ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 261-266

# Ensemble Methods for Multilayer Feedforward

† Carlos Hernández-Espinosa † Mercedes Fernández-Redondo †Mamen Ortiz-Gómez.

† Universidad Jaume I, Campus de Riu Sec, D. de Ingeniería y Ciencia de los Computadores, 12071 Castellón, Spain. e-mail: redondo@inf.uji.es, espinosa@inf.uji.es

**Abstract.** Training an ensemble of networks is an interesting way to improve the performance with respect to a single network. However there are several method to construct the ensemble and there are no results showing which one could be the most appropriate. In this paper we present a comparison of eleven different method. We have trained ensembles of a reduced number of networks (3 and 9) because in this case the computational cost is not high and the method is suitable for applications. The results show that the improvement in performance from three to nine networks is marginal. Also the improvement of performance of the different methods with respect to a simple ensemble is usually less than 1%.

## 1. Introduction

Perhaps, the most important property of a neural network is the generalization capability. The ability to correctly respond to inputs which were not used in the training set.

One technique to increase the generalization capability with respect to a single neural network consist on training an ensemble of neural network, i.e., to train a set of neural networks with different weight initialization or properties and combine the outputs of the different networks in a suitable manner to give a single output.

It is clear form the bibliography that this procedure increases the generalization capability. The error of a neural network can be decomposed into a bias and a variance [1,2]. The use of an ensemble keeps the bias constant and reduce the variance if the errors of the different networks are uncorrelated or negatively correlated. Therefore, it increases the generalization performance. The two key factors to design an ensemble are how to train the individual networks to get uncorrelated errors and how to combine the different outputs of the networks to give a single output.

Among the methods of combining the outputs, the two most popular are voting and output averaging [3]. In this paper we will normally use output averaging because it has no problems of ties.

In the other aspect, nowadays, there are several different methods in the bibliography to train the individual networks and construct the ensemble [1-9].

However, there is a lack of comparison among the different methods and it is not clear which one can provide better results.

In this paper, we present a comparison among eleven different methods of constructing the ensemble.

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 261-266

The organization of the paper is the following. In section two we briefly describe the different methods, in section three, we present the experimental results of the comparison and finally we conclude in section four.

## 2. Theory

In this section we briefly review the different ensemble methods, a full description can be found in the references.

### 2.1. Simple Ensemble

A simple ensemble can be constructed by training different networks with the same training set but with different random initialization in the weights. In this ensemble technique, we expect that the networks will converge to different local minimum and the errors will be uncorrelated.

### 2.2. Bagging

This ensemble method is described in [4]. The ensemble method consists on generating different datasets drawn at random with replacement from the original training set. After that, we train the different networks in the ensemble with these different datasets (one network per dataset). We have used datasets which have a number of training points equal to twice the number of points of the original training set, as it is recommended in the reference [1].

### 2.3. Bagging with Noise (Bagnoise)

It was proposed in [2]. It is a modification of Bagging, we use in this case datasets of size $10 \cdot N$ (number of training points) generated in the same way of Bagging, where N is the number of training points of the initial training set. And we introduce a random noise in every selected training point drawn from a normal distribution with a small variance.

### 2.4. Boosting

This ensemble method is reviewed in [3]. It is conceived for a ensemble of only three networks. It trains the three network of the ensemble with different training sets. The first network is trained with the whole training set, N input patterns. After this training, we pass all N patterns through the first network, using a subset of them such the new training set has 50% of patterns incorrectly classified by the first network and 50% classified correctly. With this new training set we train the second network. After the second machine is trained, the N original patterns are presented to both networks. If the two networks disagree in the classification, we add the training pattern to the third training set. Otherwise we discard the pattern. With this third training set we train the third network.

In the original theoretical derivation of the algorithm, evaluation of the test performance was as follows: present a test pattern to the three networks. If the first two networks aggre, use this label, otherwise use the labeling assigned by the third network. In addition to this voting scheme, we have use simple averaging of the outputs in our experiments.

### 2.5. CVC

It is reviewed in [1]. In k-fold cross-validation, the training set is divided into k subsets. Then, k-1 subsets are used to train the network and results are tested on the subset that was left out. Similarly, by changing the subset that is left out of the training process, one can construct k classifiers, each of which is trained on a slightly

different training set. This is the technique used in this method.

**2.6. Adaboost**

We have implemented the algorithm denominated "Adaboost.M1" in the reference [5]. In the algorithm the successive networks are trained with a training set selected at random from the original training set, but the probability of selecting a pattern changes depending on the correct classification of the pattern and on the performance of the last trained network. The algorithm is complex and the full description should be looked for in the reference. The method of combining the outputs of the networks is also particular to this algorithm. We have use this method and the usual output averaging in our experiments.

**2.7. Decorrelated (Deco)**

This ensemble method was proposed in [6]. It consists on introducing a penalty term added to the usual Backpropagation error function. The penalty term for network number *j* in the ensemble is in equation 1.

$$Penalty = \lambda \cdot d(i,j)(y - f_i) \cdot (y - f_j) \tag{1}$$

Where λ determines the strength of the penalty term and should be found by trial and error, $y$ is the target of the training pattern and $f_i$ and $f_j$ are the outputs of networks number $i$ and $j$ in the ensemble. The term $d(i,j)$ is in equation 2.

$$d(i,j) = \begin{cases} 1, & if\ \ i = j-1 \\ 0, & otherwise \end{cases} \tag{2}$$

**2.8. Decorrelated2 (Deco2)**

It was proposed also in reference [6]. It is basically the same method but with a different term $d(i,j)$ in the penalty. In this case the expression of $d(i,j)$ is in equation 3.

$$d(i,j) = \begin{cases} 1, & if\ \ i = j-1 \quad and \quad i\ is\ even \\ 0, & otherwise \end{cases} \tag{3}$$

**2.9. Evol**

This ensemble method was proposed in [7]. In each iteration (presentation of a training pattern), it is calculated the output of the ensemble for the input pattern by voting. If the output is correctly classified we continue with the next iteration and pattern. Otherwise, the network with an erroneous output and lower MSE (Mean Squared Error) is trained in this pattern until the output of the network is correct. This procedure is repeated for several networks until the vote of the ensemble correctly classifies the pattern. For a full description of the method see the reference.

**2.10. Cels**

It was proposed in [8]. This method also uses a penalty term added to the usual Backpropagation error function to decorrelate the output of the networks in the ensemble. In this case the penalty term for network number $i$ is in equation 4.

$$Penalty = \lambda \cdot (f_i - y) \cdot \sum_{j \neq i} (f_j - y) \tag{4}$$

Where $y$ is the target of the input pattern and $f_i$ and $f_j$ the outputs of networks number $i$ an $j$ for this pattern.

The authors propose in the paper the winner-take-all procedure to combine the outputs of the individual networks, i. e. the highest output is the output of the ensemble. We have used this procedure and the usual output averaging.

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 261-266

**2.11. Ola**

This ensemble method was proposed in [9]. In this method, first, several datasets are generated by using bagging, with a number of training patterns in each dataset equal to the original number of the training set. Every network is trained in one of this datasets and in what it is called virtual data. The virtual data for network *i* is generated by selecting randomly samples for the original training set and perturbing the sample with a random noise drawn from a normal distribution with small variance. The target for this new virtual sample is calculated by the output of the ensemble without network number *i* for this sample. For a full description of the procedure see the reference.

## 3. Experimental results

We have applied the eleven ensemble methods to nine different classification problems. They are from the UCI repository of machine learning databases. Their names are Balance Scale (BALANCE), Cylinders Bands (BANDS), Liver Disorders (BUPA), Credit Approval (CREDIT), Glass Identification (GLASS), Heart Disease (HEART), the Monk's Problems (MONK'1, MONK'2) and Voting Records (VOTE). The complete data and a full description can be found in the UCI repository (http://www.ics.uci.edu/~mlearn/MLRepository.html).

We have constructed ensembles of a low number of networks, in particular 3 and 9 networks. We think that the ensemble methodology can be useful for an application if the number of networks in the ensemble is low. Otherwise the computational cost (of an ensemble with a high number of network) would be high an the method impractical.

The first step was to determine the right parameters for each database, in the case of methods Cels (parameter lambda of the penalty), Ola (standard deviation of the noise), Deco and Deco2 (parameter lambda of the penalty) and BagNoise (standard deviation of the noise). The value of the final parameters obtained by trial and error is in Table 1.

With this parameters we trained the ensembles of three and nine networks. We repeated this process of training an ensemble ten times for different partition of data in training, cross-validation and test sets. In order to obtain a mean performance of the ensemble for each database (the mean of the ten ensembles) and and error in the performance calculated by standard error theory.

Table 1. Parameters for different ensemble methods.

| | **Cels** | | **Ola** | | **Deco** | **Deco2** | **Bag Noise** |
|---|---|---|---|---|---|---|---|
| | Networks in Ensemble | | Networks in Ensemble | | | | |
| | 3 | 9 | 3 | 9 | | | |
| **Balance** | 0.1 | 0.1 | 0.5 | 0.5 | 1 | 0.6 | 0.1 |
| **Band** | 0.5 | 0.25 | 0.5 | 0.5 | 1 | 0.6 | 0.2 |
| **Bupa** | 0.75 | 0.1 | 0.5 | 0.6 | 0.8 | 0.2 | 0.1 |
| **Credit** | 0.1 | 0.75 | 0.6 | 0.6 | 0.2 | 0.2 | 0.7 |
| **Glass** | 0.25 | 0.1 | 0.3 | 0.2 | 0.6 | 0.6 | 0.2 |
| **Hear** | 0.5 | 0.25 | 0.3 | 0.4 | 0.6 | 0.6 | 0.4 |
| **Monk1** | 0.25 | 0.1 | 0.2 | 0.2 | 0.6 | 0.6 | 0.1 |
| **Monk2** | 0.1 | 0.1 | 0.6 | 0.6 | 0.4 | 0.8 | 0.1 |
| **Vote** | 0.5 | 0.75 | 0.3 | 0.3 | 0.4 | 0.6 | 0.1 |

The results of the performance are in table 2 for the case of ensembles of three networks and in table 3 for the case of nine.

We also include in table 2 the performance of a single

network for comparison.

Table 2. Performances of the different methods for an ensemble with three networks.

| | DATABASE | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **METHOD** | **Balance** | **Band** | **Bupa** | **Credit** | **Glass** | **Hear** | **Monk1** | **Monk2** | **Vote** |
| **Single Network** | 87.6±0.6 | 72.4±1.0 | 58.3±0.6 | 85.6±0.5 | 78.5±0.1 | 82.0±0.9 | 74.3±1.1 | 65.9±0.5 | 95.0±0.4 |
| **Adaboost** | 95.9±0.5 | 73.1±1.4 | 72.4±1.7 | 85.8±1.0 | 92.8±1.6 | 81±2 | 70±7 | 79.0±1.8 | 95.6±0.7 |
| **Bagging** | 94.6±0.9 | 72.9±1.3 | 72.6±1.6 | 87.2±0.5 | 93.8±1.1 | 84.2±1.1 | 98.3±1.0 | 87±1.6 | 96.1±0.7 |
| **Bag Noise** | 90.6±0.7 | 76.2±1.0 | 64.4±1.5 | 86.7±0.6 | 81.0±1.2 | 82.9±1.2 | 98.6±0.9 | 89.1±1.7 | 96.6±0.5 |
| **Boosting** | 94.8±0.7 | 73.6±1.3 | 70.7±1.2 | 86.5±0.5 | 92.8±1.1 | 81.7±1.4 | 98.5±1.4 | 87±2 | 94.9±0.6 |
| **Cels** | 96.0±0.5 | 73.3±1.0 | 69.3±1.4 | 86.8±0.7 | 94.4±0.8 | 83.2±1.3 | 100±0 | 100±0 | 95.5±0.6 |
| **CVC** | 94.5±0.6 | 72.5±1.0 | 72.7±1.5 | 87.0±0.6 | 92.4±1.0 | 84.6±1.0 | 97.0±1.5 | 82.1±1.2 | 96.3±0.6 |
| **Deco.** | 96.6±0.3 | 86.6±0.7 | 72.5±1.4 | 86.6±0.7 | 94.6±0.8 | 82.9±1.3 | 98.9±1.1 | 90.0±1.6 | 95.9±0.6 |
| **Deco2** | 95.8±0.4 | 72.7±1.6 | 72.7±1.6 | 86.4±0.7 | 95.4±0.8 | 82.9±1.5 | 99.1±0.6 | 89.8±1.1 | 95.5±0.6 |
| **Evol** | 57±6 | 59±4 | 42.0±1.9 | 53.8±1.8 | 44±7 | 58±2 | 51.4±1.1 | 57±5 | 62±4 |
| **Ola** | 90.2±1.0 | 68±2 | 69±2 | 84.4±0.9 | 77±2 | 79.2±1.9 | 99.87±0.12 | 71.5±1.9 | 87.9±1.5 |
| **Simple. Ensemble** | 95.8±0.7 | 73.5±1.2 | 71.9±1.4 | 86.3±0.8 | 93.6±0.6 | 82.9±1.3 | 98.6±0.9 | 90.5±1.1 | 95.6±0.5 |

Table 3. Performances of the different methods for an ensemble with nine networks.

| | DATABASE | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **METHOD** | **Balance** | **Band** | **Bupa** | **Credit** | **Glass** | **Hear** | **Monk1** | **Monk2** | **Vote** |
| **Adaboost** | 96.0±0.5 | 72.0±1.9 | 72.1±1.8 | 85.1±1.0 | 94±4 | 80.5±1.5 | -- | 82±2 | 95.4±0.6 |
| **Bagging** | 95.2±0.6 | 74.2±1.4 | 73.0±1.2 | 87.2±0.6 | 95.8±0.9 | 83.7±1.1 | 99.3±0.8 | 88.4±1.3 | 96.5±0.6 |
| **Bag Noise** | 90.9±0.8 | 74.4±1.6 | 65±2 | 87.1±0.5 | 81.8±1.2 | 82.9±1.1 | 98.6±0.9 | 91.5±1.4 | 96.5±0.6 |
| **Cels** | 95.7±0.6 | 72.2±1.4 | 72.3±1.2 | 86.4±0.6 | 95.8±0.8 | 82.9±1.4 | 100±0 | 100±0 | 95.9±0.7 |
| **CVC** | 95.5±0.6 | 74.5±1.4 | 72.3±1.1 | 86.8±0.8 | 93.6±0.8 | 83.2±1.3 | 99.3±0.8 | 89.8±1.2 | 96.1±0.7 |
| **Deco.** | 96.9±0.4 | 73.1±1.2 | 71.1±1.2 | 86.5±0.7 | 95.4±1.0 | 83.2±1.4 | 99.3±0.8 | 92.1±1.0 | 95.5±0.7 |
| **Deco.2** | 96.2±0.5 | 73.8±1.2 | 72.0±1.2 | 86.3±0.7 | 94.6±0.8 | 83.6±1.5 | 99.1±0.6 | 91.4±1.0 | 95.5±0.6 |
| **Evol** | 46±6 | 58±4 | 55±3 | 54±4 | 51±8 | 63±5 | 55±2 | 62±4 | 66±7 |
| **Ola** | 89.2±1.0 | 68.5±1.7 | 69.3±1.4 | 84.9±0.9 | 60±4 | 66.6±1.5 | 94±3 | 70.6±1.3 | 60.6±0.9 |
| **Simple Ensemble** | 95.4±0.7 | 73.6±1.2 | 71.9±1.2 | 86.6±0.7 | 94.8±0.7 | 83.1±1.5 | 99.4±0.6 | 91.1±1.1 | 95.6±0.5 |

As commented before, for ensembles Adaboost, Cels, and Boosting, there is a particular method to combine the outputs of the network and we have also used output averaging. In tables 2 and 3 we have included the best results of these two methods of combining the outputs, which are output averaging for Boosting and Adaboost. Also it is output averaging for Cels except for the case of databases Monk1 and Monk2 in the ensemble of 3 networks.

By comparing the results of table 2 with the results of a single network we can see that there is a clear improvement by the use of the ensemble methods. The improvement in performance ranges from 0.6% in database Vote to 24.6% in Monk2, so it is problem dependent.

There is, however, one exception in the method Evol. This method did not work well in our experiments. In the original reference the method was tested in only one database, the database Heart. The results for a single network were 60%, for a simple ensemble 61.42% and for Evol 67.14%. Comparing with our results, the results of Evol are similar, but our results for a single network and a simple ensemble are

clearly better.
Comparing the results of the other ensemble methods with the results of a single ensemble, we can see that the differences in performance are low. They are usually around 1% and there is no a clear better method.
Now, we can compare the results of tables 2 and 3 for an ensemble of 3 and 9 networks. We can see that the results are similar and the improvement of training 9 networks is marginal. Taking into account the computational cost, we can say that the best alternative for an application is an ensemble of 3 networks.
Perhaps, the differences among the different methods will become more important for ensembles of higher number of networks. For example, in reference [2] the ensembles were of 40 networks. As we pointed out before, we thin that such ensembles are impractical for applications due to the computational cost. Anyway, a future research will go in this direction.

## 4. Conclusions

In this paper we have presented a comparison of eleven different to construct an ensemble of networks, using nine different databases. We trained ensembles of a reduced number of networks, in particular three and nine networks, because in this case the computational cost is not high and the method is suitable for applications. The results showed that there is a clear improvement by the use of the ensemble methods. Also the improvement in performance from 3 networks in the ensemble to 9 networks is marginal. Taking into account the computational cost, an ensemble of 3 networks is the best alternative. Finally, the performance of the different method is similar and there is no a clear better method. Perhaps, the differences among the different methods will become more important for ensembles of higher number of networks, for example 30 or 40. A future research will go in this direction.

## References.

1. Tumer, K., Ghosh, J., “Error correlation and error reduction in ensemble classifiers”, Connection Science, vol. 8, nos. 3 & 4, pp. 385-404, 1996.
2. Raviv, Y., Intrator, N., “Bootstrapping with Noise: An Effective Regularization Technique”, Connection Science, vol. 8, no. 3 & 4, pp. 355-372, 1996.
3. Drucker, H., Cortes, C., Jackel, D., et alt., “Boosting and Other Ensemble Methods”, Neural Computation, vol. 6, pp. 1289-1301, 1994.
4. Breiman, L., “Bagging Predictors”, Machine Learning, vol. 24, pp. 123-140, 1996.
5. Freund, .Y., Schapire, R., “Experiments with a New Boosting Algorithm”, Proc. of the Thirteenth Int. Conf. on Machine Learning, pp. 148-156, 1996.
6. Rosen, B., “Ensemble Learning Using Decorrelated Neural Networks”, Connection Science, vol. 8, no. 3 & 4, pp. 373-383, 1996.
7. Auda, G., Kamel, M., “EVOL: Ensembles Voting On-Line”, Proc. of the World Congress on Computational Intelligence, pp. 1356-1360, 1998.
8. Liu, Y., Yao, X., “A Cooperative Ensemble Learning System”, Proc. of the World Congress on Computational Intelligence, pp. 2202-2207, 1998.
9. Jang, M., Cho, S., “Ensemble Learning Using Observational Learning Theory”, Proc. of the Int. Joint Conf. on Neural Networks, vol. 2, pp. 1281-1286, 1999.