Methods of Combining Multiple Classifiers Based on Different Representations for Pen-based Handwritten Digit Recognition

Fevzi Alimoğlu, Ethem Alpaydın
Department of Computer Engineering
Boğaziçi University
TR-80815 Istanbul, Turkey
alimoglu@boun.edu.tr

Abstract. Pen-based handwriting recognition has enormous practical utility. It is different from optical recognition in that the input is a temporal signal of pen movements as opposed to a static spatial pattern. We examine various ways of combining multiple learners which are trained with different representations of the same input signal: dynamic (pen movements) and static (final 2D image). We notice that the classifiers based on different representations fail for different patterns and investigate ways to combine the two representations. We benchmark voting, stacking, mixture of experts and cascading. In voting and stacking, the two are always used together. In the mixture of experts, the gating network chooses one of the two. In cascading, the static is used only when the dynamic is not "certain". On a handwritten digit database significant increase in accuracy has been obtained.

1 INTRODUCTION

Handwritten character recognition has attracted enormous scientific interest due to its evident practical utility. In addition to optical character recognition (OCR) which is an off-line method that requires first digitizing then recognition, there is also on-line, pen-based character recognition of handwritten characters entered from a touch terminal (tablet). The touch terminal is convenient for users who prefer a touch interface over a keyboard and mouse and it is a more portable input device unlike conventional terminals and personal computers.

Compared with OCR, in pen-based recognition the input is a temporal sequence of the movements of the pen tip. Handwriting can be represented as a time-ordered sequence of coordinates of the pen tip on the tablet with varying speed and pressure at points. The

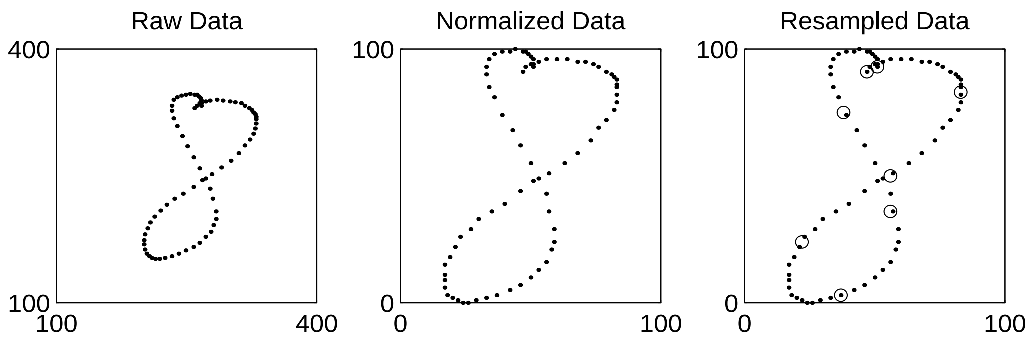

Figure 1: Preprocessing: Dynamic representation and spatial resampling to 8 points

ordering of the sequence of points gives the necessary information for recognition.

A comprehensive review of online handwritten digit recognition is given by Tappert et al. (1990). Time delay neural networks have been used for the recognition of handwritten characters (Guyon et al., 1991). This structure was later expanded to include segmentation together with recognition (Schenkel et al. 1993). Hidden Markov models are among the most popular methods used for online handwriting, especially cursive.

2 PREPROCESSING

The digits were written by people having various styles of writing. This is a normal and expected situation. The sources of variability are related to the task performed, the subjects recorded and the experimental methodology. The purpose of the preprocessing is to create an intermediate representation, incorporating, to the extent possible, the a priori knowledge of the designer. This includes two complementary processes (Kassel, 1995): (1) Reducing variability which does not help discrimination between classes, therefore building up some invariance, (2) Enhancing variability which does help discrimination between classes.

The raw data contains enormous variations in writing speed. Different writers write with different speed characteristics. Training a recognizer with such data leads to poor performance. The steps of our preprocessing, namely normalization and resampling, greatly reduce the meaningless variability. In particular, time and scale distortions are removed.

2.1 Dynamic Representation and Resampling

A commonly used technique leading to good results is resampling the (x_t, y_t) points in order to obtain constant length feature vectors. Temporal resampling (regularly spaced in time) or spatial resampling (regularly spaced in arc length) (Guyon et al., 1991) can be used here. Raw point data are already regularly spaced in time but has a variable length. Previous research (Kassel, 1995) showed that spatial resampling to obtain a constant number of regularly spaced points on the trajectory yields much better performance.

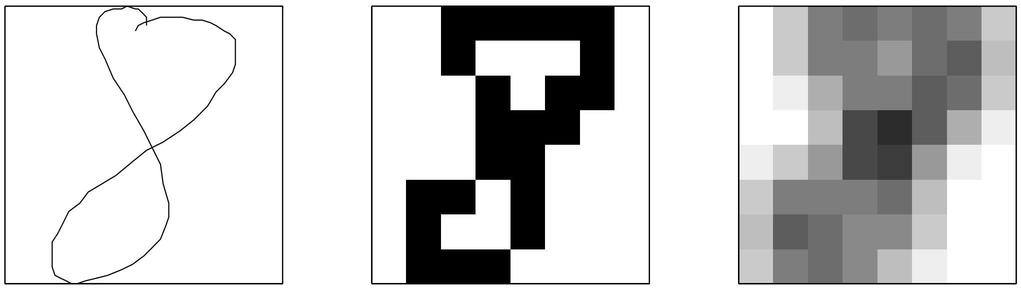

Figure 2: Preprocessing: Static representation and blurring

Our resampling algorithm uses simple linear interpolation. The resampled characters are represented as a sequence of points (x_t, y_t), regularly spaced in arc length, as opposed to the input sequence, which is regularly spaced in time (Fig. 1).

2.2 Static Representation and Blurring

In addition to the dynamic representation, we used also static representation based on the final image of pen strokes, which may be produced in various ways. The static representation differs from dynamic representation in the sense that it does not incorporate pen motion. Instead, it is based on solely the image of a character. We used Bresenham's scan conversion algorithm (Foley et al., 1990) to create bitmaps from normalized on-line handwriting data. We iterated through pairs of sequential pen samples within each stroke.

Every character in the database was normalized to a unit bounding box, so we used a square pixel array to store each image. The single parameter for this representation is the resolution of the image (Fig. 2). Numeric values of 0 and 1 correspond to white and black pixels respectively.

In preparing bitmaps for clasification the spatial relationship was not preserved. Bitmaps which are usually similar can have very dissimilar feature vectors. Small variations in pen coordinates can produce these differences due to quantization with hard thresholds. To solve this problem we choose smoothing the bitmap so that a pixel's value is distributed across its neighbors. We accomplished this by convolving a blurring kernel with the image to produce a pixmap (in which pixels can take on arbitrary floating-point values). An example is shown in Fig. 2 were we applied a blurring kernel with unit volume.

3 LEARNING AND RECOGNITION

We use two approaches: In the time-delay version, previous inputs are delayed till the final arrives and thus there is a conversion from time to space. This whole input frame is then given as one static pattern to the classifier. In this approach, we have used k-nearest neighbor (k-NN) and multi-layer perceptron (MLP). In the other approach, inputs are fed to a classifier as they arrive in time so the classifier should be able to cope with temporally

Table 1: Percentage of samples misclassified by MLP having 10 hidden units trained with dynamic (8 point) and static (8 by 8) representations

SETS	DYNAMIC	STATIC	BOTH
Training set	0.88	3.12	0.27
Cross-validation Set	2.35	5.93	0.69
Writer Indep. Test Set	1.76	6.57	0.53
Writer Dep. Test Set	4.69	7.69	2.00

changing data. In this approach, we have used a multi-layer perceptron with recurrent connections in the hidden layer.

The classifiers described above give acceptable results on both dynamic and static representations for a large number of input vector and free parameters. The dynamic representation, which keeps the order and timing of the handwriting outperforms the static representation which contains less information (based only on the final image). However, our experiments showed us the fact that classifiers trained with different representation fail for different samples (Table 1). These results show us that if we find a suitable method of combining the outputs of the learners for different representations, we can build a recognizer with higher performance.

We concentrate on combining MLPs trained on dynamic and static representations of our pen-based handwritten digits. Training a MLP with input vectors composed of the concatenation of dynamic and static features is one possibility. Another is to have two separate classifiers for the two and combine them afterwards using one of voting, stacking, mixture of experts or cascading. All dynamic MLP recognizers have 10 hidden units and are trained with 8 point spatially resampled data and all static recognizers have 10 hidden units and are trained with 8 by 8 blurred static images.

3.1 Voting

Voting (Hansen and Salamon 1990) is the simplest way to combine multiple classifiers. It corresponds to taking a linear combination of the learners. If each voter (classifier) has equal weight (i.e. we take the average), we call it *simple voting.*

3.2 Stacked Generalization

Stacked Generalization is a technique proposed by Wolpert (1992) that extends voting in that learners (called level 0 generalizers) are not necessarily combined linearly. The combination is made by a combiner system (called level 1 generalizer) that is also trained.

3.3 Mixture of Experts

We used cooperative version of mixtures of experts (Alpaydın and Jordan 1996). This method consists of two simple linear perceptrons one for dynamic one for static data, that are combined using a gating network which is again a linear perceptron to which both are given as input. The gating network thus decides on the weights of the two representations.

3.4 Boosting

Boosting (Drucker et al. 1993) is a technique for constructing for constructing a classifier which makes small error rates from classifiers which are doing just slightly better than 50 percent. Unlike previously described methods, learners are trained serially. After training the first learner, we train the second learner with the data on which the first falls making sure that the two learners complement each other. A third learner is trained with the data on which the first two learners disagree. During testing, the third learner is consulted only when the first two learners disagree. Boosting requires large training sets and is an expensive method.

3.5 Cascading

The main idea in cascading is that a great percentage of the training samples can be explained by simple rule and there are a small number of exceptions. There are two learners, one to learn the rule and the other to learn the exceptions.

In our problem, the best recognizer is the dynamic one. However the static recognizer, although less accurate, is able to classify some samples that are misclassified by the dynamic classifier. So, we can set the rule learner as dynamic MLP C_d and the exceptions learner as static MLP C_s.

The major problem is to decide when pattern is covered by the rule and thus should be learned by C_d, or is an exception and should be learned by C_s. We take this as the highest class posterior and we compare with a threshold θ. If is greater than θ, we assume that it is covered by the rule, otherwise it is an exception. During recognition, given a dynamic sample x, we first check C_d. If the highest posterior exceeds θ, x is covered by the rule C_d, otherwise by the exception learner C_s.

4 SIMULATION RESULTS

After preprocessing and elimination of nonsense input, we divided our database into two parts, one containing 30 writers and the other containing 14 writers. The first part is divided into three sets with 3,748 digits for training, 1,873 for cross-validation and 1,873 for writer-dependent testing. 3,498 examples from 14 writers are used to test performance on writers unseen during training.

First we looked for the optimal number of samplings in dynamic representation to take over the digit and tried 4, 8, 12, and 16 samples. We tried all three on both k-NN

Table 2: Results for single classifiers with dynamic (8 Point Sampling) and static (8 by 8 images) representations (i=integer, f=floating point)

METHOD	WR-DEP	WR-INDEP	FREE PARS	EPOCHS
Dyn. 5-NN	99.73, 0.00	97.91, 0.00	3748i	1, 0.00
Dyn. MLP, h=16	98.26, 0.31	95.26, 0.37	280f	15.20, 3.29
Dyn. Rec., h=16	97.93, 0.36	94.23, 0.71	474f	35.30, 4.79
Sta. 5-NN	96.26, 0.00	93.34, 0.00	3748f	1, 0.00
Sta. MLP., h=10	93.54, 0.41	92.25, 0.31	760f	17.40, 3.63

and MLP. No significant success differences were obtained and thus we preferred to use eight samples as then the system is least complex. We did similar experiments for static representation and we selected blurred 8 by 8 version for the same reason. See Table 2.

For the combination of multiple classifiers we have the results in Table 3. Our first attempt to combine the two representations was constructing a new feature vector consisting of concatenation of dynamic and static vectors this was not satisfactory.

Voting over two separately trained classifiers, one using the dynamic representation and the other the static representation, yielded good results, higher than the individual classifiers.

We tested stacked generalization with level 1 learner as simple linear perceptron and MLP. Compared with voting, the results were slightly better for writer dependent test set, but lower for writer independent test set. The complexity of the level 1 learner also did cause significant difference on the success rate. When we set level 1 learner as a simple linear perceptron with no bias and we examined its weights we noticed that for any output unit the corresponding weights coming from the two learners are the same. This means that for our data sets stacking reduces to simple voting.

We used mixture of experts where the two classifiers using two representations are linear and the gating network whose input is the concatenation of the two representations are also linear. The gating thus decides on how much to weight the two representations given the input.

We also tried cascading for different values of the certainty threshold θ. A comparison of all approaches in terms of accuracy vs. memory requirement is given in Figure 3.

5 CONCLUSIONS

By combining two classifiers using different representations, one dynamic using the movement of the pen over time and the other static using the final bitmap, we get higher accuracy than either alone. Comparing the four approaches of combining, namely voting, stacking, mixture of experts and cascading, in terms of accuracy, they are comparable but cascading is the siplest to train.

Table 3: Results for multiple classifiers based on MLPs with 10 hidden units for dynamic (8 Point Sampling) and static (8 by 8 images) representations (i=integer, f=floating point)

METHOD	WR-DEP	WR-INDEP	FREE PARS
Dyn. MLP	98.26, 0.31	95.26, 0.37	280f
Sta. MLP	93.54, 0.41	92.25, 0.31	760f
Dyn+Sta MLP, h=20	93.52, 0.36	92.02, 0.35	1830f
Voting	98.40, 0.16	96.92, 0.37	1040f
Stacking, Lin. Comb.	98.76, 0.11	96.61, 0.35	1250f
Stacking, MLP Comb.	98.82, 0.20	96.69, 0.39	1360f
Mix. of Exp.	97.80, 0.30	96.07, 0.33	1630f
Casc. θ=0.7	98.30, 0.13	95.66, 0.57	1040f

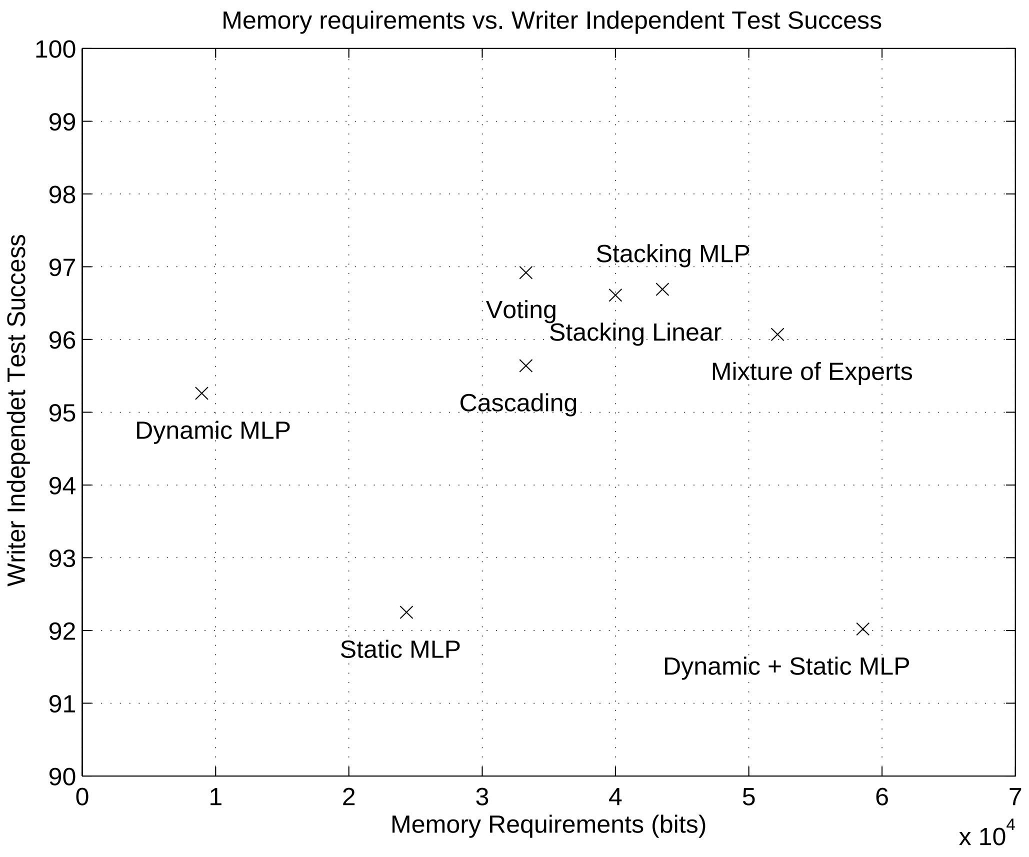

Figure 3: Memory requirements (bits) vs. Writer Independent Test Success. Here one floating point parameter is assumed to require 32 bits.

Acknowledgments

This work is supported by Tübitak Grant EEEAG-143 and Boğaziçi University Research Funds 95HA0108.

References

E. Alpaydın, M. I. Jordan, "Local Linear Perceptrons for Classification", *IEEE Transactions on Neural Networks*, vol. 7, no 3, May 1996.

H. Drucker, R. Schapire, P. Simard, "Improving Performance in Neural Networks Using a Boosting Algorithm," in *Advances in Neural Information Processing Systems*, 5, S. J. Hanson, J. Cowan, L. Giles (Eds.), 42-49, Morgan Kaufmann.

J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Reading, Massachusetts: Addison Wesley Publishing Company, Inc., second ed., 1990.

I. Guyon, P. Albrecht, Y. Le Cun, J. Denker, W. Hubbard, "Design of a Neural Network Classifier Character Recognizer for a Touch Terminal," *Pattern Recognition*, vol 24, no.2, pp. 105-119, 1991.

L. K. Hansen, P. Salamon, "Neural Network ensembles," *IEEE Transactions on Pattern Analysis and Mahcine Intelligence*, 12, 993-1001, 1990.

R. H. Kassel, "A comparison of approaches to On-line handwritten character recognition," PhD Thesis, Massachussets Institute of Technology, 1995.

M. Schenkel, H. Weissman, I. Guyon, C. Nohl, and D. Henderson, "Recognition-based segmentation of on-line hand-printed words," in *Advances in Neural Information Processing Systems* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), vol. 5, (Denver, Colorado), pp. 723-730, Morgan Kaufman, 1993.

C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in on-line handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 787-808, Aug. 1990.

D. H. Wolpert, "Stacked Generalization," Neural Networks, 5, 241-259, 1992