Combining Multiple Representations and Classifiers for Pen-based Handwritten Digit Recognition

Fevzi Alimoğlu

Dept. of Computer Engineering
Boğaziçi University
Istanbul, Turkey, TR-80815
alimoglu@boun.edu.tr

Ethem Alpaydın

Dept. of Computer Engineering
Boğaziçi University
Istanbul, Turkey, TR-80815
alpaydin@boun.edu.tr

Abstract

We investigate techniques to combine multiple representations of a handwritten digit to increase classification accuracy without significantly increasing system complexity or recognition time. We compare multiexpert and multistage combination techniques and discuss in detail in a comparative manner methods for combining multiple learners: Voting, mixture of experts, stacking, boosting and cascading. In pen-based handwritten character recognition, the input is the dynamic movement of the pentip over the pressure sensitive tablet. There is also the image formed as a result of this movement. On a real-world database, we notice that the two multi-layer perceptron (MLP) neural network-based classifiers using separately these representations make errors on different patterns implying that a suitable combination of the two would lead to higher accuracy. Thus we implement and compare voting, mixture of experts, stacking and cascading. Combined classifiers have an error percentage less than individual ones. The final combined system of two MLPs has less complexity and memory requirement than a single k-nearest neighbor using one of the representations.

1 Introduction

Handwritten character recognition has attracted enormous scientific interest due to its evident practical utility. On-line handwriting [9] is important where keyboards are difficult to use, e.g. when the writer is mobile and the device needs be portable.

To achieve high recognition accuracy, many different classification methods have been proposed using statistical, syntactical, or neural approaches [7]. These different methods need not use the same representation but may extract different features from the same raw input. The usual approach is to choose the one classifier and the particular representation that performs best, discarding the rest. However, it is generally the case that the set of patterns misclassified by different classifiers do not necessarily overlap, implying that classifiers to a certain extent complement each other. The classifiers may also be using different representations of the input thus making it possible to integrate different types of sensors/measurements or features. Different representations make different characteristics explicit allowing better identification.

The primary goal of this work is to construct a high-performance system that exploits the difference between two different representations of the same handwritten digit. One representation is dynamic, that is, the movement of the pen as the digit is written on a pressure-sensitive tablet. The other representation is static, that is, the image generated as a result of the movement of the pen tip. Two different hand movements for the same character may lead to similar images or different images of the same character may be generated by similar hand movements. Therefore, combining the two representations may improve recognition accuracy. We indeed notice during our experiments that classifiers using each of the two representations make mistakes for different samples and investigate ways to combine them efficiently.

This paper contains only a rough sketch and a subset of the work done and the interested reader is referred to [1], accessible over the Internet.

2 Preprocessing

The pressure sensitive tablet provides us the (x, y) coordinate information. In order to obtain the dynamic representation first we apply normalization. Then we perform spatial resampling to represent digits as constant length feature vectors. The static representation is based on solely the image of a character, as in off-

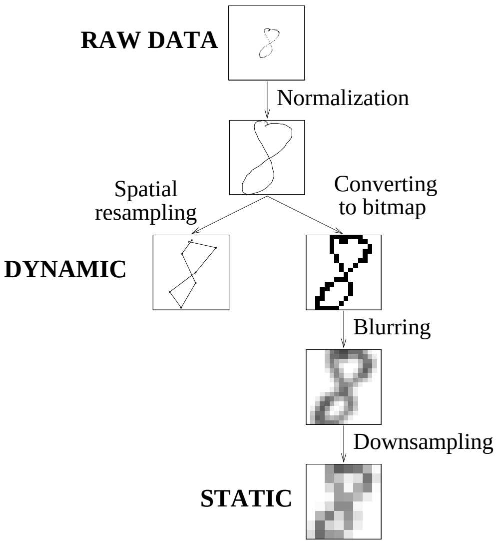

Figure 1: The stages of preprocessing.

line recognition tasks. First we convert normalized raw input to bitmaps, then we blur it and perform downsampling on it. Thus, input dimensionality is reduced keeping extra information from a higher resolution (Figure 1).

3 Multiple Classifiers

Because learning is an ill-posed problem, there is no algorithm that can converge to the optimal solution with finite data and each algorithm (or representation) depending on the biases built into it, converges to a different solution and fails under different circumstances. By suitably combining those, performance can be improved. The aim is to be able to find a set of classifiers who differ in their decisions to complement each other.

Combination approaches can be divided into two depending on the representational methodology: In *Uni-Representation*, all classifiers use the same representation. Thus the classifiers themselves should be different to cause them make different decisions. In *Multi-Representation*, different classifiers use different sensors or different features extracted from the same input data.

The approach we take is to make separate decisions using different sources and then combine the decisions. From an algorithmic point of view, there are two possibilities:

- The classifiers may use the same technique: they may use inputs in different representations, training sets may be distinct, learning may start from different initial conditions, or other parameters may be different.
- The different classifiers may be based on different techniques.

Orthogonal to these two dimensions, there are two architectural methodologies: In *Multiexpert* methods, the classifiers work in parallel. All the classifiers are trained on all patterns and given a pattern, they all give their decisions and a separate combiner computes the final decision using these. *Multistage* methods use a serial approach where the next classifier is trained/consulted only for patterns rejected by the previous classifier(s). Multistage methods suit well to multi-representation in that earlier classifiers use simple to extract features and handle a majority of the patterns. Complex features are only extracted for a small percentage without much increasing the overall complexity.

In this article, we adopt multi-representation where we have two classifiers using the two dynamic and static representations and the approaches that we examine are the following:

1. *Voting* takes a linear, weighted sum of the output of the learners. This is also known as *committees*, *ensembles*, or *linear opinion pools* and there exist different ways to compute weights [4, 11].
2. *Mixture of Experts* is similar to voting where weights are input dependent. Experts partition the input space among themselves and a separate gating model is responsible from choosing the right expert for a given input [2, 5].
3. *Stacked Generalization* uses a separate learner that is also trained and is not restricted to be linear weighted sum [10].
4. *Boosting* uses a serial approach where the second learner is trained to handle cases that cannot be learned by the first learner and a third learner learns the cases in which the first two disagree [3]
5. *Cascading* uses a first learner that is a simple classifier handling the majority of the cases and the second, more expensive classifier, is only consulted for patterns rejected by the first [6, 8].

4 Experiments

After preprocessing, we divide 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.

Table 1: Error percentages of samples misclassified by the two classifiers and by both.

Set	Dynamic	Static	Both
TR	1.02	1.24	0.25
CV	1.67	3.76	0.46
WD	1.74	4.27	0.70
WI	4.74	5.75	1.70

In the tables, we report the accuracy on the training set (TR), cross-validation set (CV), writer dependent test set (WD) and writer independent test set (WI). We also show the number of parameters (PS) to indicate the implementation complexity.

For the dynamic representation, we try with 8, 12 and 16 points where each point is composed of (x, y) coordinates, using the two classification schemes of k-NN and MLP. Though success seems to increase with more points, we adopted 8 points (16 dimensions) as difference in success is negligible but the complexity is reduced considerably with less points. Similarly for the static representation, we try several pixmap sizes of 8 by 8, 12 by 12, and 16 by 16. We decide on 8 by 8 bitmaps (64 dimensions) as, similarly, the difference in performance does not justify the increase in complexity.

We used the feedforward MLP as the base classifier in both representations, both having ten hidden units. Note that the important point in this work is not the absolute values given by the classifiers but the relative improvement gained by having multiple classifiers and representations.

We test to see if the two classifiers using the two representations fail on different samples or not (Table 1). It can be seen there that though both classifiers have around 5% error on the writer independent test set, if we know which one to use for any pattern, the error percentage can be reduced to less than 2%. Even less error may be possible by better integrating the two pieces of information than selecting one of the two.

The easiest way to combine the two pieces of knowledge is to concatenate the two feature vectors to obtain a combined feature vector (80 dimensions) and train a larger MLP with it. We see that the best is with 20 hidden units. As expected, this has better performance than the individual classifiers (Table 2).

Voting on the two classifiers is as good as concatenating but uses less parameters. We use a cooperative mixture of experts architecture where the experts are the two MLP-based classifiers and where the gating

Table 2: Results achieved using different techniques to combine the two pieces of static and dynamic representations.

Technique	WD	WI	PS
Dyn MLP	98.26, 0.31	95.26, 0.37	280
Sta MLP	95.73, 0.37	94.25, 0.25	760
Concat	98.67, 0.27	97.03, 0.41	1,830
Vote	98.41, 0.20	97.09, 0.33	1,040
ME	98.78, 0.25	96.81, 0.36	2,870
Stacking	98.65, 0.25	96.73, 0.14	1,360
Cascading	96.02, 0.57	96.41, 0.52	1,040

is another MLP with 20 hidden units which as input takes a concatenation of the two representations. This works better than the ME proper with linear experts and gating. The accuracy of competitive ME are a little less. In stacked generalization, the combiner is an MLP with 10 hidden which performs a little better than a linear one.

We could not implement boosting as the number of patterns on which the two MLPs do not agree is not sufficient to train a third one. In cascading, we use the dynamic MLP as the simple classifier and static MLP as the complex one. This is because the preprocessing required for the static MLP is time consuming. We again use 2-fold cross-validation to train the two classifiers with rejection threshold 0.99 where 30% are rejected by the dynamic MLP and train the static MLP. When the rejection threshold is less, success on writer dependent set is higher but there are not enough patterns to train the static MLP and thus success on writer independent set is less.

Visual presentation of accuracy versus the number of free parameters for the writer independent test sets is given in Fig. 2. Concatenation of the two representations increase accuracy considerably but leads to a large network. Approaches where there are two classifiers lead to simpler networks, because inputs are smaller. Voting seems to be the best. Stacking is better than voting on writer dependent set but worse on writer independent set, showing that it overfits the training sample; the combination method learned for the 30 writers does not carry over the 14 left out. Mixture of experts is also successful but needs more parameters as the gating network sees the concatenated input.

We think that cascading is the most promising approach among all as it leads to the simplest classifier. Preprocessing the static representation is the costly

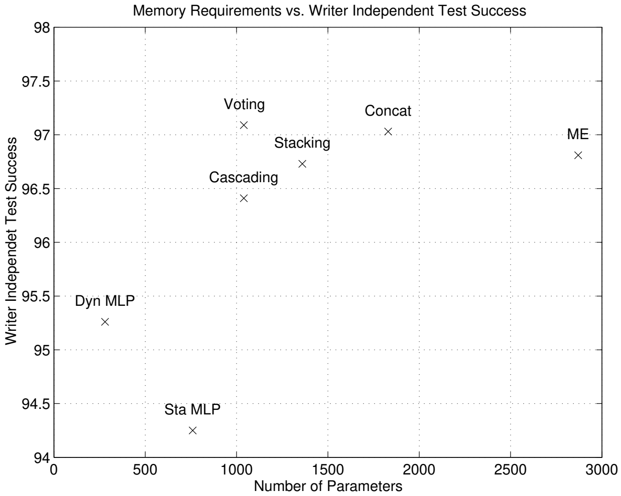

Figure 2: Average accuracy on writer independent test set versus the number of parameters.

part and by cascading we need this for only 31% of the writer independent test images (14% of the writer dependent test set), leading to the fastest classifier.

5 Conclusion

MLP classifiers we use are not very accurate individually but combining them we get higher accuracy. With the MLP using the dynamic representation, we have around 5% error on the writer independent set. The MLP using the static representation has around 6% error over the same set and seems to be worse than dynamic MLP. Combining the two, error percentage decreases to 3%. Adding a worse classifier decreases the error! This is because the two classifiers fail on different patterns. The final combined system has lower complexity and memory requirement than a single k-NN on one of the representations. That is the basic idea behind combining multiple models: The individual performances of the models do not matter much; combining a few quick and dirty classifiers, one can achieve performances that are much better than one, carefully optimized model.

According to our results given above, it seems like the actual way combining is done is not very significant. The choice of the classifiers rather than the choice of combination method is important. The greater the tendency to make correlated errors, the less impressive are the error reductions. The best way to have independent classifiers is to train them on different representations of the input thereby taking into account different data sources that provide complementary data.

Acknowledgments

This work is supported by Tübitak Grant EEEAG-143. Taylan Cemgil solved some of our problems with tablet hardware.

References

[1] F. Alimoğlu, "Combining multiple classifiers for pen-based handwritten digit recognition," M.S. Thesis, Inst. of Sciences and Engineering, Boğaziçi University, 1996. http://www.cmpe.boun.edu.tr/theses/

[2] E. Alpaydın, and M. I. Jordan, "Local Linear Perceptrons for Classification," *IEEE Trans. on Neural Networks*, Vol. 7, No. 3, pp. 788-792, 1996.

[3] H. Drucker, R. Schapire, and P. Simard, "Improving Performance in Neural Networks Using a Boosting Algorithm," in S. J. Hanson, J. Cowan, L. Giles (eds.), *Advances in NIPS*, 5, pp. 42–49, Morgan-Kaufmann, 1993.

[4] L. K. Hansen, and P. Salamon, "Neural Network Ensembles," *IEEE Trans. on PAMI*, Vol. 12, No. 10, pp. 993–1001, 1990.

[5] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, Vol. 3, pp. 79–87, 1991.

[6] C. Kaynak, "Methods of Combining Multiple Classifiers and Their Application to Handwritten Digit Recognition", M.S. Thesis, Inst. of Sciences and Engineering, Boğaziçi University, 1995.

[7] T. Pavlidis, and S. Mori, "Special Issue on Optical Character Recognition," *Proceedings of the IEEE*, vol. 80, no. 7, 1992.

[8] P. Pudil, J. Novovičová, S. Bláha, and J. Kittler, "Multistage Pattern Recognition with Reject Option," *11th IAPR ICPR*, Vol II, IEEE Computer Society Press, 92–95, 1992.

[9] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The State of the Art in On-line Handwriting Recognition," *IEEE Trans. on PAMI*, Vol. 12, No. 8, pp. 787–808, 1990.

[10] D. H. Wolpert, "Stacked Generalization," *Neural Networks*, Vol. 5, pp. 241–259, 1992.

[11] L. Xu, A. Krzyżak, and C. Y. Suen, "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Trans. on SMC*, Vol. 22, pp. 418–435, 1992.