Neural Networks for Breast Cancer Diagnosis

Xin Yao
School of Computer Science
The University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
Email: x.yao@cs.bham.ac.uk

Yong Liu
Computer Science Division
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki 305-8568
Japan

Abstract- Breast cancer diagnosis has been approached by various machine learning techniques for many years. This paper describes two neural network based approaches to breast cancer diagnosis, both of which have displayed good generalisation. The first approach is based on evolutionary artificial neural networks. In this approach, a feedforward neural network is evolved using an evolutionary programming algorithm. Both the weights and architectures (i.e., connectivity of the network) are evolved in the same evolutionary process. The network may grow as well as shrink. The second approach is based on neural network ensembles. In this approach, a number of feedforward neural networks are trained simultaneously in order to solve the breast cancer diagnosis problem cooperatively. The basic idea behind using a group of neural networks rather than a monolithic one is divide-and-conquer. The negative correlation training algorithm we used attempts to decompose a problem automatically and then solve them. We will illustrate how negative correlation helps a group of neural networks learn using a real-world time series prediction problem.

1 Introduction

Breast cancer diagnosis has been a typical machine learning benchmark problem for many years. It has been dealt with using various machine learning algorithms [1]. The problem is nontrivial and difficult to solve as the data set is noisy and relatively small. Techniques which rely on a large training data set would not work well for this problem. This paper describes two neural network approaches to breast cancer diagnosis.

Neural networks have been applied to breast cancer diagnosis. However, most of these applications assumed a predefined network architecture (including connectivity and node transfer functions) and used a training algorithm (such as back-propagation) to learn weights (including biases). The issue of designing a near optimal network architecture remains open. The optimal design of neural networks can be formulated as a search problem [2, 3]. Evolutionary algorithms are particularly suited for dealing with such a search problem as the search space is large, complex, multimodal, and very noisy. Traditional algorithms have more difficulties in handling such search spaces.

As is the case for any other machine learning algorithms, generalisation is the most important issue in designing and evolving neural networks. We want to evolve neural networks which will generalise well rather than merely memorise the training data well. One way to encourage good generalisation is to evolve small and compact neural networks, following the principle of Occam's Razor. In order to bias towards smaller neural networks in evolution, most researchers used a complexity (regularisation) term in the fitness function to penalise large neural networks. While this method may work well when we have sufficient prior knowledge about how to balance the complexity penalty and learning accuracy, it often fails for problems where little prior knowledge is available. It is usually very difficult to adjust the balance between network complexity and learning accuracy for most real world problems. If we put too much emphasis on network complexity, we might evolve a compact neural network but poorly learned. If we put too much emphasis on learning accuracy, we might get a very large neural network which (over-)learned the training data very well but generalised poorly for unseen data. We will describe an alternative method, proposed previously [4, 5], in this paper for achieving good generalisation which avoid such a dilemma. No network complexity term was introduced in this method.

While the evolutionary approach works quite well for the breast cancer diagnosis problem, it might not be suitable for very large problems where network evolution and training require a large amount of time. For large problems, a typical approach is divide-and-conquer. In other words, we divide a large problem into smaller subproblems and then solve them. It is expected that solving several smaller problems would be easier and simpler than solving a single large problem. However, decomposing a large problem into several subproblems usually requires extensive domain knowledge. Some researchers have manually decomposed a large problem into several smaller ones and then trained several neural networks separately for them. Unfortunately, this method is difficult to use when little domain knowledge is available. An automatic method for decomposing a problem into subproblems is needed. A very promising study along this line is automatic modularisation by fitness sharing [6], where a population of rule-based systems solve a complex game-playing problem cooperatively. In this paper, we will describe a recently proposed algorithm for training neural network ensembles [7, 8] and its application to breast cancer diagnosis. We will also use a real-world time-series prediction problem to illustrate how this algorithm actually works.

The rest of this paper is organised as follows: Section 2 describes the evolutionary approach to neural network design based on an evolutionary programming algorithm and its application to breast cancer diagnosis. Section 3 describes the negative correlation algorithm for training neural network ensembles and its application to breast cancer diagnosis. The negative correlation algorithm will also be explained and analysed using the Chlorophyll-a prediction problem in this section. Finally, section 4 concludes with a brief summary of the paper.

2 Evolving Neural Networks

We have previously proposed an automatic system , EPNet [5], for evolving neural network architectures and weights. EPNet evolves generalised feed-forward neural networks. It uses direct representation in order to evolve weights in the same evolutionary process as the evolution of network architectures. There is no restriction on the connectivity pattern. Any feed-forward architectures are allowed.

The evolution is driven by an evolutionary programming algorithm [9, 10] where only mutation and selection are used. There are five different mutation operators for modifying network connectivity and weights: partial training, node deletion, connection deletion, connection addition and node addition [5]. These mutations are attempted sequentially in each generation. Only one of them will be executed in a generation. A mutation is regarded as successful and will be executed if the offspring improves its parent's performance by a certain margin. Otherwise the next mutation will be attempted. The sequence of attempting these mutations is quite important and is where we encourage the evolution of compact networks. The algorithm always attempts deletion before addition. Such a bias implies that the network size will not grow if a small network can achieve good learning performance already. It grows only when the task is too complex to learn for a small network. The fitness of a network (individual) is determined solely by its error. There is no complexity (regularisation) term in the fitness function. The main steps of EPNet can be summarised by Figure 1 [5].

There is an important point which is not illustrated

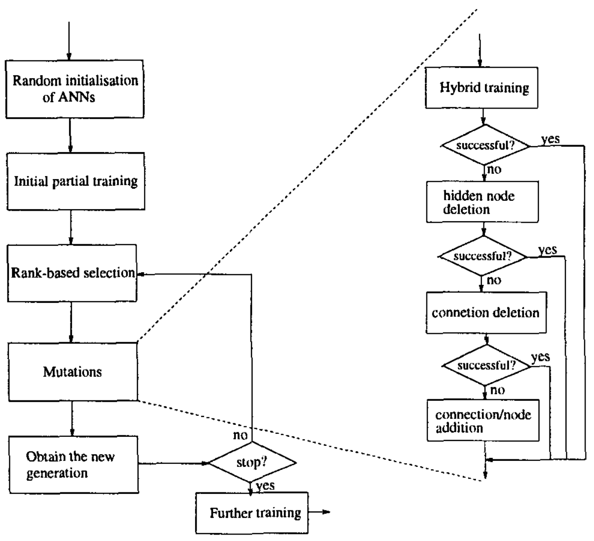


Figure 1: Major steps of EPNet for evolving feedforward neural networks.

in Figure 1. EPNet emphasizes behavioural rather than genetic evolution. In order to better maintain the behavioural link between a parent and its offspring, partial training was used after each architectural mutation (i.e., node/connection deletion/addition). In other words, a network will be trained for a fixed number of epochs using a modified back-propagation algorithm after an architectural mutation. The training may or may not lead to convergence of weights. The primary aim is to reduce the behavioural gap between the parent and its offspring. The importance of maintaining behavioural link has already been demonstrated by some experimental studies [11].

We have applied EPNet to a breast cancer diagnosis problem [5]. The data set we used was obtained from the UCI machine learning benchmark repository (anonymous ftp: `ics.uci.edu (128.195.1.1)` in the directory `/pub/machine-learning-databases`). The data set was originally obtained from W. H. Wolberg at the University of Wisconsin Hospitals, Madison. The purpose of the data set is to classify a tumour as either benign or malignant based on cell descriptions gathered by microscopic examination. The data set contains 9 attributes and 699 examples of which 458 are benign examples and 241 are malignant examples.

In our experiment, the whole data set was partitioned into three sets: a training set, a validation set, and a testing set. The training set was used to train neural networks by a modified back-propagation algorithm, the validation set was used to evaluate the fitness of the neural networks in evolution. The best neural network evolved by EPNet was further trained on the combined training and validation set before it was applied to the testing set. Considering the suggestions by Prechelt [12, 13] regarding empirical studies on neural networks, the first 349 examples of the whole data set were used for training, the following 175 examples for validation, and the final 175 examples for testing.

Tables 2 and 1 show the experimental results averaged over 30 runs [5]. The *error* in the tables refers to the error defined by Eq.(1). The *error rate* refers to the percentage of wrong classifications produced by the evolved neural networks.

The error function used in our experiment was [12]:

$$E = 100 \cdot \frac{o_{max} - o_{min}}{T \cdot n} \sum_{t=1}^{T} \sum_{i=1}^{n} (Y_i(t) - Z_i(t))^2 \quad (1)$$

where o_{max} and o_{min} are the maximum and minimum values of output coefficients in the problem representation, n is the number of output nodes, $Y_i(t)$ and $Z_i(t)$ are actual and desired outputs of node i for pattern t. The summation went over all T patterns in the *validation* set.[1]

Table 2: Architectures of evolved artificial neural networks for the breast cancer diagnosis problem. 30 runs were carried out.

	Number of connections	Number of hidden nodes	Number of generations
Mean	41.0	2.0	137.3
SD	14.7	1.1	37.7
Min	15	0	100
Max	84	5	240

Figure 2 shows the evolution of the mean of average numbers of connections and the mean of average classification accuracy of neural network over 30 runs for the breast cancer problem [5].

3 Neural Network Ensembles

While the evolutionary approach described in the previous section worked very well for the breast cancer and many other problems, it might be too computationally costly to evolve very large neural networks. This is true for any other approaches. When the problem to be solved becomes larger and more complex, designing a neural network for it becomes increasingly more difficult and time-consuming. One way to tackle large problems is to use the divide-and-conquer strategy. We can use a number of neural networks which solve a large problem collectively and cooperatively. Ideally, each individual neural network will learn a different aspect or component of a large problem so that the whole group can solve the entire problem. Negative correlation learning [8, 14] is an algorithm which encourages such cooperation among a group of neural networks, i.e., neural network ensembles.

There are several methods of designing neural network ensembles. Most of them follow the two-stage design process [15]; first generating individual networks, and then combining them. Usually, the individual networks are trained independent of each other. One of the disadvantages of such an approach is the loss of interaction among the individual networks during learning. There is no feedback from the combination stage to the individual design stage. It is possible that some of the independently designed individual networks do not make much contribution to the whole ensemble.

This section describes the negative correlation learning approach to designing neural network ensembles [8, 14]. The idea behind negative correlation learning is to encourage different individual networks to learn different parts or aspects of the training data so that the ensemble can learn the whole training data better. Negative correlation learning is different from previous work which trains the individual networks independently or sequentially [16, 17]. In negative correlation learning, the individual networks in the ensemble are trained simultaneously through unsupervised penalty terms in their error functions. Negative correlation learning attempts to train and combine individual networks in the same learning process. That is, the goal of each individual training is to generate the best result for the whole ensemble. Such an approach is quite different from other ensemble approaches which separate individual design from average procedures.

3.1 Negative Correlation Learning

Given the training data set $D = \{(\mathbf{x}(1), y(1)), \cdots, (\mathbf{x}(N), y(N))\}$, this section considers estimating y by forming an ensemble whose output is a simple averaging of outputs of a set of neural networks

$$F(n) = \frac{1}{M} \Sigma_{i=1}^{M} F_i(n) \quad (2)$$

where M is the number of the individual neural networks in the ensemble, $F_i(n)$ is the output of network i on the nth training pattern, $F(n)$ is the output of the ensemble on the nth training pattern.

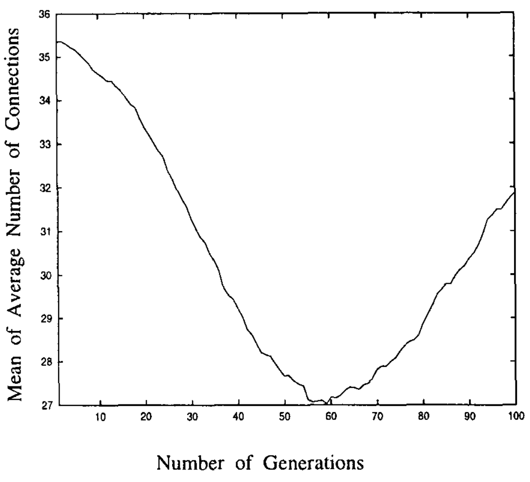
Negative correlation learning introduces a correlation penalty term into the error function of each individual network in the ensemble so that all the networks can be trained simultaneously and interactively on the same training data set D. The error function E_i for network i in negative correlation learning is defined by

$$E_i = \frac{1}{N} \Sigma_{n=1}^{N} E_i(n)$$

[1]We used the validation set to compute the error during evolution and the training set to compute the error during partial training in order to improve generalisation of evolved neural networks.

Table 1: Accuracies of evolved neural networks for the breast cancer diagnosis problem. 30 runs were carried out.

	Training set		Validation set		Test set	
	error	error rate	error	error rate	error	error rate
Mean	3.246	0.03773	0.644	0.00590	1.421	0.01376
SD	0.589	0.00694	0.213	0.00236	0.729	0.00938
Min	1.544	0.01719	0.056	0.00000	0.192	0.00000
Max	3.890	0.04585	1.058	0.01143	3.608	0.04000



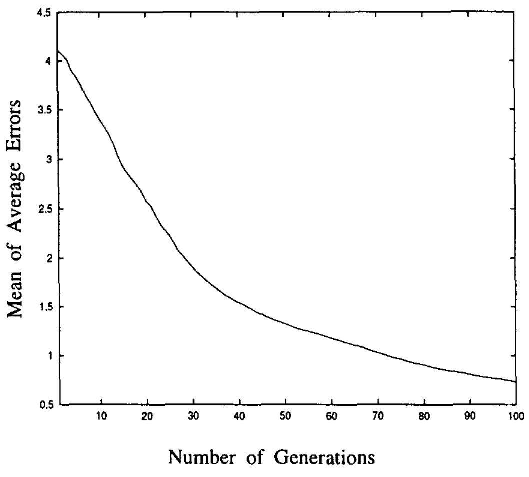


Figure 2: Evolution of neural network's connections and accuracy for the breast cancer problem.

$$= \frac{1}{N}\Sigma_{n=1}^{N}\left[\frac{1}{2}(F_i(n)-y(n))^2+\lambda p_i(n)\right] \quad (3)$$

where N is the number of training patterns, $E_i(n)$ is the value of the error function of network i at presentation of the nth training pattern, and $y(n)$ is the desired output of the nth training pattern.

The first term in the right side of Eq.(3) is the *empirical risk function* of network i. The second term p_i is a correlation penalty function. The purpose of minimising p_i is to negatively correlate each network's error with errors for the rest of the ensemble. The parameter $0 \leq \lambda \leq 1$ is used to adjust the strength of the penalty. The penalty function p_i has the form:

$$p_i(n) = (F_i(n) - F(n))\,\Sigma_{j\neq i}(F_j(n) - F(n)) \quad (4)$$

The partial derivative of E_i with respect to the output of network i on the nth training pattern is

$$\begin{aligned}\frac{\partial E_i(n)}{\partial F_i(n)} &= F_i(n) - d(n) + \lambda\frac{\partial p_i(n)}{\partial F_i(n)}\\ &= F_i(n) - d(n) + \lambda\Sigma_{j\neq i}(F_j(n) - F(n))\\ &= F_i(n) - d(n) - \lambda(F_i(n) - F(n)) \quad (5)\end{aligned}$$

where we have made use of the assumption that F(n) has constant value with respect to $F_i(n)$. The standard back-propagation (BP) algorithm [18] has been used for weight adjustments in the mode of pattern-by-pattern updating. That is, weight updating of all the individual networks is performed simultaneously using Eq.(5) after the presentation of each training pattern. One complete presentation of the entire training set during the learning process is called an *epoch*. The negative correlation learning from Eq.(5) is a simple extension to the standard BP algorithm. In fact, the only modification that is needed is to calculate an extra term of the form $\lambda(F_i(n) - F(n))$ for the ith network.

From Eqs.(3), (4), and (5), we may make the following observations:

1. During the training process, all the individual networks interact with each other through their penalty terms in the error functions. Each network F_i minimises the difference between $F_i(n)$ and $y(n)$ while maximising the difference between $F_i(n)$ and $F(n)$. That is, negative correlation learning considers errors what all other networks have learned while training a network.
2. For $\lambda = 0.0$, there are no correlation penalty terms in the error functions of the individual networks, and the individual networks are just trained independently. That is, independent training for the individual networks is a special case of negative correlation learning.

3. For $\lambda = 1$, from Eq.(5) we get

$$\frac{\partial E_i(n)}{\partial F_i(n)} = F(n) - y(n) \tag{6}$$

Note that the empirical risk function of the ensemble for the nth training pattern is defined by

$$E_{ensemble} = \frac{1}{2}(\Sigma_{i=1}^{M} F_i(n) - d(n))^2 \tag{7}$$

The partial derivative of $E_{ensemble}$ with respect to F_i on the nth training pattern is

$$\begin{aligned}\frac{\partial E_{ensemble}}{\partial F_i(n)} &= \frac{1}{M}(\Sigma_{i=1}^{M} F_i(n) - d(n)) \\ &= \frac{1}{M}(F(n) - y(n)) \end{aligned} \tag{8}$$

In this case, we get

$$\frac{\partial E_i(n)}{\partial F_i(n)} \propto \frac{\partial E_{ensemble}}{\partial F_i(n)} \tag{9}$$

The minimisation of the empirical risk function of the ensemble is achieved by minimising the error functions of the individual networks. From this point of view, negative correlation learning provides a novel way to decompose the learning task of the ensemble into a number of subtasks for each network.

3.2 Correlations among the Individual Networks

This section analyses negative correlation learning on the Chlorophyll-a prediction problem [19, 20] to show how and why negative correlation learning works.

3.3 Experimental setup

The Chlorophyll-a prediction is a highly nonlinear problem. Previous attempts using neural networks have achieved only limited success [19, 20]. The limnological time series for 3 years between 1984 and 1986 were used in our experiments. The data set was divided into two parts. The first part used the data in 1984 and 1985 as the training data to train the neural network ensemble. Then the neural network ensemble was tested on the 1986 data. Using the 1986 data as the testing data was suggested by Recknagel [20] because it represented typical year for blooms of *Microcystis.* More details about the data were given in [20].

The normalised root-mean-square (RMS) error E was used to evaluate the performance of negative correlation learning, which is determined by the RMS value of the absolute prediction error for $\Delta t = 1$, divided by the standard deviation of $x(t)$ [21, 22],

$$E = \frac{\langle [x_{pred}(t, \Delta t) - x(t + \Delta t)]^2 \rangle^{\frac{1}{2}}}{\langle (x - \langle x \rangle)^2 \rangle^{\frac{1}{2}}} \tag{10}$$

where $x_{pred}(t, \Delta t)$ is the prediction of $x(t + \Delta t)$ from the current state $x(t)$ and $\langle x \rangle$ represents the expectation of x. The ensemble architecture used in the experiments has four networks. Each individual network is a feedforward network with one hidden layer. Both hidden node function and output node function are defined by the logistic function

$$\varphi(y) = \frac{1}{1 + \exp(-y)} \tag{11}$$

All the individual networks have ten hidden nodes. The number of training epochs was set to 2000.

3.4 Experimental Results

Table 3 shows the average results of negative correlation learning and independent training (i.e., $\lambda = 0.0$ in negative correlation learning) over 25 runs for the chlorophyll-a prediction problem. Each run of negative correlation learning was from different initial weights. The value of t-test between negative correlation learning and independent training with 24 degrees of freedom is -25.08 which is significant at $\alpha = 0.005$ by a two-tailed test, i.e., negative correlation learning is statistically significantly better than independent training.

Table 3: The average results of RMS errors produced by negative correlation learning with $\lambda = 1$ and independent training (i.e., $\lambda = 0.0$ in negative correlation learning) over 25 runs for the chlorophyll-a prediction in Lake Kasumigaura from 1984 to 1 986.

λ	Year	Mean	Std Dev	Min	Max
1	1983-84	0.0161	0.0007	0.0150	0.0178
	1986	0.0815	0.0076	0.0732	0.1013
0	1983-84	0.0236	0.0029	0.0215	0.0366
	1986	0.1168	0.0043	0.1099	0.1321

In order to observe the effect of the correlation penalty terms, Table 4 shows the correlations among the individual networks trained by negative correlation learning with different λ. The correlations between network i and network j is given by

$$Cor_{ij} = \frac{\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}\left(F_i^{(k)}(n) - \overline{F}_i(n)\right)\left(F_j^{(k)}(n) - \overline{F}_j(n)\right)}{\sqrt{\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}\left(F_i^{(k)}(n) - \overline{F}_i(n)\right)^2 \Sigma_{n=1}^{N}\Sigma_{k=1}^{K}\left(F_j^{(k)}(n) - \overline{F}_j(n)\right)^2}} \quad (12)$$

where $F_i^{(k)}(n)$ is the output of the network i on the nth pattern in the testing set from the kth run, $\overline{F}_i(n)$ represents the average output of the network i on the nth pattern in the testing set, N is the number of patterns in the testing set, and K is the number of runs. There are $\begin{pmatrix} 4 \\ 2 \end{pmatrix} = 6$ correlations among different pairs of networks. The values of the correlations among the individual networks had relatively larger positive values when λ was 0. They reduced to relatively smaller positive values when λ was increased to 0.5. All of them became negative values when λ was further increased to 1. Overall, the results indicated that the neural networks trained by negative correlation learning tended to be negatively correlated. Because every individual network learns the same task in the independent training, the correlations among them are generally positive. In negative correlation learning, each individual network learns different parts or aspects of the training data so that the problem of correlated errors can be removed or alleviated. The empirical studies match the theoretical results [23]: when individual networks in an ensemble are unbiased, average procedures are most effective in combining them when errors in the individual networks are negatively correlated and moderately effective when the errors are uncorrelated. There is little to be gained from average procedures when the errors are positively correlated.

3.5 Application to the Breast Cancer Problem

This section describes the application of negative correlation learning to the breast cancer problem as described in Section 2. The data set was partitioned into two sets: a training and a testing set. The testing set was not seen by any neural network during the training phase. It is only used for testing the generalisation of neural network ensembles after they are trained. We used the first 524 examples for the training set, and the rest 175 examples for the testing set.

The input attributes were rescaled to between 0.0 and 1.0 by a linear function. The output attributes of all the problems were encoded using a 1-of-m output representation for m classes. The output with the highest activation designates the class.

The ensemble architecture used in the experiments consisted of four multilayer perceptrons with one hidden layer. All the individual networks have ten hidden nodes in the hidden layer. Both hidden node function and output node function are defined by the logistic function in Eq.(11). The number of training epochs was set to 250. The strength parameter λ was set to 1.0. These parameters were chosen after limited preliminary experiments. They are not meant to be optimal.

Table 5 shows the results of negative correlation learning over 25 independent runs. Each run of negative correlation learning was from different initial weights. The same architectures with the same initial weight setup were also independently trained using BP without the correlation penalty terms. The results are also shown in Table 5. For classification problems, the simple averaging defined in Eq.(2), was first applied to decide the output of the ensemble. For the simple averaging, it was surprising that the results of negative correlation learning with $\lambda = 1.0$ were similar to those of independent training. This phenomenon seems contradictory to the claim that the effect of the correlation penalty term is to encourage different individual networks in an ensemble to learn different parts or aspects of the training data. In order to verify and quantify this claim, we compared the outputs of the individual networks trained with the correlation penalty term to those of the individual networks trained without the correlation penalty term.

Two notions were introduced to analyses negative correlation learning. They are the correct response sets of individual networks and their intersections. The correct response set S_i of individual network i on the testing set consists of all the patterns in the testing set which are classified correctly by the individual network i. Let Ω_i denote the size of set S_i, and $\Omega_{i_1 i_2 \cdots i_k}$ denote the size of set $S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_k}$. Table 6 shows the size of the correct response sets of individual networks, which were respectively created by negative correlation learning and independent training, on the testing set and their intersections. It is evident from Table 6 that different individual networks created by negative correlation learning were able to specialise to different parts of the testing set. For instance, The sizes of correct response sets S_1, S_2, S_3, and S_4 at $\lambda = 0.0$ are from 170 to 173, while the size of their intersection set $S_1 \cap S_2 \cap S_3 \cap S_4$ is 168. There are five different patterns correctly classified by the four individual networks in the ensemble. In contrast, the individual networks in the ensemble created by independent training are quite similar. There are only one different pattern correctly classified by the four individual networks in the ensemble.

In simple averaging, all individuals have the same combination weights and are treated equally. However, not all individuals are equally important. Because different individual networks created by negative correlation learning were able to specialise to different parts of the testing set, only the outputs of these specialists should be considered to make the final decision of the

Table 4: The correlations among the individual networks trained by negative correlation learning with different λ values in the noise free condition

0	$Cor_{12} = 0.21972$	$Cor_{13} = 0.12713$	$Cor_{14} = 0.24398$
	$Cor_{23} = 0.17909$	$Cor_{24} = 0.24678$	$Cor_{34} = 0.11826$
0.25	$Cor_{12} = 0.16973$	$Cor_{13} = 0.07010$	$Cor_{14} = 0.17965$
	$Cor_{23} = 0.11519$	$Cor_{24} = 0.17438$	$Cor_{34} = 0.08683$
0.5	$Cor_{12} = 0.11468$	$Cor_{13} = 0.00593$	$Cor_{14} = 0.09827$
	$Cor_{23} = 0.03867$	$Cor_{24} = 0.10509$	$Cor_{34} = 0.06474$
0.75	$Cor_{12} = 0.05957$	$Cor_{13} = -0.02466$	$Cor_{14} = 0.12302$
	$Cor_{23} = 0.02883$	$Cor_{24} = 0.33109$	$Cor_{34} = -0.03763$
1	$Cor_{12} = -0.37871$	$Cor_{13} = -0.37402$	$Cor_{14} = -0.21235$
	$Cor_{23} = -0.22528$	$Cor_{24} = -0.39379$	$Cor_{34} = -0.41414$

ensemble for this part of the testing set. In this paper, a winner-takes-all method is applied to select such individuals. For each pattern of the testing set, the output of the ensemble is only decided by the individual whose output has the highest activation. Table 5 shows the average results of negative correlation learning over 25 runs using the winner-takes-all combination method. The winner-takes-all combination method improved negative correlation learning remarkably because there are good and poor individuals for each pattern in the testing set and winner-takes-all selects the best individual. However it did not improved the independent training much because the individual networks created by the independent training are all similar to each other.

Table 5: Comparison of error rates between negative correlation learning ($\lambda = 1.0$) and independent training (i.e., $\lambda = 0.0$ in negative correlation learning) on the breast cancer problem. The results were averaged over 25 runs. "SA" and "WTA" indicate two different combination methods, i.e., the simple averaging and the winner-takes-all.

		SA		WTA
		Training	Test	Test
$\lambda = 1.0$	Mean	0.0250	0.0114	0.0
	SD	0.0245	0.0	0.0
	Min	0.0248	0.0114	0.0
	Max	0.0267	0.0114	0.0
$\lambda = 0.0$	Mean	0.0304	0.0101	0.0096
	SD	0.0251	0.0024	0.0027
	Min	0.0286	0.0057	0.0057
	Max	0.0324	0.0114	0.0114

4 Conclusions

This paper describes two neural network approaches to breast cancer diagnosis, the evolutionary approach and the ensemble approach. Both approaches can deal with the diagnosis problem very well in terms of good generalisation of neural networks. The evolutionary approach can be used to design compact neural networks automatically by evolving network architectures and weights simultaneously. The neural network ensemble approach is aimed at tackling large problems which may not dealt with efficiently by a monolithic neural network, although the design of individual networks is still manual at this stage. Work on evolving neural network ensembles is in progress [24].

Table 6: The sizes of the correct response sets of individual networks created by negative correlation learning ($\lambda = 1.0$) and independent training (i.e., $\lambda = 0.0$ in negative correlation learning) on the testing set and their intersections for the breast cancer problem. The results were obtained from the first run among the 25 runs.

$\lambda = 1.0$	$\Omega_1 = 173$	$\Omega_2 = 173$	$\Omega_3 = 173$
	$\Omega_4 = 170$	$\Omega_{12} = 173$	$\Omega_{13} = 173$
	$\Omega_{14} = 168$	$\Omega_{23} = 173$	$\Omega_{24} = 168$
	$\Omega_{34} = 168$	$\Omega_{123} = 173$	$\Omega_{124} = 168$
	$\Omega_{134} = 168$	$\Omega_{234} = 168$	$\Omega_{1234} = 168$
$\lambda = 0.0$	$\Omega_1 = 173$	$\Omega_2 = 173$	$\Omega_3 = 173$
	$\Omega_4 = 174$	$\Omega_{12} = 173$	$\Omega_{13} = 173$
	$\Omega_{14} = 173$	$\Omega_{23} = 173$	$\Omega_{24} = 173$
	$\Omega_{34} = 173$	$\Omega_{123} = 173$	$\Omega_{124} = 173$
	$\Omega_{134} = 173$	$\Omega_{234} = 173$	$\Omega_{1234} = 173$

Bibliography

[1] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification.* London: Ellis Horwood Limited, 1994.

[2] X. Yao, "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, vol. 8, no. 4, pp. 539–567, 1993.

[3] X. Yao, "Evolutionary artificial neural networks," in *Encyclopedia of Computer Science and Technology* (A. Kent and J. G. Williams, eds.), vol. 33,

pp. 137–170, New York, NY 10016: Marcel Dekker Inc., 1995.

[4] X. Yao and Y. Liu, "Evolving artificial neural networks for medical applications," in *Proc. of 1995 Australia-Korea Joint Workshop on Evolutionary Computation*, pp. 1–16, KAIST, Taejon, Korea, September 1995.

[5] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.

[6] P. J. Darwen and X. Yao, "Speciation as automatic categorical modularization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 101–108, 1997.

[7] Y. Liu and X. Yao, "A cooperative ensemble learning system," in *1998 IEEE International Joint Conference on Neural Networks, Anchorage, USA*, (Piscataway, NJ, USA), pp. 2202–2207, IEEE Press, 1998.

[8] 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.

[9] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution.* New York, NY: John Wiley & Sons, 1966.

[10] D. B. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence.* New York, NY: IEEE Press, 1995.

[11] X. Yao, "The importance of maintaining behavioural link between parents and offspring," in *Proc. of the 1997 IEEE Int'l Conf. on Evolutionary Computation (ICEC'97), Indianapolis, USA*, pp. 629–633, IEEE Press, New York, NY, April 1997.

[12] L. Prechelt, "Proben1 — a set of neural network benchmark problems and benchmarking rules," Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, 76128 Karlsruhe, Germany, September 1994.

[13] L. Prechelt, "Some notes on neural learning algorithm benchmarking," *Neurocomputing*, vol. 9, no. 3, pp. 343–347, 1995.

[14] Y. Liu and X. Yao, "A cooperative ensemble learning system," in *Proc. of the 1998 IEEE International Joint Conference on Neural Networks (IJCNN'98)*, (Piscataway, NJ, USA), pp. 2202–2207, IEEE Press, 1998.

[15] A. Sharkey, "On combining artificial neural nets," *Connection Science*, vol. 8, pp. 299–313, 1996.

[16] M. P. Perrone and L. N. Cooper, "When networks disagree: ensemble methods for hybrid neural networks," in *Neural Networks for Speech and Image Processing* (R. J. Mammone, ed.), London: Chapman-Hall, 1993.

[17] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," *Neural Computation*, vol. 6, pp. 1289–1301, 1994.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I* (D. E. Rumelhart and J. L. McClelland, eds.), pp. 318–362, MIT Press, Cambridge, MA, 1986.

[19] F. Recknagel, "ANNA — artificial neural network model for predicting species abundance and succession of blue-green algae," *Hydrobiologia*, vol. 349, pp. 47–57, 1997.

[20] F. Recknagel, T. Fukushima, T. Hanazato, N. Takamura, and H. Wilson, "Modelling and prediction of phyto- and zooplankton dynamics in lake kasumigaura by artificial neural networks," *Lakes and Reservoirs*, vol. in press, 1998.

[21] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Physical Review Letters*, vol. 59, no. 8, pp. 845–847, 1987.

[22] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, ""neural-gas" network for vector quantization and its application to time-series prediction," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, pp. 558–569, 1993.

[23] R. T. Clemen and R. L. Winkler, "Limits for the precision and value of information from dependent sources," *Operations Research*, vol. 33, pp. 427–442, 1985.

[24] Y. Liu and X. Yao, "Towards designing neural network ensembles by evolution," in *Parallel Problem Solving from Nature (PPSN) V* (A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, eds.), vol. 1498 of *Lecture Notes in Computer Science*, (Berlin), pp. 623–632, Springer-Verlag, 1998.