ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 49-54

Combining Multi-Layer Perceptrons in classification problems[1]

E. Filippi, M. Costa, E.Pasero

Dipartimento di Elettronica, Politecnico di Torino
C.so Duca Degli Abruzzi 24 - 10129 Torino - Italy

Abstract

The impredictability of the performance after the training often limits the use of the Multi Layer Perceptron (MLP). The tuning of the gain parameter is committed to heuristic techniques and therefore these networks often tend to local minima. The choice of the training set can strongly influence the generalization ability producing overfitting and underfitting phoenomena. This paper presents a possible solution to these problems: the use of more networks to overcome the inefficiency of the single MLP. We experimented some techniques to merge the outputs of a committee of MLPs and we obtained a global performance free from the above problems. These techniques were used to classify handwritten digits from the NIST database, obtaining the best score among systems based only on the official training database.

1 Introduction

The performance of a Multi-Layer Perceptron (MLP) when used as a classifier is limited by many sources of errors. Among them, the inadequate or insufficient sampling of the training data, the sensitivity to the initial conditions and the inaccuracies in the learning procedure.

Two main approaches have been proposed to reduce the extent of these problems: statistical resampling techniques (bootstrap, cross-validation, etc.) and "smoothing" techniques (pruning, weight decay, etc.) [2]. Smoothing techniques try to improve performance by reducing the complexity of a single net, while the basic idea behind resampling techniques consists of generating multiple estimates of an unknown statistic and then combining the results.

Unfortunately MLPs cannot be combined in the parameter space: therefore, this process involves the generation of a new large "stacked" feed-forward network [8]. This is why resampling is usually limited to the choice of the best individual over a set of possible candidates through a cross-validation data set, discarding the rest. In the last years many authors have suggested that other strategies can perform better than pure winner-takes-all. Many of them [1, 4, 5] deal with methods for optimal training data distribution among the networks.

In this paper, we are concerned with two key issues in the creation of efficient network ensembles: how to optimize the design of the component networks, and how to combine them to achieve the best trade-off between performance and resource overhead. We carried out several experiments with simple MLP committees on a non-toy problem: optical recognition of handwritten numerals. Different models are compared, and their pros and cons discussed.

[1] Work partly supported by "progetto finalizzato sistemi informatici e Calcolo Parallelo" of CNR under grant n. 9100884.pf69

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 49-54

2 MLP ensembles

An MLP classifier can be thought either as a regression estimator, either as an approximated Bayes classifier [7], or as a "pure syntactic" classifier. In the following the three methods are shortly described.

2.1 Combining MLPs as regression estimators

Given a target function $T(x)$, a MLP can be trained to provide a regression estimation for it. In classification tasks, $T(x)$ is typically a multidimensional stochastic function assuming M different values T_i (where M is the number of classes) so that $T(x) = T_i$ if $x \in C_i$.

If a population of K nets share the same target function, then we can directly average their output vectors $Y_k(x)$:

$$Y_{ens}(x) = \frac{1}{K} \sum_{k=1}^{K} Y_k(x) \tag{1}$$

and then select the final winner class on these new outputs. As shown in [6], the mean square error (MSE) of the averaged estimate is less or equal to the average MSE of the individuals, and the reduction is maximal if these are mutually independent.

A natural extension of the simple average is the weighted sum of the output vectors. Some methods to find the weighting coefficients are suggested in [6].

2.2 Combining MLPs in a Bayesian framework

If we consider a MLP as an approximation of a Bayes classifier, first of all we have to take back the network output vector $Y(x)$ to a set of estimated post-probabilities $P(x \in C_i/x), i = 1,..M$. Then the class with the highest probability is chosen.

A simple way to combine a set of K classifiers is to build an average classifier with the new post-probabilities

$$P_{ens}(x \in C_i/x) = \frac{1}{K} \sum_{k=1}^{K} P_k(x \in C_i/x) \tag{2}$$

If we need some constraint on the result reliability, a rejection threshold can be easily imposed by accepting the answer only if $P(x \in C_i/x) > \alpha$ with $0 \leq \alpha \leq 1$.

A second approach is to search for the maximum upon i of the joint probabilities that $x \in C_i$ given x and given the answers of the K classifiers. If we suppose the networks to be statistically independent, then

$$P_{ens}(x \in C_i/x) = \eta \prod_{k=1}^{K} P_k(x \in C_i/x) \tag{3}$$

were η is a normalization factor.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 49-54

2.3 Combining MLPs by means of voting methods

If we consider the MLP as a syntactic classifier, then its response will be just a label corresponding to the chosen class.

When conflicts exist among the individual decisions, a plurality consensus scheme can be adopted: this is a very common procedure in fault tolerant computing. Consensus schemes can vary in the percentage of agreement required (from relative majority to unanimity): when the requested quorum is not reached, the pattern is rejected.

A theoretical basis can be provided for the analysis of the effect of coincident errors [3], which again emphasizes the role of error independence.

3 Application to handwritten digit recognition

Our experiments have been carried out on the database provided by NIST for the worldwide contest organized in 1992. It consists of a main set of 223125 samples (the suggested training set) and of a test set of 58646 samples drawn from a quite different population (we will address it as the "hard" test set).

In the first part of the work we randomly selected 20000 samples from the main NIST Database. Of these, 10000 made up a small training set, 5000 a cross-validation set, 5000 an "easy" test set. In addition, we tested our nets on the "hard" test set.

Then we checked the effectiveness of the most promising solution training a MLP ensemble on the whole main set.

We performed only scaling operations on the original data to produce 8*8 images with 256 gray levels, through two different algorithms: one (called "distorting") forces the output character to "touch" all four borders of the image, while the other (called "non-distorting") preserves the original aspect ratio.

Classification tasks provide us with an additional degree of freedom: i.e., the choice of a specific target coding. If we want to combine MLPs that are heterogeneous in this respect, we must define a more general framework. We therefore built a similarity measure $s_i, i = 1, ..M$ between the raw output vector Y and the target vector T_i corresponding to the i-th class as their dot product (normalized in the range [0,1]). We used the classical "One-Shot" encoding and random binary encoding.

If we regard s_i as a membership function for a fuzzy set, then we can estimate the post-probabilities according to the following formula:

$$P(x \in Ci/x) = \eta s_i \prod_{j \neq i} (1 - s_j) = \eta' \frac{s_i}{1 - s_i} \tag{4}$$

where η, η' are normalizing constants.

In OCR tasks usually a wrong answer is worse than a character rejection: as in [9], a useful reliability measure is defined:

$$\text{reliability} = \frac{1 - \text{error} - \text{rejection}}{1 - \text{rejection}} \tag{5}$$

3.1 Experimental results

Using the small training set, we generated several population of 5 up to 10 MLPs. Individuals in each population could differ in the training data subset, or in the weight initialization, or in the input data features, or in the output

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 49-54

target encoding. On the whole, the individual nets achieved widely varying results (at 0% rejection rate): from 2.53% to 5.2% error rate on the easy test set, and from 9.74% to 15.38% error rate on the hard test set. Then we built different ensembles with the methods described in section 2.

We observed very small differences between averaging in the output (1) or in the probability (2) space. Moreover, in our experiments we never obtained an appreciable improvement by using weighted sum instead of simple average. Thus we selected simple average in the probability space as our reference rule for the family of averaging methods.

	ensemble A		ensemble B		ensemble C	
rule	easy	hard	easy	hard	easy	hard
output average	17%	11%	64%	28%	38%	27%
probab. prod.	8%	6%	67%	28%	38%	23%
voting	9%	3%	32%	9%	19%	13%

Table 1: Relative improvement % achieved with different combination rules.

Table 1 compares averaging, product-of-probability and voting rules. We show the relative improvement (defined as the percentage error reduction with respect to the best individual in the population), that we obtained with 3 ensembles of 5 components. Ensemble A consists of 5 nets with different weight initialization, ensemble B consists of 5 nets with different random target encoding and ensemble C consists of 5 nets with different training data subsets. All the nets were trained with "distorted" data.

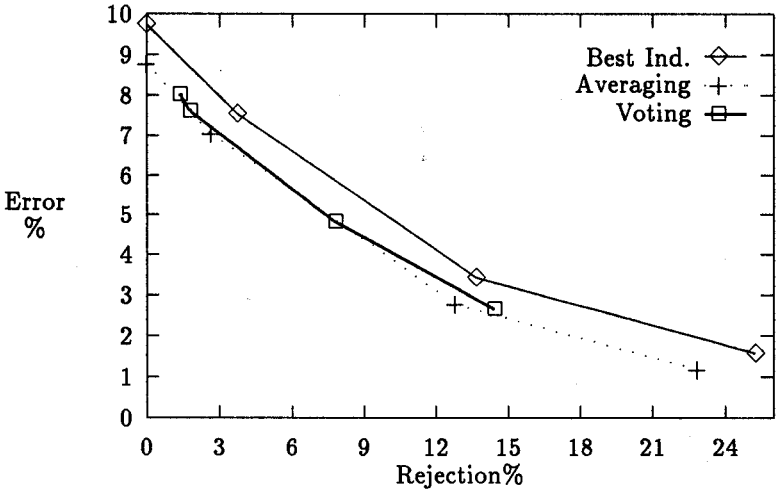

Figure 1: error rate versus rejection rate with different combination rules (hard test)

The relative improvement is very variable with respect to either the combining rule or the nature of the ensemble. It tends to be higher when the individual performance of the components is not very good: for example, combining nets with different disjoint subsets of the training data (table 1, ensemble C) is quite

effective, but in our experiments the improvement cannot compensate the individual worsening due to the training set reduction. This problem is sometimes solved in the literature by an ad-hoc generation of additional synthetic training prototypes [1].

In general, we obtained the best results with average, followed by probability product and voting. The difference between the first two can be explained by supposing that the product is more sensitive than average to network mutual dependence. Voting is the least effective due to the smaller quantity of information carried by each net.

Anyway, we noted that for certain rejection rates no significant difference appears (see fig. 1, plotting an example of the error-rejection curve for ensemble A).

We also extensively analyzed the trade-off between individual performance and mutual independence. Some results are reported in figure 2, showing the error rate versus the number of nets, for different ensembles built with the averaging rule.

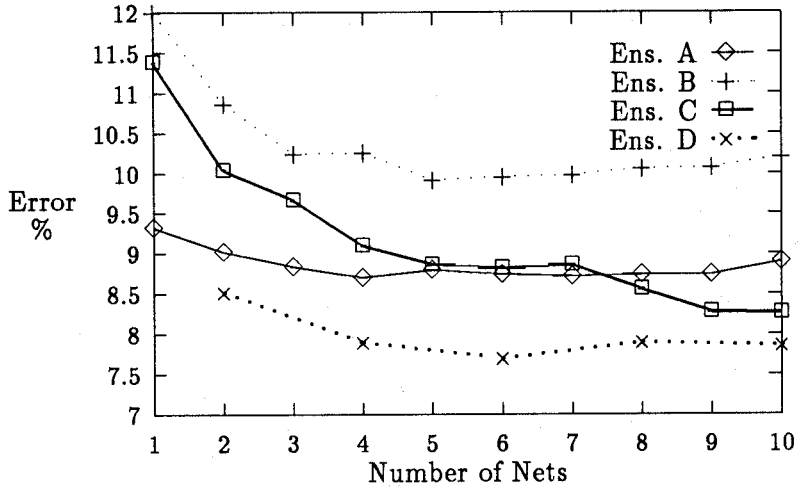

Figure 2: error rate on the hard test set versus number of nets for different ensembles.

The ensembles A and B consist of nets trained with different initial weights, with distorted and non-distorted data respectively. Ensemble C consists of nets with different random target encodings (trained with distorted data). Ensemble D consists for one half of individuals from population A and for the other half of individuals from population B. Nets are added to the ensemble according to their performance on the cross-validation set: thus the ensemble of one net is the best individual of the population. The comparison of ensembles A and B versus B and D shows that combining nets with different initial weights (A and B) is not as effective as combining nets with different output (C) or input (D) features, whatever is the individual net performance. This suggests that the effectiveness of the ensemble methods relies upon component independence as much as upon individual performance. Note that while the generation of more input features may be very expensive (or infeasible), the cost of differentiating the output targets is equivalent to the cost of differentiating the initial weights.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 49-54

Moreover, a critical number of components exists which limits the "useful" size of the ensemble, which is smaller if nets are strongly correlated.

Finally, on the basis of the above results, we trained two 2-layer Perceptrons (with $\approx$ 5000) synapses each on the whole training database, with standard plain backpropagation. We then combined them by averaging their outputs. The first one was fed with the "distorted" characters and achieved 4.1% error rate on the hard test set, while the second one was fed with the "non-distorted" characters and achieved 5.4%. The whole system is in effect a 3-layer locally connected feed-forward network with 10000 synapses, and reached 3.4%. This result equals the best one obtained in the NIST 1992 world contest by systems based only on the official training database.

4 Conclusions

MLP ensembles can be very useful to improve performance at a reasonable cost, providing to the designer an additional degree of freedom, in order to fit implementation constraints and better utilize available data. The resulting hybrid network is still a feed-forward network, with the interesting features of being locally connected and hierarchically organized.

A limit exists to the "useful" size of the ensemble, and often satisfactory results can be achieved with only two or three components.

Theoretical considerations and simulation results show that, whatever the combination rule is, the effectiveness of the solution relies on the network mutual independence at least as much as on their individual performance. Therefore, novel optimization criteria arise that may have a strong impact on the design of effective neural architectures.

References

[1] H. Drucker, R. Shapire, P. Simard: Improving Performance in Neural Networks Using a Boosting Algorithm. NIPS 5, p.43-49 , S.J. Hanson et al. eds., San Mateo, CA: Morgan Kaufmann Publishers (1993).

[2] W. Finnoff, F.Hergert, H.G.Zimmermann, "Improving Model Selection by Nonconvergent Methods", Neural Networks, vol.6, pp. 771-783, 1993.

[3] L.K Hansen, P.Salamon: Neural Network Ensembles. IEEE Trans.on Pattern Analysis and Machine Intelligence , 12(10):993-1001 (1990).

[4] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton: Adaptive mixtures of local experts. Neural Computation, 3(2):79-87 (1991).

[5] G. Paass: Assessing and Improving Neural Networks Predictions by the Bootstrap Algorithm. NIPS 5, p.197-203 , S.J Hanson et al. eds., San Mateo, CA: Morgan Kaufmann Publishers (1993).

[6] M.P. Perrone, L.N. Cooper: When Network Disagree: Ensemble Method for Neural Networks. "Neural Networks for Speech and Image processing", R.J. Mammone ed., Chapman-Hall (1993).

[7] E.A. Wan: Neural Network Classification: A Bayesian Interpretation. IEEE Trans. on Neural Networks 1(4):303-304 (1990).

[8] D.H. Wolpert: Stacked Generalization. Neural Networks, (5):241-259 (1992).

[9] L. Xu, A. Krzyzak, C.Y. Suen: Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition. IEEE Trans. on Systems, Man, and Cybernetics, 22(3):418-435 (1992).