Ensembles of Nonconformist Neural Networks

M.C. van Wezel M.D. Out W.A. Kosters

Leiden Institute of Advanced Computer Science
Universiteit Leiden
P.O. Box 9512, 2300 RA Leiden, The Netherlands
Email: {michiel,mout,kosters}@liacs.nl

Abstract

In this paper a novel scheme is proposed for training feed-forward networks in an ensemble. In this approach we do not only minimize the error of each network on the training set, but we also reduce the correlation of the errors among the networks. Experimental results on various datasets show that the performance of bagging (i.e., taking the unweighted mean of the networks) improves when compared to the traditional training scheme.

1 Introduction

Recently ensembles of feed-forward networks receive a lot of attention. In this approach multiple feed-forward networks which are trained for the same problem using different initial weight configurations, optimization procedures, training sets, or even architectures are combined into a joined forecast. This forecast, a linear combination of the network outputs, has a lower error variance in comparison to single networks (for the statistical background, see [2]) and experimental results show that this technique gives a significant improvement in predictions.

In the literature various methods of determining suitable coefficients of the linear combination have been proposed. We mention the technique *bagging*, i.e., take the unweighted mean of the network outputs, and the technique which finds the coefficients from a maximum likelihood perspective (see [7]).

The performance of the ensemble however does not only depend on the individual performance of the trained feed-forward networks, but also on the rate of dependence among the network predictions. In fact, it can be shown that conflicting behavior between the networks is exploited in an ensemble to arrive at more accurate predictions (see [4]). However, this observation is not taken into account during the training process of the feed-forward networks (which consists solely of minimizing an error term containing the mean squared error on the training set and possibly a weight decay term to prevent overfitting). We therefore feel that it might be helpful to anticipate the later use of the network predictions, and to stimulate different behavior of the networks in an ensemble. One could look upon the networks as being *nonconformists*, in the sense that all networks attempt to express an 'opinion' that disagrees as much as possible from the opinions of the colleage networks.

In order to achieve this we propose a different training scheme in which we do not only minimize the mean squared error of each network, but also the covariance

$i, j : 1, \ldots, N$	Index for the N input-target data.
$\mathbf{x}_i = (x_{i1}, \ldots, x_{in})^T$	i-th input data vector (n-dimensional).
y_i	i-th target value (scalar).
$\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})^T$	Vector of outputs of the K ensemble members on input vector $\mathbf{x}_i$.
$\mathbf{e}_i = (e_{i1}, \ldots, e_{iK})^T$	Vector of errors of the ensemble members on input vector $\mathbf{x}_i$: $e_{ik} \equiv (z_{ik} - y_i)$.
$\mathrm{cov}(e_k, e_\ell)$	Covariance of errors of networks k and ℓ.
Σ	$K \times K$ covariance matrix of the errors of the ensemble members.
$k, \ell : 1, \ldots, K$	Indices for the K members of the ensemble.
$\mathbf{w} = (w_1, \ldots, w_K)^T$	Vector of ensemble weights. Weight w_k denotes the weight of ensemble member k in the linear combination.
λ	Scale parameter that determines the influence of the covariance cost term during training.

Table 1: Summary of notations.

of the errors among the networks. This will lead to less correlated predictions and, as we will see, to an improvement in the performance of the bagging technique.

The outline of this paper is as follows. First we describe the new training scheme after giving the formal definitions to make the notion of neural network ensembles and the rate of dependence between the networks more precise. Then we present and discuss the experimental results obtained by applying this technique to various datasets. We conclude this paper with some remarks, conclusions and recommendations for further research. The notations used in this paper are summarized in Table 1.

2 Neural Network Ensembles

A neural network ensemble is usually a linear combination of individual neural networks: the 'members' of the ensemble. This means that Q_i, the output of the ensemble on the i-th input vector $\mathbf{x}_i$, is given by $Q_i = \sum_k w_k z_{ik}$. Here, z_{ik} denotes the output of ensemble member k on the i-th input vector, and w_k is the weight that is assigned to ensemble member k in the linear combination. The weights $w_k, k = 1, \ldots, K$ are collectively denoted by $\mathbf{w}$. Furthermore, the error e_{ik} of ensemble member k on the i-th input vector is given by $e_{ik} = (z_{ik} - y_i)$, where y_i denotes the target value for input vector $\mathbf{x}_i$.

Before the K ensemble members are combined into a single forecast, they are trained using the backpropagation algorithm to minimize the mean squared error function E^{old}, given by $E_k^{\mathrm{old}} = 1/N \sum\limits_i e_{ik}^2$ for ensemble member k.

3 Optimizing Ensemble Performance

Two commonly used methods of combining the networks in an ensemble are *bagging* and *bumping*. In bagging, $w_k = 1/K, k = 1, \ldots, K$, so all networks are equally important in the combination. In bumping, only the network ℓ with the lowest error on the training data is used for the combined forecast, so $w_\ell = 1$ and $w_k = 0, k \neq \ell$.

More sophisticated schemes (see [1, 2, 5, 4, 7]) attempt to minimize a cost function that expresses the quality of the combination as a function of the ensemble weights $\mathbf{w}$. This cost function contains the covariance matrix Σ of the errors of the ensemble members as a component. In fact, it can be written as $\mathbf{w}^T \Sigma \mathbf{w}$ (see [7]). This is just the variance of the error of the combined forecast (see [2]). If the expectation of the error is 0, the variance of the error equals the mean squared error (MSE), so variance minimization is equivalent to error minimization.

Besides choosing optimal values for the components of $\mathbf{w}$ in order to minimize the variance $\mathbf{w}^T \Sigma \mathbf{w}$ of the error of the combined forecast, it is a logical step to attempt to minimize the elements of Σ, the covariances of the errors of the ensemble members, in the network training process. Therefore, we train each network k in the ensemble using a modified error function:

$$E_k^{\text{new}} = E_k^{\text{old}} + \lambda CT_k \ . \tag{1}$$

Here E_k^{old} is the usual mean squared error cost function, λ controls the weight of the new cost term and CT_k is given by the sum of the squared error covariances of ensemble member k with members $1, \ldots, K$ (with k itself omitted)

$$CT_k = \sum_{\substack{\ell=1 \\ \ell \neq k}}^{K} \operatorname{cov}(e_k, e_\ell)^2 \ . \tag{2}$$

The covariance of the errors of ensemble members k and ℓ is given by

$$\begin{aligned} \operatorname{cov}(e_k, e_\ell) &= \frac{1}{N} \sum_{i=1}^{N} e_{ki} e_{\ell i} - \frac{1}{N^2} (\sum_{i=1}^{N} e_{ki}) (\sum_{j=1}^{N} e_{\ell j}) \\ &= B_{k\ell} - C_{k\ell}, \end{aligned} \tag{3}$$

with $B_{k\ell}$ and $C_{k\ell}$ the first and second term of (3).

The minimization of (1) is performed in two stages. In the first stage, the networks are trained using only the MSE cost function E^{old}. After each network has been trained in this manner, Σ is estimated, CT is added to the cost function and the networks are retrained using E^{new}.

This 'retraining' is repeated for a number of iterations. This is necessary because the errors of the ensemble members, and thus also the covariances of the errors, change when the outputs of the network change. So, if two networks are trained subsequently using E^{new}, the minimum of E^{new} found for the first network is not up to date anymore after the second network has been trained, and therefore the first network will have to be retrained. Usually, after a few iterations the changes that result from the retraining process become negligible, so the training process can be stopped. This would suggest that the method does not suffer from convergence problems, but at this moment we have no formal proof of convergence. In the experiments described in Section 5 we repeat the second training step for 1000 cycles.

4 Partial Derivatives of the New Cost Function

In order to minimize cost function (1) by gradient techniques, we have to calculate the partial derivatives of CT_k with respect to the parameters (network weights and biases) of ensemble member k. These partial derivatives are computed in this section. We assume a three layer network but the derivations are easily extended to more layers. For a network weight or bias w of network k this partial derivative can be computed using the chain rule

$$\frac{\partial CT_k}{\partial w} = \sum_{\ell \neq k} 2 \operatorname{cov}(e_k, e_\ell) \left\{ \frac{\partial \left(B_{k\ell} - C_{k\ell} \right)}{\partial w} \right\}. \tag{4}$$

The partial derivatives $\partial B_{k\ell}/\partial w$ and $\partial C_{k\ell}/\partial w$ can be rewritten as follows:

$$\frac{\partial B_{k\ell}}{\partial w} = \frac{1}{N} \sum_i \frac{\partial e_{ki} e_{\ell i}}{\partial w} = \frac{1}{N} \sum_i e_{ki} \frac{\partial e_{\ell i}}{\partial w}, \tag{5}$$

$$\frac{\partial C_{k\ell}}{\partial w} = \frac{1}{N^2} (\sum_i e_{ki}) (\sum_j \frac{\partial e_{\ell j}}{\partial w}), \tag{6}$$

$$\frac{\partial e_{\ell j}}{\partial w} = \frac{\partial z_{\ell j}}{\partial w}. \tag{7}$$

Finally, the partial derivatives of the network outputs $\partial z_{\ell i}$ with respect to a network weight w can be computed as follows. Let a, b, c index input, hidden and output units respectively. First define

$$\delta_{ci} = g_c{}'(h_{ci}), \tag{8}$$

$$\delta_{bi} = \left(\sum_{\text{outputs } c} \delta_{ci} w_{bc} \right) g_b{}'(h_{bi}), \tag{9}$$

where $g_u(\,)$ denotes the transfer function of unit u of ensemble member ℓ, h_u is the input to unit u, w_{bc} is the weight of a hidden to output connection from hidden unit b to output unit c and w_{ab} is the weight of a connection from an input unit a to a hidden unit b. Note that the summation in Equation 9 allows for more than one output. In our experiments we only use networks with one output, but in principle multiple outputs are possible.

Now, for the partial derivatives of the network outputs we have:

$$\frac{\partial z_{\ell i}}{\partial w_{bc}} = \delta_{ci} V_{bi} \tag{10}$$

$$\frac{\partial z_{\ell i}}{\partial w_{ab}} = \delta_{bi} x_{ai} \tag{11}$$

where V_b denotes the output of unit b in the hidden layer and x_{ai} denotes the a-th element of the i-th input pattern. The algorithm needed to calculate of $\partial CT_\ell/\partial \mathbf{w}$ is given in pseudo-code in Figure 1.

(0) **initialize** all partial derivatives and $B's$ and $C's$ to 0.
(1) **for all** patterns i **do**
(2) {
(3) **compute** δ's for all units **for this** i using Eqs. 8, 9
(4) **compute** partial derivatives $\partial e_{\ell i}/\partial w$ $\forall w$ **for this** i
(.) using Eqs. 7, 10, 11
(5) **for all** networks $k = 1, \ldots, K$ **do**
(6) {
(7) **update** $\partial C_{k\ell}/\partial w$ $\forall w$ **for this** i using Eq. 6
(8) **update** $\partial B_{k\ell}/\partial w$ $\forall w$ **for this** i using Eq. 5
(9) **update** $B_{k\ell}, C_{k\ell}$ **for this** i
(10) }
(11) }
(12) **for all** networks $k = 1, \ldots, K$ **do**
(13) {
(14) **compute** the covariance $\mathrm{cov}(e_k, e_\ell)$
(15) **compute** partial derivatives of $CT_{\ell k}$ using Eq. 4
(..) and add to CT_ℓ
(16) }

Figure 1: Algorithm in pseudo-code for the computation of $\partial CT_\ell/\partial \mathbf{w}$. $CT_{\ell k}$ denotes the contribution of networkpair (ℓ, k) to the cost term CT_ℓ.

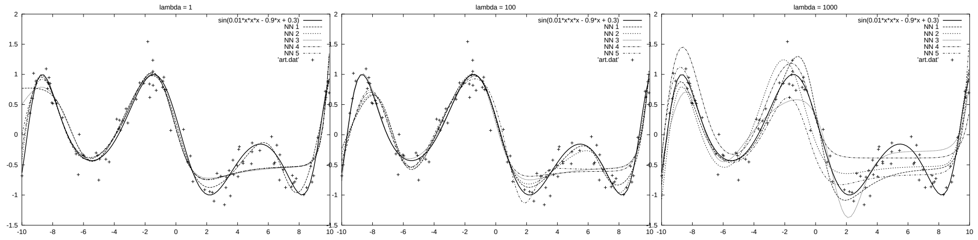

Figure 2: The artificial dataset. Shown are the regression (thick line), the actual data used in the experiments (crosses) and the functions represented by the five ensemble members. From left to right: $\lambda = 1, 100, 1000$.

5 Experiments and Results

We implemented the training method described in the previous section and experimented with it on several datasets.

5.1 Datasets

In our experiments we used three datasets: two real-life data sets and one artificial data set. These datasets are described below.

The artificial dataset: This dataset contains 100 patterns, of which 75 are used for training and 25 for testing. The input and output dimensionalities are 1. The 100 x-values were drawn from a uniform distribution on $[-10, 10]$. The correspond-

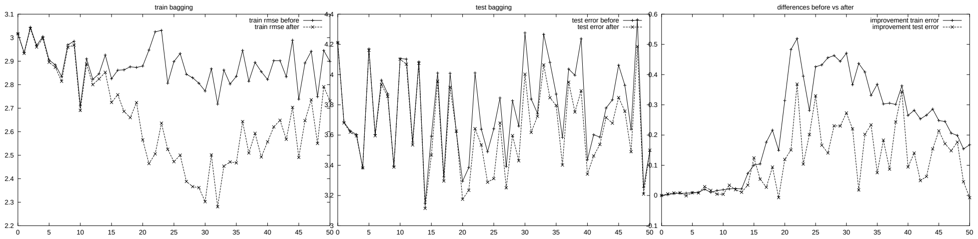

Figure 3: Results for the Boston housing dataset. These plots show the RMSE-values (vertical axis) for increasing λ-values. The left graph concerns training data, the middle graph test data and the right graph shows the improvement in RMSE.

ing y values were computed by

$$y = \sin(0.01x^3 - 0.9x + 0.3) + \epsilon \tag{12}$$

where ϵ represents added Gaussian noise with a standard-deviation of 0.2. The data in this dataset are depicted as the scattered points in Figure 2.

The marketing dataset: This dataset is concerned with predicting market shares for a brand of a fast moving consumer good. It contains 255 patterns with 6 inputs and 1 target; 150 randomly selected patterns were used for training, the rest for testing. The inputs represent marketing variables such as price and product-availability. The output represents the market share as a percentage. Knowing the functional relationship between these variables and the market share has obvious advantages.

The 'Boston Housing Dataset': This is a well known dataset concerning tracts in Boston. Thirteen (mainly socio-economic) variables are available as input for each tract. The target value is the median price for owner-occupied houses in the tract. The number of samples in the dataset is 506. See [3] for more information.

5.2 Results

To validate that the above described method was correctly implemented we applied the program to the artificial dataset using various values of λ, the parameter that controls the contribution of the CT part to the total error function. One would expect that as λ increases, the mappings that are implemented by the various ensemble members become more and more different. This is confirmed by the results shown in Figure 2. In this figure, it can clearly be seen that the output functions diverge for large values of λ.

Next, we applied the method to the marketing and the Boston dataset: the two real-life datasets. All results shown here are averaged over five runs. All trained networks had two hidden nodes. All ensembles consisted of five networks. Our experiments had the following setup. First, an ensemble of networks was generated and randomly initialized. The members of the ensemble were then trained using only the MSE error function E^{old}, and subsequently retrained for 1000 iterations using the extended cost function E^{new}. The RMSE values (obtained by taking the root of the MSE) of the combined prediction that was obtained using bagging

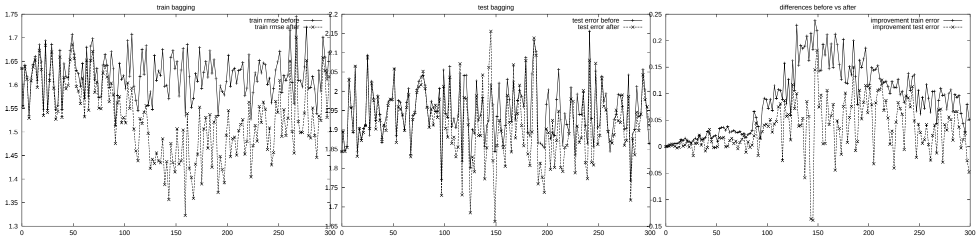

Figure 4: Results for the marketing dataset. For explanation see Figure 3.

	bagging				maximum likelihood			
	before		after		before		after	
λ value	train	test	train	test	train	test	train	test
0	3.017	4.212	3.017	4.212	2.921	4.269	2.921	4.268
5	2.905	4.170	2.895	4.161	2.713	4.286	2.702	4.389
10	2.710	4.106	2.691	4.102	2.631	4.178	2.616	4.209
15	2.825	3.592	2.725	3.468	2.714	3.415	2.652	3.368
20	2.879	3.295	2.565	3.175	2.820	3.282	2.547	3.175
25	2.898	3.641	2.472	3.312	2.723	3.595	2.424	3.227
30	2.773	4.276	2.302	4.003	2.673	4.216	2.263	3.946
35	2.835	3.871	2.467	3.796	2.794	3.822	2.449	3.827
40	2.822	3.436	2.557	3.342	2.715	3.462	2.525	3.303
45	2.739	4.062	2.491	3.848	2.676	4.054	2.469	3.819
50	2.899	3.493	2.731	3.500	2.737	3.522	2.685	3.544

Table 2: RMSE values for the Boston dataset

were saved before the second training phase and after the second training phase. These before/after RMSE values are reported here.

The results for the Boston dataset can be seen in Figure 3. Note that the difference between the RMSE value before and after the second training phase is negligible for small λ-values, but becomes significant for larger λ-values. Although the graphs depicted here are very peaky, there is a clear improvement for λ-values around 30. The RMSE values using bagging and the maximum likelihood method introduced in [7] are shown numerically in Table 2. This table should be interpreted by comparing the 'before' with the 'after' values in each row. The RMSE value in the 'before' columns represent the ensemble errors obtained after the first training stage (networks trained without CT), the RMSE value in the 'after' columns represent the ensemble errors that are obtained after re-training with CT added to the cost function. A clear improvement can be noted for λ values around 25. It is interesting to see that the improvement for both weighting methods is approximately equal. Intuitively, one would expect a more substantial improvement for bagging than for the maximum likelihood weighting, because the latter method squeezes more performance gain out of the optimal choice for $\mathbf{w}$, suggesting that there remains less to be gained from training with the adjusted cost function.

The results for the marketing dataset can be seen in Figure 4. These results are less convincing, but there is still an improvement visible. Note that the largest improvement is achieved at a λ-value of around 150. The best λ value for the Boston dataset was approximately 30, so the 'ideal' value for λ is obviously dependent on the dataset. Due to space limitations we do not tabulate the numerical results.

6 Conclusions and Further Research

We may conclude that the approach provides promising results. Reducing the covariance of the errors among the ensemble members during the training process contributes in a positive way to the performance of the bagging-technique.

We also examined a variation of the described technique in which the ensemble was stepwise expanded with a new network. The training process of the new network consisted of minimizing the error on the trainingset and reducing the error covariance with the other networks. The performance of this approach however was not satisfactory.

For further research we are interested in the following. The effectiveness of the proposed technique was only tested on bagging. It might be interesting to examine its potential when used in other combination techniques, for example as in [7] and [6]. Furthermore, reducing the covariance of the errors is not the only way to stimulate the networks in generating less correlated predictions. We are also examining ensembles of which the networks have different architectures or have been trained using different learning schemes, for example different learning rates or weight decay parameters.

References

[1] J.M. Bates and C.W.J. Granger. The combination of forecasts. *Operational Research Quarterly*, 20:451–467, 1969.

[2] J.P. Dickinson. Some statistical results in the combination of forecasts. *Operational Research Quarterly*, 24(2):253–260, 1973.

[3] H. Harrison and D.L. Rubinfeld. Hedonic prices and the demand for clean air. *J. Environ. Economics & Management*, 5:81–102, 1978.

[4] T. Heskes. Balancing between bagging and bumping. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing 9*, pages 466–472. Morgan Kaufmann, 1997.

[5] A. Krogh and J. Vedelsby. Neural networks, cross validation and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing 7*, pages 231–238. Morgan Kaufmann, 1995.

[6] M.D. Out and W.A. Kosters. A Bayesian approach to combined neural networks forecasting. In M. Verleysen, editor, *Proceedings ESANN'00*, pages 323–328. D-Facto, 2000.

[7] M.C. van Wezel, W.A. Kosters, and J.N. Kok. Maximum likelihood weights for a linear ensemble of regression neural networks. In S. Usui and T. Omori, editors, *Proceedings ICONIP'98*, pages 498–501. IOS Press, 1998.