Published in: Proceedings of 2004 Congress on Evolutionary Computation. pp.1-8, Portland, June 19-23, 2004

Neural Network Regularization and Ensembling Using Multi-objective Evolutionary Algorithms

Yaochu Jin
Honda Research Institute Europe
Carl-Legien-Str.30
63073 Offenbach, GERMANY
Email: yaochu.jin@honda-ri.de

Tatsuya Okabe
Honda Research Institute Europe
Carl-Legien-Str.30
63073 Offenbach, GERMANY

Bernhard Sendhoff
Honda Research Institute Europe
Carl-Legien-Str.30
63073 Offenbach, GERMANY

***Abstract*— Regularization is an essential technique to improve generalization of neural networks. Traditionally, regularization is conduced by including an additional term in the cost function of a learning algorithm. One main drawback of these regularization techniques is that a hyperparameter that determines to which extension the regularization influences the learning algorithm must be determined beforehand.**

This paper addresses the neural network regularization problem from a multi-objective optimization point of view. During the optimization, both structure and parameters of the neural network will be optimized. A slightly modified version of two multi-objective optimization algorithms, the dynamic weighted aggregation (DWA) method and the elitist non-dominated sorting genetic algorithm (NSGA-II) are used and compared. An evolutionary multi-objective approach to neural network regularization has a number of advantages compared to the traditional methods. First, a number of models with a spectrum of model complexity can be obtained in one optimization run instead of only one single solution. Second, an efficient new regularization term can be introduced, which is not applicable to gradient-based learning algorithms.

As a natural by-product of the multi-objective optimization approach to neural network regularization, neural network ensembles can be easily constructed using the obtained networks with different levels of model complexity. Thus, the model complexity of the ensemble can be adjusted by adjusting the weight of each member network in the ensemble. Simulations are carried out on a test function to illustrate the feasibility of the proposed ideas.

I. Introduction

One of the most essential issues in neural network training is to improve generalization of the neural network models. In other words, neural network models should not only have a high approximation accuracy on the data samples used in the training, but also show good performance on unseen data. A class of commonly used techniques for improving generalization of neural networks is known as regularization, which aims to prevent the learning algorithm from over-fitting the training data. Several regularization techniques have been suggested in the literature, such as early stopping, weight decay and curvature-driven smoothing [1].

A popular approach to regularization is to include an additional term in the cost function of learning algorithms, which penalizes overly high model complexity. A hyperparameter, known as the regularization parameter determines to what extent the regularization will influence the learning algorithm. That is to say, this parameter will determine the model complexity of the trained neural network. The larger the parameter is, the higher the penalty will be on the model complexity. However, it is usually not trivial to determine a suitable model complexity that is optimal for the problem at hand. Very often, this has been done by minimizing an estimated generalization error [2]. To estimate the generalization error, it is usually necessary to split the available data into a training data set and a validation data set. Unfortunately, selecting models by minimizing an estimate of the generalization error is not always consistent, as it has been mentioned in [3].

From the multi-objective optimization point of view, including 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 first been reported in [4]. In that paper, a variance of the ϵ-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 in [5], 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.

This paper presents a method for regularizing neural networks using multi-objective evolutionary algorithms. Thus, no hyperparameter needs to be specified beforehand. A number of Pareto-optimal neural networks, instead of one single network will be generated in the evolutionary optimization. Two existing multi-objective algorithms, the dynamic weighted aggregation (DWA) method [6], [7] and the elitist non-dominated sorting genetic algorithms (NSGA-II) [8] are slightly modified to adapt them to the structure and parameter optimization of neural networks. Life-time learning of the parameters, i.e., the weights of the neural network is carried out using a fast and robust learning algorithm known as the RProp$^+$ algorithm [9]. The weights trained by the RProp$^+$ algorithm are inherited by the individuals, which is often known as Lamarkian evolution [10]. Since evolutionary algorithms are used in optimization, a more direct model complexity measure, the number of connections in the network can be used as the regularization

term.

The performance of the DWA-based and the NSGA-II based multi-objective optimization algorithms are compared on the Ackley function [11]. It is shown that the non-dominated solutions in the final generation are often dominated by other solutions obtained in the history of optimization, although NSGA-II is basically an elitist algorithm. This becomes more serious when the population size is small. This indicates that the population may not be able to maintain the found non-dominated solutions. On the other hand, it is shown DWA works quite well for neural network structure and parameter optimization, although it has been often used for continuous parameter optimization problems. However, the accuracy of the DWA is worse than that of the NSGA-II if a large population size is used.

Neural network ensembles can be constructed straightforwardly using the non-dominated solutions generated in the multi-objective optimization. Since they are quite diversified in structure, these networks are natural candidates for constructing network ensembles. Illustrative examples on constructing neural network ensembles from the obtained non-dominated neural networks are also provided.

II. Neural Network Regularization

The most common error 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. In this work, we consider multi-layer perceptron (MLP) neural networks with one output. Refer to [1] for other error functions, such as the Minkowski error or cross-entropy.

It has been found that neural networks can often overfit 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. To improve generalization of neural networks, regularization techniques are often adopted by including an additional term in the error function:

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

where λ is a hyperparameter that controls the strength of the regularization and Ω is known as the regularizer. A most popular regularization method is known as weight decay:

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

where k is an index summing up all weights.

One weakness of the weight decay method is that it is not able to drive small irrelevant weights to zero, which may result in many small weights [12]. The following regularization term has been proposed to address this problem [12]:

$$\Omega = \sum_{i} |w_i|. \tag{4}$$

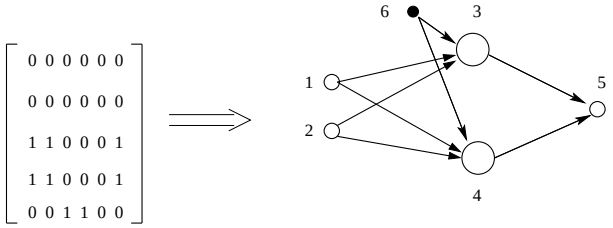


Fig. 1. A connection matrix and the corresponding network structure.

This regularization was used for structure learning, because it is able to drive irrelevant weights to zero.

Both regularization terms in equations (3) and (4) have also been studied from the Bayesian learning point of view, which are known as the Gaussian regularizer and the Laplace regularizer, respectively.

III. Evolutionary Learning and Regularization

A. 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 MLP 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 the 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.

B. Multi-objective Optimization Formulation of Regularization

It is quite straightforward to see that the neural network regularization 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 Ω is one of the regularization terms defined in equation (3) or (4).

Since evolutionary algorithms are used to implement regularized learning of neural networks, a new and more direct index for measuring complexity of neural networks can be employed, which is the number of connections in the neural network:

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

where c_{ij} equals 1 if there is connection from neuron j to neuron i, and 0 if not. Obviously, the smaller the number of connections in a network is, the less complex the network. Note that this regularizer is well suitable for evolutionary optimization although it is not applicable to gradient-based learning algorithms due to its discrete nature. We term this *evolutionary regularizer* for convenience.

C. Mutation and Life-time 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 [10]. A Gaussian-type mutation is applied to 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 [9] has been employed to train the weights. This can be seen as a kind of 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 to the chromosome, which is known as the Lamarkian type of inheritance.

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 than other gradient-based learning algorithms.

Let w_{ij} denotes the weight connecting neuron j and neuron i, then the change of the weight (Δw_{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 Δ_0 for all weights. The step-size for each weight is adjusted as follows:

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

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

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.

In reference [9], it is argued that the condition for weight retraction in equation (11) is not always reasonable. 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:

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

It has been shown on several benchmark problems in [9] that the modified Rprop (termed as Rprop$^+$ in [9]) exhibits consistent better performance than the Rprop algorithm.

D. Fitness assignment and selection: DWA versus NSGA-II

The two multi-objective optimization algorithms are the same except that different strategies in fitness assignment and selection have been adopted. In the first algorithm, the two objectives are aggregated into one single fitness function, as suggested in [6]:

$$F = \eta E + (1 - \eta)\Omega. \tag{13}$$

As the evolution proceeds, the weight η is gradually changed from 1 to 0. In this way, a number of Pareto-optimal solutions can be obtained. Since the DWA method reduces a multi-objective optimization problem to a dynamic single objective optimization problem, the tournament selection method used in [9] can directly be adopted.

In the second algorithm, the fitness assignment and selection proposed in NSGA-II [8] is employed. At first, the off-spring 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. After that, the crowded tournament selection 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, the one with a better crowding distance (the one locating in a less crowded area) is selected. Compared to fitness sharing techniques, the crowded tournament selection guarantees that always the individual 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.

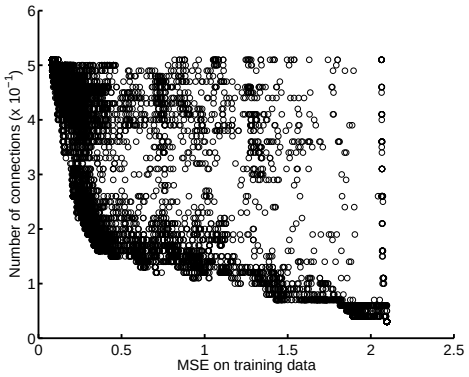


Fig. 2. All solutions found using the DWA method. Population size is 100.

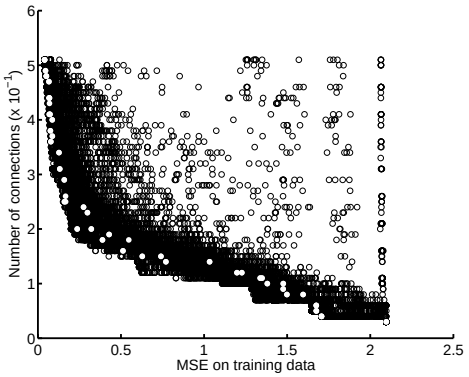


Fig. 3. All solutions found using the NSGA-II method. The solutions maintained in the population of the final generation are denoted using circles. Population size is 100.

IV. Illustrative Examples on Regularization

To illustrate the feasibility to carry out neural network regularization using multi-objective optimization, simulation studies have been carried out on a three-dimensional Ackley function. 100 data samples have been generated, of which 80 samples are used for training and the remaining 20 data samples for test. The population size of the algorithms used for evolving neural networks is 100 and the optimization is run for 200 generations, wherever not explicitly explained. 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 [9]. In our work, we use the default values recommended in [9] and 50 iterations are implemented in each life-time learning.

A. Comparison of DWA and NSGA-II

We first compare the performance of the algorithms based on DWA and NSGA-II, respectively, using the evolutionary regularizer. The results are shown in Fig. 2 and Fig. 3. Note that an archive is needed in DWA to store the non-dominated solutions.

It can be found that the DWA works quite well although it is not a purely parameter optimization problem. It can be seen from Fig. 2 that the DWA itself is able to approximate the Pareto front properly, and the performance of the final non-dominated solutions largely depends on the archiving method. It can also be found that although the population size is 100, the number of Pareto-optimal solutions maintained in the population of the final generation of the NSGA-II algorithm is relatively small. To see if the NSGA-II is able to maintain the found non-dominated solutions, we also show all solutions found by the NSGA-II algorithm in Fig. 3. It can be seen that the solutions contained in the population of the final generation are not really non-dominated, if we look at all solutions found during the search. This indicates that archiving of the found non-dominated solutions might be necessary, despite that an elitist selection strategy has been used in the NSGA-II algorithm.

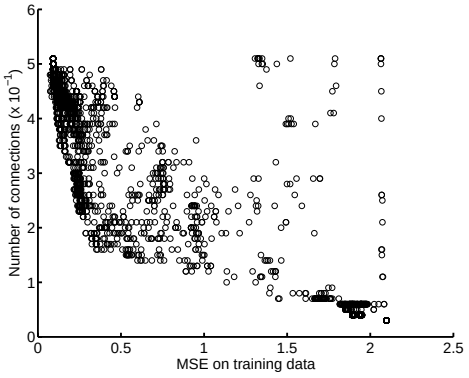


Fig. 4. All solutions found using the DWA method when the population size is 15.

Next, let us compare the two algorithms when a smaller population size is used. Figure 4 shows all solutions obtained from the DWA. It is obvious that the quality of the solutions degrades, when the population size decreases. Similarly, the performance of the NSGA-II also degrades, particularly in terms of the non-dominated solutions maintained in the population of the final generation, refer to Fig. 5.

B. Training against Validation Error

The necessity of regularization is based on the assumption that the learning algorithm could over-fit the training data, which leads to bad performance on validation data. This could be avoided by controlling the model complexity of the network during learning. To verify this assumption empirically, the relationship between model complexity and validation error is shown in Figures 6, 7 and 8, whereas the evolutionary, the Gaussian and the Laplace regularizers are used, respectively.

From these figures, we see that over-fitting does occur when the neural network is overly complex. This trend is relatively clear when the evolutionary and the Laplace regularizers are

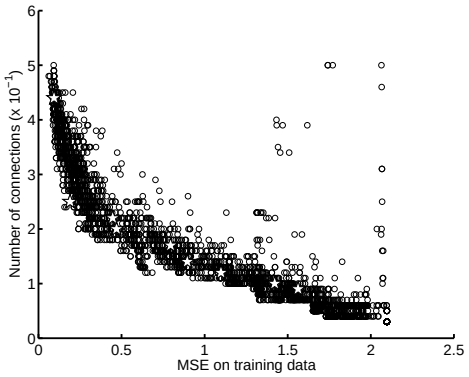


Fig. 5. All solutions found in the search using the NSGA-II method. The solutions maintained in the population of the final generation are denoted using stars. Population size is 15.

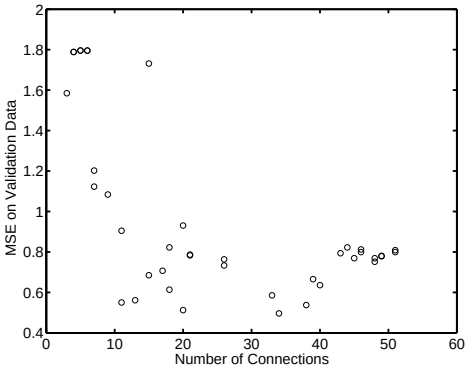


Fig. 6. Number of connections versus validation error.

used. In the Gaussian regularizer case, it seems that the higher the complexity is, the smaller the approximation error on the validation data. However, we also notice that a model with a medium complexity turns out to be the best on the validation data.

A general conclusion from these results is that the relationship between model complexity and generalization is not as simple as we might imagine. This is why an optimal early stopping is often difficult.

C. Gaussian or Laplace Regularizer

When gradient-based learning algorithms are employed for regularization, the Laplace regularizer is usually believed to be better than a Gaussian regularizer in that the Laplace regularizer is able to drive irrelevant weights to zero. In this way, "structural learning" is realized with the help of the Laplace regularizer [12]. In this subsection, we show that there is no big difference between the Gaussian and the Laplace regularizers in terms of their ability to realize structural learning, when evolutionary algorithms are used as an optimizer. From Figures 9 and 10, we can see that they show similar relationship between the sum of squared or absolute weights and the number of connections. In other words, even when the Gaussian regularizer is used, the number of connections can also be reduced to the minimum when the sum of squared

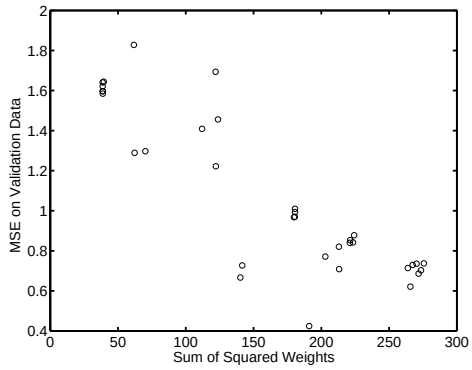


Fig. 7. Sum of squared weights versus validation error.

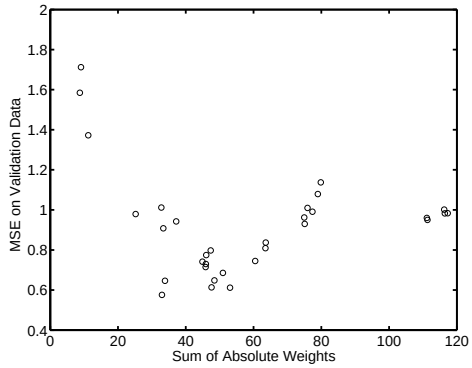


Fig. 8. Sum of absolute weights versus validation error.

weights is minimized. It does not result in many small weights as when gradient-based learning algorithms are used.

V. Constructing Ensembles from Regularized Networks

Traditionally, individual neural networks for an ensemble can be trained either independently, sequentially and simultaneously [13]. In the first case, neural networks are generated separately and no interaction between the networks will be taken into account during 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.

It is well recognized that diversity plays an important role in constructing neural network ensembles [1]. Basically, diversity of ensemble members can be achieved or enhanced by using various initial random weights, varying the network architecture, employing different training algorithms or supplying different training data[14]. 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.

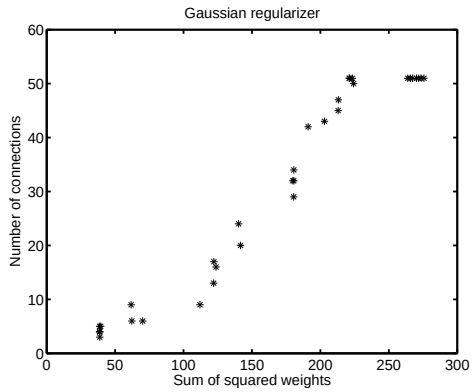


Fig. 9. Relationship between the number of connections and the sum of squared weights.

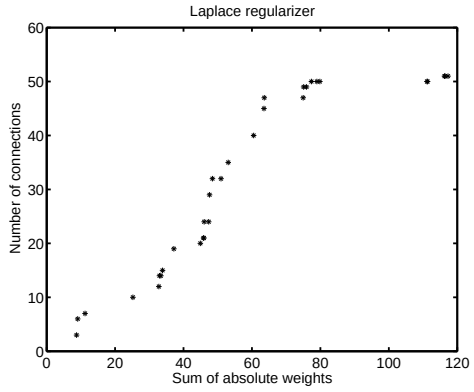


Fig. 10. Relationship between the number of connections and the sum of absolute weights.

In contrast to the above-mentioned methods where diversity is achieved 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 [15], degree of decorrelation [16], or degree of negative correlation [17], [18] between the output of the candidate networks.

In the multi-objective approach to neural network regularization, a number of neural networks can be obtained. Since the structure of the neural network varies from one to another, they can be very good candidates for constructing neural network ensembles. The question is, which networks should be used and how they should be combined.

A straightforward idea is to use all non-dominated solutions obtained by the multi-objective optimizer. Alternatively, a subset of the non-dominated solutions that is well distributed could be used as representatives.

To select a subset from a given number of networks can also be regarded as to find out the optimal weight for each candidate network based 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:

$$y^{EN} = \sum_{k=1}^{N} a^{(k)} y^{(k)}, \tag{14}$$

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:

$$E^{EN} = \sum_{i=1}^{N} \sum_{j=1}^{N} a^{(i)} a^{(j)} C_{ij}, \tag{15}$$

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

$$C_{ij} = E[(y_i - y_i^d)(y_j - y_j^d)], \tag{16}$$

where $E[\cdot]$ denotes the mathematical expectation.

It has been shown [19] that there exists an optimal set of weights that minimizes the expected prediction error of the ensemble:

$$a^{(k)} = \frac{\sum_{j=1}^{N} (C_{kj})^{-1}}{\sum_{i=1}^{N} \sum_{j=1}^{N} (C_{ij})^{-1}}, \tag{17}$$

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 [20], where a genetic algorithm is also used to select a subset of the solutions for constructing ensembles by minimizing the training error. Other methods have also been proposed to minimize the validation error using a genetic algorithm [21].

In this investigation, a canonical evolution strategy is employed to find the optimal weights to minimize the expected error in equation 15.

VI. Illustrative Examples on Ensemble Construction

A standard (15,100)-ES has been used to optimize the ensemble weights in equation (14) based on the expected error on the validation data. A standard evolution strategy can be briefly described as follows:

$$\sigma_i(t) = \sigma_i(t-1)\exp(\tau' z)\exp(\tau z_i) \tag{18}$$

$$\mathbf{x}(t) = \mathbf{x}(t-1) + \tilde{\mathbf{z}} \tag{19}$$

where $\mathbf{x}$ is an n-dimensional parameter vector to be optimized, $\tilde{\mathbf{z}}$ is an n-dimensional random number vector with $\tilde{\mathbf{z}} \sim N(\mathbf{0}, \sigma(t)^2)$, z and z_i are normally distributed random numbers with $z, z_i \sim N(0,1)$. Parameters τ, τ' and σ_i are the

strategy parameters, where σ_i is mutated as in equation (18) and τ, τ' are constants as follows:

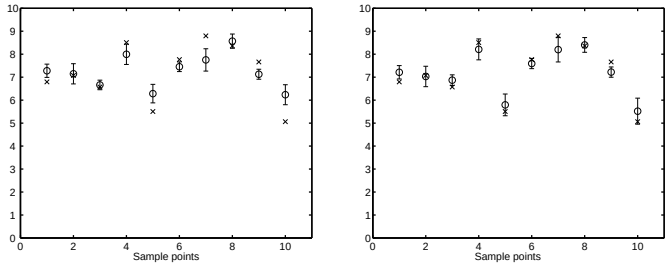
$$\tau = \left(\sqrt{2\sqrt{n}}\right)^{-1} \; ; \; \tau' = \left(\sqrt{2n}\right)^{-1} \tag{20}$$

The initial step-sizes of the evolution strategy are set to 0.00001 and the weights are initialized randomly between 0.005 and 0.01. The weight optimization has been run for 200 generations. The non-dominated solutions that are used in the simulations in this section are those obtained using the NSGA-II algorithm as shown in Fig. 3.

A. Use All Found Non-dominated Solutions

The most straightforward approach is to use all obtained non-dominated solutions to construct an ensemble. In the final generation of the optimization, 40 solutions have been found to be non-dominated, as shown in Fig. 3.

Fig. 11 (a) shows the average prediction of 10 validation samples from the 40 networks with the standard deviation, whereas the results using weighted average are given in Fig. 11 (b). These 10 samples have not been used during the training, however, they are used to estimate the expected prediction error for optimizing the weights of the ensemble members in equation (14). The optimization process (change of the expected prediction error) is shown in Fig. 12. Obviously, the prediction performance of the ensemble has been improved significantly on the validation data using the optimized weights.



(a) (b)

Fig. 11. Prediction of the validation data using (a) simple average and (b) weighted average.

However, the weights optimized by minimizing the expected prediction error on the validation data are not necessarily optimal for test data. This can be seen from the results shown in Fig. 13.

The MSE of the best and worst single networks from the 40 solutions, the MSE of the simple average ensemble, and the MSE with the weights being optimized by minimizing the cost function defined in equation (15) are given in Table I. 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.

Some remarks can be made on the results. On the one hand, it can be seen that the simple average can be better or worse than the simple worst, although it is not necessarily better

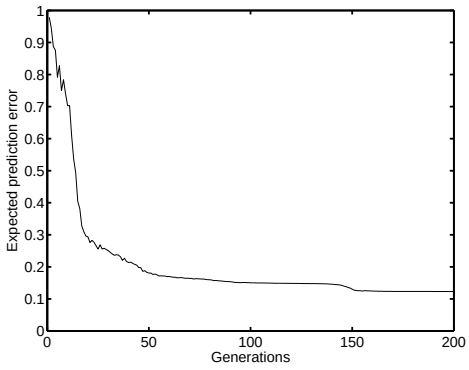
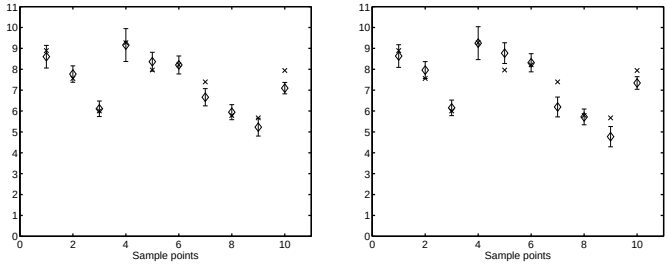


Fig. 12. Optimization of ensemble weights by minimizing the expected prediction error.



(a) (b)

Fig. 13. Prediction of the test data using (a) simple average and (b) weighted average.

than the simple best. Besides, optimization of the ensemble weights can improve the performance significantly on the validation data, which however does not necessarily imply a better performance on the test data. The reason is that the statistics of the validation data may be somewhat different from that of the test data. Thus, if it is unclear whether the validation data are able to represent the full statistics of the test data, the use of simple average is still recommended.

B. Use a Subset of the Non-dominated Solutions

It is suggested in [19], [20] that it might be better to use a subset of available neural networks than to use all. For this purpose, we select a "representative" subset from the non-dominated solutions to construct a neural network ensemble. Fig. 14 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 validation as well as the test data are shown in Table II. Due to space limit, the details of the prediction will not be presented.

It seems that in this example, the MSE of the ensemble using simple average of the 14 selected representatives is worse than that using all non-dominated solutions. However, this conclusion should not be generalized to other cases.

TABLE I

MSE OF THE ENSEMBLE CONSISTING OF ALL 40 NON-DOMINATED SOLUTIONS.

	best	worst	average	weighted average
validation	0.121	2.285	0.401	0.123
test	0.348	2.069	0.179	0.353

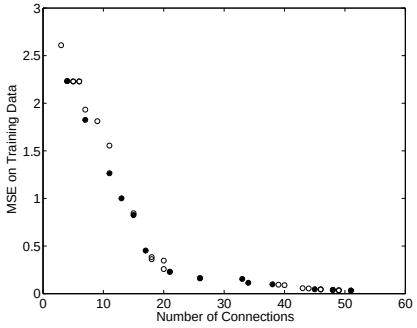


Fig. 14. 14 selected representatives. The selected networks are marked with filled circles.

VII. CONCLUSIONS

The main purpose of the paper is to show that neural network regularization can be addressed from the multi-objective optimization point of view. This approach exhibits two advantages over traditional regularization techniques. First, a number of neural networks of a spectrum of model complexity instead of one single neural network can be obtained in one optimization run. Second, a new and more direct regularizer can be used, which can be attributed to the fact that evolutionary algorithms instead of gradient-based learning algorithms are used. Finally, the obtained neural networks, which are structurally diversified, are ready to be used for constructing neural network ensembles. These ideas are demonstrated to work successfully on a test problem using the DWA and NSGA-II algorithms.

Further research effort is to compare the proposed method for generating neural network ensembles with the existing methods, particularly when neural network ensembles are used for reducing fitness evaluations in evolutionary optimization [22], [23]. Besides, learning algorithms for incorporating a priori knowledge into neural networks [24] can also be further investigated using the proposed method.

ACKNOWLEDGMENT

The authors would like to thank E. Körner for his support.

REFERENCES

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[2] J. Larsen, C. Svarer, L. N. Anderson, and L. K. Hansen. *Adaptive regularization in neural network modeling*, chapter 5, pages 113–132. Springer, 1998.

[3] J. Shao and D. Tu, editors. *The Jackknife and Bootstrap*. Springer, 1996.

[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.

TABLE II

MSE OF THE ENSEMBLE CONSISTING OF 14 HEURISTICALLY SELECTED MEMBERS.

	best	worst	average	weighted average
validation	0.160	2.28	0.279	0.074
test	0.468	2.07	0.236	0.449

[5] H.A. Abbass. Speeding up back-propagation using multiobjective evolutionary algorithms. *Neural Computation*, 15(11):2705–2726, 2003.

[6] Y. Jin, T. Okabe, and B. Sendhoff. Adapting weighted aggregation for multi-objective evolution strategies. In *Proceedings of The 1st Int. Conf. on Evolutionary Multi-Criterion Optimization*, pages 96–110, Berlin, 2001. Springer.

[7] Y. Jin, M. Olhofer, and B. Sendhoff. Evolutionary dynamic weighted aggregation for multi-objective optimization: Why does it work and how? In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1042–1049, San Francisco, CA, 2001.

[8] 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.

[9] 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.

[10] 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.

[11] D. H. Ackley. *A connectionist machine for genetic hillclimbing*. Kluwer, Boston, 1987.

[12] R.D. Reed and R.J. Marks II. *Neural Smithing*. The MIT Press, 1999.

[13] Md. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Trasactions on Neural Networks*, 14(4):820–834, 2003.

[14] 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.

[15] 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.

[16] B. E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3–4):373–384, 1996.

[17] 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.

[18] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[19] M.P. Perrone and L.N. Cooper. When networks disgree: 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.

[20] 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.

[21] 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.

[22] 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.

[23] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural network ensembles. In *Genetic and Evolutionary Computation Conference*, Seattle, 2004. Springer. Acceped.

[24] Y. Jin and B. Sendhoff. Knowledge incorporation into neural networks from fuzzy rules. *Neural Processing Letters*, 10(3):231–242, 1999.