# Evolutionary multiobjective approach to constructing neural network ensembles for regression

**Yaochu Jin, Tatsuya Okabe, Bernhard Sendhoff**

**2004**

# CHAPTER 1

# Evolutionary Multi-objective Optimization Approach to Constructing Neural Network Ensembles for Regression

Yaochu Jin, Tatsuya Okabe and Bernhard Sendhoff
*Honda Research Institute Europe*
*Carl-Legien-Str.30, 63073 Offenbach, Germany*
*E-mail: yaochu.jin@honda-ri.de*

Neural network ensembles have shown to be very effective in improving the performance of neural networks when they are used for classification or regression. One essential issue in constructing neural network ensembles is to ensure that the ensemble members have sufficient diversity in their behavior. This chapter suggests a multi-objective approach to generating diverse neural network ensemble members. A genetic algorithm is used to evolve both the weights and structure of the neural networks. Besides, the R-prop learning algorithm is employed for efficient life-time learning of the weights. Different complexity criteria, such as the number of connections, the sum of absolute weights and the sum of squared weights have been adopted as an additional objective other than the approximation accuracy. Ensembles are constructed using the whole set or a subset of found non-dominated solutions. Various methods for selecting a subset from the found non-dominated solutions are compared. The proposed multi-objective approach is compared to the random approach on two regression test problems. It is found that using a neural network ensemble can significantly improve the regression accuracy, especially when a single network is not able to predict reliably. In this case, the multi-objective approach is effective in finding diverse neural networks for constructing neural network ensembles.

## 1. Introduction

It has been shown that neural network ensembles are able to improve the generalization performance both for classification and regression[17,3]. The benefit of using a neural network ensemble originates from the diversity of the behavior of the ensemble members. Basically, diversity of ensemble members can be enhanced by using various initial random weights, varying the network architecture, employing different training algorithms or supplying different training data[20]. In some cases, it is also possible to increase network diversity by generating training data from different sources. For example, the geometry of an object can be represented by parametric or non-parametric methods. Thus, different sources of training data can be obtained for describing certain performance of the same object.

Compared to the above-mentioned methods that achieve diversity implicitly, methods for explicitly encouraging diversity among ensemble members have been widely studied in the recent years. Measures for increasing diversity include a diversity index[16], degree of decorrelation[19], or degree of negative correlation[14,15] between the output of the candidate networks.

Individual neural networks in an ensemble can be trained either independently, sequentially and simultaneously[11]. In the first case, neural networks are generated separately and no interaction between the networks will be taken into account in training. In the second case, neural networks are generated sequentially. However, the correlation between the current network and the existing ones will be considered too to encourage diversity. In the third case, neural networks are trained simultaneously, not only minimizing the approximation error, but also encouraging diversity among individual networks. Obviously, in the latter two approaches, diversity is taken into account explicitly. It is believed that one possible disadvantage of simultaneous training is that the networks in the population could be competitive[11].

In training single neural networks, regularization techniques have widely been employed to improve the generalization performance of neural networks[3]. A general idea is to include an additional term in the cost function of learning algorithms, often known as the regularization, to avoid overfitting the training data. Actually, most diversity based methods for generating ensembles can also be seen as a kind of regularization techniques.

From the multi-objective optimization point of view, adding a regularization term in the cost function is equivalent to combining two objectives using a weighted aggregation formulation. Thus, it is straightforward to re-formulate the regularization techniques as multi-objective optimization problems. Such ideas have been reported[4]. In that chapter, a variation of the $\epsilon$-constraint algorithm was adopted to obtain one single Pareto-optimal solution that simultaneously minimizes the training error and the norm of the weights. Similar work has also been reported[1], where a multi-objective evolutionary algorithm is used to minimize the approximation error and the number of hidden nodes of the neural network. Again, only the one with the minimal approximation error has been selected for final use. In addition, multi-objective optimization has been employed to evolve neural network modules in a cooperative co-evolution framework to increase diversity of the modules[7].

This chapter presents a method for generating a set of Pareto-optimal neural networks for constructing neural network ensembles. The genetic algorithm with Lamarckian inheritance for evolving neural networks[9] is adapted to the multi-objective optimization purpose. To this end, the elitist non-dominated sorting and the crowded tournament selection suggested[6] are adopted for fitness assignment and selection. The whole obtained non-dominated set or a subset of it is used to construct neural ensembles. The performance of the ensembles are compared on two test problems. Ensembles whose members are generated using the multi-objective approach is also compared to those whose member networks are generated independently. It is shown that the performance of the ensembles depends to a large degree on the features of the training, validation and test data.

## 2. Multi-objective Optimization of Neural Networks

### 2.1. *Parameter and Structure Representation of the Network*

A connection matrix and a weight matrix are employed to describe the structure and the weights of the neural networks. Obviously, the connection matrix specifies the structure of the network whereas the weight matrix determines the strength of each connection. Assume that a neural network consists of $M$ neurons in total, including the input and output neurons, then the size of the connection matrix is $M \times (M+1)$, where an element in last column indicates whether a neuron is connected to a bias value. In the matrix, if element $c_{ij}, i = 1, ..., M, j = 1, ..., M$ equals 1, it means that there is a connection between the $i$-th and $j$-th neuron and the signal flows from neuron $j$ to neuron $i$. If $j = M + 1$, it indicates that there is a bias in the $i$-th neuron. Obviously, for a purely feedforward network, the upper part of the matrix, except the $M + 1$-th column is always zero. Fig. 1 illustrates a connection matrix and the corresponding network structure. It can be seen from the figure that the network has one input neuron, two hidden neurons, and one output neuron. Besides, both hidden neurons have a bias.

The strength (weight) of the connections is defined in the weight matrix. Accordingly, if the $c_{ij}$ in the connection matrix equals zero, the corresponding element in the weight matrix must be zero too.

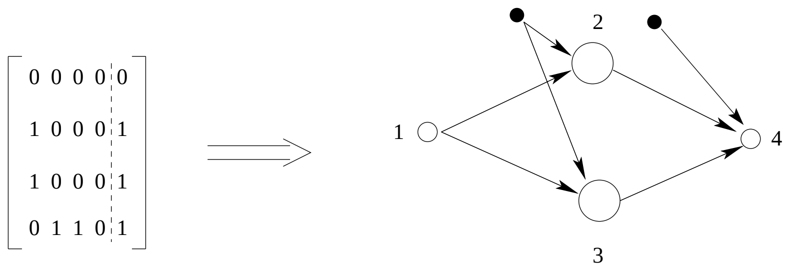

Fig. 1. A connection matrix and the corresponding network structure.

## 2.2. *Objectives in Network Optimization*

The most common objective function (also known as the error function or the cost function) in training or evolving neural networks is the mean squared error (MSE):

$$E = \frac{1}{N}\sum_{i=1}^{N}(y^d(i) - y(i))^2, \tag{1}$$

where $N$ is the number of training samples, $y^d(i)$ is the desired output of the $i$-th sample, and $y(i)$ is the network output for the $i$-th sample. For the sake of clarity, we assume here that the neural network has only one output. Other error functions, such as Minkowski error or cross-entropy can also be used.[3]

It has been found that neural networks can often over-fit the training data, which means that the network has a very good approximation accuracy on the training data, but a very poor one on unseen data. Many methods have been developed to improve the generalization performance of neural networks. [3] A very popular technique to improve the generalization performance is known as regularization, which usually adds a penalty term to the error function:

$$J = E + \lambda\Omega, \tag{2}$$

where $\lambda$ is a coefficient that controls the extent to which the regularization influences the optimal solution, and $\Omega$ is known as the regularizer. A simple class of regularizers is to penalize the sum of squared weights, also known as the Gaussian regularizer, which favors smooth output of the network:

$$\Omega = \frac{1}{2}\sum_{k} w_k^2, \tag{3}$$

where $k$ is an index summing up all weights. Alternatively, the sum of absolute weights, also known as the Laplace regularizer can be used:

$$\Omega = \sum_i |w_i|. \tag{4}$$

The Gaussian regularizer and the Laplace regularizer are also known as weight decay in neural network training.

In neural network training using regularization techniques, it is often a matter of trial-and-error to determine the coefficient $\lambda$, although methods have been developed to optimize the coefficient based on empirical, algebraic or Bayesian estimation of the generalization error on the validation data.[21] This situation is quite easy to understand from the multi-objective point of view. For each given $\lambda$, one single Pareto-optimal solution will be obtained. Obviously, the regularization technique in equation (2) can be reformulated as a bi-objective optimization problem:

$$\min \{f_1, f_2\} \tag{5}$$

$$f_1 = E, \tag{6}$$

$$f_2 = \Omega, \tag{7}$$

where $E$ is defined in equation (1), and $\Omega$ is one of the regularization term defined in equation (3) or (4).

Basically, the weight decay techniques try to reach a good trade-off between the complexity of neural networks and the approximation accuracy to avoid overfitting the training data. Another straightforward index for measuring the complexity of neural networks is the sum of connections in the network:

$$\Omega = \sum_i \sum_j c_{ij}. \tag{8}$$

Obviously, the smaller the number of connections in a network is, the less complex the network. Note that this regularizer is well suited for evolutionary optimization although it is not applicable to gradient-based learning algorithms due to its discrete nature. For this reason, we term it as *evolutionary regularization.*

In the following study, the sum of connections, the sum of absolute weights and the sum of squared weights are employed as the second objective in optimization.

## 2.3. *Mutation and Learning*

A genetic algorithm with a hybrid of binary and real-valued coding has been used for optimizing the structure and weights of the neural networks. The genetic operators used are quite specific. Four mutation operators are implemented on the chromosome encoding the connection matrix, namely, insertion of a hidden neuron, deletion of a hidden neuron, insertion of a connection and deletion of a connection.[9] A Gaussian-type mutation is applied on the chromosome encoding the weight matrix. One of the five mutation operators is randomly selected and performed on each individual. No crossover has been employed in this algorithm.

After mutation, an improved version of the Rprop algorithm[10] has been carried out to train the weights. This can be seen as a life-time learning within a generation. After learning, the fitness of each individual with regard to the approximation error ($f_1$) is updated. In addition, the weights modified during the life-time learning are also encoded back into the chromosome, which is known as the Lamarckian type of inheritance[2].

In the life-time learning, only the first objective, i.e., the approximation error will be minimized. The Rprop learning algorithm is employed in this work because it is believed that the Rprop learning algorithm is faster and more robust compared with other gradient-based learning algorithms.

Let $w_{ij}$ denotes the weight connecting neuron $j$ and neuron $i$, then the change of the weight ($\Delta w_{ij}$) in each iteration is as follows:

$$\Delta w_{ij}^{(t)} = -\text{sign}\left(\frac{\partial E^{(t)}}{\partial w_{ij}}\right) \cdot \Delta_{ij}^{(t)}, \tag{9}$$

where $sign(\cdot)$ is the sign function, $\Delta_{ij}^{(t)} \geq 0$ is the step-size, which is initialized to $\Delta_0$ for all weights. The step-size for each weight is adjusted as follows:

$$\Delta_{ij}^{(t)} = \begin{cases} \xi^+ \cdot \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \xi^- \cdot \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)} & , \text{ otherwise} \end{cases} \tag{10}$$

where $0 < \xi^- < 1 < \xi^+$. To prevent the step-sizes from becoming too large or too small, they are bounded by $\Delta_{\mathrm{m}in} \leq \Delta_{ij} \leq \Delta_{\mathrm{m}ax}$.

One exception must be considered. After the weights are updated, it is necessary to check if the partial derivative changes sign, which indicates that the previous step might be too large and thus a minimum has been missed. In this case, the previous weight change should be retracted:

$$\Delta w^{(t)} = -\Delta_{ij}^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0. \tag{11}$$

Recall that if the weight change is retracted in the $t$-th iteration, the $\partial E^{(t)}/\partial w_{ij}$ should be set to 0.

It is argued that the condition for weight retraction in equation (11) is not always reasonable.[10] The weight change should be retracted only if the partial derivative changes sign and if the approximation error increases. Thus, the weight retraction condition in equation (11) is modified as follows:

$$\Delta w^{(t)} = -\Delta_{ij}^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \textbf{ and if } E^{(t)} > E^{(t-1)}. \tag{12}$$

It has been shown on several benchmark problems that the modified Rprop (termed as Rprop$^+$) exhibits consistent better performance than the Rprop.[10]

### 2.4. *Elitist Non-dominated Sorting and Crowded Tournament Selection*

After mutation and life-time learning, the offspring and the parent populations are combined. Then, a non-domination rank ($r_i$) and a local crowding distance ($d_i$) are assigned to each individual in the combined population as suggested.[6] After that, the crowded tournament selection[6] is implemented. In the crowded tournament selection, two individuals are randomly picked out from the combined population. If individual $A$ has a higher (better) rank than individual $B$, individual $A$ is selected. If they have the same rank, then, the one with a better crowding distance (the one locating in a less crowded area) is selected. Compared to the fitness sharing techniques, the crowded tournament selection gurantees that the one with a better rank is selected. The crowding distance can be calculated either in the parameter or objective space. In this work, the distance is computed in the objective space.

## 3. Selecting Ensemble Members

So far, the size of ensembles is often determined empirically, with a few exceptions.[22,23] A genetic algorithm is used to select a subset of the final population as ensemble members.[22] In another work[23], a genetic programming has been employed to search for an optimal ensemble size.

Selecting a subset from a given number of networks can also be converted into finding out the optimal weight for each candidate network based on a certain criterion. Given $N$ neural networks, the final output of the ensemble can be obtained by averaging the weighted outputs of the ensemble members:

$$y^{EN} = \sum_{k=1}^{N} a^{(k)} y^{(k)}, \tag{13}$$

where $y^{(k)}$ and $a^{(k)}$ are the output and its weight of the $k$-th neural network in the ensemble. Usually, all weights are equally set to $1/N$, and the overall output is known as *simple average*. If the weights are optimized based on a certain criterion, the overall output is then called *weighted average*. Given a set of validation data, the expected error of the weighted output of the ensemble can be calculated by:

$$E^{EN} = \sum_{i=1}^{N} \sum_{j=1}^{N} a^{(i)} a^{(j)} C_{ij}, \tag{14}$$

where $C_{ij}$ is the error correlation matrix between network $i$ and network $j$ in the ensemble:

$$C_{ij} = E[(y_i - y_i^d)(y_j - y_j^d)], \tag{15}$$

where $E(\cdot)$ denotes the mathematical expectation.

It has been shown[17] that there exists an optimal set of weights that minimizes the expected prediction error of the ensemble:

$$a^{(k)} = \frac{\sum_{j=1}^{N} (C_{kj})^{-1}}{\sum_{i=1}^{N} \sum_{j=1}^{N} (C_{ij})^{-1}}, \tag{16}$$

where $1 \leq i, j, k \leq N$.

However, a reliable estimation of the error correlation matrix is not straightforward because the prediction errors of different networks in an ensemble are often strongly correlated. Alternatively, the recursive least-square method can be employed to search for the optimal weights. [22] Other methods have also been proposed to solve this problem. [12,24]

In this investigation, a canonical evolution strategy is employed to find the optimal weights to minimize the expected error in equation 14.

In the multi-objective optimization approach to generating neural network ensemble members, the easiest way is to select all non-dominated solutions found in the optimization as ensemble members. In the following empirical investigations, we compare three cases. In the first case, all non-dominated solutions found in the final population are used to construct an ensemble. In the second case, a well distributed subset of the non-dominated solutions are selected by hand. Finally, the criterion in equation (14) is minimized using an evolution strategy based on a validation data set.

## 4. Case Studies

### 4.1. *Experimental Settings*

The population size of the GA used for evolving neural networks is 100 and the optimization is run for 200 generations. In mutating the weights, the standard deviation of the Gaussian noise is set to 0.05. The weights of the network are initialized randomly in the interval of $[-0.2, 0.2]$ and the maximal number of hidden neurons is set to 10. In the Rprop$^+$ algorithm, the step-sizes are initialized to 0.0125 and bounded between $[0, 50]$ in the adaptation, and $\xi^- = 0.2$, $\xi^+ = 1.2$. Note that a number of parameters needs to be specified in the Rprop$^+$ algorithm, however, the performance of the algorithm is not very sensitive to these values.[10] In our work, we use the default values suggested in reference 10 and 50 iterations are implemented in each life-time learning.

A standard (15,100)-ES has been used to optimize the ensemble weights in equation (13) based on the expected error on the validation data. The initial step-sizes of the evolution strategy are set to 0.0001 and the weights are initialized randomly between 0.005 and 0.01. The weight optimization has been run for 200 generations.

### 4.2. *Results on the Ackley Function*

The simulation study has been first conducted on the 3-dimensional Ackley function.[13] 100 samples are generated randomly between $[-3, 3]$, of which the first 80 samples are used as training data, another 10 data are used as validation data, and the remaining 10 data samples are used as test data.

In the first case, the approximation error and the number of connections described in equation (8) are used as two objectives in evolving the neural network. The non-dominated solutions in the 200-th generation are plotted in Fig. 2.

The most straightforward approach is to use all obtained non-dominated solutions to construct the ensemble. In the final generation, 40 solutions have been found to be non-dominated. The MSE of the best and worst

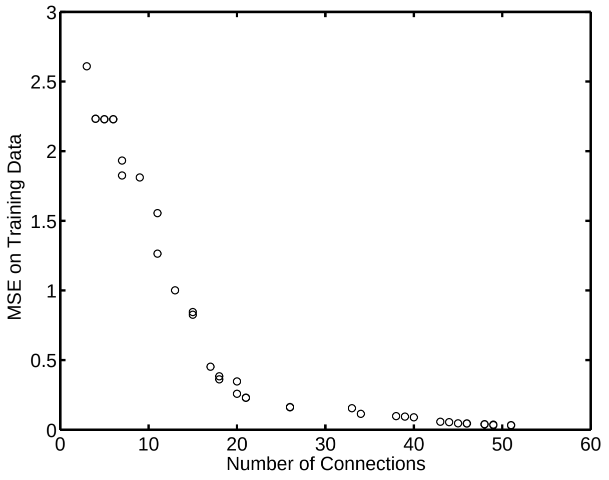

Fig. 2. Non-dominated solutions when number of connections is used as the second objective.

single networks from the 40 solutions, the MSE of the simple average ensemble, and the MSE with the weights being optimized using the algorithm presented in Section 3 are given in Table 1. Notice that in calculating the MSE of the ensemble on the test data, the weights are those optimized on the basis of the validation data.

Table 1. MSE of the ensemble consisting of all 40 non-dominated solutions.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.121 | 2.29 | 0.409 | 0.118 |
| test | 0.348 | 2.07 | 0.179 | 0.361 |

It is suggested that it might be better to use a subset of available neural networks than to use all.[17,22] For this purpose, different strategies have been tried. For example, we can select a "representative" subset from the non-dominated solutions to construct a neural network ensemble. Another possibility is to select the non-dominated solutions whose MSE error on training data is smaller than a specified value, or to select those whose MSE on the validation data is smaller than a given value. Fig. 3 shows the 14 heuristically selected representative solutions (filled circles).

The MSE of the best and worst single networks, the MSE of the ensemble using simple average and weighted average of the 14 representatives on the

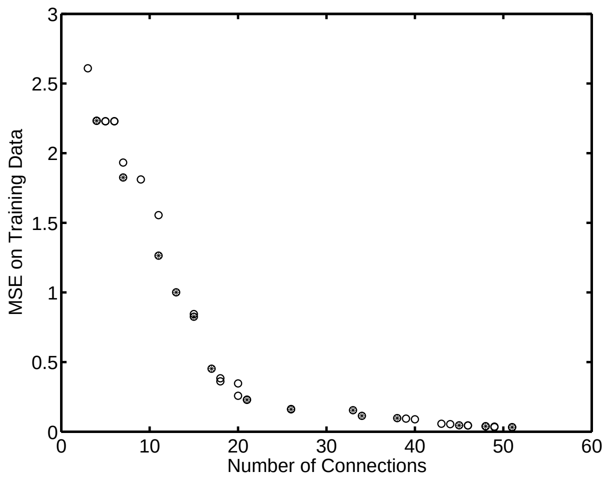

Fig. 3. 14 selected representatives.

validation as well as the test data are shown in Table 2.

Table 2. MSE of the ensemble consisting of 14 heuristically selected members.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.160 | 2.28 | 0.279 | 0.074 |
| test | 0.468 | 2.07 | 0.236 | 0.449 |

Some observations can be made from the results. First, the MSE of the ensemble using simple average of the 14 selected representatives is worse than that using all non-dominated solutions. Second, the ensemble with optimized weights on the basis of the validation data exhibits better performance on the validation data than the one with simple average. Unfortunately, its MSE on the test data is larger than that of the ensemble using simple average. This implies that validation data set and the test data set might not have the same statistical characteristics. In this case, it might be not practical to optimize the weights based on the validation data for predicting unseen data sets.

The results for ensemble members selected according to the MSE on the training and validation data, respectively, are shown in Table 3 and 4.

From Tables 3 and 4, it can be seen that the MSE on the test data of both ensembles are larger than that of the ensemble consisting of the 14 representative networks. Furthermore, good performance on data training

Table 3. MSE of the ensemble consisting of 14 networks whose MSE on the training data is smaller than 0.01.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.57 | 1.09 | 0.84 | 0.50 |
| test | 0.34 | 0.61 | 0.43 | 0.42 |

Table 4. MSE of the ensemble consisting of 12 networks whose MSE on validation data is smaller than 0.50.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.12 | 0.57 | 0.073 | 0.038 |
| test | 0.50 | 1.40 | 0.535 | 0.618 |

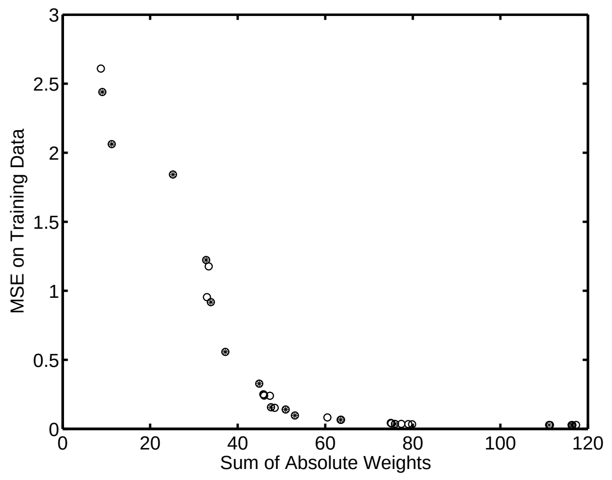

Fig. 4. Non-dominated solutions when the sum of absolute weights is used as the second objective. The shaded circles denotes those selected as a subset for constructing an ensemble.

or validation data does not mean good performance on test data. Optimization of the weights of the ensemble members do not necessarily reduces the error on the test data.

Next, the sum of absolute weights in equation (4) is adopted as the second objective in the evolution. The obtained non-dominated solutions are shown in Fig. 4.

Similar to the above simulations, we calculate the best and worst MSE of a single network, the MSE of the ensemble with simple and weighted average over all the 32 non-dominated solutions, or over a heuristically selected representative subset (the circles filled with a star in Fig. 4), a subset whose MSE on the training data is smaller than 0.1, or a subset whose MSE on the validation data is smaller than 0.5. The results are presented in Tables 5, 6, 7 and 8, respectively.

Table 5. MSE of the ensemble using all 32 non-dominated solutions.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.174 | 1.52 | 0.336 | 0.152 |
| test | 0.637 | 1.91 | 0.491 | 0.558 |

Table 6. MSE of the ensemble consisting of 14 heuristically selected members.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.410 | 1.52 | 0.363 | 0.152 |
| test | 0.637 | 1.91 | 0.361 | 0.453 |

Table 7. MSE of the ensemble consisting of 15 networks whose MSE on the training data is smaller than 0.1.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.460 | 0.62 | 0.524 | 0.460 |
| test | 0.636 | 1.72 | 1.21 | 1.38 |

Table 8. MSE of the ensemble consisting of 9 networks whose MSE on the validation data is smaller than 0.5.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.174 | 0.46 | 0.172 | 0.150 |
| test | 0.770 | 1.46 | 0.560 | 0.570 |

From the above results, it can be seen that the use of ensemble is a reliable way to reduce the prediction error, although the ensemble quality must not be better than the best member in it. The results also suggest that it is still an open question how to properly select an optimal subset from a set of obtained non-dominated solutions to construct an ensemble. Networks with good performance on either training or validation data sets are not necessarily good candidates for the test data set. The optimization algorithm presented in Section 3 is very effective in minimizing the ensemble prediction error on the validation data. However, this does not imply that the MSE on the test data will be reduced too using the optimal weights obtained on the validation data.

Finally, a single objective optimization has been run for 14 times, where the MSE on the training data is used as the fitness function. The individual networks are generated randomly and no interactions between the networks have been considered. In generating the networks, all parameter settings are the same as in the multi-objective case. These 14 neural networks are then used to construct a neural ensemble and the results on validation and test data are presented in Table 9. The results seem worse than those from the ensembles consisting of 14 networks that are generated using multi-objective optimization, as shown in Tables 2 and 6.

Table 9. MSE of the ensemble consisting of 14 networks randomly generated using the single objective optimization.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.270 | 1.79 | 0.320 | 0.220 |
| test | 0.655 | 1.81 | 0.532 | 0.595 |

Finally, a number of non-dominated solutions are obtained using the MSE and the sum of squared weights as two objectives, which are shown in Fig. 5. Simulations have been conducted to study the different methods for selecting ensemble members and very similar results are obtained. Thus, these results will not be presented in detail here.

## 4.3. *Results on the Macky-Glass Function*

In this subsection, neural network ensembles are used to predict the output of the Mackey-Glass series:

$$\dot{x}(t) = \frac{\alpha x(t-\tau)}{1 + x^{10}(t-\tau)} - \beta x(t), \tag{17}$$

where $\alpha = 0.2$, $\beta = 0.1$, $\tau = 17$. The task of the neural ensemble is to predict $x(t+85)$ using $x(t)$, $x(t-6)$, $x(t-12)$, and $x(t-18)$. According to reference 8, 500 samples are generated for training, 250 samples for validation and the another 250 samples for test.

In the 200-th generation, 34 non-dominated solutions have been found, which are illustrated in Fig. 6. All the non-dominated solutions are used to construct an ensemble. The results from the best and the worst single

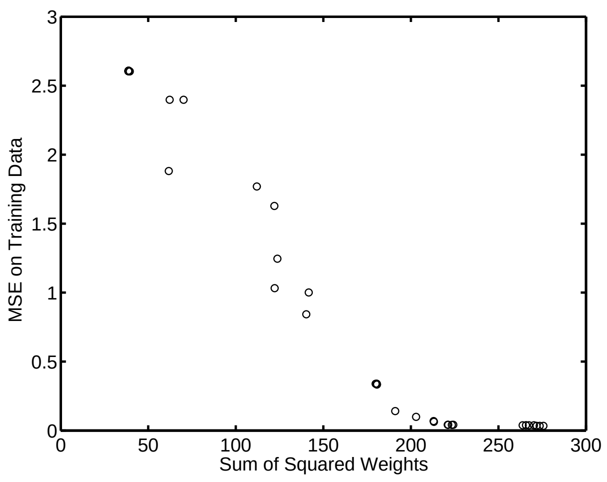

Fig. 5. Non-dominated solutions when the sum of squared weights is used as the second objective.

networks, and those from simple average and weighted average of the ensemble members are provided in Table 10. From these results, we notice first that the performance of the simple average ensemble is better than the worst member, but worse than the best one. Another important factor is that the performance of the ensemble using weighted averaging exhibits better performance than the one with simple averaging not only on validation data, but also on the test data. This indicates that in this example, the validation data are able to reflect the feature of the test data.

Table 10. MSE of the ensemble consisting of all 34 non-dominated solutions.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.0111 | 0.0488 | 0.0134 | 0.0117 |
| test | 0.0097 | 0.0518 | 0.0118 | 0.0104 |

As done in the previous Section, a second ensemble is constructed by selecting 14 representative solutions from the 34 non-dominated solutions, which are the filled circles in Fig. 6. The results of this ensemble are presented in Table 11.

Next, we construct another two ensembles by selecting the networks having the MSE smaller than 0.012 on the training data and on the validation data, respectively. According to this criterion, 6 and 7 networks have

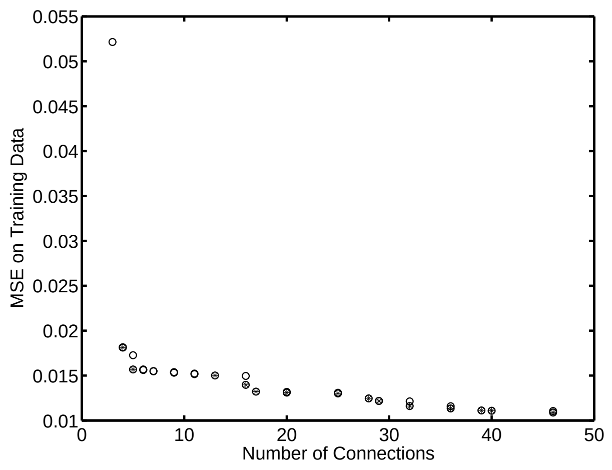

Fig. 6. Non-dominated solutions when the number of connections is used as the second objective. The filled circles are the representatives.

Table 11. MSE of the ensemble consisting of 14 representatives.

| | single best | single worst | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.0112 | 0.0488 | 0.0129 | 0.0112 |
| test | 0.0097 | 0.0518 | 0.0111 | 0.0099 |

been selected and the results are given in Tables 12 and 13.

Table 12. MSE of the ensemble consisting of 6 networks whose MSE on the training data is smaller than 0.012.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.0112 | 0.0116 | 0.0113 | 0.0112 |
| test | 0.0097 | 0.0105 | 0.0102 | 0.0097 |

Table 13. MSE of the ensemble consisting of 7 networks whose MSE on the validation data is smaller than 0.012.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.0112 | 0.0117 | 0.0114 | 0.0112 |
| test | 0.0097 | 0.0109 | 0.0102 | 0.0097 |

From these results, it can be seen that no big differences exist between the various methods for selecting ensemble members. Besides, the ensemble with weighted average shows consistent better performance than the one using simple average. However, the performance of the best single network is better than that of the ensemble with simple average, and thus the performance of the ensemble with optimized weighted average is almost the same as that of the single best, which makes sense. Nevertheless, the ensembles with simple average or optimized weighted average show consistently better performance than that of the single worst network. Furthermore, ensembles consisting of the selected networks based on training or validation error are better than those consisting of all or a heuristically selected subset of the non-dominated solutions. This implies that no significant overfitting occurs during the training.

Finally, 14 networks are generated randomly using single objective optimization. The results of this ensemble are shown in Table 14. It can be seen that the performance of the ensemble is better than that of the single worst network but worse than that of the single best. Obviously, diversity does not help to improve the performance of the ensemble if no significant overfitting occurs.

Table 14. MSE of the ensemble consisting of 14 randomly generated networks.

| | best single | worst single | simple average | weighted average |
|---|---|---|---|---|
| validation | 0.01 | 0.0143 | 0.0115 | 0.01 |
| test | 0.0095 | 0.0133 | 0.0111 | 0.0095 |

Simulations have also been conducted when the sum of absolute weights or the sum of squared weights serves as the second objective on the Macky-Glass series data. The non-dominated solutions from these optimization runs are plotted in Fig. 7 and Fig. 8, respectively. Notice that non-dominated solutions whose MSE on the training data is larger than 0.05 are missing from the 200-th generation. This does not mean that such solutions do not exist. Rather, this is due to the randomness of multi-objective optimization algorithm introduced by the crowded tournament selection. As discussed in reference 5, such randomness occurs when the number of non-dominated solutions in the combined population is larger than the population size.

The prediction results of the ensembles constructed from these solutions are omitted here because they are very similar to those presented above when the number of connections is used as the second objective.

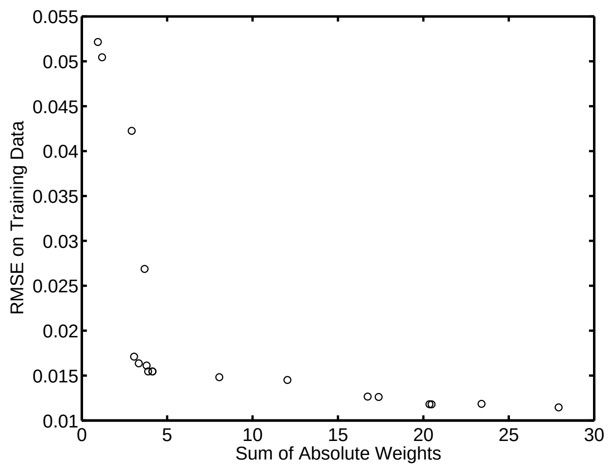

Fig. 7. Non-dominated solutions when the sum of absolute weights is used as the second objective.

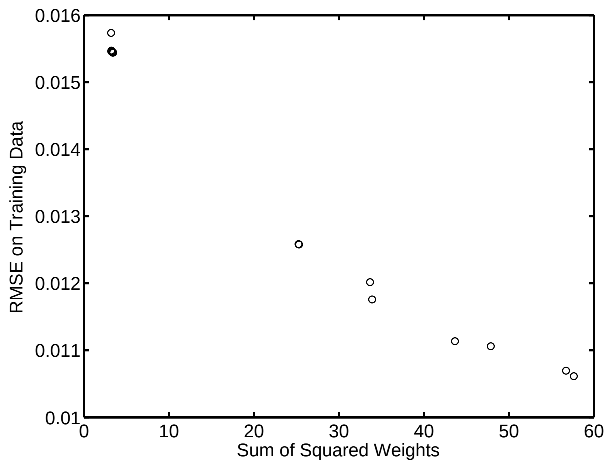

Fig. 8. Non-dominated solutions when the sum of squared weights is used as the second objective.

## 5. Discussions and Conclusions

Approximation accuracy and complexity have been used as two objectives to generate neural networks for constructing ensembles. In the algorithm, *ad hoc* mutations such as node/connection addition and deletion are employed without crossover. The Rprop learning algorithm is adopted in life-time

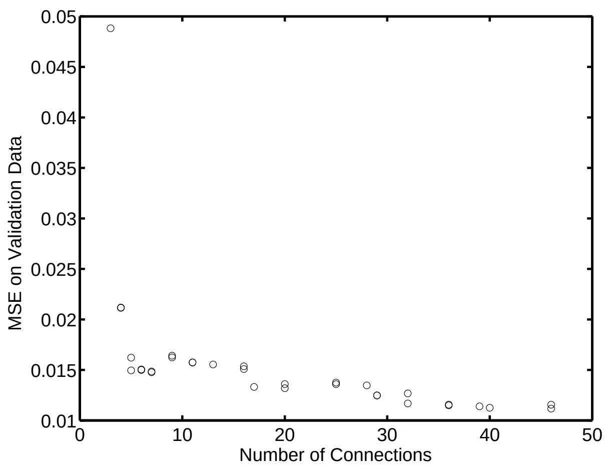

Fig. 9. Trade-off between the MSE on the validation data and the complexity of the neural networks.

learning and a Lamarckian inheritance has been implemented. In selection, the elitist non-dominated sorting and the crowded tournament selection techniques have been used. This algorithm has proved to be effective in generating neural networks trading off between accuracy and complexity through two test problems.

Whereas it is able to improve the performance of the ensemble whose members have a trade-off between complexity and accuracy if overfitting occurs, no performance improvement can be expected by use of network ensembles when the networks do not overfit the training data. In fact, it seems that in this case, the network that has the best accuracy on the training data also exhibits the best performance on the test data. Thus, ensembles with different degrees of accuracy will degrade its performance. Note that the proposed method for individual network training belongs to the simultaneous approach. Due to the explicit trade-off between the complexity and accuracy, the individuals in a population are competitive, which is harmful to the performance of the ensemble if the test data has the same feature as the training data. This can easily be observed by plotting the relationship between the MSE on the validation data and the complexity, as shown in Fig. 9. It can be seen from the figure that the higher the complexity of the network is, the better.

As argued in reference[11], it is equally important that the ensemble members cooperate with each other. To this end, the concept of cooperative co-evolution[18] could play a significant role in generating ensemble members. This will be our next research direction.

**Acknowledgments**

The authors would like to thank Edgar Körner and Andreas Richter for their kind support.

## References

1. H.A. Abbass. Speeding up back-propagation using multiobjective evolutionary algorithms. *Neural Computation*, 15(11):2705–2726, 2003.
2. D.H. Ackley and M.L. Littman. A case for lamarckian evolution. In C.G. Langton, editor, *Artificial Life*, volume 3, pages 3–10. Addison–Wesley, Reading, Mass., 1994.
3. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.
4. R. de A. Teixeira, A.P. Braga, R. H.C. Takahashi, and R. R. Saldanha. Improving generalization of MLPs with multi-objective optimization. *Neurocomputing*, 35:189–194, 2000.
5. K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, 2001.
6. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature*, volume VI, pages 849–858, 2000.
7. N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez. Multi-objective cooperative co-evolution of artificial neural networks (multi-objective cooperative networks). *Neural Networks*, 15:1259–1278, 2003.
8. E. Hartmann and J.D. Keeler. Predicting the future: Advantages of semilocal units. *Neural Computation*, 3(4):566–578, 1991.
9. M. Hüsken, J. E. Gayko, and B. Sendhoff. Optimization for problem classes – Neural networks that learn to learn. In Xin Yao and David B. Fogel, editors, *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN 2000)*, pages 98–109. IEEE Press, 2000.
10. C. Igel and M. Hüsken. Improving the Rprop learning algorithm. In *Proceedings of the 2nd ICSC International Symposium on Neural Computation*, pages 115–121, 2000.
11. Md. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training copperative neural network ensembles. *IEEE Trasactions on Neural Networks*, 14(4):820–834, 2003.
12. D. Jimenez. Dynamically weighted ensemble neural networks for classification. In *Proceedings of International Joint Conference on Neural Networks*, pages 753–756, Anchorage, 1998. IEEE Press.
13. Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.

14. Y. Liu and X. Yao. Negatively correlated neural networks can produce best ensemble. *Australian Journal of Intelligent Information Processing System*, 4(3–4):176–185, 1997.
15. Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.
16. D.W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural network ensemble. In *Advances in Neural Information Processing Systems*, volume 8, pages 535–541, Cambridge, MA, 1996. MIT Press.
17. M.P. Perrone and L.N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman & Hall, London, 1993.
18. M.A. Potter and K.A. De Jong. Coperative coevolution: An architechture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
19. B. E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3–4):373–384, 1996.
20. A.J.C. Sharkey and N. E. Sharkey. Diversity, selection and ensembles of artificial neural nets. In *Proceedings of Third International Conference on Neural Networks and their Applications*, pages 205–212, March 1997.
21. S. Sigurdsson, J. Larsen, and L. K. Hansen. On comparison of adaptive regularization methods. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, volume 10, pages 221–230, 2000.
22. X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part B:Cybernetics*, 28(3):417–425, 1998.
23. B.-T. Zhang and J.G. Joung. Building optimal committee of genetic programs. In *Parallel Problem Solving from Nature*, volume VI, pages 231–240. Springer, 2000.
24. Z.-H. Zhou, J.-X. Wu, Y. Jiang, and S.-F. Chen. Genetic algorithm based selective neural network ensemble. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 797–802, Seattle, 2001. Morgan Kaufmann.