Electronic Letters on Computer Vision and Image Analysis 3(1):25-41, 2004

Optimization of Weights in a Multiple Classifier Handwritten Word Recognition System Using a Genetic Algorithm

Simon Günter and Horst Bunke

Department of Computer Science, University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland

Received 21 November 2002; revised 18 November 2003; accepted 22 January 2004

Abstract

Automatic handwritten text recognition by computer has a number of interesting applications. However, due to a great variety of individual writing styles, the problem is very difficult and far from being solved. Recently, a number of classifier creation methods, known as ensemble methods, have been proposed in the field of machine learning. They have shown improved recognition performance over single classifiers. For the combination of these classifiers many methods have been proposed in the literature. In this paper we describe a weighted voting scheme where the weights are obtained by a genetic algorithm.

Key Words: Handwritten Text Recognition, Classifier Combination, Hidden Markov Models (HMM), Weighted Voting, Genetic Algorithm.

1 Introduction

The field of off-line handwriting recognition has been a topic of intensive research for many years. First only the recognition of isolated handwritten characters was investigated [1], but later whole words [2] were addressed. Most of the systems reported in the literature until today only consider constrained recognition problems based on small vocabularies from specific domains, e.g. the recognition of handwritten check amounts [3] or postal addresses [4]. Free handwriting recognition, without domain specific constraints and large vocabularies, was addressed only recently in a few papers [5, 6]. The recognition rate of such systems is still low, and there is a need to improve it.

The combination of multiple classifiers has become a very active area of research recently [7, 8]. It has been demonstrated in a number of applications that using more than a single classifier in a recognition task can lead to a significant improvement of the system's overall performance. Hence multiple classifier systems seem to be a promising approach to improve the recognition rate of current handwriting recognition systems. Concrete examples of multiple classifier systems in handwriting recognition include [9, 10, 11, 12, 13, 14, 15, 16].

To build a multiple classifier system, one needs a number of basic classifiers first. Very often, the design of these basic classifiers is guided by intuition and heuristics. Sometimes, different sources of information, which are redundant or partly redundant to each other, are exploited, for example, zip code and city name in address

Correspondence to: bunke@iam.unibe.ch

Recommended for acceptance by Josep Lladós
ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

reading [4], or legal and courtesy amount in bankcheck processing [3]. Recently, a number of procedures for classifier generation, called ensemble creation methods, were proposed in the field of machine learning. A summary of these methods is given in [17]. They are characterized by the fact that they produce several classifiers out of one given base classifier automatically. Given a base classifier, an ensemble of different classifiers can be generated by changing the training set [18], the input features [19], the input data by injecting randomness [20] or the parameters and architecture of the base classifier [21].

In a multiple classifier system for handwriting recognition, each of the basic classifiers first generates, as its output, one or several hypotheses about the identity of the unknown word to be recognized. Next, these outputs need to be appropriately combined to derive the final recognition result. There are many ways to combine the results of a set of classifiers, depending on the type of the classifiers' output [22, 23]. If the output is only the best ranked class then majority voting can be applied. More sophisticated voting schemes also look at the probability of the classification error for a specific class (Bayesian Combination Rule [24]), or dependencies between the classifiers (Behavior-Knowledge Space [25]). Some classifiers have a ranked list of classes as output. In this case often Borda count [26] or related methods are used. In the most general situation, a classifier generates a score value for each class. Then the sum, product, maximum, minimum, or the median of the scores of all classifiers can be calculated and the class with the highest value is regarded as the combined result [24]. It is also possible to first weight each classifier according to its individual performance and then apply a combination rule [27].

Automatic classifier ensemble generation methods together with related combination schemes have rarely been applied in the field of cursive handwriting recognition until now. In this paper we propose a framework where the individual base classifiers are given by hidden Markov Models (HMMs) [28]. This kind of classifier has shown superior performance over other approaches in many handwriting recognition tasks. The proposed multiple classifier system is distinguished from many other classifiers described in the literature in that it has to deal with a large number of classes. (In the experiments described in Section 6 a recognition problem with over 2000 words, i.e. pattern classes, was considered.) This restricts the number of possible classifier combination schemes. For example, considering class specific error rates in the combination method, as it was proposed in [23], is no longer feasible because of its low reliability in case of a high number of classes. Further constraints on possible combination schemes are imposed by the use of HMMs as base classifiers. In our framework, only the class on the first rank together with its score is returned by each individual HMM classifier. Therefore, Borda count, as well as sum, product, and median rule can't be applied. Yet weighted voting is feasible for this problem. It is, in fact, the most general form of classifier combination available in the proposed framework.

In weighted voting, each classifier has a single vote for its top ranked class, and this vote is given a weight. To derive the final decision in a multiple classifier system using weighted voting, the weights assigned to each class by the different classifiers are summed up and the class with the highest score is selected as the final result. Under a weighted voting scheme, the weights assigned to the individual classifiers are free parameters. Sometimes these weights are chosen proportional to the recognition performance of individual classifiers. In this paper, we apply a more general approach where the weights are considered as parameters which are to be selected in such a way that the overall performance of the combined system is optimized. A genetic algorithm is used to actually determine an optimal (or suboptimal) combination of weight values. Also in [29] a genetic algorithm was used for weight optimization in a multiple classifier system. However, an easier recognition problem was considered there, i.e. the application was the recognition of handwritten digits and the combined classifiers were not created by an ensemble creation method, but were each separately designed by hand. In [30] a genetic algorithm was used for the selection of a subset of classifiers from an ensemble, which is equivalent to weight optimization using only the weights 0 and 1. Another application of a genetic algorithm in a multiple classifier framework has been proposed in [16]. In this work, a genetic algorithm was used to select individual classifiers from a pool for the different modules of a multiple classifier framework.

The remainder of this paper is organized as follows. In Section 2 our base classifier, which is a handwritten word recognizer based on hidden Markov Models (HMMs), is introduced. The following section describes the methods used to produce classifier ensembles from the base classifier. Then the classifier combination

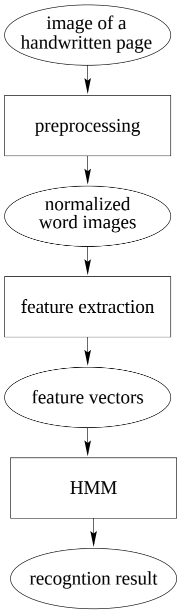


Figure 1: System overview

schemes used in this work are introduced in Section 4. The genetic algorithm for the calculation of the weights applied in the weighted voting combination scheme is described in Section 5. In Section 6 experimental results comparing genetic weight optimization with other combination schemes are presented. Finally the last section draws conclusions from this work.

2 Handwritten word recognizer

The basic handwritten text recognizer used in the experiments of this paper is similar to the one described in [6]. It follows the classical architecture and consists of three main modules (see Fig. 1): the preprocessing, where noise reduction and normalization take place, the feature extraction, where the image of a handwritten text is transformed into a sequence of numerical feature vectors, and the recognizer, which converts these sequences of feature vectors into a word class.

The first step in the processing chain, the preprocessing, is mainly concerned with text image normalization. The goal of the different normalization steps is to produce a uniform image of the writing with less variations of the same character or word across different writers. The aim of feature extraction is to derive a sequence of feature vectors which describe the writing in such a way that different characters and words can be distinguished, but avoiding redundant information as much as possible. In the presented system the features are based on geometrical measurements. At the core of the recognition procedure is an HMM. It receives a sequence of feature vectors as input and outputs a word class. In the following these modules are described in greater detail. In the Appendix a small subset of the words used in the experiments described in Section 6 are shown.

2.1 Preprocessing

Each person has a different writing style with its own characteristics. This fact makes the recognition task complicated. To reduce variations in the handwritten texts as much as possible, a number of preprocessing operations are applied. The input for these preprocessing operations are images of words extracted from the database described in [31, 32]. In the presented system the following preprocessing steps are carried out:

- Skew Correction: The word is horizontally aligned, i.e. rotated, such that the baseline is parallel to the

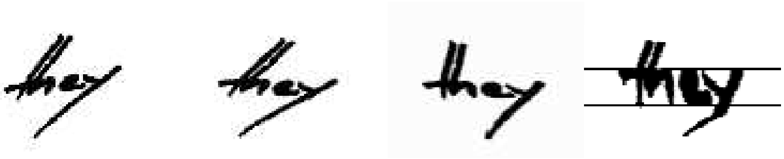


Figure 2: Preprocessing of the images. From left to right: original, skew corrected, slant corrected and positioned. The two horizontal lines in the right most picture are the two baselines.

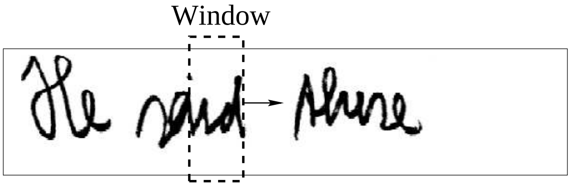


Figure 3: Illustration of the sliding window technique. A window is moved from left to right and features are calculated for each position of the window. (For graphical representation purposes, the window depicted here is wider than one pixel.)

x-axis of the image.

- Slant Correction: Applying a shear transformation, the writing's slant is transformed into an upright position.
- Line Positioning: The word 's total extent in vertical direction is normalized to a standard value. Moreover, applying a vertical scaling operation the location of the upper and lower baseline are adjusted to a standard position.

An example of these normalization operations is shown in Fig. 2. For any further technical details see [6].

2.2 Feature Extraction

To extract a sequence of feature vectors from a word, a sliding window is used. The width of the window used in the current system is one pixel and its height is equal to the word's height. The window is moved from left to right over each word. (Thus there is no overlap between two consecutive window positions.) Nine geometrical quantities are computed and used as features at each window position. A graphical representation of this sliding window technique is shown in Fig. 3.

The first three features are the weight of the window (i.e. the number of black pixels), its center of gravity, and the second order moment of the window. This set characterizes the window from the global point of view. It includes information about how many pixels in which region of the window are, and how they are distributed. The other features represent additional information about the writing. Features four and five define the position of the upper and the lower contour in the window. The next two features, number six and seven, give the orientation of the upper and the lower contour in the window by the gradient of the contour at the window's position. As feature number eight the number of black-white transitions in vertical direction is used. Finally, feature number nine gives the number of black pixels between the upper and lower contour. Notice that all these features can be easily computed from the binary image of a text line. However, to make the features robust against different writing styles, careful preprocessing, as described in Subsection 2.1, is necessary.

To summarize, the output of the feature extraction phase is a sequence of 9-dimensional feature vectors. For each word to be recognized there exists one such vector per pixel along the x-axis, i.e. along the horizontal extension of the considered word.

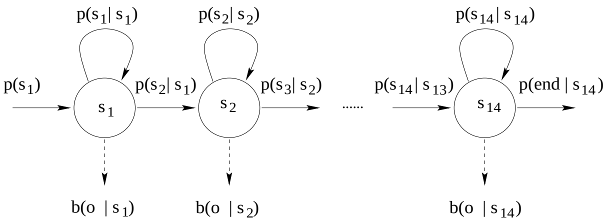


Figure 4: HMM for a single character with linear transition structure

2.3 Hidden Markov Models

Hidden Markov models (HMMs) are widely used in the field of pattern recognition. Their original application was in speech recognition [33]. But because of the similarities between speech and cursive handwriting recognition, HMMs have become very popular in handwriting recognition as well [34].

When using HMMs for a classification problem, an individual HMM is constructed for each pattern class. For each observation sequence, i.e. for each sequence of feature vectors, the likelihood that this sequence was produced by an HMM of a class can be calculated. The class whose HMM achieved the highest likelihood is considered as the class that produced the actual sequence of observations.

An HMM consists of a set of states and transitions probabilities between those states. One or several of the states are defined as final states. For each state a likelihood value for each possible observation is defined. If there is a finite number of observations then a probability for each observation, i.e. feature vector, is defined, but if we have continuous observation vectors a probability distribution is used. A valid sequence of states for a observation sequence $o_{seq} = o_1, o_2, \ldots, o_n$ is $s_{seq} = s_1, s_2, \ldots, s_n$ where s_n is a final state. Note that the number of states in s_{seq} is the same as the number of observations in o_{seq}. The likelihood of the sequence of states s_{seq} is the product of the likelihoods of observing o_i in state s_i for all observations, multiplied by the probabilities of the transitions from state s_i to s_{i+1} for all $i \in \{1, \ldots, n-1\}$. There are two possibilities to define the likelihood of an observation sequence o_{seq} for a given HMM. Either the highest likelihood of all possible state sequences is used (Viterbi recognition), or the sum of the likelihoods of all possible state sequences is considered as the likelihood of the observation sequence (Baum-Welch recognition). In the system described in this paper the first possibility is used. For details see [33], for example.

In word recognition systems with a small vocabulary, it is possible to build an individual HMM for each word. But for large vocabularies this method doesn't work anymore because of the lack of enough training data. Therefore, in our system an HMM is build for each character. The use of character models allows us to share training data. Each instance of a letter in the training set has an impact on the training and leads to a better parameter estimation.

To achieve high recognition rates, the character HMMs have to be fitted to the problem. In particular the number of states, the possible transitions and the type of the output probability distributions have to be chosen. In our system each character model consists of 14 states. This number has been found empirically. (The rather high number can be explained by the fact that the sliding window used for feature extraction is only one pixel wide and that many different writing styles are present in the used database.) Because of the left to right direction of writing, a linear transition structure has been chosen for the character models. From each state only the same or the succeeding state can be reached. (A graphical representation of the HMMs used in our system is shown in Fig. 4.) Because of the continuous nature of the features, probability distributions for the features are used. Each feature has its own probability distribution and the likelihood of an observation in a state is the multiplication of the likelihoods calculated for all features. This separation of the elements of the feature vector reduces the number of free parameters, because no covariance terms must be calculated. The probability distribution of all states and features are assumed to be Gaussians, so that only two free parameters per distribution exist, namely, the mean and the variance. The initialization of the models is done by Viterbi alignment to segment the training observations and recompute the free parameters of the models, i.e. the mean

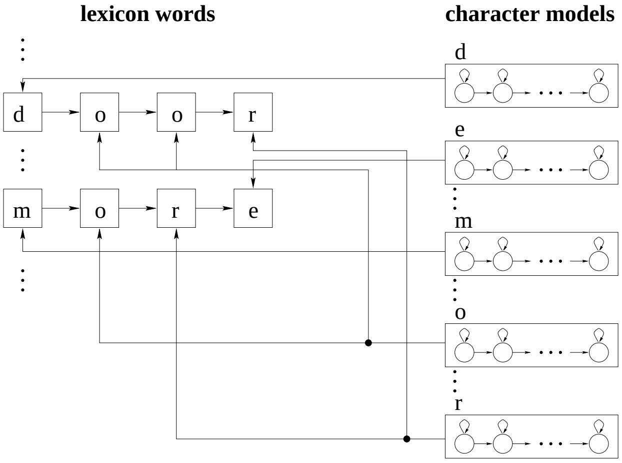


Figure 5: Concatenation of character models yields the word models

and variance of each probability distribution and the transition probabilities between the states. To adjust these free parameters during training, the Baum-Welch algorithm [33] is used.

To model entire words, the character models are concatenated with each other. Thus a recognition network is obtained (see Fig. 5). Note that this network doesn't include any contextual knowledge on the character level, i.e., the model of a character is independent of its left and right neighbor. In the network the best path is found with the Viterbi algorithm [33]. It corresponds to the desired recognition result, i.e., the best path represents the sequence of characters with maximum probability, given the image of the input word. The architecture shown in Fig. 5 makes it possible to avoid the difficult task of segmenting a word into individual characters. More details of the handwritten text recognizer can be found in [6].

3 Ensemble creation methods

In this section the ensemble creation methods used in this paper are described. Each ensemble creation method takes a base classifier and a training set as input and returns a number of trained instances of the base classifier as a result. In the first subsection general aspects of ensemble creation are discussed. Then details of the various methods are given.

3.1 Issues in ensemble creation

A good performing ensemble creation method should have at least two properties. First, the method should create diverse classifiers, which means that the misclassification of patterns should have a low correlation across different classifiers (or in other words, the recognition rate of a classifier C_i on the patterns misclassified by another classifier C_j should be close to the average recognition rate of C_i). In the ideal case independent classifiers are created, but this is almost impossible in real world applications. The diversity of classifiers is crucial, because all of the known combination rules can only increase the performance of single classifiers if they are used with an ensemble of diverse classifiers. It is well known that a high correlation between the errors committed by individual classifiers may lead to a decreasing performance of the ensemble when compared to the best individual classifier. For a more detailed discussion of classifier diversity the reader is referred to [35].

The second requirement is that an ensemble creation method should produce individual classifiers whose recognition rate is not much lower than that of the trained base classifier. It is obvious that the recognition rate of an ensemble using a combination rule depends on the performance of its individual members. There are some ensemble creation methods that have the potential of creating classifiers which outperform the best base classifier. But if many members of an ensemble have a poor performance they may eventually become dominant over the well-performing classifiers. To avoid performance degradation an ensemble creation method should particularly avoid to overfit the training data.

In the following, four ensemble creation methods, namely, Bagging, AdaBoost, random subspace, and architecture variation are introduced. These methods were originally proposed in the area of machine learning. Note that their quality with regard to the two properties discussed above is application dependent and can't be guaranteed a priori.

3.2 Bagging

Bagging [18], an acronym for **b**ootstrapping and **agg**regat**ing**, was among the first methods proposed for ensemble creation. Given a training set S of size n, bagging generates N new training sets $S_1, \ldots, S_N$, each of size n, by randomly drawing elements of the original training set, where the same element may be drawn multiple times. If the probability of being drawn is equally distributed over S, as it is the case here, about two third of all training elements are contained in each modified training set S_i, some of them multiple times. Each of the new sets S_i is used to train exactly one classifier. Hence an ensemble of N individual classifiers is obtained from N new training sets.

3.3 AdaBoost

Similarly to Bagging, AdaBoost [36] modifies the original training set for the creation of the ensemble. To each pattern of the training set a selection probability is assigned, which is equal for all elements of the training set in the beginning. Then elements for a new training set are randomly drawn from the original training set taking the selection probabilities into account. The size of the new training set is equal to the size of the original one. After the creation of a new training set, a classifier is trained on this set. Then the new classifier is tested on the original training set. The selection probabilities of correctly classified patterns in the original training set are decreased and the selection probabilities of misclassified patterns are increased. During the execution of the AdaBoost procedure the selection probabilities are dynamically changing. Hence, unlike Bagging, where the classifiers are created independently, the classifiers generated by AdaBoost are dependent on selection probabilities, which in turn depend on the performance of previously generated classifiers.

The main idea of AdaBoost is to concentrate the training on "difficult" patterns. Note that the first classifier is trained in the same way as the classifiers in Bagging. The classical AdaBoost algorithm can only be used for two-class problems, but AdaBoost.M1 [36], a simple extension of AdaBoost, can cope with multi-class problems. Consequently, AdaBoost.M1 was applied in the system described in this paper.

3.4 Random subspace method

In the random subspace method [19] an individual classifier uses only a subset of all features for training and testing. The size of the subset is fixed and the features are randomly chosen from the set of all features.

For the handwritten text recognizer described in Section 2 the situation is special in the sense that the number of available features is rather low. (As described in Section 2, only nine features are extracted at each position of the window.) Therefore, the features are not completely randomly chosen. If the number of classifiers which use feature f_i is denoted by $n(f_i)$, then the following relation holds: $\forall i, j \ |n(f_i) - n(f_j)| <= 1$. This means that each individual feature is used in approximately the same number of classifiers. Therefore, all features have approximately the same importance. By means of this condition it is enforced that the information of every feature is exploited as much as possible. By contrast, when choosing completely random feature sets, it is possible that certain features are not used at all.

In the experiments described in Section 6, always subsets of six features were used. This number was experimentally determined as a suitable value. The whole training set with feature vectors of reduced dimensionality was used for the training of each individual classifier.

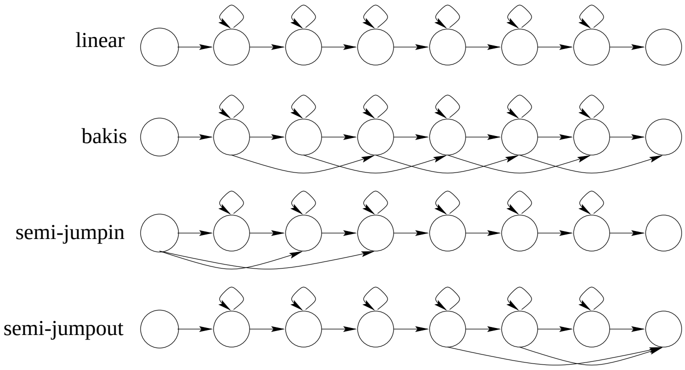


Figure 6: HMM topologies (for a small HMM with 6 emitting states. Note that the HMMs of the classifier in Section 2 have 14 emitting states.)

3.5 Architecture variation

Another way to create an ensemble out of a base classifier is to vary its architecture. In a feed-forward neural network, for example, one may change the number of hidden layers or the number of neurons in each layer [21]. Similar possibilities exist for HMM. Our base classifier was changed as follows.

First, the linear topology was replaced by the Bakis model (see Fig. 6). This topology allows more flexibility in the decoding process by skipping certain states. Next, two additional architectures were implemented. The HMM models used in our base classifier don't include any ligature states[*]. But the transition from one character to the next is often context dependent. Therefore, if certain character pairs are not sufficiently well represented in the training set, misalignments at the beginning and at the end of a character model during decoding may be expected. To account for this kind of problem, the semi-jumpin and semi-jumpout architecture shown in Fig. 6 were introduced. Here the first or last $\frac{n-4}{2}$ states of a linear model may be skipped (with n denoting the total number of states of the considered HMM).

Normally the columns of a word image are read form left to right. Another possibility is to read them from right to left. Because the Viterbi search used in the decoding phase is a suboptimal procedure that prunes large portions of the search space, the results of a forward and a backward scan of the word are not necessarily the same. To implement a right-to-left scan of the image, only the concatenation of character HMMs needs to be changed appropriately.

Apparently, left-to-right as well as right-to-left scanning can be combined with any of the architectures shown in Fig. 6. Therefore, a total of eight different classifiers were generated. Each of these classifiers was trained on the full training set.

4 Combination schemes

In this section the combination schemes used in our multiple classifier system for handwriting recognition are described.

4.1 Maximum score rule

In this scheme the word class with the highest score among all word classes and all classifiers is the output of the combined classifier. This combination scheme is denoted as *max* in the following.

[*] Here the term ligature denotes a connection stroke between two consecutive characters

4.2 Performance weighted voting

In the weighted voting combination scheme a weight is assigned to each classifier. For all word classes the sum of the weights of the classifiers which output this class is calculated and the combined result is the word class that has the largest sum of weights. In the performance weighted voting scheme, which is denoted as *perf voting* in the following, the weight of the classifier is equal to the classifier's performance (i.e. recognition rate) on the training set. The system described in Section 2 was found to have a good generalization power, i.e. the results on the training set allow a good estimation of the behavior of the system on test data. So the training set was used for the evaluation of the performance of the classifiers. For other classifiers it may be necessary to use a separate validation set to evaluate the performance of the created classifiers. (The Nearest-Neighbor classifier, for example, has always a recognition rate of 100 % on its training set.)

4.3 Weighted voting using weights calculated by a genetic algorithm

Using the performance of a classifier as its weight is based on the intuitive assumption that classifiers with a high recognition rate are more trustworthy than classifiers that perform poorly. However, there is no objective proof that this strategy is optimal. Under a more general approach, one considers the set of weights as free parameters in a multiple classifier system, and tries to find the combination of values that lead to the best performance of the whole system. Out of many possible optimization procedures it was decided to use a genetic algorithm [37] for weight optimization. Among the reasons to favor a genetic approach over other methods was the simplicity and elegance of genetic algorithms as well as their demonstrated performance in many other complex optimization problems [38, 39, 40].

The training set used to find the individual classifiers was also used to derive the optimal combination of weights, assuming that the obtained values lead to a good performance on the test set as well. The genetic algorithm for weight optimization will be described in Section 5 in greater detail. In the following this algorithm will be denoted as *ga voting*.

4.4 Voting with Ties Handling

Under this scheme a normal voting procedure is executed first, i.e., if the occurrence of a class among the results of the classifiers is higher than the occurrence of any other class then this class is output as the combined result. A tie occurs if no unique result is obtained. For some applications it may be sufficient to just reject a pattern if a tie occurs, but here we use a more general approach. In case of a tie we focus our attention on those classes that compete under the tie and apply one the above mentioned combination schemes. So there is voting with tie handling by maximum score rule, by performance weighted voting, and by weighted voting using weights calculated by a genetic algorithm. Theses schemes are denoted as *ties max*, *ties perf voting*, and *ties ga voting*, respectively.

5 Genetic algorithm for the calculation of the weights used by weighted voting

First proposed in [37], genetic algorithms have been found to be robust and practical optimization methods. In a genetic algorithm a possible solution of the problem under consideration is represented by a chromosome. In the initialization step of the algorithm a set of chromosomes is created randomly. The actual set of chromosomes is called the population. A fitness function is defined to represent the quality of the solution given by a chromosome. Only the chromosomes with the highest values of this fitness function are allowed to reproduce. In the reproduction phase new chromosomes are created by fusing information of two existing chromosome (crossover) and by randomly changing them (mutation). Finally the chromosomes with the lowest values of the fitness function are removed. This reproduction and elimination step is repeated until a predefined termination condition is become true. In the following we describe the genetic algorithm that is used in our multiple classifier system in more details.

5.1 Chromosome representation and Fitness

The representation of a set of weights for the individual classifiers by a chromosome is straightforward. Each chromosome is represented by an array of real numbers between 0 and 1. The i-th position of the array corresponds to the weight of the i-th classifier of the ensemble. The number of elements in the array is equal to the number of classifiers. The fitness of a chromosome is defined as the recognition rate of the ensemble when using weighted voting with the weights represented by the chromosome. Note that by using the performance of the whole ensemble as fitness the diversity of the individual classifiers is also taken into account.

5.2 Initialization and termination

A population of size 50 was used in the algorithm. All positions of the chromosomes are set to random real values between 0 and 1 at the beginning of the algorithm. If the fitness value of the ten best chromosomes is the same the algorithm is terminated. Alternatively, if this condition doesn't become true, the algorithm is terminated after 100 generations. The weights of the chromosome with the highest fitness value encountered during all generations (not only the last one) are the final result and are used for the weighted voting combination.

5.3 Crossover operator

A normal one point crossover operator is used. First a position i in the chromosome is randomly selected. Then the values the first parent chromosome from position 1 to position i are copied to the corresponding positions in the first child chromosome. Moreover, the values of the remaining positions of the first parent chromosome are copied to the corresponding positions of the second child chromosome. Then the positions of the children chromosomes not yet defined are set to the corresponding values of the second parent chromosome. The probability of the crossover was set to 90 %.

5.4 Mutation operator

The mutation operator is applied to all new chromosomes produced by the crossover operator. This operator changes only one random position of the array in the following manner. The value at this position is changed by a constant multiplied with a random number between 0 and 1. Under this procedure, the chance of an increase or a decrease is both equal to 50%. If the value after this modification is higher than 1 or lower than 0 it is set to 1 and 0, respectively. In the experiments the constant was set to 0.2.

5.5 Generation of a new population

First 25 chromosomes are produced by the crossover operator. Each of the 50 chromosomes of the old generation may be selected as a parent of a new chromosome. The selection probability of a chromosome is proportional to its fitness value minus the minimal fitness value of the old generation (i.e. the chromosome with the lowest fitness in the old population has a selection probability equal to 0). The mutation operator is applied to all new 25 chromosomes. Then the 50 old and the 25 new chromosomes are combined into one population. To reduce this population to the original size, the 25 chromosomes with the smallest score values are removed. Note that also newly created chromosomes may be removed.

6 Experiments

All ensemble creation methods discussed in Section 3 were implemented and tested on a part of the IAM database. This database is publicly available[†] and has been used by several research groups meanwhile [31]. The original version of the database contains complete lines of text as its basic entities, without any segmentation of a line of text into individual words [31]. Meanwhile, however, part of this database has been segmented into individual words [31, 32]. A subset of these words was used in the experiments described in this section.

The training set used in the experiments contains 9861 and the test set 1066 word instances over a vocabulary of size 2296. The test set was chosen in such a way that none of its writers was represented in the training set. Hence all experiments described in this paper are writer independent. The total number of writers who contributed to the training and test set is 81. A small sample of words from this database is shown in the Appendix.

Table 1 shows the results of the experiments. The recognition rate of the classifier with the original architecture and training set was 66.23 %. Bagging, AdaBoost and random subspace method each created 10 classifiers while the architecture variation method generated only 8 (see Subsection 3.5).

At first glance, the recognition performance of all systems under consideration may appear quite low. One has to keep in mind, however, that a very difficult classification problem is considered. First of all, we are faced with a pattern recognition task that involves 2296 pattern classes. Secondly, there were almost no constraints imposed on the writers. Hence all kinds of different writing styles and writing instruments are represented in the data set. Thirdly, a large number of writers contributed to the database, and all experiments were run in a writer-independent fashion, i.e. the writers of the training and validation set are disjoint from the writers of the test set.

In the following the results of the different combination schemes are discussed. Obviously the *max* combination performed rather poorly. A possible explanation of this poor performance is the different ranges of score values returned by the classifiers. Because the score is only a likelihood value, it depends on the specific HMM, and identical score values from different HMMs don't necessarily imply that the word classes which correspond to these values have the same probability of being correct. Note that the performance of the maximum combination rule is especially poor for *architecture var* and *random subspace*, where the HMMs of the classifiers are very different. A possibility to overcome this problem is to normalize the score values for each classifier. However, this possibility has not been explored in the context of this paper and is left to future research.

All other combination schemes lead to an increase of the recognition rate for all ensemble creation methods when compared to the original classifier. The proposed *ga voting* combination was the best scheme for three out of the four ensemble creation methods considered in the tests. The quality of the other schemes relative to each other varied among the tests.

Please note that with the simple weighting mechanism of *perf voting* also good results were achieved. The superior performance of *ga voting* over *perf voting* doesn't hold true any longer for voting with tie handling. Here *ties ga voting* is outperformed by *ties perf voting* for two ensemble creation methods. The reason for this behavior is that the weights calculated by the genetic algorithm are optimized for weighted voting and not for voting with ties handling by weighted voting. Nevertheless *ties ga voting* is clearly superior to the original classifier.

To compare the different ensemble methods in more detail the average performance and the standard deviation of the performances of the individual classifiers were calculated. Those values are shown in Table 2.

Bagging produced classifiers with very similar performances and which were in average almost as good as the original classifier. As the performance of the ensemble is not much higher than the performance of the original classifier in respect to the other ensemble methods it may be concluded that the diversity of the classifiers is low.

The classifiers produced by AdaBoost had a wider range of performance than Bagging. Although the average performance of the individual classifier is slightly lower than in Bagging, a much better ensemble performance

[†] http://www.iam.unibe.ch/~zimmerma/iamdb/iamdb.html

was achieved. This indicates that the classifiers are quite diverse. AdaBoost was the only ensemble method were *ga voting* did not produce the best result. A possible reason for this is the following. In AdaBoost the performance of the ensemble on the training set is optimized by focusing on “difficult” patterns. Such optimization on the training set normally leads to classifiers which are much better on the training set than on the test set. As the genetic algorithm works with the results of the training set, it may overestimate the performance of some classifiers and produce suboptimal weights. This problem may be overcome by using a separate validation set for calculating the weights.

The average performance of the classifiers produced by *random subspace* was much lower than the performance of AdaBoost, yet the ensemble performance was still quite good. So the diversity of classifiers increased again. For *random subspace* the best performance of *ga voting* in respect to the other combination schemes was achieved (*ga voting* had a 0.56 % higher performance than the second best scheme). An analysis of the calculated weights showed that the weights of three out of the ten classifiers were so low that in fact those classifiers were almost irrelevant for the combination. This means that *ga voting* was capable to discover the classifiers which lower the ensemble performance and to exclude them from the combination.

The classifiers produced by *architecture var.* had in average a very low performance (20.26 % lower than the performance of the original classifier). Yet good ensemble results were achieved by this method which leads to the conclusion that the classifiers must be very diverse. For all ensemble methods but *architecture var. perf voting* and *ties perf voting* produced te same results.

When using *ga voting* or *ties ga voting*, in addition to the testing of all classifiers on the training set also the genetic algorithm must be executed. Yet the time consumption of the genetic algorithm is over 1000 times lower than that of the tests on the training set so that this additional overhead is not significant.

7 Conclusions

In this paper the recognition of cursively handwritten words was considered. Because of the large number of classes involved and the great variations of words from the same class, which is due to the considerable number of individual handwriting styles, this is regarded a difficult problem in pattern recognition. Multiple classifier systems have demonstrated very good performance in many pattern recognition problems recently. In this paper we have explored a number of classifier ensemble generation methods and related combination schemes. As hidden Markov Models (HMMs) are considered to be one of the most powerful methods for cursive handwriting recognition today, we have focused on those classifier ensemble generation method and combination procedures that are applicable in situations where the base classifiers of a multiple classifier system are given by HMMs.

The combination schemes considered in this paper are based on the assumption that each base classifier only outputs its top-ranked class, together with a score value. Among other combination schemes, two versions of weighted voting were considered. In the first version the weight of each individual base classifier was set equal to its recognition rate on the test set. By contrast a genetic algorithm was used for weight optimization in the second version, using the recognition performance of the whole ensemble as fitness function. In a series of experiments it was shown that for all but one combination scheme all multiple classifier systems could improve the performance of the original, single HMM-based classifier. Among all combination schemes tested in the experiments, for three out of four creation methods, the highest recognition rate was obtained with weighted voting using genetic weight optimization.

The results reported in this paper confirm the suitability of genetic algorithms to find optima or near optima of functions in complex situations. Future works will address the problem of genetic weight optimization for systems including significantly more classifiers. These classifiers may be generated from a single base classifier using methods similar to those considered in the present paper. Alternatively, it is possible to produce the base classifiers by the simultaneous application of several classifier generation procedures. Topic of future work will also be the use of a separate validation set for the calculation of the weights of the classifiers to avoid overfitting problems.

Acknowledgment

The research was supported by the Swiss National Science Foundation (Nr. 20-52087.97). The authors thank Dr. Urs-Victor Marti for providing the handwritten word recognizer and Matthias Zimmermann for the segmentation of a part of the IAM database. Additional funding was provided by the Swiss National Science Foundation NCCR program "Interactive Multimodal Information Management (IM)2" in the Individual Project "Scene Analysis".

References

[1] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Proc. of the IEEE*, 80(7):1162–1180, 1992.

[2] J.-C. Simon. Off-line cursive word recognition. *Proc. of the IEEE*, 80(7):1150–1161, July 1992.

[3] S. Impedovo, P. Wang, and H. Bunke, editors. *Automatic Bankcheck Processing*. World Scientific Publ. Co, Singapore, 1997.

[4] A. Kaltenmeier, T. Caesar, J.M. Gloger, and E. Mandler. Sophisticated topology of hidden Markov models for cursive script recognition. In *2nd Int. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan*, pages 139–142, 1993.

[5] G. Kim, V. Govindaraju, and S.N. Srihari. Architecture for handwritten text recognition systems. In S.-W. Lee, editor, *Advances in Handwriting Recognition*, pages 163–172. World Scientific Publ. Co., 1999.

[6] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int. Journal of Pattern Recognition and Art. Intelligence*, 15:65–90, 2001.

[7] J. Kittler and F. Roli, editors. *1st International Workshop on Multiple Classifier Systems*, Cagliari, Italy, 2000. Springer.

[8] J. Kittler and F. Roli, editors. *2nd International Workshop on Multiple Classifier Systems*, Cambridge, UK, 2001. Springer.

[9] A. Bellili, M. Gilloux, and P. Gallinari. An hybrid MLP-SVM handwritten digit recognizer. In *6th International Conference on Document Analysis and Recognition*, pages 28–32, 2001.

[10] A. Brakensiek, J. Rottland, A. Kosmala, and G. Rigoll. Off-line handwriting recognition using various hybrid modeling techniques and character n-grams. In *7th International Workshop on Frontiers in Handwritten Recognition*, pages 343–352, 2000.

[11] Y. Huang and C. Suen. Combination of multiple classifiers with measurement values. In *Second International Conference on Document Analysis and Recognition*, pages 598–601, 1993.

[12] D. Lee and S. Srihari. Handprinted digit recognition: A comparison of algorithms. In *Third International Workshop on Frontiers in Handwriting Recognition*, pages 153–162, 1993.

[13] A. Rahman and M. Fairhurst. An evaluation of multi-expert configurations for the recognition of handwritten numerals. *Pattern Recognition*, 31(9):1255–1273, 1998.

[14] X. Wang, V. Govindaraju, and S. Srihari. Multi-experts for touching digit string recognition. In *5.International Conference on Document Analysis and Recognition*, pages 800–803, 1999.

[15] L. Xu, A. Krzyzak, and C. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, 1992.

[16] A. Rahman and M. Fairhurst. Automatic self-configuration of a novel multiple-expert classifier using a genetic algortihm. In *Proceedings of the 7th International Conference on Image Processing and its Applications*, pages 57–61.

[17] T. G. Dietterich. Ensemble methods in machine learning. In *[7]*, pages 1–15, 2000.

[18] Leo Breiman. Bagging predictors. *Machine Learning 2*, pages 123–140, 1996.

[19] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[20] T.G. Dietterich and E.B. Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Departement of Computer Science, Oregon State University, 1995.

[21] D. Partridge and W. B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8(4):869–893, 1996.

[22] R. Duin and D. Tax. Experiments with classifier combination rules. In *[7]*, pages 16–29, 2000.

[23] C. Suen and L. Lam. Multiple classifier combination methodologies for different output level. In *[7]*, pages 52–66, 2000.

[24] J. Kittler, R. Duin, and M. Hatef. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20:226–239, 1998.

[25] T. Huang and C. Suen. Combination of multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17:90–94, 1995.

[26] T.K. Ho, J.J. Hull, and S.N. Srihari. Decision combination in multiple classifier systems. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16:66–75, 1994.

[27] G. Houle, D. Aragon, R. Smith, M. Shridhar, and D. Kimura. A multilayered corroboration-based check reader. In J. Hull and S. Taylor, editors, *Document Analysis System 2*, pages 495–546. World Scientific, 1998.

[28] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–285, 1989.

[29] L. Lam, Y.-S. Huang, and C. Suen. Combination of multiple classifier decisions for optical character recognition. In *[41]*, pages 79–101.

[30] K. Sirlantzis, M. Fairhurst, and M. Hoque. Genetic algorithms for multiclassifier system configuration: A case study in character recognition. In *[8]*, pages 99–108.

[31] U. Marti and H. Bunke. The IAM-database: an English sentence database for off-line handwriting recognition. *Int. Journal of Document Analysis and Recognition*, 5:39–46, 2002.

[32] M. Zimmermann and H. Bunke. Automatic segmentation of the IAM off-line database for handwritten English text. In *Proc. of the 16th Int. Conference on Pattern Recognition*, volume 4, pages 35–39, Quebec, Canada, 2002.

[33] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[34] A. Kundu. Handwritten word recognition using hidden Markov model. In *[41]*, pages 157–182.

[35] A. Krogh and J. Vedelsby. Neural networks ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. MIT Press, 1995.

[36] Yoav Freund and Robert E. Schapire. A descision-theoretic generalisation of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.

[37] J. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975.

[38] F.J. Ferri, V. Kadirkamanathan, and J. Kittler. Feature subset search using genetic algorithms. In *IEE/IEEE Workshop on Natural Algorithms in Signal Processing (NASP 93)*, pages 23/1–23/7, 1993.

[39] J. Morris, D. Deaven, and K. Ho. Genetic-algorithm energy minimization for point charges on on a sphere. *Physical Review B*, 53(4):1740–1743, 1996.

[40] H. Zhang, B.-T. Mühlenbein. Genetic programming of minimal neural nets using Occam's razor. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 342–349, 1993.

[41] H. Bunke and P. Wang, editors. *Handbook of Character Recognition and Document Image Analysis*. World Scientific, 1997.

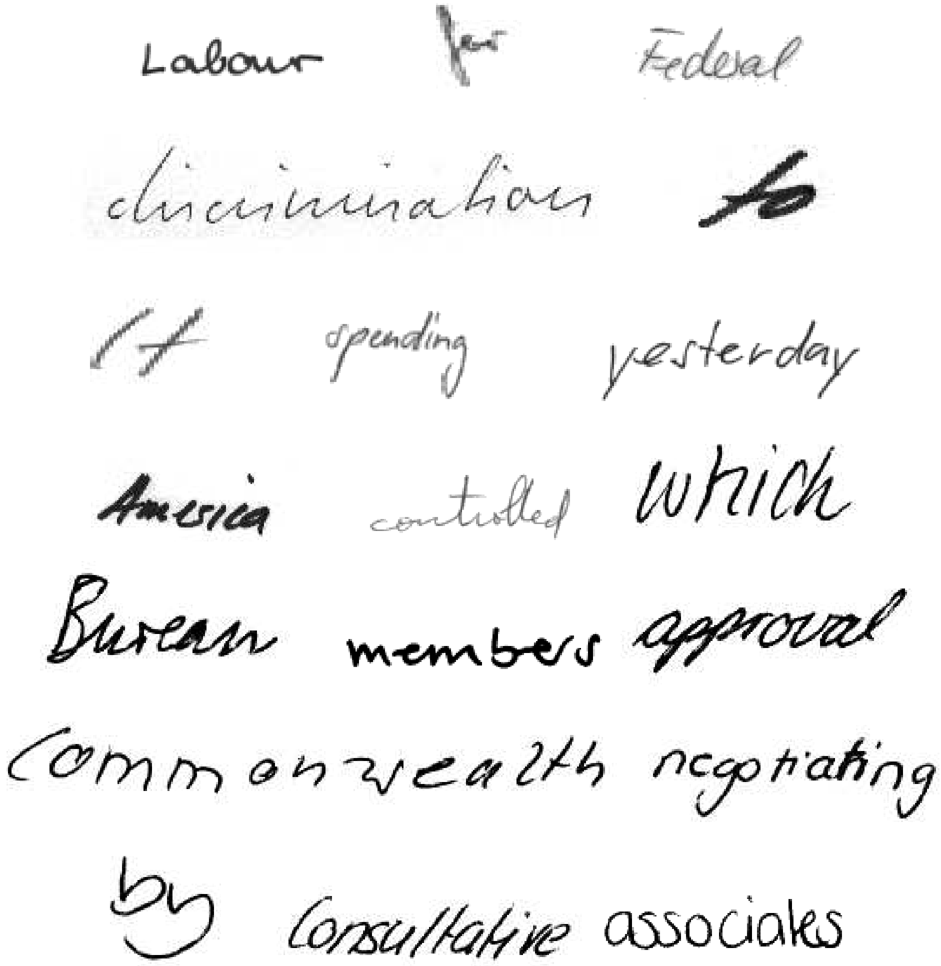
	ensemble creation method			
combination	Bagging	AdaBoost	random subspace	architecture var.
max	65.2 %	63.51 %	62.10 %	45.97 %
perf voting	67.64 %	**68.86 %**	68.11 %	68.39 %
ga voting	**67.92 %**	68.29 %	**68.67 %**	**68.76 %**
ties max	67.35 %	68.29 %	67.54 %	66.98 %
ties perf voting	67.64 %	**68.86 %**	68.11 %	68.57 %
ties ga voting	67.82 %	68.39 %	68.01 %	68.57 %
original	66.23 %			

Table 1: Recognition rates achieved by the ensemble creation methods under different combination rules. The best result for each ensemble creation method is printed in bold face.

	ensemble creation method			
measure	Bagging	AdaBoost	random subspace	architecture var.
average	66.02 %	65.82 %	60.76 %	52.49 %
std. deviation	0.58 %	1.82 %	4.66 %	9.71 %

Table 2: Average performance and the standard deviation of the performance of the individual classifiers. The performances of the original classifier is 66.23 %.

Appendix


Labour
for
Federal
discrimination
to
of
spending
yesterday
America
controlled
which
Bureau
members
approval
Commonwealth
negotiating
by
Consultative
associates