Simultaneous generation of accurate and interpretable neural network classifiers

Yaochu Jin, Bernhard Sendhoff, Edgar Körner

2006

Simultaneous Generation of Accurate and Interpretable Neural Network Classifiers

Yaochu Jin and Bernhard Sendhoff and Edgar Körner

Honda Research Institute Europe
Carl-Legien-Str. 30
63073 Offenbach/Main, Germany
yaochu.jin@honda-ri.de

Summary. Generating machine learning models is inherently a multi-objective optimization problem. Two most common objectives are accuracy and interpretability, which are very likely conflicting with each other. While in most cases we are interested only in the model accuracy, interpretability of the model becomes the major concern if the model is used for data mining or if the model is applied to critical applications. In this chapter, we present a method for simultaneously generating accurate and interpretable neural network models for classification using an evolutionary multi-objective optimization algorithm. Lifetime learning is embedded to fine-tune the weights in the evolution that mutates the structure and weights of the neural networks. The efficiency of Baldwin effect and Lamarckian evolution are compared. It is found that the Lamarckian evolution outperforms the Baldwin effect in evolutionary multi-objective optimization of neural networks. Simulation results on two benchmark problems demonstrate that the evolutionary multi-objective approach is able to generate both accurate and understandable neural network models, which can be used for different purpose.

1 Introduction

Artificial neural networks are linear or nonlinear systems that encode information with connections and units in a distributed manner, which are more or less of biological plausibility. Over the past two decades, various artificial neural networks have successfully been applied to solve a wide range of tasks, such as regression and classification, among many others.

Researchers and practitioners are mainly concerned with the accuracy of neural networks. In other words, neural networks should exhibit high prediction or classification accuracy on both seen and unseen data. To improve the accuracy of neural networks, many efficient learning algorithms have been developed [8, 37].

Nevertheless, there are also many cases in which it becomes essential that human users are able to understand the knowledge that the neural network has learned from the training data. However, conventional learning algorithms do not take the interpretability of neural networks into account and thus the knowledge acquired by neural networks is often not transparent to human users. To address this problem, it is often necessary to extract symbolic or fuzzy rules from trained neural networks [7, 15, 24, 25]. A common drawback of most existing rule extraction method is that the rules are extracted after the neural network has been trained, which incurs additional computational costs.

This chapter presents a method for generating accurate and interpretable neural models simultaneously using the multi-objective optimization approach [28]. A number of neural networks that concentrate on accuracy and interpretability to a different degree will be generated using a multi-objective evolutionary algorithm combined with life-time learning, where accuracy and complexity serve as two conflicting objectives.

It has been shown that evolutionary multi-objective algorithms are well suited for and very powerful in obtaining a set of Pareto-optimal solutions in one single run of optimization [11, 12].

The idea of introducing Pareto-optimality to deal with multiple objectives in constructing neural networks can be traced back to the middle of 1990's [29]. In this work, two objectives, namely, the mean squared error and the number of hidden neurons are considered. In another work [39], the error on training data and the norm of the weights of neural networks are minimized.

The successful development in evolutionary multi-objective optimization [11, 12] has encouraged many researchers to address the conflicting, multiple objectives in machine learning using multi-objective evolutionary algorithms [2, 3, 4, 10, 16, 26, 27, 30, 32, 41]. Most of the work focuses on improving the accuracy of a single or an ensemble of neural networks. Multi-objective approach to support vector machines [21] and fuzzy systems [22, 40] has also been studied. Objectives in training neural networks include accuracy on training or test data, complexity (number of hidden neurons, and number of connections, and so on), diversity, and interpretability. It should be pointed out that a trade-off between the accuracy on the training data and the accuracy on test data does not necessarily result in a trade-off between accuracy and complexity.

Section 2 discusses different aspects in generating accurate and interpretable neural networks. General model selection criteria in machine learning are also briefly described. Section 3 shows that any formal neural network regularization methods can be treated as multi-objective optimization problems. The details of the evolutionary multi-objective algorithm, together with the life-time learning method will be provided in Section 4. The proposed method is verified on two classification examples in Section 5, where a population of Pareto-optimal neural networks are generated. By analyzing the Pareto front, both accurate neural network (in the sense of approximation accuracy on test data) and interpretable neural networks can be identified. A short summary of the chapter is given in Section 5.

2 Tradeoff between Accuracy and Interpretability

2.1 Accurate Neural Networks

Theoretically, feedforward neural networks with one hidden layer are universal approximators, see e.g., [18]. However, a large number of hidden neurons is required to approximate a given complex function to an arbitrary degree of accuracy.

A problem in creating accurate neural networks is that a very good approximation of the training data can result in a poor approximation of unseen data, which is known as overfitting. To avoid overfitting, it is necessary to control the complexity of the neural network, which in turn will limit the accuracy that can be achieved on the training data. Thus, when we talk about accurate neural network models, we mean that the neural network should have good accuracy on both training data and unseen test data.

2.2 Interpretability of Neural Networks

The knowledge acquired by neural networks is distributed on the weights and connections and is therefore not easily understandable to human users. To improve the interpretability of neural networks, the basic approach is to extract symbolic or fuzzy rules from trained neural networks [7, 15, 24]. According to [7], approaches to rule extraction from trained neural networks can be divided into two main categories. The first category is known as decompositional approach, in which rules are extracted from individual neurons in the hidden and output layer. In the second approach, the pedagogical approach, no analysis of the trained neural network is undertaken and the neural network is treated as a “black-box” that provides input-output data pairs for generating rules.

We believe the decompositional approach is more interesting, because the resulting neural network architecture reflects the problem structure in a way, particularly if structure optimization, structural regularization or pruning has been performed during learning [23, 25, 36], which often reduces the complexity of neural networks. Complexity reduction procedure prior to rule extraction is also termed skeletonization [14]. Decompositional approaches to rule extraction from trained neural network can largely be divided into three steps: neural network training, network skeletonization, and rule extraction [15].

2.3 Model Selection in Machine Learning

It is interesting to notice that we need to control the complexity of neural networks properly, no matter when we want to generate accurate or interpretable neural network models. However, the complexity for an accurate model and an interpretable model is often different. Usually, the complexity of an interpretable model is lower than an accurate model due to the limited cognitive capacity of human beings. Unfortunately, the complexity of interpretable neural networks may be insufficient for neural networks to achieve a good accuracy. Thus, a trade-off between accuracy and interpretability is often unavoidable.

Model selection is an important topic in machine learning. Traditionally, model selection criteria have been proposed for creating accurate models. The task of model selection is to choose the best model in the sense of accuracy for a set of given data, assuming that a number of models are available. Several criteria have been proposed based on the Kullback-Leibler Information Criterion [9]. The most popular criteria are Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC). For example, model selection according to the AIC is to minimize the following criterion:

$$AIC = -2\ log(\mathcal{L}(\theta|y,\ g) + 2\ K, \tag{1}$$

where, $\mathcal{L}(\theta|y,\ g)$ is the maximized likelihood for data y given a model g with model parameter θ, K is the number of effective parameters of g.

The first term of Equation (1) reflects how good the model approximates the data, while the second term is the complexity of the model. Usually, the higher the model complexity is, the more accurate the approximation on the training data will be. Obviously, a trade-off between accuracy and model complexity has to be taken into account in model selection.

Model selection criteria have often been used to control the complexity of models to a desired degree in model generation. This approach is usually known as regularization in the neural network community [8]. The main purpose of neural network regularization is to improve the generalization capability of a neural network by control its complexity. By generalization, it is meant that a trained neural network should perform well not only on training data, but also on unseen data.

3 Multi-objective Optimization Approach to Complexity Reduction

3.1 Neural Network Regularization

Neural network regularization can be realized by including an additional term that reflects the model complexity in the cost function of the training algorithm:

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

where E is an error function, Ω is the regularization term representing the complexity of the network model, and λ is a hyperparameter that controls the strength of the regularization. The most common error function in training or evolving neural networks is the mean squared error (MSE):

$$E = \frac{1}{N} \sum_{i=1}^{N} (y^d(i) - y(i))^2, \tag{3}$$

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 that the neural network has only one output. Refer to [8] for other error functions, such as the Minkowski error or cross-entropy.

Several measures have also been suggested for denoting the model complexity Ω. A most popular regularization term is the squared sum of all weights of the network:

$$\Omega = \frac{1}{2} \sum_{k} w_k^2, \tag{4}$$

where k is an index summing up all weights. This regularization method has been termed *weight decay.*

One weakness of the weight decay method is that it is not able to drive small irrelevant weights to zero, when gradient-based learning algorithms are employed, which may result in many small weights [37]. An alternative is to replace the squared sum of the weights with the sum of absolute value of the weights:

$$\Omega = \sum_{i} |w_i|. \tag{5}$$

It has been shown that this regularization term it is able to drive irrelevant weights to zero [31].

Both regularization terms in equations (4) and (5) have also been studied from the Bayesian learning point of view, which are known as the Gaussian regularizer and the Laplace regularizer, respectively.

A more direct measure for model complexity of neural networks is the number of weights contained in the neural network:

$$\Omega = \sum_{i} \sum_{j} c_{ij}, \tag{6}$$

where c_{ij} equals 1 if there is connection from neuron j to neuron i, and 0 if not. It should be noticed that the above complexity measure is not generally applicable to gradient-based learning methods.

A comparison of the three regularization terms using multi-objective evolutionary algorithms has been implemented [27]. Different to the conclusions reported in [31] where the gradient-based learning method has been used, it has been shown that regularization using the sum of squared weights is able to change (reduce) the structure of neural networks as efficiently as using the sum of absolute weights. Thus, no significant difference has been observed from the Gaussian and Laplace regularizers when evolutionary algorithms are used for regularization. In other words, the difference observed in [31] is mainly due to the gradient learning algorithm, but not the regularizers themselves.

3.2 Multi-objective Optimization Approach to Regularization

It is quite straightforward to notice that neural network regularization in equation (2) can be reformulated as a bi-objective optimization problem:

$$\min \{f_1, f_2\} \tag{7}$$
$$f_1 = E, \tag{8}$$
$$f_2 = \Omega, \tag{9}$$

where E is defined in equation (3), and Ω is one of the regularization terms defined in equation (4), (5), or (6).

Note that regularization is traditionally formulated as a single objective optimization problem as in Equation (2) rather than a multi-objective optimization problem as in equation (7). In our opinion, this tradition can be mainly attributed to the fact that traditional gradient-based learning algorithms are not able to solve multi-objective optimization problems.

3.3 Simultaneous Generation of Accurate and Interpretable Models

If evolutionary algorithms are used to solve the multi-objective optimization problem in Equation (7), multiple solutions with a spectrum of model complexity can be obtained in a single optimization run. Neural network models with a sufficient but not overly high complexity are expected to perform a good approximation on both training and test data. Meanwhile, interpretable symbolic rules can be extracted from those of a lower complexity [28].

An important issue is how to choose models that are of good accuracy. One possibility is to employ model selection criteria we mentioned above, such as AIC, BIC, cross-validation, and bootstrap.

With the Pareto front that trades off between training error and complexity at hand, an interesting question arisen is that if it is possible to pick out the accurate models without resorting to conventional model selection criteria. In this chapter, a first step towards this direction is made by analyzing the performance gain with respect to complexity increase of the obtained Pareto-optimal solutions.

In order to generate interpretable neural networks, rules are extracted from the simplest Pareto-optimal neural networks based on the decompositional approach.

To illustrate the feasibility of the suggested idea, simulations are conducted on two widely used benchmark problems, the breast cancer diagnosis data and the iris data.

$$\begin{bmatrix} 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 0\ 0\ 1 \\ 1\ 1\ 0\ 0\ 0\ 1 \\ 0\ 0\ 1\ 1\ 0\ 0 \end{bmatrix} \Longrightarrow$$

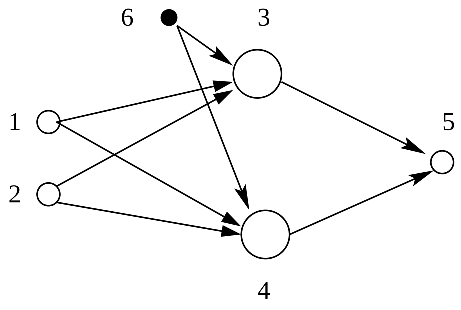

Fig. 1. A connection matrix and the corresponding network structure.

4 Evolutionary Multi-objective Model Generation

4.1 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. 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. Fig. 1 illustrates a connection matrix and the corresponding network structure. It can be seen from the figure that the network has two input neurons, 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.

4.2 Evolution and Life-time Learning

A genetic algorithm has been used for optimizing the structure and weights of the neural networks. Binary coding is adopted representing the neural network structure and real-valued coding for encoding the weights. Five genetic operations have been introduced in the evolution of the neural networks, four of which mutate the connection matrix (neural network structure) and one of which mutates the weights. The four mutation operators are the insertion of a hidden neuron, deletion of a hidden neuron, insertion of a connection and deletion of a connection [19]. A Gaussian-type mutation is applied to mutate the weight matrix. No crossover has been employed in this algorithm.

After mutation, an improved version of the Rprop algorithm citeIgel00 has been employed to train the weights. This can be seen as a kind of life-time learning (the first objective only) within a generation. Both Baldwin effect and Lamarckian evolution have been examined in this work, which will be discussed further in Section 4.3.

The Rprop learning algorithm [34] is believed to be a fast and robust learning algorithm. 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:

$$\Delta w_{ij}^{(t)} = -\text{sign}\left(\frac{\partial E^{(t)}}{\partial w_{ij}}\right) \cdot \Delta_{ij}^{(t)}, \tag{10}$$

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:

$$\Delta_{ij}^{(t)} = \begin{cases} \xi^{+} \cdot \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \xi^{-} \cdot \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)} & , \text{ otherwise} \end{cases}, \tag{11}$$

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

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:

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

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 [20], it is argued that the condition for weight retraction in equation (12) 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 (12) is modified as follows:

$$\Delta w^{(t)} = -\Delta_{ij}^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \textbf{ and } E^{(t)} > E^{(t-1)}. \tag{13}$$

It has been shown on several benchmark problems in [20] that the modified Rprop (termed as Rprop^{+} in [20]) exhibits consistently better performance than the Rprop algorithm.

4.3 Baldwin Effect and Lamarckian Evolution

When life-time learning is embedded in evolution, two strategies can often be adopted. In the first approach, the fitness value of each individual will be replaced with the new fitness resulted from learning. However, the allele of the chromosome remains unchanged. In the second approach, both the fitness and the allele are replaced with the new values found in the life-time learning. These two strategies are known as the Baldwin effect and the Lamarkian evolution, which are two terms borrowed from biological evolution [5, 6].

4.4 Elitist Non-dominated Sorting

In this work, the elitist non-dominated sorting method proposed in the NSGA-II algorithm [13] has been adopted. Assume the population size is N. At first, the offspring and the parent populations are combined. Then, a non-domination rank and a local crowding distance are assigned to each individual in the combined population. In selection, all non-dominated individuals (say there are N_1 non-dominated solutions) in the combined population are passed to the offspring population, and are removed from the combined population. Now the combined population has $2N - N_1$ individuals. If $N_1 < N$, the non-dominated solutions in the current combined population (say there are N_2 non-dominated solutions) will be passed to the offspring population. This procedure is repeated until the offspring population is filled. It could happen that the number of non-dominated solutions in the current combined population (N_i) is larger than the left slots ($N - N_1 - N_2 - ... - N_{i-1}$) in the current offspring population. In this case, the $N - N_1 - N_2 - ... - N_{i-1}$ individuals with the largest crowding distance from the N_i non-dominated individuals will be passed to the offspring generation.

5 Simulation Results

5.1 Benchmark Problems

To illustrate the feasibility of the idea of generating accurate and interpretable models simultaneously using the evolutionary multi-objective optimization approach, the breast cancer data and the iris data in the UCI repository of machine learning database [33] are studied in this work. The breast cancer data contains 699 examples, each of which has 9 inputs and 2 outputs. The inputs are: clump thickness (x_1), uniformity of cell size (x_2), uniformity of cell shape (x_3), marginal adhesion (x_4), single epithelial cell size (x_5), bare nuclei (x_6), bland chromatin (x_7), normal nucleoli (x_8), and mitosis (x_9). All inputs are normalized, to be more exact, $x_1, ..., x_9 \in \{0.1,\ 0.2,\ ..., 0.8,\ 0.9,\ 1.0\}$. The two outputs are complementary binary value, i.e., if the first output is 1, which means "benign", then the second output is 0. Otherwise, the first output is

0, which means “malignant”, and the second output is 1. Therefore, only the first output is considered in this work. The data samples are divided into two groups: one training data set containing 599 samples and one test data set containing 100 samples. The test data are unavailable to the algorithm during the evolution.

The iris data has 4 attributes, namely, Sepal-length, Sepal-width, Petal-length and Petal-width. The 150 data samples can be divided into 3 classes, Iris-Setosa (class 1), Iris-Versicolor (class 2), and Iris-Virginica (class 3), each class having 50 samples. In this work, 40 data samples for each class are used for training and the rest 10 data samples are for test.

5.2 Experimental Setup

The population size is 100 and the optimization is run for 200 generations. One of the five mutation operations is randomly selected and performed on each individual. The standard deviation of the Gaussian mutations applied on the weight matrix is set to 0.05. The weights of the network are initialized randomly in the interval of $[-0.2, 0.2]$. In the Rprop$^+$ algorithm, the step-sizes are initialized to 0.0125 and bounded between $[0, 50]$ during the adaptation, and $\xi^- = 0.2$, $\xi^+ = 1.2$, which are the default values recommended in [20] and 50 iterations are implemented in each local search.

Although a non-layered neural network can be generated using the coding scheme described in Section 3, only feedforward networks with one hidden layer are generated in this research. The maximum number of hidden nodes is set to 10. The hidden neurons are nonlinear and the output neurons are linear. The activation function used for the hidden neurons is as follows,

$$g(z) = \frac{x}{1 + |x|}, \tag{14}$$

which is illustrated in Fig. 2.

5.3 Comparison of Baldwin Effect and Lamarckian Evolution

To compare the performance of the Baldwin effect and Lamarkian evolution in the context of evolutionary multi-objective optimization of neural networks, the Pareto fronts achieved using the the two strategies are plotted in Fig. 3 for the breast cancer data and in Fig. 4 for the iris data.

From these results, it can be seen that the solution set from the Lamarkian evolution shows clearly better performance in all aspects including closeness, distribution and spread. The spread of the Pareto-optimal solutions is particularly poor for the breast cancer data, which might indicate that efficient life-time learning becomes more important for harder learning tasks, recalling that the breast cancer data has more input attributes and more data samples. Thus, only the results using the Lamarckian evolution will be considered in the following sections.

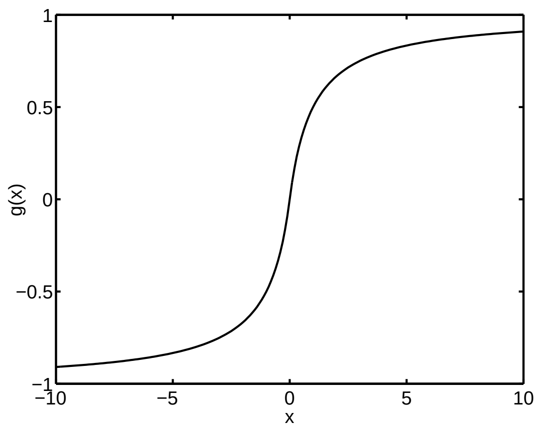

Fig. 2. The activation function of the hidden nodes.

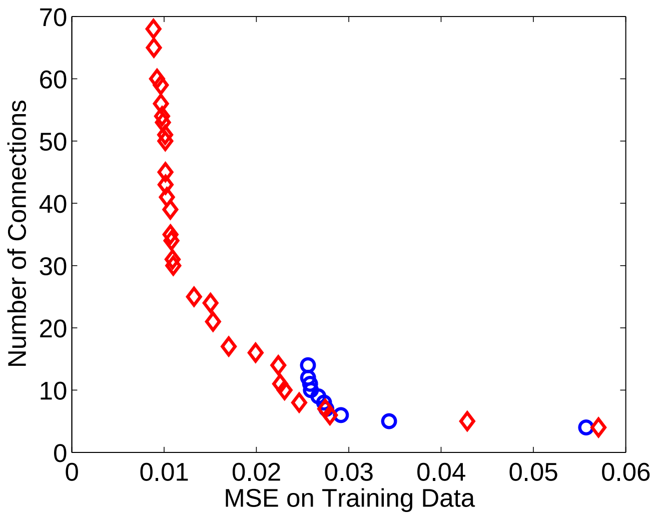

Fig. 3. Pareto fronts obtained using the Baldwin effect (circles) and Lamarckian evolution (diamonds) for the breast cancer data.

5.4 Rule Extraction from Simple Neural Networks

To obtain interpretable neural networks, we attempt to extract symbolic rules from simple neural networks. For the breast cancer data, the simplest Pareto-optimal neural network has only 4 connections: 1 input node, one hidden node and 2 biases, see Fig 5(a). The mean squared error (MSE) of the network on training and test data are 0.0546 and 0.0324, respectively.

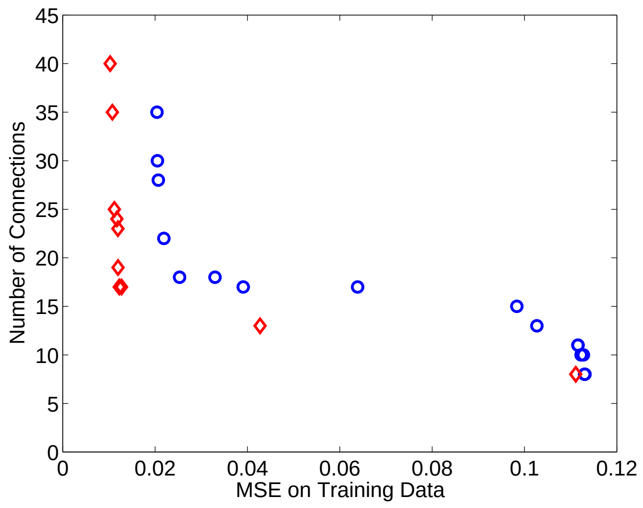

Fig. 4. Pareto fronts obtained using the Baldwin effect (circles) and Lamarckian evolution (diamonds) for the iris data.

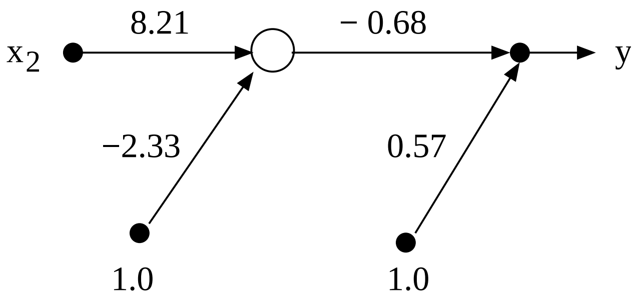

Fig. 5. The simplest neural network for the breast cancer data.

Assuming that a case can be decided to be "malignant" if $y < 0.25$, and "benign" if $y > 0.75$, We can then derive that if $x_2 > 0.4$, which means that $x_2 \geq 0.5$, then "malignant" and "benign" if $x_2 < 0.22$, i.e., then $x_2 \leq 0.2$.

From such a simple network, the following two symbol-type rules can be extracted (denoted as MOO_NN1):

$$\text{R1: If } x_2 \text{ (uniformity) } \geq 0.5, \text{ then malignant;} \tag{15}$$

$$\text{R2: If } x_2 \text{ (uniformity) } \leq 0.2, \text{ then benign.} \tag{16}$$

Based on these two simple rules, only 2 out of 100 test samples will be misclassified, and 4 of them cannot be decided with a predicted value of 0.49, which is very ambiguous. The prediction results on the test data are presented in Fig 5(b).

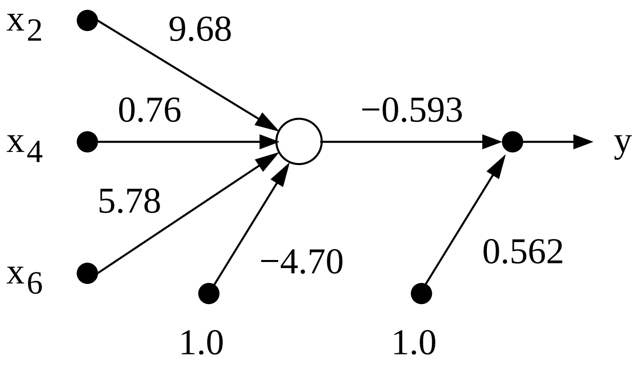

Fig. 6. The next simplest neural network on the breast cancer data.

Now let us look at the second simplest network, which has 6 connections in total. The connection and weights of the network are given in Fig. 6(a), and the prediction results are provided in Fig. 6(b). The MSE of the network on training and test data are 0.0312 and 0.0203, respectively.

In this network, x_2, x_4 and x_6 are present. If the same assumptions are used in deciding whether a case is benign or malignant, then we could extract the following rules: (denoted as MOO_NN2)

$$\begin{aligned}
\text{R1: If} \quad & x_2 \text{ (uniformity) } \geq 0.6 && \text{or} \\
& x_6 \text{ (bare nuclei) } \geq 0.9 && \text{or} \\
& x_2 \text{ (uniformity) } \geq 0.5 \wedge x_6 \text{ (bare nuclei) } \geq 0.2 && \text{or} \\
& x_2 \text{ (uniformity) } \geq 0.4 \wedge x_6 \text{ (bare nuclei) } \geq 0.4 && \text{or} \\
& x_2 \text{ (uniformity) } \geq 0.3 \wedge x_6 \text{ (bare nuclei) } \geq 0.5 && \text{or} \\
& x_2 \text{ (uniformity) } \geq 0.2 \wedge x_6 \text{ (bare nuclei) } \geq 0.7, \text{ then malignant;}
\end{aligned} \tag{17}$$

$$\begin{aligned}
\text{R2: If} \quad & x_2 \text{ (uniformity) } \leq 0.1 \wedge x_6 \text{ (bare nuclei) } \leq 0.4 \quad \text{or} \\
& x_2 \text{ (uniformity) } \leq 0.2 \wedge x_6 \text{ (bare nuclei) } \leq 0.2, \text{ then benign;}
\end{aligned} \tag{18}$$

Compared to the simplest network, with the introduction of two additional features x_6 and x_4 (although the influence of x_4 is too small to be reflected in the rules), the number of cases that are misclassified has been reduced to 1, whereas the number of cases on which no decision can be made remains to be 4, although the ambiguity of the decision for the four cases did decrease.

Similar analysis has been done on the Iris data. The simplest neural network has 8 connections, where only attribute 1 has been selected, refer to Fig. 7.

From the neural network structure, it can be seen that the network can only distinguish the class Iris-Setosa (class 3) from others, but is still too

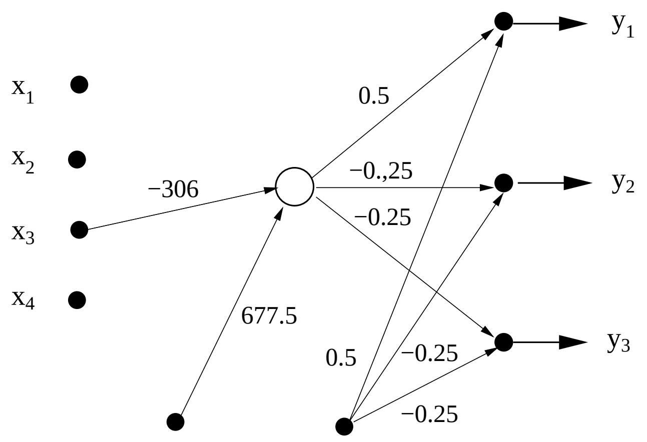

Fig. 7. The simplest neural network for the iris data.

simple to distinguish between Iris-Versicolor and Iris-Virginica. One rule can be extracted from the neural network:

$$\text{R1: If } x_3(\text{Petal-Length}) < 2.2, \text{ then Setosa;} \tag{19}$$

The second simple Pareto-optimal neural network has 13 connections with 2 inputs, see Fig. 8. Though the structure is still very simple, the network is able to separate three classes correctly. By analyzing the structure of the neural network, the following three rules can be extracted:

$$\text{R1:If } x_3(\text{Petal-Length}) \leq 2.2, x_4(\text{Petal-Width}) < 1.0, \text{ then Setosa;} \tag{20}$$
$$\text{R2:If } x_3(\text{Petal-Length}) > 2.2, x_4(\text{Petal-Width}) < 1.4, \text{ then Versicolor;} \tag{21}$$
$$\text{R3:If } x_4(\text{Petal-Width}) > 1.8, \text{ then Virginica;} \tag{22}$$

From the above analyses, we have been successful in extracting understandable symbolic rules from the simple neural networks generated using the evolutionary multi-objective algorithm. Thus, we demonstrate that the idea to generate interpretable neural networks by reducing the complexity of the neural networks is feasible.

5.5 Identifying Accurate Neural Network Models

One advantage of evolutionary multi-objective optimization approach to neural network generation is that a number of neural networks with a spectrum of complexity can be obtained in one single run. With these neural networks that trade off between training error and model complexity, we hope that we can identify those networks that are most likely to perform well both on training data and unseen test data.

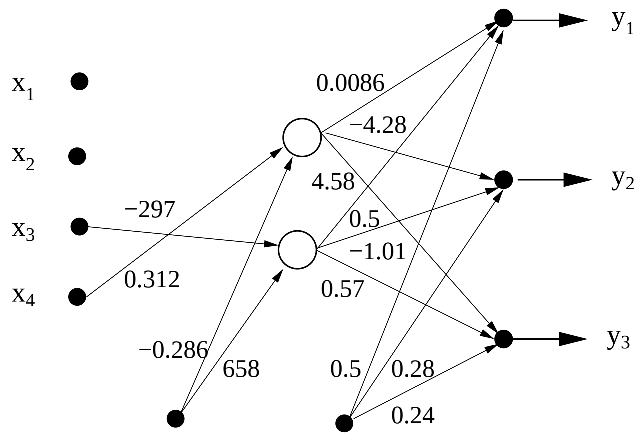

Fig. 8. The second simplest neural network for the iris data.

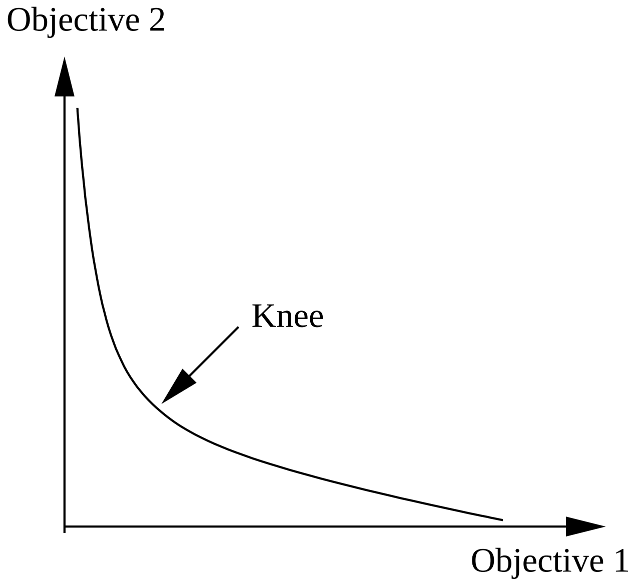

Fig. 9. Knee of a Pareto front.

In multi-objective optimization, the “knee” of a Pareto front is often considered to be interesting, as illustrated in Fig. 9. The reason is that for the solutions at the knee, a small improvement in one objective will cause large deterioration in the other objective. However, such a knee is not always obvious in some applications. Thus, it is important to define a quantitative measure for the knee according to the problem at hand. In [17], some interesting work has been presented to determine the number of clusters by exploit the Pareto front in multi-objective data clustering.

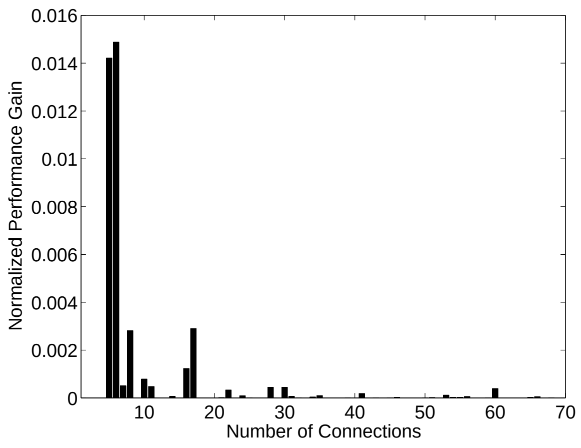

Fig. 10. Normalized performance gain for the breast cancer data.

If we take a closer look at the Pareto fronts that we obtained for the two benchmark problems, we notice that when the model complexity increases, the accuracy on training data also increases. However, the amplitude in accuracy increase is not always the same when the number of connections increases. Thus, we introduce the concept of "normalized performance gain (NPG)", which is defined as follows:

$$NPG = \frac{MSE_i - MSE_j}{N_i - N_j}, \tag{23}$$

where MSE_i, MSE_j, N_i, N_j are the MSE on training data, and the number of connections of the i-th and j-th Pareto optimal solutions.

The normalized performance gain on the breast cancer data is illustrated in Fig. 10. We notice that the largest NPG is achieved when the number of connections is increased from 4 to 6 and from 6 to 7. After that, the NPG decreases almost to zero when the number of connections is increased to 14, though there is some small fluctuations. Thus, we denote the Pareto optimal solution with 7 connections as the knee of the Pareto front, see Fig 11. To investigate if there is any correlation between the knee point and the approximation error on test data, we examine the test error of the neighboring five Pareto-optimal solutions right to the knee, which are circled in Fig.11. Very interestingly, we can see from Fig 15 that all the five solutions belong to the cluster that has the smallest error on test data.

We do the same analysis on the iris data. The NPG on the iris data is presented in Fig. 13. Similarly, the NPG is large when the number of connections increases from 8 to 13 and from 13 to 17. After that, the NPG decreases dramatically. Thus, the solution with 14 connections is considered as the knee of the Pareto front, see Fig. 14.

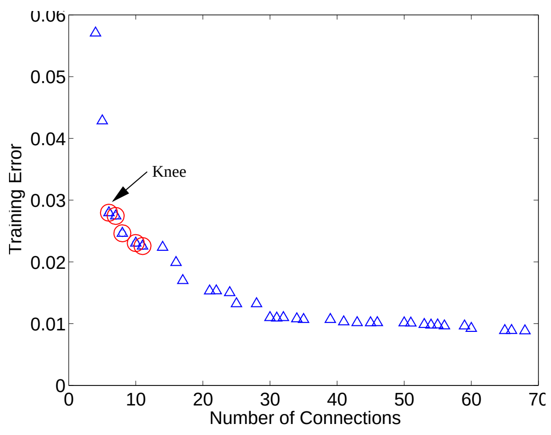

Fig. 11. Complexity versus training error for the breast cancer data. The five solutions right to the knee are circled.

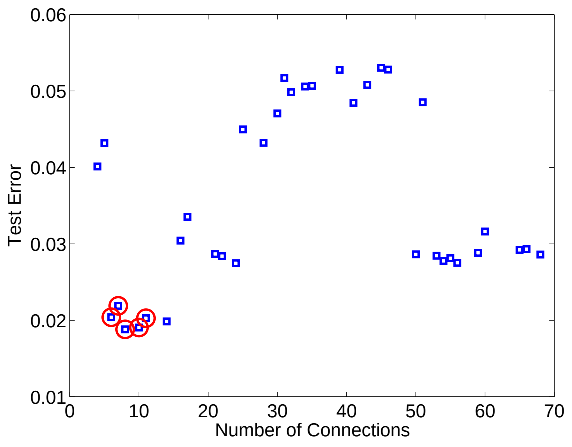

Fig. 12. Complexity versus test error for the breast cancer data. The circled solutions are those neighbor solutions right to the knee on the Pareto front.

However, when we look at the test error of all the solutions, we find that no observable overfitting has occurred with the increase in complexity. Actually, the test error shows very similar behavior to the training data with respect to the increase in complexity. In fact, the test errors are even smaller than the training error, refer to Fig. 15. If we take a look at the training and test data as shown in Fig. 16, this turns out to be natural, because none of the test samples (denoted by filled shapes) are located in the overlapping part of the different classes.

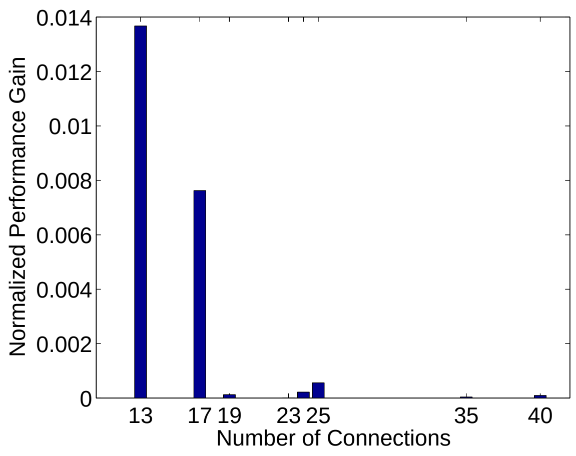

Fig. 13. Normalized performance gain for the iris data.

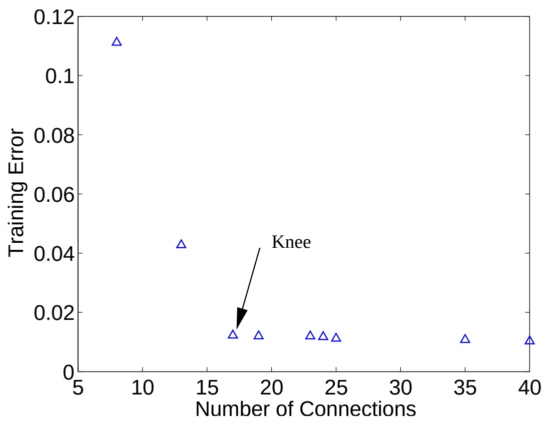

Fig. 14. Complexity versus training error for the iris data.

6 Conclusions

Generating accurate and interpretable neural networks simultaneously is very attractive in many applications. This work presents a method for achieving this goal using an evolutionary multi-objective algorithm. The accuracy on the training data and the number of connections of the neural network are taken as the two objectives in optimization. We believe the complexity is a very important objective for learning models in that both generalization performance and interpretability are closely related to the complexity.

By analyzing the achieved Pareto-optimal solutions that trade off between training accuracy and complexity, we are able to choose neural networks with a simpler structure for extracting interpretable symbolic rules. In the meantime,

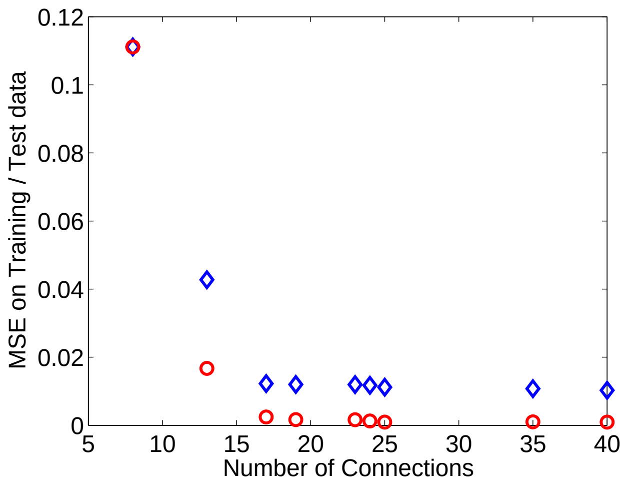

Fig. 15. Complexity versus training and test error for the iris data. The training errors are denoted by diamonds and the test error by circles.

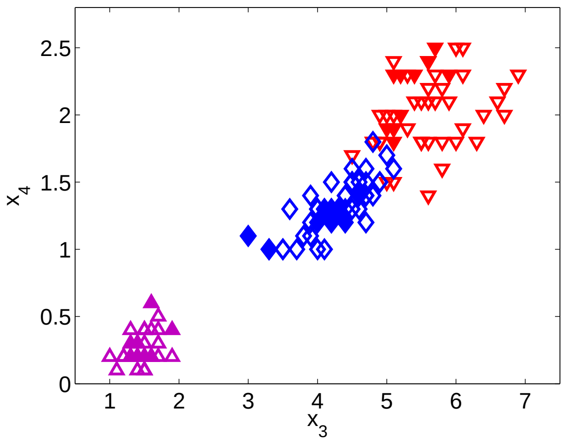

Fig. 16. Location of the training and test samples in the $x_3 - x_4$ space of the iris data. Training and test data of class 1 are denoted by triangles and filled triangles, class 2 by diamonds and filled, and class 3 by triangles down and filled triangles down, respectively.

neural networks that have good accuracy on unseen data can also be picked out by choosing those solutions in the neighborhood of the knee of the Pareto front, if overfitting occurs.

Much work remains to be done in the future. First it is interesting to compare our method with model selection criteria such as cross-validation or bootstrap methods. Second, this method should be verified on more benchmark problems.

References

1. H.A. Abbass. A memetic Pareto evolutionary approach to artificial neural networks. *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, pages 1–12, 2001
2. H.A. Abbass. An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 25(3):265–281, 2002.
3. H.A. Abbass. Pareto neuro-ensemble: Constructing ensembles of neural networks using multi-objective optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2074–2080
4. H.A. Abbass. Speeding up back-propagation using multi-objective evolutionary algorithms. *Neural Computation*, 15(11):2705–2726, 2003.
5. D.H. Ackley, M.L. Littman. Interactions between learning and evolution. *Artificial Life*, 2:487–509, 1992
6. D.H. Ackley, M.L. Littman. A case for Lamarckian evolution. *Artificial Life*, 3:3–10, 1994
7. R. Andrews, J. Diederich, and A. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Systems*, 8(6):373–389, 1995.
8. C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1995.
9. K.P. Burnham and D.R. Anderson. *Model Selection and Multimodel Inference.* Springer, New York, second edition, 2002.
10. Evolutionary framework for the construction of diverse hybrid ensembles. In: *Proceedings of 13th European Symposium on Artificial Neural Networks.* pages 253–258, 2005
11. C. Coello Coello, D. Veldhuizen, and G. Lamont. *Evolutionary algorithms for solving multi-objective problems.* Kluwer Academic, New York, 2002.
12. K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms.* Wiley, Chichester, 2001.
13. 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.
14. W. Duch, R. Adamczak, and K. Grabczewski. Extraction of logical rules from backpropagation networks. *Neural Processing Letters*, 7:1–9, 1998.
15. W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for rule-based data understanding. *Proceedings of the IEEE*, 92(5):771–805, 2004.
16. J.E. Fieldsend, S. Singh. Optimizing forecast model complexity using multi-objective evolutionary algorithms. In *Applications of Multi-objective Evolutionary Algorithms*, C. Coello Coello, G.B. Lamont (eds.), pages 675–700. World Scientific, 2004
17. J. Handl and J. Knowles. Exploiting the trade-off - The benefits of multiple objectives in data clustering. *Evolutionary Multi-Criterion Optimization*, LNCS 3410, pages 547–560, 2005

18. K. Hornik, M. Stinchcombe, H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
19. 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.
20. 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.
21. C. Igel. Multi-objective model selection for support vector machines. *Evolutionary Multi-Criterion Optimization*, LNCS 3410, pages 534–546, 2005
22. H. Ishibuchi, T. Yamamoto. Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1077–188, 2003.
23. M. Ishikawa. Rule extraction by successive regularization. *Neural Networks*, 13:1171–1183, 2000.
24. Y. Jin. *Advanced Fuzzy Systems Design and Applications.* Springer, Heidelberg, 2003.
25. Y. Jin, B. Sendhoff. Extracting interpretable fuzzy rules from RBF networks. *Neural Processing Letters*, 17(2):149–164, 2003.
26. Y. Jin, T. Okabe, B. Sendhoff. Evolutionary multi-objective optimization approach to constructing neural network ensembles for regression. In *Applications of Multi-objective Evolutionary Algorithms*, C. Coello Coello, G.B. Lamont (eds.), pages 653–673. World Scientific, 2004
27. Y. Jin, T. Okabe, B. Sendhoff. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation*, pages 1–8. IEEE, 2004.
28. Y. Jin, B. Sendhoff, E. Körner. Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations. *Evolutionary Multi-Criterion Optimization*, LNCS 3410, pages 752–766, 2005
29. K. Kottathra, Y. Attikiouzel. A novel multicriteria optimization algorithm for the structure determination of multilayer feedforward neural networks. *Journal of Network and Computer Applications*, 19:135–147, 1996.
30. M.A. Kupinski, M.A. Anastasio. Multiobjective genetic optimization of diagnostic classifiers with implementations for generating receiver operating characteristic curves. *IEEE Transactions on Medical Imaging*, 18(8):675–685, 1999.
31. D.A. Miller, J.M. Zurada. A dynamical system perspective of structural learning with forgetting. *IEEE Transactions on Neural Networks*, 9(3):508–515, 1998.
32. S. Park, D. Nam, C.H. Park. Design of a neural controller using multiobjective optimization for nonminimum phase. *Proceedings of IEEE International Conference on Fuzzy Systems* , I:533-537, 1999.
33. L. Prechelt. PROBEN1 - a set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.
34. M. Riedmiller, H. Braun. A direct adaptive method for faster backpropgation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, volume 1, pages 586–591, New York, 1993. IEEE.
35. R. Setiono. Generating concise and accurate classification rules for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 18:205–219, 2000.

36. R. Setiono, H. Liu. Symbolic representation of neural networks. *IEEE Computer*, 29(3):71–77, 1996.
37. R.D. Reed, R.J. Marks II. *Neural Smithing.* The MIT Press, 1999.
38. I. Taha, J. Ghosh. Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):448–463, 1999.
39. R. de A. Teixeira, A.P. Braga, R. H.C. Takahashi, R. R. Saldanha. Improving generalization of MLPs with multi-objective optimization. *Neurocomputing*, 35:189–194, 2000.
40. H. Wang, S. Kwong, Y. Jin, W. Wei, K.F. Man. Agent-based evolutionary approach for interpretable rule-based knowledge extraction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(2):143–155, 2005
41. S. Wiegand, C. Igel, U. Handmann. Evolutionary multi-objective optimization of neural networks for face detection. *International Journal of Computational Intelligence and Applications*, 4:237–253, 2004