Maximum Likelihood Weights for a Linear Ensemble of Regression Neural Networks

Michiel C. van Wezel, Walter A. Kosters and Joost. N. Kok

Email:{michiel,kosters,joost}@wi.leidenuniv.nl

Department of Computer Science, Leiden University
P.O. Box 9512, 2300 RA Leiden, The Netherlands

ABSTRACT

In this we paper study the problem of combining the outputs of the members of an ensemble of neural networks. We review the commonly used methods and thoroughly derive a cost function from a maximum likelihood perspective, that can be minimized in order to obtain maximum likelihood weights. The solution is shown to be closely related to a well known statistical method. The various combination methods are tested on several data-sets, with encouraging results.

KEYWORDS: ensembles, maximum likelihood, neural networks

1. Introduction

Recently, neural network researchers have devoted a lot of attention to neural network ensembles [4, 6, 5]. In neural network ensembles, the outputs of different neural networks trained for the same problem are used to form a combined prediction. This combination is usually linear.

Much effort has been put into determining the 'best' coefficients of the linear combination, but the currently existing methods are more or less ad-hoc and lack a probabilistic basis. It is our goal to provide a method with a theoretical basis by deriving the coefficients of the linear combination from a maximum likelihood perspective. We do this after making the notion of neural network ensembles more precise (Section 2) and reviewing commonly used methods for finding the weights in an ensemble (Section 3). After the derivation of maximum likelihood coefficients (Section 4) and explanation of the relation to the mentioned statistical method (Section 5) we give some experimental results (Section 6). We conclude the paper with some conclusions in Section 7. The symbols used in this paper are summarized in Table 1.

2. Neural Network Ensembles

As stated, a neural network ensemble is usually a linear combination of 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. 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 $\mathbf{x}_i$.

$i : 1, \ldots, N$	Index for input-target data.
$k, \ell : 1, \ldots, K$	Index for members of the ensemble.
$\mathbf{x}_i = (x_{i1}, \ldots, x_{in})^T$	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 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 $\mathbf{x}_i . e_{ik} \equiv (z_{ik} - t_i)$.
$\mathbf{w} = (w_1, \ldots, w_K)^T$	Vector of ensemble weights. Weight w_k denotes the weight of ensemble member k in the linear combination.
Q_i	Ensemble output on input vector i. $Q_i = \sum_k w_k z_{ik}$.
^eQ_i	Error of the ensemble on input vector i. $^eQ_i = \sum_k w_k z_{ik} - y_i$.
Σ	$K \times K$ covariance matrix of the errors of the ensemble members.
$\boldsymbol{\omega} = (w_1, \ldots, w_{K-1})^T$	$(K-1)$ dimensional vector with ensemble weights for the first $K-1$ ensemble members.
$\boldsymbol{\sigma} = (\Sigma_{k\ell})\ k, \ell = 1, \ldots, K-1$	Sub-matrix of Σ with last row and last column left out.
$\boldsymbol{\varsigma} = (\Sigma_{1K}, \ldots, \Sigma_{(K-1)K})^T$	$(K-1)$ dimensional vector with covariances of ensemble member K with ensemble members $1, \ldots, (K-1)$.
$\mathbf{I}$	Vector consisting of 1's of the appropriate dimensionality.

Table 1: Used symbols

3. Finding the Ensemble Weights

To optimize ensemble performance, it is important that good values for the weights $w_1, \ldots, w_K$ of the K ensemble members are selected. Two commonly used methods are:

1. Taking $w_1, \ldots, w_K$ to be equal. This way, the ensemble output is the average of the member outputs. This method is sometimes refereed to as bagging.

2. Taking $w_\ell = 1, w_k = 0, k \neq \ell$, where network ℓ has the lowest error on the training data. This way, all but one of the outputs are discarded. This method is sometimes referred to as bumping.

Despite of the obvious simplicity, these schemes often yield a significant improvement over the average performance of the ensemble members. A third method, introduced in [6], is more sophisticated. It attempts to find optimized values for the elements of $\mathbf{w}$. This is done by introducing two constraints on the weights, namely the constraint that the weights sum to one ($\sum_k w_k = 1$) and a positivity constraint on the weights ($\forall k : w_k \geq 0$), and then writing the error of the ensemble on input vector i:

$$e_{Q_i} = \sum_k w_k z_{ik} - y_i = \sum_k w_k e_{ik} \ . \qquad (1)$$

Now, the sum of squared errors of the ensemble can be written as

$$E(\mathbf{w}) = \sum_{i=1}^{N} e_{Q_i}^2 = \sum_{i,k,\ell} w_k w_l (z_{ik} - y_i)(z_{il} - y_i) \ , \qquad (2)$$

which is a quadratic objective function that has to be minimized under the above mentioned constraints. This can be done by a quadratic programming package. This method will be referred to as 'balancing', as in [5].

4. Derivation of a Maximum Likelihood Cost Function

The three methods in the previous section yield a significant improvement over the individual performance of the ensemble members. Unfortunately, a theoretical basis is lacking and it is not immediately clear 'what happens' on a probabilistic level. In this section, we show that the ensemble error e_Q has a Gaussian distribution in the case of regression problems. This distribution is used to construct a likelihood function, which in turn is used to derive maximum likelihood estimates for $\mathbf{w}$.

It is a standard assumption in regression methods where a quadratic loss function is minimized that the target values are corrupted by Gaussian noise (see e.g. [7]). That is, the error $(f - y)$, where f is the output of the regression model and y is the target value, is assumed to have a Gaussian distribution with mean 0. In the case of a neural network ensemble, the errors of the individual ensemble members e_k all have Gaussian distributions. These errors are not independent, so the proper distribution to describe the vector of errors $\mathbf{e}_i = (e_{i1}, \ldots, e_{iK})^T$ is a multivariate Gaussian with common mean 0:

$$f(e_1, \ldots, e_K) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} \exp\left\{ -\frac{\mathbf{e}^T \Sigma^{-1} \mathbf{e}}{2} \right\} \ , \qquad (3)$$

with Σ the error covariance matrix. If we write $e_Q = \sum_k w_k e_k = \mathbf{w}^T \mathbf{e}$, thereby introducing the constraint $\sum_k w_k = 1$, it is clear that e_Q is a linear sum of K dependent Gaussian variates, and it is a natural question what the distribution of e_Q is. Note that the w_k are not necessarily positive.

Lemma 1 *The ensemble error e_Q has a Gaussian distribution with mean 0 and variance $\mathbf{w}^T \Sigma \mathbf{w}$:*

$$f(e_Q) = \frac{1}{\sqrt{2\pi \mathbf{w}^T \Sigma \mathbf{w}}} \exp\left\{ -\frac{e_Q^2}{2\mathbf{w}^T \Sigma \mathbf{w}} \right\} . \qquad (4)$$

Proof We use moment generating functions for this proof. Recall that the moment generating function of a random variable, if it exists, is unique and completely determines the distribution of that random variable. The moment generating function of the multivariate Gaussian distribution of the e_k is given by

$$M(t_1, \ldots, t_K) = \mathsf{E}(e^{\mathbf{t}^T \mathbf{e}}) = \exp\left\{ \frac{\mathbf{t}^T \Sigma \mathbf{t}}{2} \right\} , \qquad (5)$$

where E denotes expectation. Obviously, since $e_Q = \mathbf{w}^T \mathbf{e}$, the moment generating function of e_Q is given by

$$M(t) = \mathsf{E}(e^{t e_Q}) = \mathsf{E}(e^{t \mathbf{w}^T \mathbf{e}}) = \exp\left\{ \frac{\mathbf{w}^T \Sigma \mathbf{w} t^2}{2} \right\} . \qquad (6)$$

This is the moment generating function of a Gaussian distributed random variable with mean 0 and variance $\mathbf{w}^T \Sigma \mathbf{w}$. □

If we assume independence between the samples in the dataset the likelihood function $\mathcal{L}(\mathbf{w})$ can be written as the product of the individual probability densities:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{\sqrt{2\pi \mathbf{w}^T \Sigma \mathbf{w}}^N} \exp\left\{ -\frac{\sum_{i=1}^{N} e_{Q_i}^2}{2\mathbf{w}^T \Sigma \mathbf{w}} \right\} . \qquad (7)$$

Since $\sum_{i=1}^{N} e_{Q_i}^2 = \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{e}_i)^2$, and, under the assumption that the ensemble members are unbiased, $\Sigma_{k\ell} = \frac{1}{N} \sum_{i=1}^{N} e_{ik} e_{i\ell}$, and thus $\mathbf{w}^T \Sigma \mathbf{w} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{e}_i)^2$, this likelihood function can be rewritten as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{\sqrt{2\pi \mathbf{w}^T \Sigma \mathbf{w}}^N} \exp\left\{ -\frac{N}{2} \right\} . \qquad (8)$$

This leads to the log likelihood function $\mathcal{LL}(\mathbf{w})$

$$\mathcal{LL}(\mathbf{w}) = -\frac{N}{2} \left\{ \log(2\pi) + \log(\mathbf{w}^T \Sigma \mathbf{w}) + 1 \right\} . \qquad (9)$$

Maximization of $\mathcal{LL}(\mathbf{w})$ is thus identical to minimization of the cost function

$$E_{\text{ml}} = \mathbf{w}^T \Sigma \mathbf{w} \ . \qquad (10)$$

which can be recognized as the variance of (4). This minimization has to be performed under the constraint $\sum_k w_k = 1$.

5. Minimization of the cost function

5.1. Minimization by the Bates-Granger Formula

The Bates-Granger formula due to Bates and Granger [1] and Dickinson [3] is

$$\mathbf{w} = (\Sigma^{-1} \mathbf{I})(\mathbf{I}^T \Sigma^{-1} \mathbf{I})^{-1} \ . \qquad (11)$$

This formula was derived by Lagrange minimization of $\mathbf{w}^T \Sigma \mathbf{w}$. However, neither in [1] nor in [3] the derivation of the probability distribution of the combined forecast or the derivation of a likelihood function was made with rigor. In fact, in [2] the formula was called ad-hoc.

There is a second problem with this method, namely that there is a high risk that the error covariance matrix is almost singular and thus is very difficult to invert. It requires the use of sophisticated techniques. In Section 6, we elaborate on this problem.

5.2. Minimization by Conjugate Gradient

If we want to use a conjugate gradient algorithm to minimize (10), the constraint $\sum_k w_k = 1$ poses a problem. To guarantee constraint satisfaction, we explicitly eliminated the $\sum_k w_k = 1$ constraint. Thus, we substitute

$$w_K = 1 - \sum_{k=1}^{K-1} w_k \ . \tag{12}$$

Hereto, we introduce some new terminology. Let $\boldsymbol{\sigma} = (\Sigma)_{k\ell}\ k, \ell = 1, \ldots, K-1$, denote the $(K-1) \times (K-1)$ sub-matrix of Σ which is identical to Σ except that the last row and the last column are left out. Let $\boldsymbol{\varsigma} = (\Sigma_{1K}, \ldots, \Sigma_{(K-1)K})^T$ be the $(K-1)$ dimensional vector denoting the covariance of ensemble member K with experts $1, \ldots, (K-1)$. Let $\boldsymbol{\omega} = (w_1, \ldots, w_{K-1})^T$ denote the vector of ensemble weights for the first $K-1$ networks in the ensemble. w_K can be computed from this vector. All these notations are also summarized in Table 1. Now, using elementary algebra, we can rewrite

$$\begin{aligned} E_{\text{new}} &= \mathbf{w}^T \Sigma \mathbf{w} \qquad (13) \\ &= \boldsymbol{\omega}^T \boldsymbol{\sigma} \boldsymbol{\omega} + 2\boldsymbol{\omega}^T \boldsymbol{\varsigma}(1 - \boldsymbol{\omega}^T \mathbf{I}) + (1 - \boldsymbol{\omega}^T \mathbf{I})^2 \Sigma_{KK} \end{aligned}$$

where $\mathbf{I}$ is a vector consisting of 1's of the appropriate dimensionality. Minimization of this E_{new} is an unconstrained problem that can be solved with fast numerical techniques.

We used the conjugate gradient code from Numerical Recipes to minimize (13). Therefore, we needed the vector of partial derivatives of (13). This vector is given by

$$\frac{1}{2}\frac{\partial E_{\text{new}}}{\partial \boldsymbol{\omega}} = \boldsymbol{\sigma}\boldsymbol{\omega} + \boldsymbol{\varsigma} - \boldsymbol{\omega}^T \boldsymbol{\varsigma} \mathbf{I} - \boldsymbol{\omega}^T \mathbf{I} \boldsymbol{\varsigma} - \Sigma_{KK}(1 - \boldsymbol{\omega}^T \mathbf{I})\mathbf{I} \tag{14}$$

Results of the minimization by conjugate gradient are given in Section 6.

6. Experiments and Results

In this section we describe performed experiments and the obtained results for three different datasets. All methods described in this paper were implemented, i.e., bagging, bumping, balancing, maximum likelihood weights with conjugate gradient (cjg), and the Bates-Granger formula. The datasets used are:

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 share price dataset: This dataset concerns predicting the price of 'Heineken' shares on the Amsterdam stock exchange. The value of the share is to be modelled on the basis of the values of nine other shares (the so-called 'leaders') in the five preceding days. These shares are Nedlloyd, Nutricia, Philips, Unilever, Gist Brocades, ING, AKZO, DSM and Elsevier. We used 1301 patterns from the period 1992–1996, of which 1000 (randomly selected) were used for training and the rest for testing. Each pattern has 45 inputs and one output.

The water level dataset: This dataset concerns predicting the water level during storm situations in the Netherlands. Predicting the correct water level is important for access regulation to harbors. The input variables are water levels, air pressure and wind speed at various locations along the Dutch shore, at a number of time steps before the point in time at which the water level has to be predicted. The total number of input variables is 22. There are 350 patterns in the dataset, of which 70 were used for testing and 280 for training.

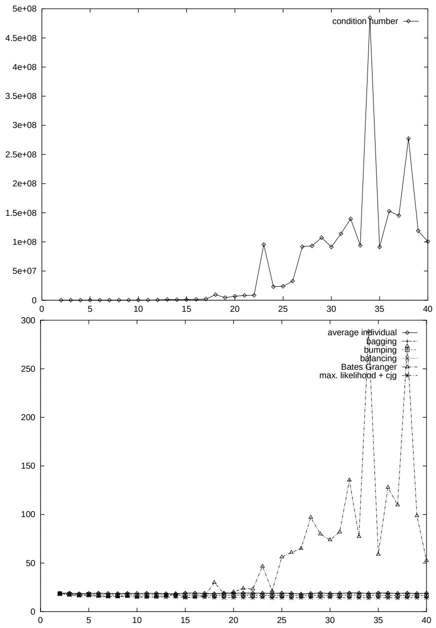

Figure 1: Condition numbers (upper graph) and test set error (rmse) for the share price dataset.

Table 2 shows the average errors over 20 runs yielded by all weighting methods in the case of a 25 network ensemble. In Figure 2 it is shown how the root mean squared error (rmse) develops for increasing ensemble sizes for all weighting methods. The graphs show the average results over 20 runs for all methods except the Bates-Granger formula. The maximum likelihood weighting yields the lowest error on the test data and the training data for all datasets. The performance gain of the maximum likelihood methods over bagging, bumping and balancing is significant.

	marketing		share price		water level	
	test	train	test	train	test	train
avg. indiv. networks	4.09	3.86	19.07	18.81	15.12	12.29
bagging	4.03	3.80	18.85	18.60	15.01	12.14
bumping	3.37	3.22	16.74	16.50	14.33	11.49
balancing	3.37	3.22	16.89	16.67	14.27	11.49
Bates-Granger	29.5	28.8	56.17	55.56	161.0	160.1
max. likelihood + cjg	2.97	2.53	14.50	14.13	12.72	10.48

Table 2: The rmse-values yielded by various weighting methods averaged over 20 runs. The trained ensembles all contained 25 networks.

The Bates-Granger formula is not included in the graphs because it suffers from instability if the ensembles contain more than 14 networks. This can be explained as follows. The Bates-Granger formula requires the inversion of Σ. This matrix can become near singular. This happens if two networks have almost linearly dependent errors. Of course, the chance of this happening increases with the ensemble size. Near singular matrices are very hard to invert, and thus lead to poor results for the Bates-Granger formula. These statements are supported by plotting the condition numbers of the error covariance matrices. A large condition number indicates that a matrix is close to singular. A near singular matrix is called ill conditioned. The condition numbers for the share price dataset and the corresponding test set errors are plotted in Figure 1, from which one can see that the Bates-Granger formula tends to fail when Σ is ill conditioned.

7. Summary and Conclusions

We have summarized several methods for finding coefficients of a linear ensemble of neural networks. We have derived a cost function for these coefficients from a maximum likelihood perspective and considered two ways of minimizing this cost function. We tested all described methods on various datasets. The maximum likelihood methods gave the best results for all problems.

References

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

[2] R.F. Bordley. The combination of forecasts: a bayesian approach. *Journal of the operational research society*, 33(2):171–174, 1982.

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

[4] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

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

[6] 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.

[7] D.J.C. MacKay. Bayesian methods for neural networks: Theory and applications. In *Course notes of neural network summer school*, Cambridge, 1995.

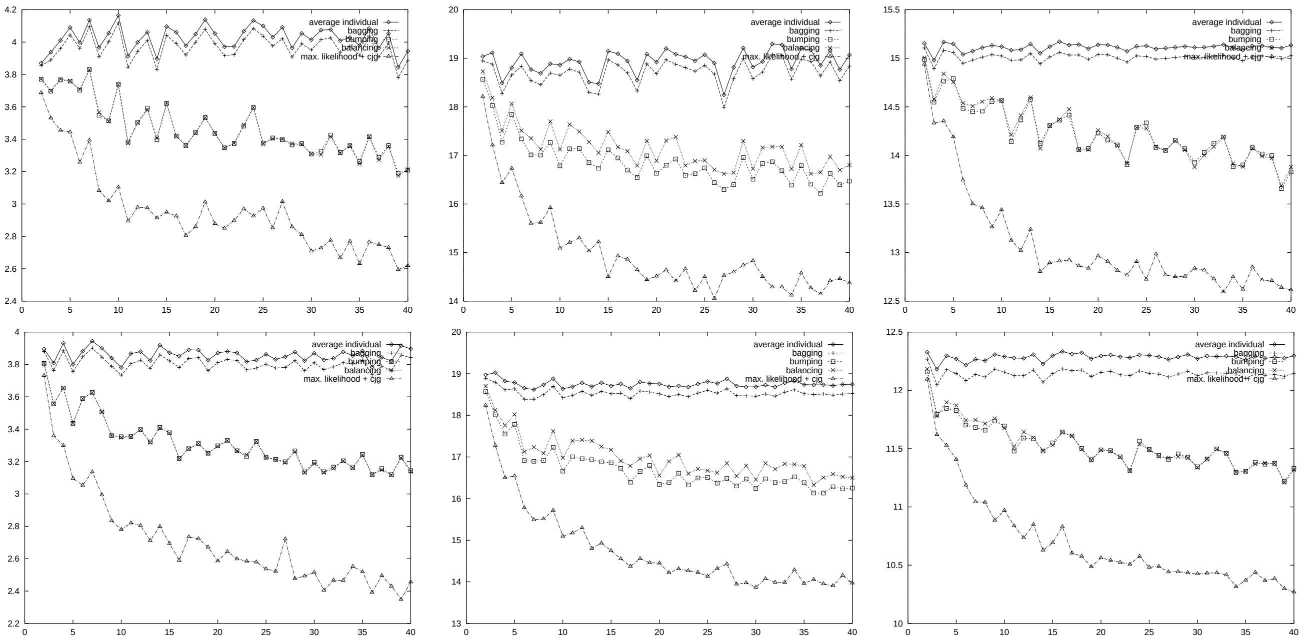

Figure 2: Graphs showing how the error (rmse) on test data (upper graphs) and training data (lower graphs) develops for increasing ensemble sizes. Left: marketing dataset, middle: share price dataset, right: water level dataset.