Neural Network Ensembles and Their Application to Traffic Flow Prediction in Telecommunications Networks

Xin Yao[†], Manfred Fischer[‡] and Gavin Brown[†]

[†]School of Computer Science, the University of Birmingham
Edgbaston, Birmingham B15 2TT, U.K.
Email: {x.yao,g.brown}@cs.bham.ac.uk
[‡]Department of Economic Geography and Geoinformatics
Vienna University of Economics and Business Administration
Rossauer Lände 23/1, A-1090 Vienna, Austria
Email: Manfred.Fischer@wu-wien.ac.at

Abstract

It is well-known that large neural networks with many unshared weights can be very difficult to train. A neural network ensemble consisting of a number of individual neural networks usually performs better than a complex monolithic neural network. One of the motivations behind neural network ensembles is the divide-and-conquer strategy, where a complex problem is decomposed into different components each of which is tackled by an individual neural network. A promising algorithm for training neural network ensembles is the negative correlation learning algorithm which penalizes positive correlations among individual networks by introducing a penalty term in the error function. A penalty coefficient is used to balance the minimization of the error and the minimization of the correlation. It is often very difficult to select an optimal penalty coefficient for a given problem because as yet there is no systematic method available for setting the parameter. This paper first applies negative correlation learning to the traffic flow prediction problem, and then proposes an evolutionary approach to deciding the penalty coefficient automatically in negative correlation learning. Experimental results on the traffic flow prediction problem will be presented.

1 Introduction

Neural network ensembles offer a number of advantages over a single neural network system. They have the potential for improved generalization, lower dependence on the training set, and reduced training time. Sharkey [6] provides a good summary of the literature, and states *"Combining a set of imperfect estimators can be thought of as a way of managing the recognized limitations of the individual estimators"*. It is this *management* that we address here.

Training a neural network generally involves a delicate balance of various factors. The *bias-variance decomposition* [7] states that the mean square error (MSE) of a neural network is equal to the bias squared plus the variance. There is a trade-off between them — with more training, it is possible to achieve lower bias, but at the cost of a rise in variance. Krogh and Vedelsby [8] extend this concept to ensemble errors, showing how the bias can be seen as the extent to which the averaged output of the ensemble members differs from the target function, and the variance is the extent to which the ensemble members disagree. Ueda and Nakano [9] further provide a detailed proof of how the decomposition can be extended to a *bias-variance-covariance* one. From this result, one way to decrease the error is clear: decrease the covariance, ideally making it strongly negative. This means that an ideal ensemble consists of highly correct classifiers that disagree as much as possible, empirically verified in [10] among others.

Ensembles have been successfully applied for both regression and classification problems in varied domains, such as time series prediction [11], robotics [12], and medical diagnosis [13]. Here we apply them to prediction of network traffic flow in a telecommunications system [1].

Various algorithms have been proposed for training ensembles to achieve better generalisation. They can be broadly classified as manipulating the initial conditions, the network architectures, the training data, or the learning algorithm.

Early work trained networks independently, then averaged the results, hoping to achieve higher performance simply through differences in initial conditions (different weight initializations). The idea was that starting the networks in different areas of the weight space, they would follow different trajectories in the functional space. However, if a random initialization by chance gives a set of weights that are far from a solution, convergence can be exceedingly slow.

Manipulation of training data has been the most widely investigated method. Boosting [14], bagging, disjoint input sources [15], nonlinear transformations of input [15], and noise injection [16] have all proved their worth.

Manipulating the network topologies would mean having hybrid ensembles, consisting of estimators that work in different search spaces entirely. Different areas of functional (solution) space will be more accessible in certain search spaces than in others. Although this seems a promising path, not much work seems to have been done in the area.

The three methods mentioned so far are all implicit methods for achieving diverse errors, the nets may still converge to be highly correlated, regardless of your efforts. Explicit methods manipulate the training algorithm itself to produce decorrelated errors. Within this category ensemble methods are divided into those which train networks independently, then attempt to combine them, and those methods which take regard of the need for decorrelated errors during the training. Rosen [17] used a penalty function, training an ensemble sequentially, to decorrelate nets from ones that had been trained before, although this did not guarantee negative correlation of all the networks. A recent advancement, *Negative Correlation Learning* [2, 3, 4], trained the networks in parallel and negatively correlated the networks. This had the advantage of removing any bias in manipulation of the training data, as well as elimination of the need for a gating network. The negative correlation learning algorithm has shown marked improvements over other ensemble learning algorithms [2, 3, 4], but it does have a disadvantage in the need to adjust the penalty coefficient.

The rest of this paper is organised as follows. Section 2 introduces the basic ideas behind negative correlation learning. Section 3 explains how to evolve the penalty coefficient in negative correlation learning using the improved fast evolutionary programming (IFEP) [19]. Section 4 presents our experimental work. Finally, Section 5 concludes the paper.

2 Negative Correlation Learning

Negative correlation (NC) learning [2, 3, 4] is an efficient ensemble training method which can easily be implemented on top of standard backpropagation in feedforward networks. Take a set of neural networks N and a training pattern set P, each pattern in P is presented and backpropagated on, *simultaneously*, by the networks. A penalty term is introduced to the error function for the individual networks, which takes into account the error of the other networks in the ensemble. In the standard backpropagation algorithm, the error function for the output layer nodes is

$$\frac{1}{2}(F_i(n) - d(n))^2,$$

where $F_i(n)$ is the output of network i on pattern n, and $d(n)$ is the desired response for that pattern. In NC learning, the error function becomes

$$\frac{1}{2}(F_i(n) - d(n))^2 + \lambda p_i(n), \tag{1}$$

where $p_i(n)$ is

$$(F_i(n) - F(n)) \sum_{j \neq i} (F_j(n) - F(n)), \tag{2}$$

and $F(n)$ is the output of the ensemble on pattern n. A common ensemble output function is a simple average of the networks in the ensemble, i.e.,

$$F(n) = \frac{1}{N} \sum_{i=1}^{N} F_i(n).$$

In this case we have an overall error function of

$$\frac{1}{2}(F_i(n) - d(n))^2 - \lambda (F_i(n) - F(n))^2 \tag{3}$$

where λ is an adjustable parameter for the penalty. As can be seen from the equation, each network receives lower error for moving its response closer to the target response, and away from the mean response of all the other networks — this is a trade-off, controlled by the penalty parameter λ.

3 An Evolutionary Approach to Selecting λ

Although NC learning has been shown to be very effective and efficient in solving many problems [2, 3, 4], it can be time-consuming to select a near optimal λ. If it is too small, individual networks will not be sufficiently different and negatively correlated. The advantage of using ensembles will not be fully exploited. If λ is too large, individual networks may sacrifice the accuracy in exchange of negative correlation. Hence we may end up with an ensemble of very different networks, but none of them is good at the task to be learned.

Obviously, the answer to the question of which λ is the optimum is highly problem-dependent. The setting of λ is the classic problem of finding the trade-off between objective and penalty functions, and could be tackled effectively by an evolutionary approach. In this paper, we will apply an improved fast evolutionary programming (IFEP) algorithm [19] to evolve λ for a real world problem — the traffic flow prediction problem in a telecommunications network. The IFEP algorithm can be described as follows:

1. Generate the initial population of μ individuals, and set $k = 1$. Each individual is taken as a pair of real-valued vectors, $(\mathbf{x}_i, \eta_i)$, $\forall i \in \{1, \cdots, \mu\}$, where $\mathbf{x}_i$'s are objective variables and η_i's are strategy parameters that determine the search step size of mutation.

2. Evaluate the fitness score for each individual $(\mathbf{x}_i, \eta_i)$, $\forall i \in \{1, \cdots, \mu\}$, of the population based on the objective function, $f(\mathbf{x}_i)$.

3. Each parent $(\mathbf{x}_i, \eta_i)$, $i = 1, \cdots, \mu$, creates a single offspring $(\mathbf{x}_i', \eta_i')$ by: for $j = 1, \cdots, n$,

$$\eta_i'(j) = \eta_i(j) \exp(\tau' N(0,1) + \tau N_j(0,1)) \quad (4)$$
$$x_i'(j) = x_i(j) + \eta_i'(j) D_j(0,1), \quad (5)$$

where $x_i(j)$, $x_i'(j)$, $\eta_i(j)$ and $\eta_i'(j)$ denote the j-th component of the vectors $\mathbf{x}_i$, $\mathbf{x}_i'$, η_i and η_i', respectively. $N(0,1)$ denotes a normally distributed one-dimensional random number with mean 0 and standard deviation 1. $N_j(0,1)$ indicates that the random number is generated anew for each value of j. The factors τ and τ' are commonly set to $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$ [5]. $D_j(0,1)$ means either $N_j(0,1)$ or $C(1)$ (i.e., Cauchy distributed random number with scaling parameter $t = 1$).

4. Calculate the fitness of each offspring $(\mathbf{x}_i', \eta_i')$, $\forall i \in \{1, \cdots, \mu\}$.

5. Conduct pairwise comparison over the union of parents $(\mathbf{x}_i, \eta_i)$ and offspring $(\mathbf{x}_i', \eta_i')$, $\forall i \in \{1, \cdots, \mu\}$. For each individual, q opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win."

6. Select the μ individuals out of $(\mathbf{x}_i, \eta_i)$ and $(\mathbf{x}_i', \eta_i')$, $\forall i \in \{1, \cdots, \mu\}$, that have the most wins to be parents of the next generation.

7. Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to Step 3.

IFEP has been shown to be effective and efficient for many benchmark problems [19]. In evolving λ by IFEP, fitness evaluation will be based on the training error of ensemble learning.

4 Traffic Flow Prediction

4.1 The Dataset

The experiments were conducted using Austrian telecommunication flow data [1]. The data set was constructed from three data sources: a $(32, 32)$-interregional telecommunication flow matrix, a $(32, 32)$-distance matrix, and gross regional products for the 32 telecommunication regions. It contains 992 4-tuples (x_1, x_2, x_3, y), where the first three components represent the input vector $\mathbf{x} = (x_1, x_2, x_3)$ and the last component the target output, i.e. the telecommunication intensity from one region of origin to another region of destination. Input and target output data were preprocessed to be within $[0, 1]$. The telecommunication data stem from network measurements of carried telecommunication traffic in Austria in 1991, in terms of erlang, which is defined as the number of phone calls (including facsimile transmission) multiplied by the average length of the call (transfer) divided by the duration of measurement.

The neural network ensemble performance is measured in this study by the *average relative variance ARV(S)* of a set S of patterns [1]:

$$ARV(S) = \frac{\sum_{(x^k, y^k) \in S} (y^k - \Omega_L(x^k, \mathbf{w}))^2}{\sum_{(x^k, y^k) \in S} (y^k - \bar{y})^2} =$$
$$= \frac{1}{N_S} \frac{1}{\hat{\sigma}^2} \sum_{(x^k, y^k) \in S} (y^k - \Omega_L(x^k, \mathbf{w}))^2 \quad (6)$$

where y^k denotes the target value and $\bar{y}$ the average over the K desired values in S. The averaging, i.e. division by N_S makes $ARV(S)$ independent of the size of the set S. Thus $ARV(S)$ provides a normalized mean squared error metric for assessing the in-sample and out-of-sample performance of trained neural network ensembles. $ARV(S) = 1$ if the estimate is equivalent to the mean of the data (i.e, $\Omega_L(x^k, \mathbf{w}) = \bar{y}$). The division by the estimated variance $\hat{\rho}^2$ of the data removes the dependence on the dynamic range of the data.

λ	mean ARV	min	max
0.0	0.273486	0.258218	0.293902
0.5	0.271116	0.256660	0.283344
1.0	0.327312	0.262347	0.404953

Table 1: Performance of the ensemble with 3 networks and 2 hidden nodes in each. Results were averaged over 30 trials.

λ	mean ARV	min	max
0.0	0.271567	0.250356	0.300830
0.5	0.271058	0.255322	0.289233
1.0	0.312576	0.273501	0.401188

Table 2: Performance of the ensemble with 3 networks and 3 hidden nodes in each. Results were averaged over 30 trials.

4.2 Experimental Results with Fixed λ

Various ensemble architectures were tested, all consisted of feedforward multilayer perceptrons, all nodes using the logistic activation function. Learning rate 0.1 and momentum 0.8 were used. The error on the validation set was measured every 64 epochs. Fitness of an individual is defined as the negative of the ARV averaged over 30 random weight initialisations.

Six different ensemble architectures were tested using three different λ values in order to evaluate the impact of the architecture and λ on the ensemble performance. Tables 1 to 6 summarise the training results obtained using NC learning. It is clear from these results that different λ may lead to very different ensemble performance. It is also noticeable that ensembles trained with $\lambda = 1$ did not give the best performance in all cases. There is no clear winner between ensembles trained with $\lambda = 0$ and those trained with $\lambda = 0.5$, although it appeared that ensembles trained with $\lambda = 0$ tended to perform slightly better than those trained with $\lambda = 0.5$ when a larger ensemble and larger networks were used. This seems to indicate that negative correlation is likely to be most useful when a task is too complex to learn by any single networks. If a single network is sufficiently powerful in solving a problem, negative correlation is less useful to an ensemble.

4.3 Experimental Results with Evolving λ

In order to find a near optimal λ for NC learning, the IFEP algorithm described in Section 3 was applied to evolve λ. The IFEP algorithm was used with population size 4. Simple truncation selection was used, tak-

λ	mean ARV	min	max
0.0	0.271376	0.263476	0.300811
0.5	0.277429	0.265054	0.301111
1.0	0.307329	0.271044	0.393888

Table 3: Performance of the ensemble with 3 networks and 4 hidden nodes in each. Results were averaged over 30 trials.

λ	mean ARV	min	max
0.0	0.271189	0.256071	0.295975
0.5	0.273534	0.259535	0.292354
1.0	0.302880	0.253121	0.372399

Table 4: Performance of the ensemble with 4 networks and 2 hidden nodes in each. Results were averaged over 30 trials.

λ	mean ARV	min	max
0.0	0.273549	0.261471	0.297198
0.5	0.273715	0.264733	0.287570
1.0	0.292625	0.257993	0.359107

Table 5: Performance of the ensemble with 4 networks and 3 hidden nodes in each. Results were averaged over 30 trials.

λ	mean ARV	min	max
0.0	0.270545	0.259860	0.298174
0.5	0.272294	0.258569	0.291968
1.0	0.287234	0.256833	0.357884

Table 6: Performance of the ensemble with 4 networks and 4 hidden nodes in each. Results were averaged over 30 trials.

H	λ	mean ARV	min	max
2	0.746758525	0.269623	0.257804	0.287183
3	-4.960063612	0.268442	0.256066	0.275976
4	-16.45343198	0.268519	0.262368	0.288764

Table 7: Near optimal values for λ, for an ensemble with 3 networks, as discovered by IFEP. H is the number of hidden nodes in each network.

H	λ	mean ARV	min	max
2	0.714460568	0.268436	0.252264	0.289648
3	-1.553750871	0.266609	0.254209	0.282119
4	-1.327186364	0.266315	0.252610	0.274487

Table 8: Near optimal values for λ, for an ensemble with 4 networks, as discovered by IFEP. H is the number of hidden nodes in each network.

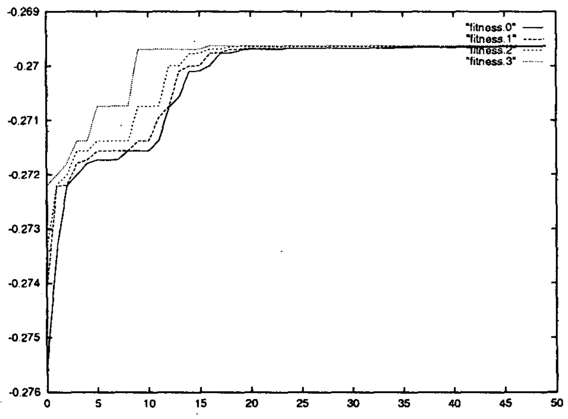

Figure 1: Fitness trace for ensembles of 3 networks with 2 hidden nodes in each.

ing the place of steps 4 and 5 in the IFEP algorithm. All ensemble architectures started with the same random initial population between -1 and $+1$.

Tables 7 and 8 show the near optimal λ values found by evolution using different ensemble architectures. Figures 1 to 4 show the evolution of fitness, λ, bias, variance and covariance for an ensemble of 3 networks with 2 hidden nodes in each.

5 Conclusions

This work has presented an evolutionary approach for setting the penalty coefficient in NC learning [2, 3, 4]. This technique has been applied to a network traffic flow prediction problem, allowing better performance than a monolithic neural network solution.

A highly desirable future work would be to adapt λ *during* the training, according to the characteristics of the dataset and the other networks' current performances. Even for a single problem, different learning stages may require different λ values. For example, in the initial stage of ensemble learning, we may prefer a small λ to encourage all individual networks to perform above

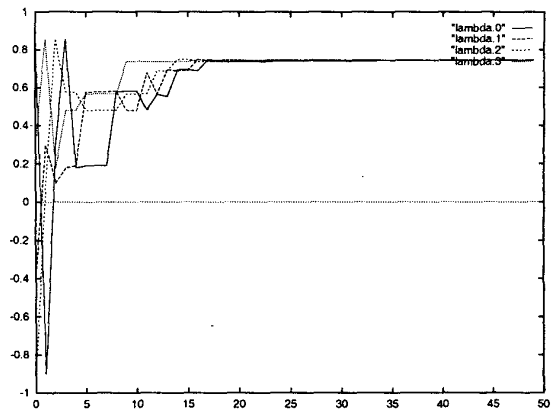

Figure 2: λ trace for ensembles of 3 networks with 2 hidden nodes in each.

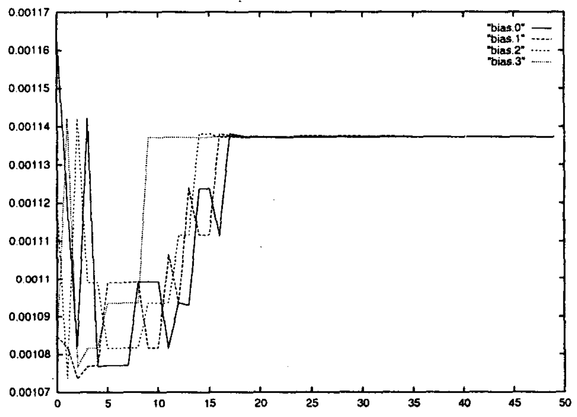

Figure 3: Bias trace for ensembles of 3 networks with 2 hidden nodes in each.

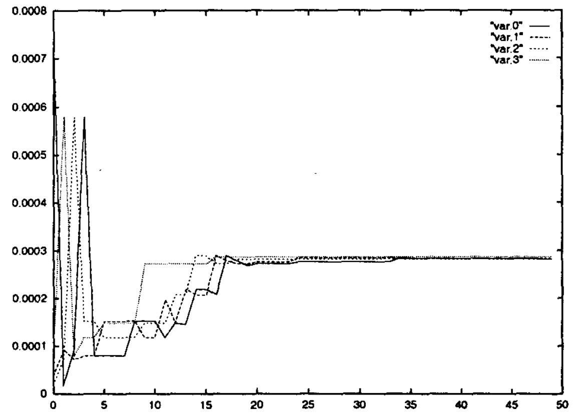

Figure 4: Variance trace for ensembles of 3 networks with 2 hidden nodes in each.

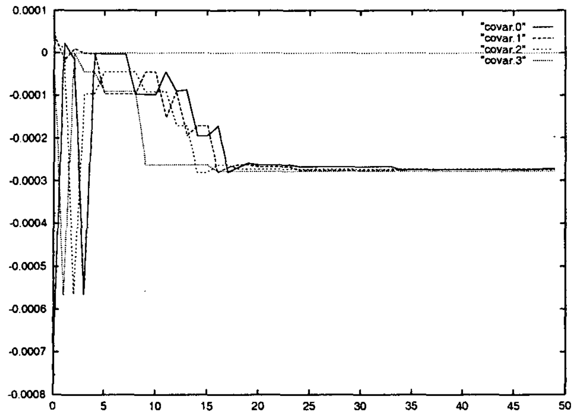

Figure 5: Covariance trace for ensembles of 3 networks with 2 hidden nodes in each.

a minimum standard (in terms of accuracy). As the learning progresses, we may want to increase λ such that individual networks can specialise without sacrificing individual accuracies. An adaptive method that can adjust λ automatically is needed to achieve this.

Another extension would be to allow asymmetric correlations between networks, which would be desirable if the networks moved in different search spaces, this could be due to differing architectures. In terms of NC learning, this would mean a different λ for each network's relationship with each other net. These extensions are work-in-progress.

References

[1] M. M. Fischer and S. Gopal, "Artificial neural networks: a new approach to modelling interregional telecommunication flows," *Journal of Regional Science*, vol. 34, no. 4, pp. 503–527, 1994.

[2] Y. Liu and X. Yao, "Negatively correlated neural networks can produce best ensembles," *Australian Journal of Intelligent Information Processing Systems*, vol. 4, no. 3/4, pp. 176–185, 1997.

[3] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, pp. 1399–1404, December 1999.

[4] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, pp. 716–725, December 1999.

[5] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[6] A. Sharkey, *Multi-Net Systems*, ch. Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, pp. 1–30. Springer-Verlag, 1999.

[7] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, pp. 1–58, 1992.

[8] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation and active learning," *Advances in Neural Information Processing Systems (NIPS-7)*, vol. 7, 1995.

[9] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *Proceedings of International Conference on Neural Networks (ICNN96)*, pp. 90–95, 1996.

[10] D. Opitz and J. Shavlik, "Generating accurate and diverse members of a neural-network ensemble," 1996.

[11] A. S. Weigend and M. Mangeas, "Nonlinear gated experts for time series: discovering regimes and avoiding overfitting," *International Journal of Neural Systems*, vol. 6, pp. 373–399, 1995.

[12] M. Meng and A. C. Kak, "Mobile robot navigation using neural networks and nonmetrical environment models," *IEEE Control Systems*, pp. 30–39, October 1993.

[13] A. Sharkey, N. Sharkey, and S. Cross, "Adapting an ensemble approach for the diagnosis of breast cancer," pp. 281–286, 1998.

[14] H. Drucker, C. Cortes, L. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," 1994.

[15] A. Sharkey, N. Sharkey, and G. Chandroth, "Diverse neural net solutions to a fault diagnosis problem," *Neural Computing and Applications*, vol. 4, pp. 218–227, 1996.

[16] Y. Raviv and N. Intrator, "Bootstrapping with noise: An effective regularisation technique," *Connection Science*, vol. 8, pp. 355–372, 1996.

[17] B. E. Rosen, "Ensemble learning using decorrelated neural networks," *Connection Science - Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, vol. 8, no. 3 and 4, pp. 373-384, 1996.

[18] Y. Liu and X. Yao, "Negatively correlated neural networks can produce best ensembles," *Australian Journal of Intelligent Information Processing Systems*, vol. 4, no. 3/4, pp. 176–185, 1997.

[19] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102, July 1999.