Network: Comput. Neural Syst. **8** (1997) 283–296. Printed in the UK PII: S0954-898X(97)78637-8

Optimal ensemble averaging of neural networks

Ury Naftaly†, Nathan Intrator‡ and David Horn§

Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel

Received 7 October 1996

Abstract. Based on an observation about the different effect of ensemble averaging on the bias and variance portions of the prediction error, we discuss training methodologies for ensembles of networks. We demonstrate the effect of variance reduction and present a method of extrapolation to the limit of an infinite ensemble. A significant reduction of variance is obtained by averaging just over initial conditions of the neural networks, without varying architectures or training sets. The minimum of the ensemble prediction error is reached later than that of a single network. In the vicinity of the minimum, the ensemble prediction error appears to be flatter than that of the single network, thus simplifying optimal stopping decision. The results are demonstrated on sunspots data, where the predictions are among the best obtained, and on the 1993 energy prediction competition data set B.

1. Introduction

In recent years, the use of artificial neural networks (NN) for time series prediction has gained popularity and nowadays NN can compete with the best time series methods [1]. In this paper we re-examine one of the major techniques for NN performance improvement—ensemble averaging [2, 3]. We argue that it requires a special training methodology, and can be more effective when *not* combined with popular training constraints such as weight decay and early stopping‖.

The theoretical setting of the method is provided by the bias/variance decomposition. Within this framework, we will define a particular bias/variance decomposition for networks differing by their initial conditions only. This is a particularly useful subset of the general set of all sources of variance. We show that while the bias of the ensemble of networks with different initial conditions remains unchanged, the variance error decreases considerably. The theoretical background is presented in the next section. We then describe the sunspots data, on which our technique is demonstrated. This includes a simple method of extrapolation to the infinite ensemble. We show that the minimal prediction error of the ensemble is reached *later* in training than that of single networks, and the ensemble error curve appears to be flatter in the vicinity of the minimum error. Results for the sunspots problem are presented in section 5, where we also evaluate the combination of ensemble averaging with other popular techniques. Our results outperform the best published results [5].

† School of Physics and Astronomy. E-mail: ury@tarazan.tau.ac.il.
‡ School of Mathematical Sciences. E-mail: nin@math.tau.ac.il.
§ School of Physics and Astronomy. E-mail: horn@vm.tau.ac.il.

‖ Under a different setup (training with input noise) weight decay was found useful in conjunction with ensemble averaging [4].

0954-898X/97/030283+14$19.50

Our method is further evaluated with a data set from the 1993 energy competition (section 6). The extrapolation method works somewhat differently in this case, signifying the existence of correlations among networks with different initial conditions. The qualitative behaviour of the error minima are the same as those of the sunspots analysis.

2. The bias/variance decomposition and ensemble averaging

The motivation of our approach follows from a key observation regarding the bias/variance decomposition, namely the fact that ensemble averaging does not affect the bias portion of the error, but reduces the variance, when the estimators on which averaging is done are independent.

The classification problem is to estimate a function $f_{\mathcal{D}}(x)$ of observed data characteristics x, predicting class label y, based on a given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_L, y_L)\}$, using some measure of the estimation error on $\mathcal{D}$. A good estimator will perform well not only on the training set, but also on new *validation* sets which were not used during estimation.

Evaluation of the performance of the estimator is commonly done via the mean squared error (MSE) distance by taking the expectation with respect to the (unknown) probability distribution P of y:

$$E[(y - f_{\mathcal{D}}(x))^2 | x, \mathcal{D}].$$

This can be decomposed into

$$E[(y - f_{\mathcal{D}}(x))^2 | x, \mathcal{D}] = E[(y - E[y|x])^2 | x, \mathcal{D}] + E[(f_{\mathcal{D}}(x) - E[y|x])^2].$$

The first term does not depend on the training data $\mathcal{D}$ or on the estimator $f_{\mathcal{D}}(x)$; it measures the amount of noise or variability of y given x. Hence f can be evaluated using

$$E[(f_{\mathcal{D}}(x) - E[y|x])^2].$$

The empirical mean squared error of f is given by

$$E_{\mathcal{D}}[(f_{\mathcal{D}}(x) - E[y|x])^2]$$

where $E_{\mathcal{D}}$ represents expectation with respect to all possible training sets $\mathcal{D}$ of fixed size.

To further investigate the performance under MSE, we decompose the error to bias and variance components [6] to get

$$E_{\mathcal{D}}[(f_{\mathcal{D}}(x) - E[y|x])^2] = (E_{\mathcal{D}}[f_{\mathcal{D}}(x)] - E[y|x])^2 + E_{\mathcal{D}}[(f_{\mathcal{D}}(x) - E_{\mathcal{D}}[f_{\mathcal{D}}(x)])^2]. \quad (1)$$

The first term on the RHS is called the bias of the estimator and the second term is called the variance. When training on a fixed training set $\mathcal{D}$, reducing the bias with respect to this set may increase the variance of the estimator and contribute to poor generalization performance. This is known as the trade-off between variance and bias. Typically variance is reduced by smoothing; however, this may introduce bias (since, for example, it may blur sharp peaks). Bias is reduced by prior knowledge. When prior knowledge is used also for smoothing, it is likely to reduce the overall MSE of the estimator.

When training neural networks, the variance arises from two terms. The first term comes from inherent data randomness and the second term comes from the non-identifiability of the model, namely, the fact that for a given training data, there may be several (local) minima of the error surface†.

† An example of an identifiable model is (logistic) regression.

Consider the ensemble average $\bar{f}$ of Q predictors, which in our case can be thought of as neural networks with different random initial weights which are trained on a fixed training set:

$$\bar{f}(x) = \frac{1}{Q}\sum_{i=1}^{N} f_i(x).$$

These predictors are identically distributed and thus, the variance contribution (second term on the RHS of equation (1)) becomes (we omit x and $\mathcal{D}$ for simplicity)

$$\begin{aligned}\mathrm{Var}(\bar{f}) = E[(\bar{f} - E[\bar{f}])^2] &= E\left[\left(\frac{1}{Q}\sum f_i - E\left[\frac{1}{Q}\sum f_i\right]\right)^2\right] \\ &= E\left[\left(\frac{1}{Q}\sum f_i\right)^2\right] - \left(E\left[\frac{1}{Q}\sum f_i\right]\right)^2. \qquad (2)\end{aligned}$$

The first term on the RHS can be rewritten as

$$E\left[\left(\frac{1}{Q}\sum f_i\right)^2\right] = \frac{1}{Q^2}\sum E[f_i^2] + \frac{2}{Q^2}\sum_{i<j} E[f_i f_j]$$

and the second term gives

$$\left(E\left[\frac{1}{Q}\sum f_i\right]\right)^2 = \frac{1}{Q^2}\sum E[f_i] + \frac{2}{Q^2}\sum_{i<j} E[f_i]E[f_j].$$

Substituting these equalities in equation (2) gives

$$E[(\bar{f} - E[\bar{f}])^2] = \frac{1}{Q^2}\sum\{E[f_i^2] - (E[f_i])^2\} + \frac{2}{Q^2}\sum_{i<j}\{E[f_i f_j] - E[f_i]E[f_j]\}. \qquad (3)$$

It follows that

$$\frac{1}{Q}\mathrm{Var}(f_i) \leqslant \mathrm{Var}(\bar{f}) \leqslant \frac{\mathrm{Var}(f_i) + \max_{i,j}(E[f_i f_j] - E[f_i]E[f_j])}{Q} \leqslant \max_i \mathrm{Var}(f_i). \qquad (4)$$

More specifically, when replacing $f(x)$ by $\bar{f}(x)$, the reduction in the variance portion of the error is proportional to the degree of independence between the predictors in the ensemble. Due to random initial conditions only, a certain level of independence may be achieved. The independence can be increased by various ways such as different architectures, input noise injection, and different training times. Thus the variance portion of the error for ensemble has the form

$$E_{\mathcal{D}}[(\bar{f}_d - E_{\mathcal{D}}[\bar{f}_d])^2] \simeq \frac{E_{\mathcal{D}}[(f_d - E_{\mathcal{D}}[f_d])^2] + \gamma}{Q}$$

where $\gamma \geqslant 0$ is given by

$$\gamma = E[f_i f_j] - E[f_i]E[f_j] = E\Big(\{f_i - E[f_i]\}\{f_j - E[f_j]\}\Big).$$

Thus, the notion of independence can be understood as independence of the deviations of each predictor from the expected value of the predictor, which can be replaced (due to linearity) by

$$E\Big(\{f_i - E[\bar{f}]\}\{f_j - E[\bar{f}]\}\Big)$$

and is thus interpreted as an independence of the prediction variation around a common mean.

We wish to find the optimal training procedure for reducing the error of our ensemble average. Traditional training algorithms aim at reducing the error of the individual NN, i.e. the total expression of equation (1), including both bias and variance. Typically the bias decreases and the variance increases as one employs more and more training epochs, and one aims to stop when their sum reaches a minimum. Since in our algorithm the predictor is defined by an ensemble average, we have to search for a different minimum, as we are able to eliminate some portion of the variance of the estimator via ensemble averaging. We should, therefore, search for a point with a smaller bias (longer training time) as the optimal trade-off for ensemble predictor.

3. The sunspots problem

Yearly sunspot statistics have been gathered since 1700. The data are plotted in figure 1. These data have been extensively studied and have served as a benchmark in the statistical literature [7–9].

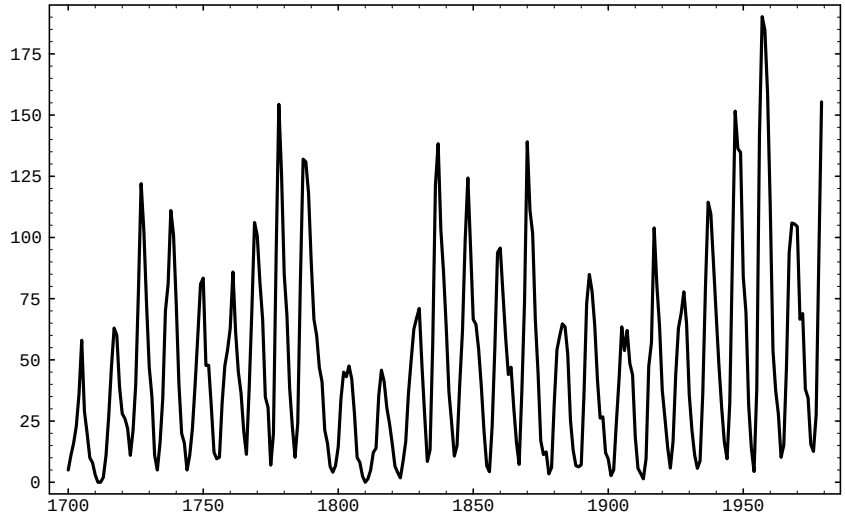

Figure 1. Sunspots data: average sunspot activity from 1700 to 1979.

Following previous publications [5, 8, 10], we choose the training set to contain the period between 1701 and 1920, and the test set to contain the years 1921 to 1955. Following [8], we calculate the prediction error according to the average relative variance (ARV)

$$\mathrm{ARV} = \frac{\sum_{k\in S}(y_k - f(\boldsymbol{x}_k))^2}{\sum_{k\in S}(y_k - E[y_k])^2} \tag{5}$$

which is the MSE divided by the variance of the data set S. The denominator is $\sigma^2 = 1535$ for the training set. The same value is used for the test set.

3.1. Previous results

In a survey of sunspots prediction models [11], the threshold autoregressive (TAR) model of Tong and Lim [12, 13] was favoured. The TAR model is a combination of two linear autoregressive models with an activity threshold above which one autoregression model is

used and below which the other is used [14]. The TAR estimator gave a training ARV of 0.097 and the same result for the prediction set.

Weigend *et al* [8] used a standard multilayer perceptron architecture with 12 input units, 8 sigmoidal hidden units and a linear output unit. Possible over-fitting was addressed by the use of weight decay [15]. Their best result was an ARV of 0.082 on the training set and 0.086 on the prediction set.

Nowlan and Hinton [5] imposed a mixture-of-Gaussians prior on the weights which they called 'soft weight sharing' to get an ARV of 0.072 on the test set. Pi and Peterson [10] introduced the δ-test which establishes the dependence of a sequence of numbers on previous element(s) of the sequence. They found that x_{t-1}, x_{t-2}, x_{t-3}, x_{t-4}, x_{t-9} and x_{t-10} are the most important variables in the sunspots series. Thus, the functional form $y = f(x_{t-1}, x_{t-2}, x_{t-3}, x_{t-4}, x_{t-9}, x_{t-10})$ is expected to lead to good prediction results. These lags were used as inputs to a (6,8,1) network. The ARV obtained on the test set was 0.073.

4. Ensemble averaging over initial conditions

In this section we will demonstrate the method we use for ensemble averaging over initial conditions by applying it to the sunspots problem. We use neural networks with 12 inputs (as in [8]). All nets have one sigmoidal hidden layer consisting of 4 units and a linear output. They are then enlarged to form simple recurrent networks (SRN) [16] in which the input layer is increased by adding to it the hidden layer of the previous point in the time series. This favours ordered temporal application.

The learning algorithm consists of back propagation applied to an error function which is the MSE of the training set. A learning rate of 0.003 is employed. A validation set containing 35 randomly chosen points was left out during training to serve for performance validation.

The training procedure of an NN starts out with some choice of initial values of the connection weights. We consider then an ensemble of networks that differ from one another just by these initial values. This defines the ensemble that we wish to average over. Since the space of initial conditions is very large, we develop a technique which allows us to approximate averaging over the whole space.

Our technique consists of constructing groups of a fixed number of networks, Q. Choosing several different groups of the same size Q, and averaging over them, we define a finite size average. Then we try to estimate the limit $Q \to \infty$. If we regard the specific choice of initial conditions to be equivalent to some random error added to the predictor, we may expect this error to decrease as $1/Q$. It turns out that performing a simple regression in $1/Q$ indeed suffices to obtain this limit.

Figure 2 displays our results on the validation set and figure 3 shows the results on the test set. In both figures we show ARV values as function of the number of training epochs. The highest curves in both figures correspond to $Q = 1$, i.e. the case of single networks. Below it appear the curves of $Q = 2, 4, 10, 20$ followed by the extrapolation to $Q \to \infty$. To demonstrate how the extrapolation is carried out we display in figure 4 the points obtained for $t = 70$ kE (kilo epochs) and $t = 140$ kE for the test set as a function of $1/Q$. It is quite clear that a linear extrapolation is very satisfactory. Moreover, the results for $Q = 20$ are not far from the extrapolated $Q \to \infty$ results.

In this example (figure 3) the ARV is quite flat as a function of t, suggesting that most of the variance was eliminated by the averaging process. The variance that is due to initial conditions can be found by subtracting the $Q \to \infty$ result from $Q = 1$ (figure 5). Although

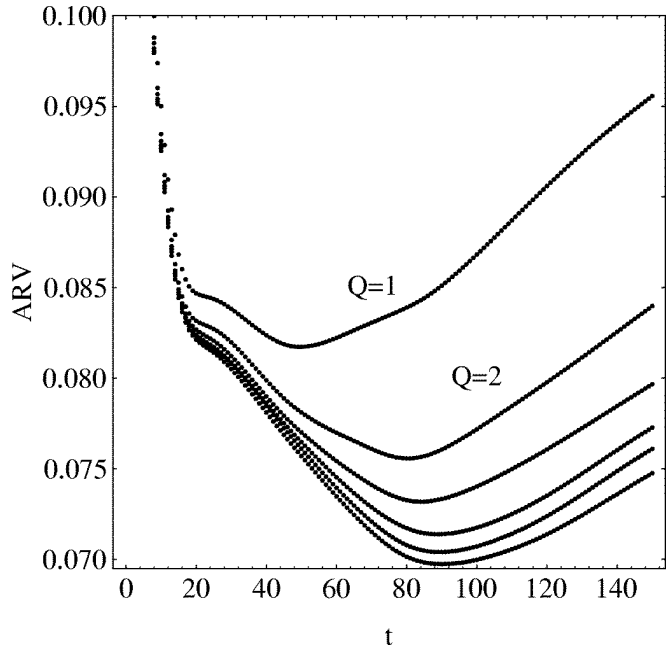

Figure 2. ARV of cross-validation sets. ARV is plotted versus training time in kilo epochs (kE). These are results for the cross-validation set of 35 points in the sunspots problem. The curves are shown for different choices of group sizes: $Q = 1, 2, 4, 10, 20$ from top to bottom. The lowest curve is the extrapolation to $Q \to \infty$.

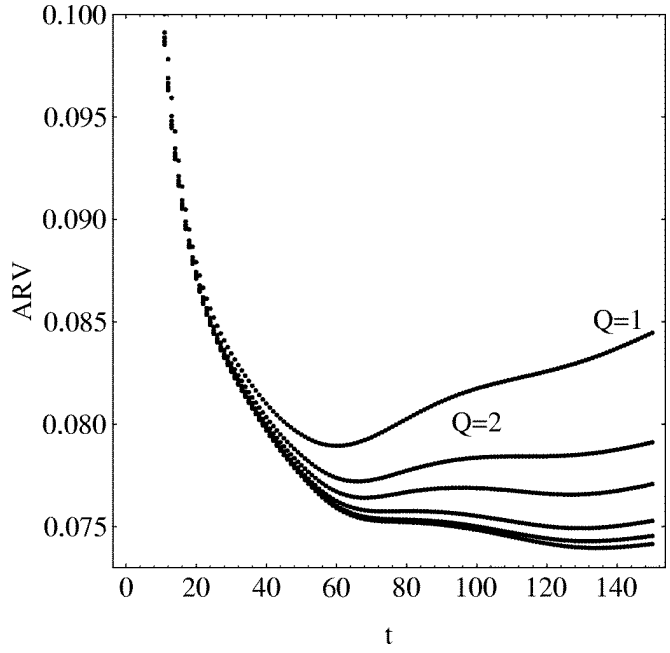

Figure 3. ARV of test set. Results for the test set show a shallow $Q \to \infty$ curve, and the two minima. The set-up is the same as in figure 2.

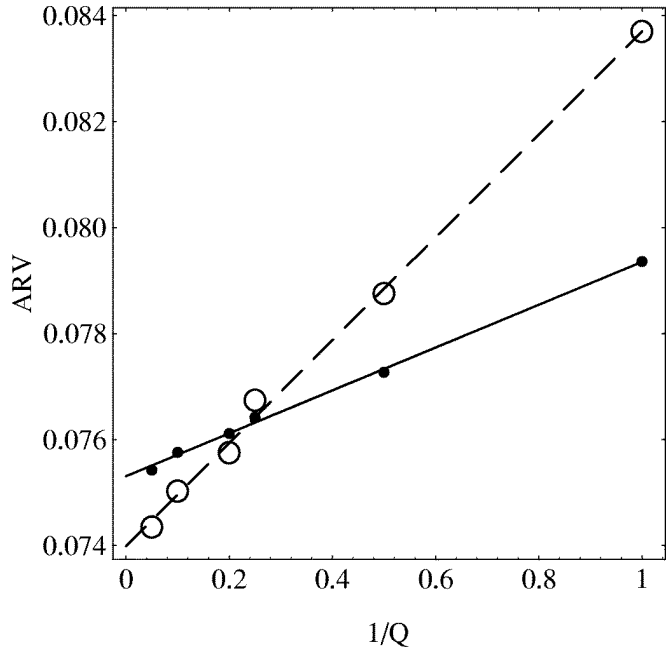

Figure 4. Extrapolation method used for extracting the $Q \to \infty$ prediction. The results for different Q at two different training periods, $t = 70$ kE (top) and 140 kE (bottom), can be extrapolated by a linear regression in $1/Q$.

this variance is due to initial conditions and not to different training sets, it is increasing in time (as would be expected from the variance due to training sets).

Some interesting conclusions follow from this numerical study:

(i) The true minimum of the final predictor is obtained at larger t values than that of the $Q = 1$ curve. Note that if we were to use canonical techniques of single network training, we would have stopped around $t = 50$ kE, whereas the validation set shows minimal ARV for $Q = 1$. This is, however, very different from where the $Q \to \infty$ curve goes through minimum, both in the cross-validation set and in the test set. This property of ensemble training follows from the fact that one portion of the variance contribution to the error is removed or reduced by the ensemble averaging, and thus less smoothing (bias) is needed for regularizing the predictor.

(ii) The error function of the final predictor is shallower than that of the single network, which is obviously due to the reduction in variance.

The final $Q \to \infty$ error function is quite flat, and therefore ARV differences between different stopping points are not large. Nonetheless, for accurate estimates one should beware of the fact that the stopping criterion for the $Q = 1$ problem is very different from the $Q \to \infty$ one. The latter is the one which is relevant to our problem.

The curves shown in figures 2–5 were obtained with a learning rate of 0.003. We will see in the next section that better results can be obtained for slower learning rates, as well as for other choices of architecture. When lower errors are obtained, the variation between the single networks and the ensemble is less dramatic. Yet in all cases we find the general characteristics of variance reduction and shift in the temporal structure of the error functions.

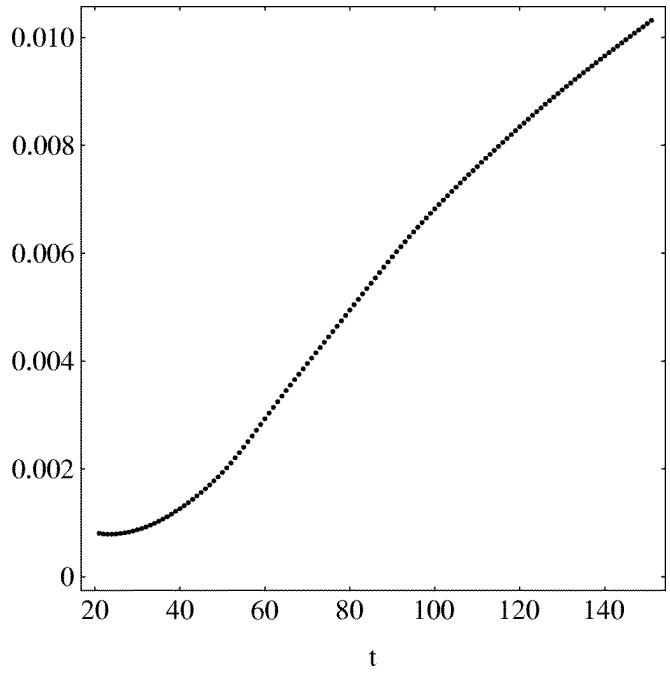

Figure 5. Variance error due to initial conditions is estimated by the difference of the ARV values for $Q = 1$ and for $Q \to \infty$.

5. Analysis of the sunspots problem

From the curves of figures 2 and 3, we may read the performance values of the predictors which can be defined through different stopping criteria. Using $Q = 20$ data and the conventional stopping criterion, i.e. the minimum of the $Q = 1$ curve for the cross-validation set of data at $t = 49$ kE, we are led to ARV=0.0796 on the test set. Stopping at $t = 90$ kE, which is the minimum point of $Q = 20$ on cross-validation data, leads to ARV $= 0.0752$ on the test set. The absolute minimum of the $Q = 20$ curve on the test set, ARV $= 0.075$, is reached at $t = 132$ kE.

Table 1. The minimal ARV on the sunspots test set as a function of the learning rate for simple networks.

Learning rate	ARV
0.010	0.0868
0.003	0.0731
0.002	0.0720
0.001	0.0713

We repeat the whole exercise for different learning rates and find that the results are very sensitive to this parameter. This is demonstrated in table 1 which is based on $Q = 20$ groups of networks in each case. Clearly lower learning rates lead to better results. Unfortunately, they require an immense number of training epochs and are very costly.

5.1. Recurrent networks

Recurrent neural networks have been successful as time-series predictors. Simple recurrent networks (SRN) [16] are a practical compromise between fully recurrent dynamics and computational overload, and are being widely used [17].

In such networks, the connections are mainly feedforward, but include some feedback connections. The recurrency lets the network remember cues from the recent past, but does not appreciably complicate the training. We have already presented in the previous section the results of a simple recurrent network [18]. Here we compare it to a simple feedforward network (table 2). We see that recurrency helps for lowering the ARV on both ensemble and single nets, but its influence on single nets is about four times stronger than on the ensemble. We note that the improvement of the SRN ensemble is smaller than that of the feedforward network. Since the reduction of errors is due to variance elimination, this means that SRN networks are less independent than feedforward nets. This sounds plausible because of the extra dependence of an SRN on the previous data point.

Table 2. The predicted ARV of recurrent and non-recurrent networks trained with the same learning and prediction conditions, using a learning rate of 0.001.

	SRN	Feedforward
Ensemble	0.0706	0.0713
Single	0.0739	0.0764

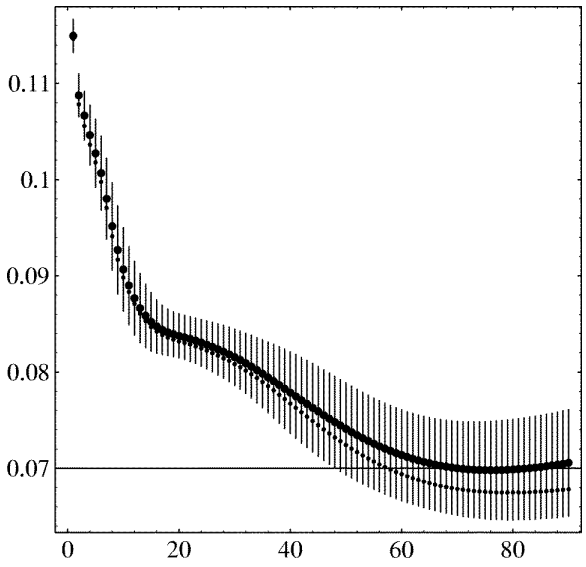

Figure 6. Our best results for the test set of the sunspots problem. Plotted here are $Q = 1$ results for various choices of initial conditions, represented by their averages with error bars extending over a standard deviation, and $Q = 20$ results (the smaller points), as a function of training time in kilo epochs. The network is based on the Pi and Peterson variables, and the learning rate is 0.0005.

5.2. Selected inputs

Pi and Peterson [10] report very good results using a (6,8,1) network where the input vectors consist of the variables x_{t-1}, x_{t-2}, x_{t-4}, x_{t-9} and x_{t-10}. Applying the same inputs to our paradigm, indeed improved the prediction results of the ensemble (table 3). The result of 0.0674 is better than any previously reported result.

Table 3. The predicted ARV of recurrent networks trained with full input vectors (12 inputs) and of δ chosen lags (6 inputs) using a learning rate of 0.0005.

	12 inputs	6 inputs
Ensemble	0.0700	0.0674
Single	0.0730	0.0698

Note that we employed here a very low learning rate which, together with the choice of the Pi and Peterson variables, led to this remarkable result. We have already remarked before that in such a case the variance is no longer as strong as in the examples studied above, where a learning rate that is six times faster was employed. For completeness we present in figure 6 the $Q = 1$ and $Q = 20$ curves for this case. Error reduction due to ensemble averaging is small; nonetheless it is significant.

5.3. Weight decay

It is well known [8, 17] that a large network cannot generalize well. Therefore, it is advisable to use the smallest network that is able to fit the training data. Many algorithms were used to achieve this goal [5, 8, 17]. Here we use weight decay [8] which lets the network itself decrease non-useful connections during training. The cost function is

$$\mathrm{MSE} + \frac{1}{2}\lambda \sum_i \frac{w_i^2}{1 + w_i^2} \tag{6}$$

where the sum is over the network weights, and λ is the smoothing parameter to be adjusted.

The authors of [8] use this cost function to predict the sunspots series. We repeat their experiment using a $Q = 20$ ensemble of (12,8,1) networks, and obtain the results presented in table 4. The main effect of the weight decay algorithm is to reduce the ARV of the single networks, while it has almost no effect on the ensemble ARV.

Table 4. The predicted ARV of networks with and without weight-decay trained with the same learning and prediction conditions.

	With decay	Without decay
Ensemble	0.08138	0.08142
Single	0.08331	0.08578

A simple weight decay which adds a penalty of the form $\lambda \sum_i w_i^2 f(w_0^2 + w_i^2)$ to the energy function, gave no significant improvement on the ensemble prediction, although it did improve the results of single networks.

6. Analysis of energy competition data

Here we use another data set to demonstrate our findings: data set B of the 1993 energy competition, which consists of 3344 measurements of four input variables at hourly intervals during daylight (see figure 7). The physical source of the data were measurements of solar flux from five outdoor devices. Four of the devices were fixed. The fifth, whose output was to be predicted, was driven by motors so that it pointed at the sun. For more information and results see [19].

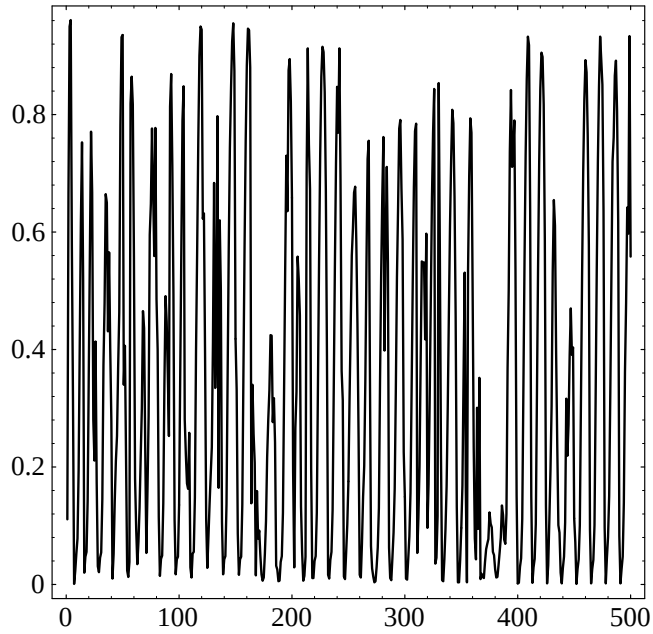

Figure 7. 500 elements of the 1993 energy competition data set B.

Our training set consists of 1000 vectors containing the four variables, the decimal date and the five last values of the target. Test and cross-validation sets that contain 500 vectors each are formed in the same manner. A similar methodology to the one used for the sunspots data was applied to these data: we use 60 simple (non-recurrent) feedforward networks of 10 inputs, one sigmoidal hidden layer of 8 hidden units and a linear output unit. Setting the learning rate to 0.02, we get the MSE of ensembles consisting of 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 and 60 nets, depicted in figure 8. Again we see the same effects that were observed on the sunspots data set: (i) a shift of the minimum, and (ii) a shallower curve of the ensemble MSE.

There is one notable difference between this analysis and the former one on sunspots: we no longer obtain a simple $1/Q$ dependence. This is demonstrated in figure 9 where we plot the behaviour of the MSE as a function of $1/Q$. Since a linear regression does not capture the trend of the data, we have employed a power law dependence:

$$\mathrm{MSE} = aQ^{-k} + b. \tag{7}$$

The results of the fit indicate that $k = 0.81$. This serves as the basis for producing the dots in figure 8 that represent the extrapolation to infinite Q. The fact that $k < 1$ indicates that there exists a correlation between networks with different initial conditions.

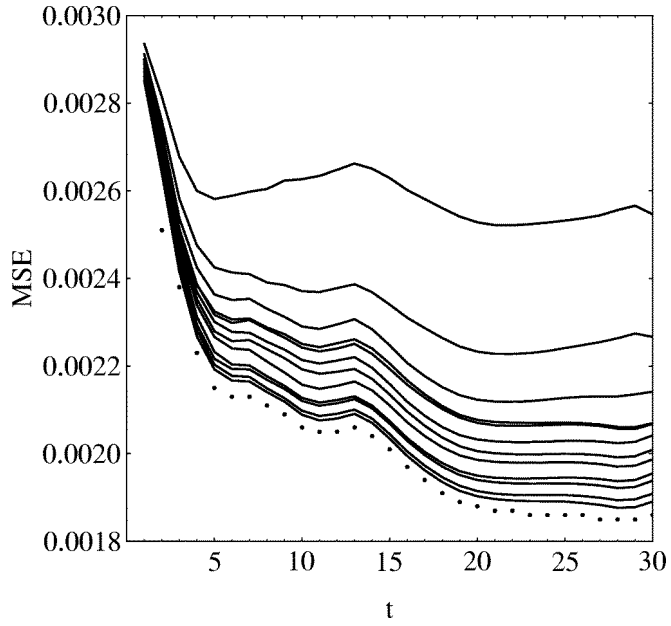

Figure 8. Prediction for the 1993 energy competition data set B. Results for a test set of 500 vectors. The curves are shown for the different choices of ensemble size: $Q = 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30$ and 60 from top to bottom. The dots are the extrapolation to $Q \to \infty$.

7. Conclusions

We have shown that ensemble averaging is a powerful procedure which, when used correctly, improves on single network performance. Ensemble averaging is not an alternative to methods for introducing bias and reducing variance, such as smoothing or early stopping, as it does not eliminate variance that is due to training on a limited training set. When the portion of the variance due to initial conditions is large, ensemble averaging is most effective. By using large ensembles we may eliminate it altogether.

As demonstrated on the sunspots data set, we were able to perform a simple extrapolation to infinite ensemble size. The variance turned out to be inversely proportional to the size, demonstrating that, in this case, some independence between predictors can be achieved by varying the initial conditions.

Our experience suggests that when training for optimal ensemble performance, the training method and stopping criteria have to be chosen carefully. This is because stopping criteria based on single network training may not be useful for the ensemble. In fact, ensemble results are improved if single nets are over-trained. Instead of early stopping or even stopping when the validation error reaches a minimum, further training has to be done, so that the bias portion of the error will be further reduced, while paying the price of higher variance for individual networks. Later, the variance portion of the error is reduced by the ensemble average with no effect on the bias. Finally, the reduction of variance, that is inherent in this method, leads to a flattening of the error curve as a function of training time.

We have applied our method both to the sunspots data and to the energy competition data. For both we have obtained a considerable decrease of variance by averaging over

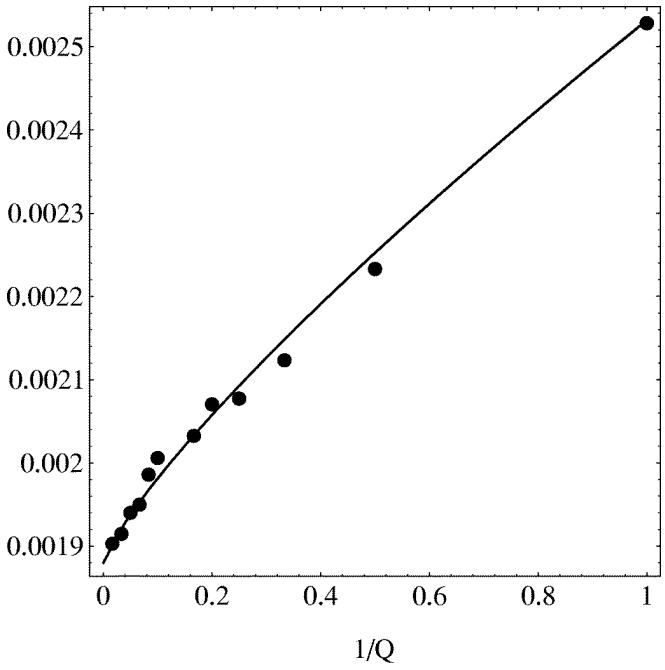

Figure 9. Extrapolation method used in the energy competition data set. Shown here are results for different Q at $t = 20$ kE. The solid line represents the best fit to a power law, described in the text, which leads to a decreasing component of $Q^{-0.81}$. This indicates that there exist correlations between networks having different initial conditions.

the space of initial conditions. In the energy competition case we found the Q dependence to be more complicated, signifying non-trivial correlations between the different networks. Some decorrelation methods could turn out to be useful here. In the sunspots case, the Q dependence was as expected from statistical independence. Using the δ-test of Pi and Peterson [10] together with our method, we obtained the best predictions yet for this problem.

References

[1] Weigend A S and Gershenfeld N A (ed) 1994 *Time Series Prediction* (Reading, MA: Addison-Wesley)
[2] Lincoln W P and Skrzypek J 1990 Synergy of clustering multiple back propagation networks *Advances in Neural Information Processing Systems 2* ed D S Touretzky (San Mateo, CA: Morgan Kaufmann) pp 650–7
[3] Wolpert D H 1992 Stacked generalization *Neural Networks* **5** 241–59
[4] Raviv Y and Intrator N 1996. Bootstrapping with noise: an effective regularization technique *Connection Sci.* Special issue on combining estimators, in press
[5] Nowlan S J & Hinton G E 1992 Simplifying neural networks by soft weight-sharing Neural Comput. **4** 473–93
[6] Geman S, Bienenstock E and Doursat R 1992 Neural networks and the bias/variance dilemma *Neural Comput.* **4** 1–58
[7] Priestley M B 1981 *Spectral Analysis and Time Series* (New York: Academic Press)
[8] Weigend A S, Huberman B A and Rumelhart D 1990 Predicting the future: a connectionist approach *Int. J. Neural Syst.* **1** 193–209
[9] Morris J (1977) Forecasting the sunspot cycle *J. R. Stat. Soc.* ser. A **140** 437–47
[10] Pi H and Peterson C 1994 Finding the embedding dimension and variable dependencies in time series *Neural Comput.* **6** 509–20
[11] Priestley M B 1988 *Non-linear and Non-stationary Time Series Analysis* (New York: Academic Press)
[12] Tong H and Lim K S 1980 Threshold autoregression, limit cycles and cyclical data *J. R. Stat. Soc.* **42** 245

[13] Tong H 1983 *Threshold Models in Non-linear Time Series Analysis (Lecture Notes in Statistics 21)* (Berlin: Springer)
[14] Tong H 1990 *Non-linear Time Series: A Dynamical Systems Approach* (Oxford: Oxford University Press)
[15] Hinton G E 1986 Learning distributed representations of concepts *Proc. 8th Ann. Conf. of the Cognitive Science Society* (Hillsdale, NJ: Erlbaum) pp 1–12
[16] Elman J L and Zipser D 1988 Learning the hidden structure of speech *J. Acoust. Soc. Am.* **83** 1615–26
[17] Hertz J, Krogh A and Palmer R G 1991 *Introduction to the Theory of Neural Computation (Lecture Notes of the Santa Fe Institute, vol 1)* (Reading, MA: Addison-Wesley)
[18] Elman J L 1990 Finding structure in time *Cognitive Sci.* **14** 179–211
[19] MacKay D J C 1994 Bayesian non-linear modeling for the prediction competition *ASHRAE Trans.* **100** part 2, pp 1053–62