactivating Prototypes 2*. If ten percent of the learning samples were erroneous, the initial error rate was 13.3% and 14.4% and the final error rate was 9.5% and 7.8% with the inactivation methods *1* and *2*, respectively.

8 Conclusions

We have described and typified sources of erroneous learning samples in the adaptation of an on-line character recognizer. Also, we have discussed the problems they cause in the learning process. Erroneous learning samples of two kinds, namely characters mislabeled due to unnoticed recognition errors and wrongly interpreted writing mistake corrections, have been concerned more deeply. We have described an experiment setting with which we have been able to simulate and test the effects of such spurious learning samples on our recognition system. We believe that the experimental results presented here for a handwritten character recognition problem can to some extend be generalized to other tasks too. The labeling of learning samples without any direct interaction with the user and the problem of erroneous learning samples are common to all prototype- or case-based recognition systems which are adapted with supervised learning techniques.

According to the results of the performed experiments as well as our earlier studies [9], the system is able to learn new writing styles well if all input characters are used as learning samples and none of them is erroneously labeled. In such a situation, the average recognition accuracy for 24 subjects was over 98% after adaptation with any of the proposed strategies while the average accuracy prior to adaptation was only 85%. The achieved accuracy is high enough to be accepted by a user in a practical application. However, the system's ability to improve its accuracy via adaptation decreases significantly if some of the learning samples are erroneous.

If the the risk of erroneous learning samples is minimized by adapting the system conservatively with only the characters whose recognition result the user has accepted, the recognition accuracy does not improve enough. Even if the user noticed and corrected all recognition errors, the recognition accuracy was at the end only 93%. If it is assumed that the user makes no writing mistake corrections, the results are always better when characters which were not correctly recognized in the first input attempt are also used in the adaptation even though the risk of using erroneous learning samples simultaneously increases. This result can be justified by the fact that the misrecognized characters contain more valuable information on the user's writing style than the characters already familiar to the recognition system.

Erroneous learning samples caused by the user when he makes writing mistake corrections, were found to be more harmful than those caused by unnoticed recognition errors. If the probability of writing mistake corrections was too high, more than approximately 3-4 percent, the recognition accuracy improved at the beginning of the adaptation but started to deteriorate after a while and was at the end even worse than without adaptation. This problem and other unwanted effects of erroneous learning samples could be avoided fairly well with the proposed prototype inactivation methods. The experiments showed that it is equally important to inactivate the mislabeled prototypes introduced into the prototype set due to erroneous learning samples and the initial correctly-labeled but otherwise confusing prototypes. In this way, the recognition accuracy continued improving even though the probability of erroneous learning samples was as high as ten percent of all the training samples. We found out that the proposed prototype inactivation methods work best when they are applied aggressively: prototypes should be inactivated right after they have contributed to only one or two unsuccessful recognitions.

References

[1] Stéphanie Clergeau and Réjean Plamondon. Integration of lexical and syntactical knowledge in a handwriting-recognition system. *Machine Vision and Applications*, 8:249–259, 1995.

[2] Jianying Hu, Sok Gek Lim, and Michael K. Brown. Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition*, 33:133–147, 2000.

[3] I. Scott MacKenzie, Blair Nonnecke, Stan Riddersma, Craig McQueen, and Malcom Meltz. Alphanumeric entry on pen-based computers. *International Journal of Human-Computer Studies*, 41:775–792, 1994.

[4] Jorma Laaksonen, Vuokko Vuori, Erkki Oja, and Jari Kangas. Adaptation of prototype sets in on-line recognition of isolated handwritten latin characters. In Seong-Whan Lee, editor, *Advances in Handwriting Recognition*, pages 489–497. World Scientific Publishing, 1999.

[5] 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.

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

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

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

[9] Vuokko Vuori. Adaptation in on-line recognition of handwriting. Master's thesis, Helsinki University of Technology, January 1999. http://www.cis.hut.fi/ vuokkov/hcr/dtyo.ps.

[10] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmark. In *Proceedings of International Conference on Pattern Recognition*, pages 29–33, 1994.