Rule extraction from compact Pareto-optimal neural networks

Yaochu Jin, Bernhard Sendhoff, Edgar Körner

2007

Preprint:

This is an accepted article published in Multi-objective Evolutionary Algorithms for Knowledge from Databases. The final authenticated version is available online at: https://doi.org/[DOI not available]

1

Rule Extraction from Compact Pareto-Optimal Neural Networks

Yaochu Jin, Bernhard Sendhoff, and Edgar Körner

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

Summary. Rule extraction from neural networks is a powerful tool for knowledge discovery from data. In order to facilitate rule extraction, trained neural networks are often pruned so that the extracted rules are understandable to human users. This chapter presents a method for extracting interpretable rules from neural networks that are generated using an evolutionary multi-objective algorithm. In the algorithm, the accuracy on the training data and the complexity of the neural networks are minimized simultaneously. Since there is a tradeoff between accuracy and complexity, a number of Pareto-optimal neural networks, instead of one single optimal neural network, are obtained. We show that the Pareto-optimal networks with a minimal degree of complexity are often interpretable in that understandable logic rules can be extracted from them straightforwardly. The proposed approach is verified on two benchmark problems.

1.1 Introduction

Knowledge acquired by neural networks is not easily understandable to human beings since the acquired knowledge is distributed among the weights and nodes of the neural networks. Many efforts have been made to extract symbolic or fuzzy rules from trained neural networks to gain a deep insight into the neural networks [1, 2, 4, 5, 6], which is of particular importance when the neural networks are employed for critical applications, or when neural networks are used for knowledge discovery from data [7].

As indicated in [1], two main issues must be taken into account in extracting rules from trained neural networks. First, the extracted rules should contain the same information that is learned by the original neural network. No significant information loss should occur, nor should additional information be introduced into the rules. Second, the extracted rules should be understandable to human users. As discussed in [8, 9, 3], one important aspect that is closely related to the interpretability of the extracted rules is the complexity of the extracted rules, including the number of rules and the number

of premises in the rules. It is suggested that for the rules to be easily understandable, the number of rules should be limited, and the number of premises should be small (usually less than 10). To improve the comprehensibility of the extracted rules, it is a common practice to prune the neural networks using regularization techniques.

Two basic approaches, namely, the de-compositional approach and the pedagogical approach, have been developed for extracting rules from trained neural networks [1]. The de-compositional approach translates each node in the neural network into a rule, in which the inputs of the node are the premises of the rule, and the output is the consequence of the rule. The pedagogical approach, by contrast, treats the neural network as a blackbox and considers rule extraction as a learning process.

This chapter presents an approach to extracting interpretable rules from neural networks that are generated using an evolutionary multi-objective approach. The basic idea is to generate a number of Pareto-optimal neural networks that reflect a tradeoff between accuracy and complexity. Multiple neural networks of a variety of model complexities, instead of either a single signal-type or symbol-type model, will be generated using a multi-objective evolutionary algorithm combined with a local search, where accuracy and complexity serve as two conflicting objectives. It has been shown in [10, 11] that by analyzing the Pareto front, we are able to identify neural networks that are interpretable and those that are able to generalize on unseen data.

The use of evolutionary multi-objective algorithms [16, 15] for addressing machine learning problems has attracted increasing interest over the past few years. On the one hand, evolutionary algorithms are well suited for solving multi-objective problems mainly because they are population based search methods, which are able to achieve an approximation of the Pareto-front in one single run. On the other hand, machine learning problems are inherently multi-objective problems, where tradeoffs between accuracy and complexity, between accuracy and diversity, between accuracy and interpretability, and between stability and plasticity have to be taken into account. The marriage of evolutionary multi-objective optimization and machine learning has shown to be very fruitful, see [12] for a complete and updated overview of the research results.

Section 1.2 discusses very briefly the existing methods for controlling model complexity in the context of model selections in machine learning. Methods for extracting rules from neural networks are introduced. Section 1.3 shows that any formal neural network regularization method that simultaneously minimizes the accuracy and complexity can be treated as a multi-objective optimization problem. The details of the evolutionary multi-objective algorithm, together with the local search method will be provided in Section 1.4. Two illustrative examples are given in Section 1.5, where a number of Pareto-optimal neural networks are generated using the evolutionary multi-objective optimization algorithm. It will be shown that among the models generated by

the multi-objective evolutionary algorithm, interpretable logic rules can be extracted from the compact Pareto-optimal neural networks.

1.2 Rule Extraction and Complexity Control

1.2.1 Model Selection in Machine Learning

By model selection, we mean to choose the best model for a set of given data, assuming that a number of models is available. Here, the meaning of "best" needs to be further explained. In general, a model is the best if the prediction error on the unseen data is minimal. Several criteria have been proposed based on the Kullback-Leibler Information Criterion [14]. Two 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:

$$AIC = -2\ log(\mathcal{L}(\theta|y,\ g) + 2\ K, \tag{1.1}$$

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 two terms in in Equation (1.1) indicate that model selection has to deal with two criteria. The first criterion minimizes the approximation error on the training data, whereas the second one minimizes the complexity of the model. There is a conflict between the two objectives, i.e., we cannot minimize the two objectives simultaneously. Usually, the smaller the error on the training data, the larger the complexity of the neural network. Obviously, a trade-off between accuracy and model complexity has to be handled in model selection.

Consequently, model selection criteria have often been employed to control the complexity of models to a desired degree. This approach is usually known as regularization in the neural network learning [13]. The main purpose of neural network regularization is to avoid the overfitting of the training data by means of controlling the complexity of the neural network. In this way, the generalization capability of a neural network is improved. By generalization, it is meant that a trained neural network should perform well not only on the training data, but also on unseen data.

1.2.2 Complexity Reduction in Rule Extraction

Generally, neural networks are difficult to understand for human users. Due to this reason, many efforts have been made to extract symbolic or fuzzy rules from trained neural network [1, 2, 4]. Two assumptions are often made during rule extraction from trained neural networks. First, units in the neural network are either maximally active or inactive. To meet this requirement, regularization techniques such as structural regularization [24], weight sharing [3] or

network pruning [31] are usually implemented before rule extraction, which in effect reduces the complexity of neural networks and thus also contributes to the comprehensibility of the extracted rules. The complexity reduction procedure prior to rule extraction is also termed skeletonization [18]. The second assumption is that a label, or in other words, a meaning needs to be associated with each unit. This is of less concern if rules are extracted from the output neurons.

Rule extraction from trained neural networks can be seen as a model selection process that trades off between accuracy and interpretability, where preference is put on the interpretability of the model. Thus, the trade-off between accuracy and complexity in model selection also reflects a trade-off between accuracy and interpretability.

Existing methods for extracting symbolic rules from trained neural network can largely be divided into three steps: neural network training, network skeletonization, and rule extraction [4].

1.3 Multi-objective Optimization Approach to Complexity Reduction

1.3.1 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:

$$J = E + \lambda \Omega, \tag{1.2}$$

where E is the approximation error on the training data, 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):

$$E = \frac{1}{N} \sum_{i=1}^{N} (y^d(i) - y(i))^2, \tag{1.3}$$

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 [13] 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:

$$\Omega = \frac{1}{2} \sum_{k} w_k^2, \tag{1.4}$$

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 [28]. An alternative is to replace the squared sum of the weights with the sum of absolute value of the weights:

$$\Omega = \sum_i |w_i|. \tag{1.5}$$

It has been shown that this regularization term is able to drive irrelevant weights to zero [26].

Note, however, that the weakness of weight decay is more related to the learning method (e.g., a gradient-based learning algorithm) other than the complexity measure itself. Different to the conclusions reported [26], where a gradient-based learning method has been used, it has been shown in [25] 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, when an evolutionary algorithm is employed to minimize the structure of neural networks.

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

$$\Omega = \sum_i \sum_j c_{ij}, \tag{1.6}$$

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

1.3.2 Multi-objective Optimization Approach to Regularization

It is quite straightforward to notice that neural network regularization in equation (1.2) can be reformulated as a bi-objective optimization problem:

$$\min \{f_1, f_2\} \tag{1.7}$$

$$f_1 = E, \tag{1.8}$$

$$f_2 = \Omega, \tag{1.9}$$

where E is defined in equation (1.3), and Ω is one of the regularization terms defined in equation (1.4), (1.5), or (1.6).

It is noticed that regularization is traditionally formulated as a single objective optimization problem as in Equation (1.2) rather than a multi-objective optimization problem as in equation (1.7). 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.

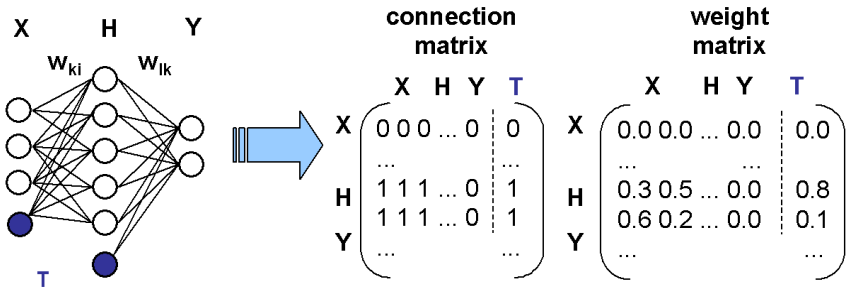

Fig. 1.1. Coding of the structure and parameters of neural networks using a connection matrix and a weight matrix.

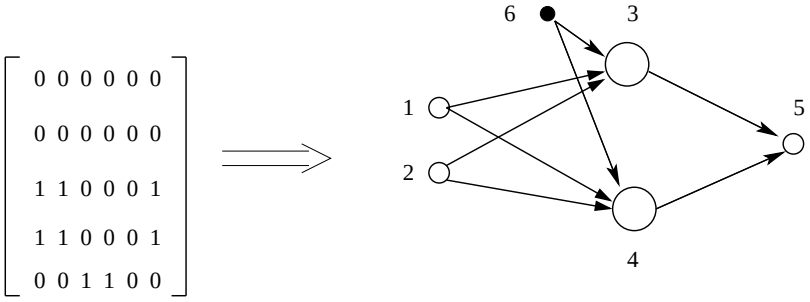

Fig. 1.2. An example of a connection matrix and its corresponding neural network structure.

1.4 Evolutionary Multi-objective Optimization of Neural Networks

1.4.1 Coding the Structure and Parameters of Neural Networks

A connection matrix and a weight matrix are employed to describe the structure and the weights of the neural networks, see Fig. 1.1. 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. 1.2 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 c_{ij} in the connection matrix equals zero, the corresponding element in the weight matrix must be zero too.

1.4.2 Evolutionary Variations and Life-time Learning

A genetic algorithm has been used for optimizing the structure and weights of the neural networks. Binary coding has been used representing the neural network structure and real-valued coding for encoding the weights. Five genetic operations have been introduced in the global search, four of which mutate the connection matrix (neural network structure) and one of which mutates the weights. The four mutation operators are insertion of a hidden neuron, deletion of a hidden neuron, insertion of a connection and deletion of a connection [21]. 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 [22] 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. 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 encoded back to the chromosome, which is known as the Lamarckian type of inheritance.

The Rprop learning algorithm [29] 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:

$$\Delta w_{ij}^{(t)} = -\text{sign}\left(\frac{\partial E^{(t)}}{\partial w_{ij}}\right) \cdot \Delta_{ij}^{(t)}, \tag{1.10}$$

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:

$$\Delta_{ij}^{(t)} = \begin{cases} \xi^+ \cdot \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \xi^- \cdot \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)} & , \text{ otherwise} \end{cases}, \tag{1.11}$$

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:

$$\Delta w^{(t)} = -\Delta_{ij}^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0. \tag{1.12}$$

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 [22], it is argued that the condition for weight retraction in equation (1.12) 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 (1.12) is modified as follows:

$$\Delta w^{(t)} = -\Delta_{ij}^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \textbf{ and } E^{(t)} > E^{(t-1)}. \tag{1.13}$$

It has been shown on several benchmark problems in [22] that the modified Rprop (termed as Rprop$^+$ in [22]) exhibits consistently better performance than the Rprop algorithm.

1.4.3 Crowded Tournament Selection

To select the offspring for the next generation, we employ the crowded tournament selection method proposed in the NSGA-II algorithm [17]. At first, the offspring and the parent populations are combined. Then, a non-dominated rank and a local crowding distance are assigned to each individual in the combined population. In the non-dominated ranking, the non-dominated solutions are found out and assigned a rank 1. These solutions consist of the first non-dominated front. After that, the non-dominated solutions with rank 1 are removed from the population. Then, non-dominated solutions in the rest of the individuals are identified, which is the second non-dominated front. A rank of 2 is assigned to these solutions. This procedure repeats until all the individuals are assigned to a non-dominated front. In the next step, a crowding distance is calculated for each individual with regard to the non-dominated front it belongs to. The crowding distance of solution i in the non-dominated front j is the distance of the two neighboring of solution s_i^j in the objective space.

$$d_i^j = \sum_{k=1}^{m} |f_k(s_{i-1}^j) - f_k(s_{i+1}^j)|, \tag{1.14}$$

where m is the number of objectives in the multi-objective optimization problem, solutions s_{i-1}^j and s_{i+1}^j are the two neighboring solutions of solution s_i^j. A large distance is assigned to the boundary solutions in each non-dominated front. Here, the larger the crowding distance is, the less crowded around the solution s_i^j.

In selection, two solutions are chosen randomly. The solution with the better (lower) rank wins the tournament. If the two solutions have the same rank, the one with the larger crowding distance wins. If the two solutions with the same rank and the same crowding distance, choose a winner randomly.

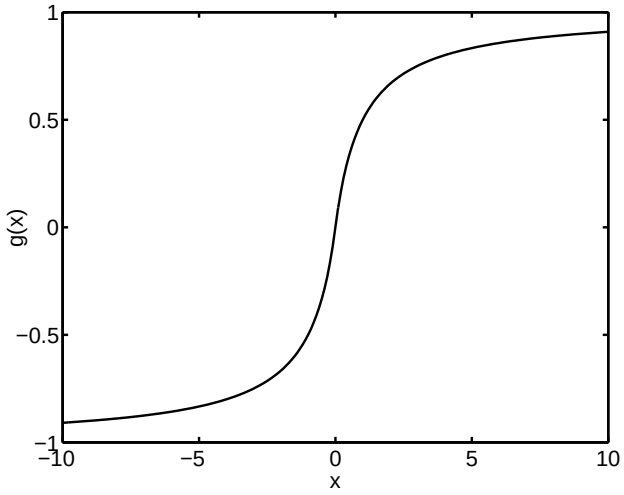

Fig. 1.3. The activation function of the hidden nodes.

1.5 Illustrative Examples

1.5.1 Experimental Setup

To demonstrate that interpretable logic rules can be extracted from the compact Pareto-optimal solutions, the evolutionary multi-objective algorithm is applied to two benchmark problems, namely, the Breast Cancer Diagnosis data and the Iris data.

Although a non-layered neural network can be generated using the coding scheme described in Section 1.3, feedforward networks with one hidden layer will be generated. 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,

$$g(z) = \frac{x}{1 + |x|}, \tag{1.15}$$

which is illustrated in Fig. 1.3.

In this study, the complexity measure defined in Equation (1.6) has been used as the objective describing the complexity of the neural networks.

The population size of the evolutionary algorithm is 100 and the optimization is run for 200 generations. One of the five mutation operations is randomly chosen and performed on each individual. The standard deviation of the Gaussian mutations applied to 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 [22] and 50 iterations are implemented in each lifetime learning.

1.5.2 Breast Cancer Diagnosis

The breast cancer benchmark problem in the UCI repository of machine learning database was collected by Dr. W.H. Wolberg at the University of Wisconsin-Madison Hospitals [27]. Studies have been carried out to extract symbolic rules from trained neural network using the three-step procedure for rule extraction on this benchmark problem [32, 30]. The benchmark problem 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 non-dominated solutions obtained at the 200-th generation are plotted in Fig. 1.4. Note that many solutions in the final population are the same and finally 41 non-dominated solutions have been generated.

Among the 41 neural networks, the simplest one has only 4 connections: 1 input node, one hidden node and 2 biases, see Fig. 1.5. The mean squared error (MSE) of the network on the training and test data are 0.0546 and 0.0324, respectively.

Assuming that a case can be decided to be "malignant" if $y < -0.75$, and "benign" if $y > 0.75$. For the neural network in Fig. 1.5, if the output of the hidden node is z, we have:

$$-0.68z + 0.57 > 0.75, \tag{1.16}$$

which means that

$$z < -0.28. \tag{1.17}$$

Let a denote the summed input of the hidden node, we obtain

$$\frac{a}{1 + |a|} < -0.28, \tag{1.18}$$

which implies that

$$a < -0.39. \tag{1.19}$$

Therefore, we get:

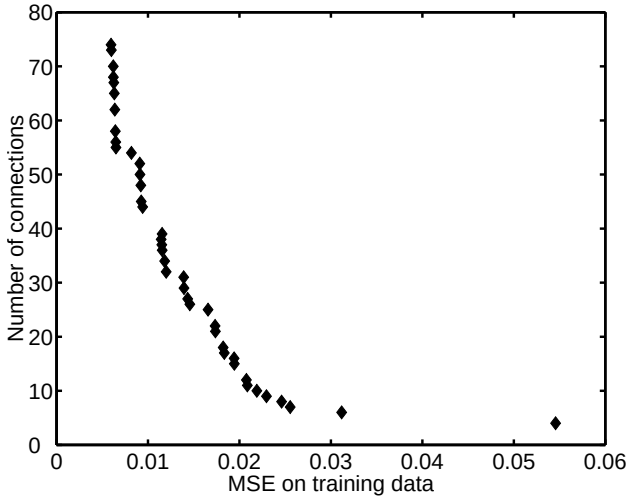

Fig. 1.4. The Pareto front containing 41 non-dominated solutions representing neural networks of a different model complexity.

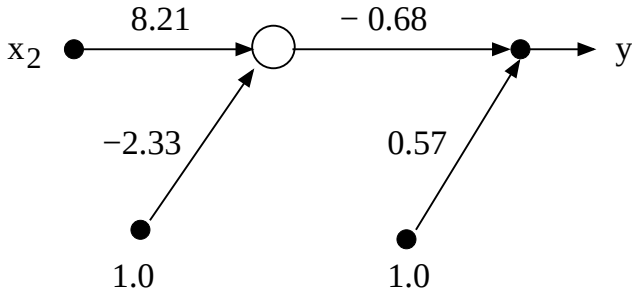

Fig. 1.5. The structure of the simplest Pareto-optimal neural network.

$$8.21x_2 - 2.33 < -0.39, \tag{1.20}$$

and finally we have:

$$x_2 < 0.21. \tag{1.21}$$

Thus, we can derive that if $x_2 < 0.21$, then the case is "benign". The same procedure can be applied to the malignant case.

As a result, the following two logic rules can be extracted from the simplest Pareto-optimal neural network (denoted as MOO_NN1):

$$\text{R1: If } x_2 \text{ (uniformity) } \geq 0.5, \text{ then malignant;} \tag{1.22}$$

$$\text{R2: If } x_2 \text{ (uniformity) } \leq 0.2, \text{ then benign.} \tag{1.23}$$

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,

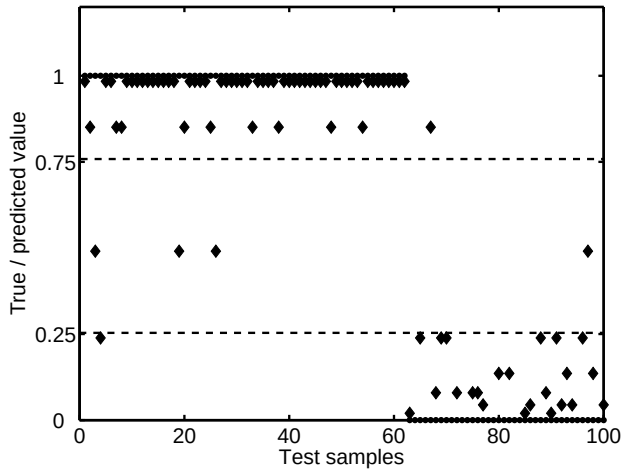

Fig. 1.6. The prediction results of the simplest Pareto-optimal neural network on test data.

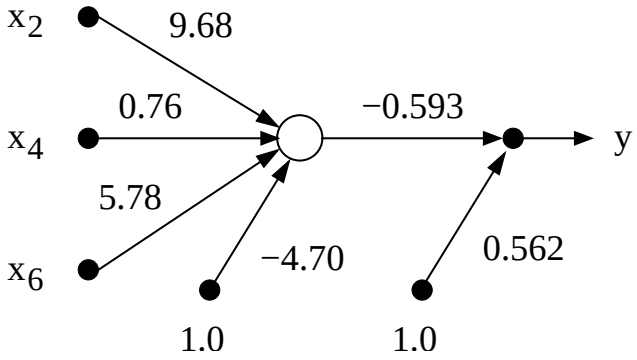

Fig. 1.7. The structure of the second simplest Pareto-optimal neural network.

which is very ambiguous. The prediction results on the test data are presented in Fig 1.6.

Now let us look at the second simplest Pareto-optimal neural network, which has 6 connections in total. The connection and weights of the network are given in Fig. 1.7, and the prediction results are provided in Fig. 1.8. 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)

R1: If	x_2 (uniformity) ≥ 0.6	or
	x_6 (bare nuclei) ≥ 0.9	or

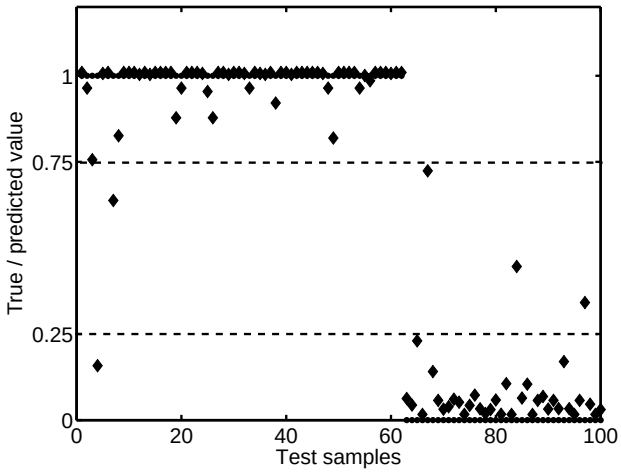

Fig. 1.8. The prediction results of the second simplest Pareto-optimal neural network on test data.

$$\begin{aligned} & x_2 \text{ (uniformity)} \ \geq 0.5 \wedge x_6 \text{ (bare nuclei)} \ \geq 0.2 \quad \text{or} \\ & x_2 \text{ (uniformity)} \ \geq 0.4 \wedge x_6 \text{ (bare nuclei)} \ \geq 0.4 \quad \text{or} \\ & x_2 \text{ (uniformity)} \ \geq 0.3 \wedge x_6 \text{ (bare nuclei)} \ \geq 0.5 \quad \text{or} \\ & x_2 \text{ (uniformity)} \ \geq 0.2 \wedge x_6 \text{ (bare nuclei)} \ \geq 0.7, \text{ then malignant;} \end{aligned} \tag{1.24}$$

$$\begin{aligned} \text{R2: If} \quad & x_2 \text{ (uniformity)} \ \leq 0.1 \wedge x_6 \text{ (bare nuclei)} \ \leq 0.4 \quad \text{or} \\ & x_2 \text{ (uniformity)} \ \leq 0.2 \wedge x_6 \text{ (bare nuclei)} \ \leq 0.2, \text{ then benign;} \end{aligned} \tag{1.25}$$

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 do decrease.

The above two neural networks are very simple in structure. We have shown that for such networks of a low model complexity, interpretable logic rules can be extracted. In the following, we will take a look at two neural networks obtained in the multi-objective optimization, which are of better accuracy but are of more signal-type quality, i.e., no interpretable rules can be extracted.

The first network of relatively higher model complexity has 16 connections, whose structure and weights are described in Fig. 1.9. The prediction results are plotted in Fig. 1.10. In this network, only x_3 is absent and there are 2 hidden nodes. The MSE on training and test data sets are 0.019 and 0.014, respectively. From Fig. 1.10, we can see that the classification accu-

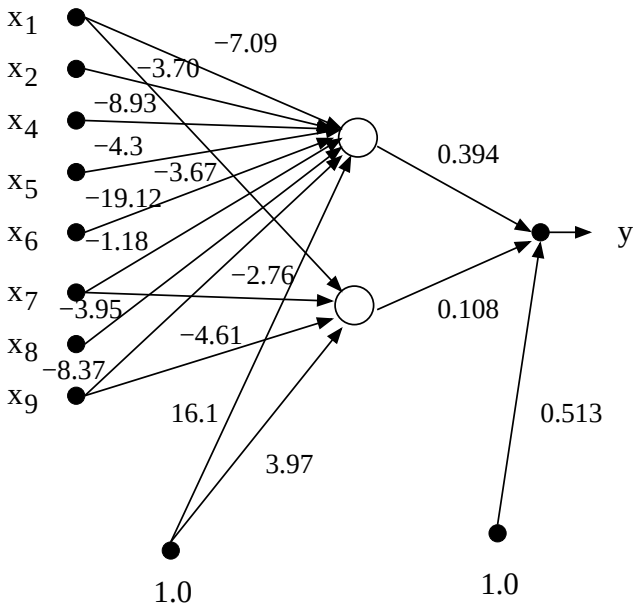

Fig. 1.9. The Pareto-optimal neural network with 16 connections.

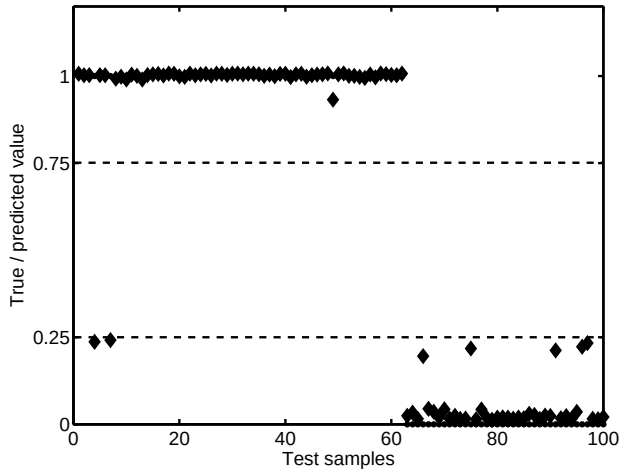

Fig. 1.10. The prediction results of the Pareto-optimal neural network on test data.

racy is better: only two cases are misclassified. However, extracting symbolic rules from the network becomes much more difficult. Besides, although the architecture of the two simple networks still exist in the current network, it not longer shows a dominating influence. Thus, the "skeleton" defined by the simple networks has been lost.

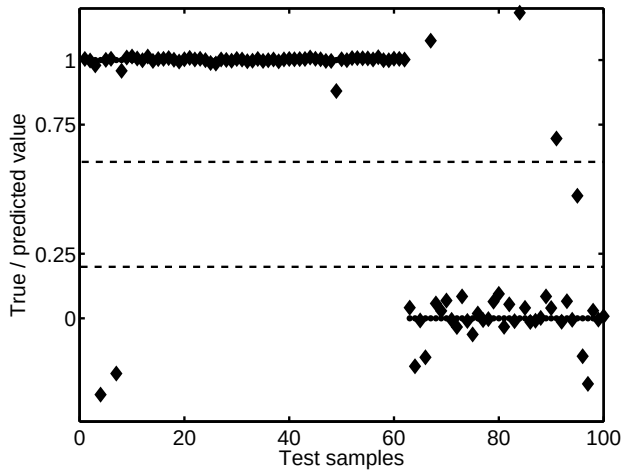

Fig. 1.11. The prediction results of the most complex Pareto-optimal neural network obtained in the simulation.

The most complex network obtained in the run has 74 connections. All input features are included in the network and the number of hidden nodes is 9. The MSE on the training data set is 0.0060, however, the MSE on the test data set increases to 0.066 with 5 samples misclassified and 1 undetermined. It seems that the network has over-fitted the training data and the understanding of the network is difficult. The prediction results of this neural network are provided in Fig. 1.11.

1.5.3 The Iris Data

The second data set we looked at is the Iris data which was originated from references [33, 27]. The data set contains 3 classes of 40 instances each, where each class refers to a type of iris plant. The three classes are: Iris Setosa (class 1, represented by -1), Iris Versicolor (class 2, represented by 0), and Iris Virginica (class 3, represented by 1). Four attributes are used to predict the iris class, i.e., sepal length (x_1), sepal width (x_2), petal length (x_3), and petal width (x_4), all in centimeter. Among the three classes, class 1 is linearly separable from the other two classes, and class 2 and 3 are not linearly separable from each other, refer to Fig. 1.12.

The same parameter settings have been used to generate the Pareto-optimal neural networks that minimize the accuracy and the complexity. The final population contains 11 non-nominated solutions, which are plotted in Fig. 1.13. The most compact Pareto-optimal solution has 8 connections in total, and only attribute x_3 is used for prediction, see Figs. 1.14 and 1.15 for the structure of the neural network and the prediction results. Although this simple network is not able to separate all the three classes, we can extract

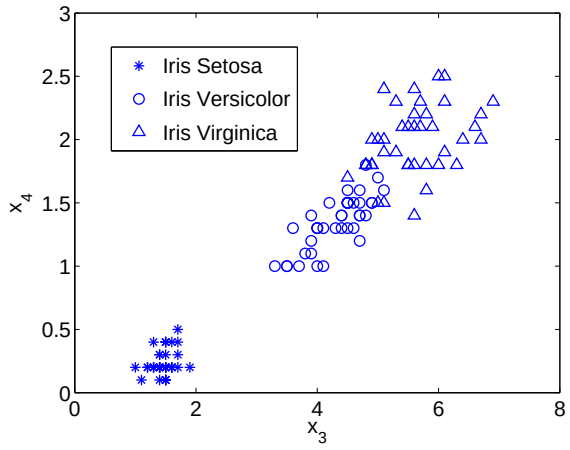

Fig. 1.12. The distribution of the Iris data set.

the following single rule, which is able to separate class 1 from the other two classes.

$$\text{R1: If } x_3 \text{ (petal length) } \leq 2.2, \text{ then Iris Setosa.} \tag{1.26}$$

If we look at the class distribution in Fig. 1.13, we find that this simple rule effectively captures the condition under which class 1 can be separated from the other 2 classes. Let us now take a closer look at the input-output relation, refer to Fig.1.18, we find that attribute x_3 and x_4 alone are able to separate class 1. However, it is clearly more advantageous to choose attribute x_3 than x_4, and our optimization method has selected the optimal feature for separating class 1 from classes 2 and 3.

Now let us investigate the second simplest Pareto-optimal neural network. Similarly, we can extract the following two logic rules:

$$\begin{aligned}
&\text{R1: If } x_3 \text{ (petal length) } \leq 2.2 \wedge x_4 \text{ (petal width) } \leq 1.0, \text{ then Iris Setosa; or} \\
&\text{R2: If } x_3 \text{ (petal length) } > 2.2 \wedge x_4 \text{ (petal width) } \leq 1.4, \text{ then Iris Versicolor; or} \\
&\text{R3: If } \quad x_4 \text{ (petal width) } \geq 1.8, \quad \text{then Iris Virginica;}
\end{aligned} \tag{1.27}$$

The correctness of the rules can be checked by looking at the input-output distribution of the data in Fig. 1.18.

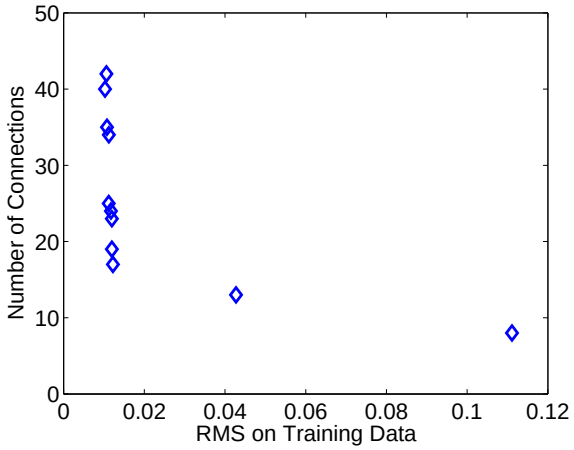

Fig. 1.13. Pareto-optimal solutions generated from the Iris data.

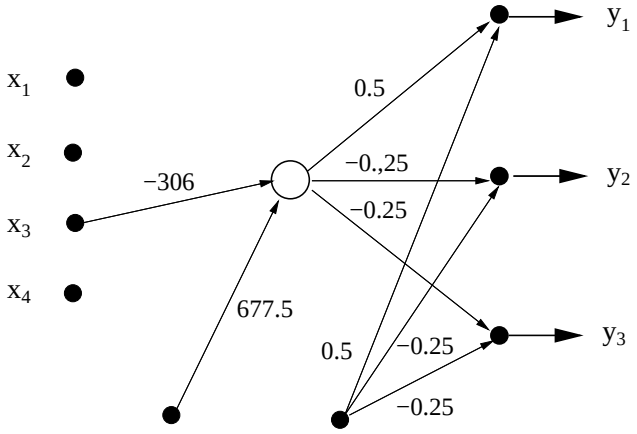

Fig. 1.14. The neural network structure of the simplest Pareto-optimal solution.

1.6 Conclusions

Extracting correct and interpretable rules from data is of great interest for data mining. This chapter suggests a method for generating a set of Pareto-optimal using an evolutionary multi-objective optimization algorithm. Among the Pareto-optimal neural networks, we show that interpretable logic rules can be easily extracted from the compact Pareto-optimal solutions. This idea has been verified on the Breast Cancer Diagnosis data set and the Iris data set. We show that the compact Pareto-optimal neural networks, though very simple and not perfect in terms of approximation accuracy, are able to select the most relevant attributes and effectively separate the classes in the data set.

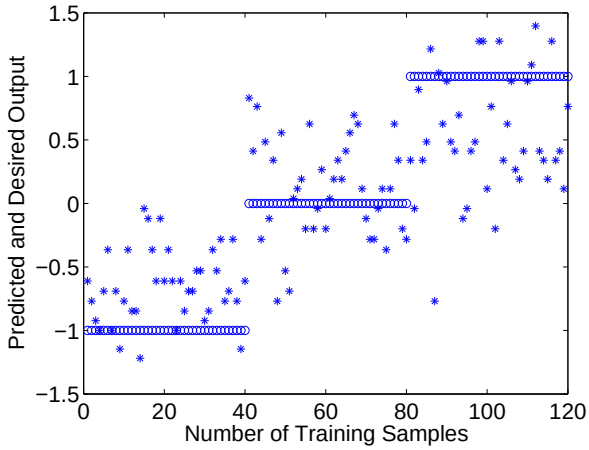

Fig. 1.15. The desired and predicted results of the simplest Pareto-optimal neural network on the training data.

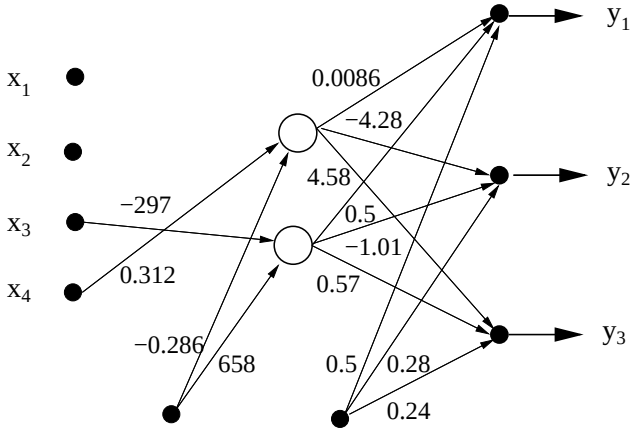

Fig. 1.16. The neural network structure of the second simplest Pareto-optimal solution.

When the complexity increase, it is then difficult to extract understandable logic rules from the neural networks, though the approximation accuracy is better.

As discovered in our previous study [11], the Pareto-optimal solutions that are located in the knee part of the Pareto-front are the most likely the ones that generalize well on unseen data. Together with the findings in this work, we show that Pareto-based multi-objective approach to machine learning is advantageous over traditional approaches where different objective functions are summed up. This advantage is made possible by achieving a Pareto front

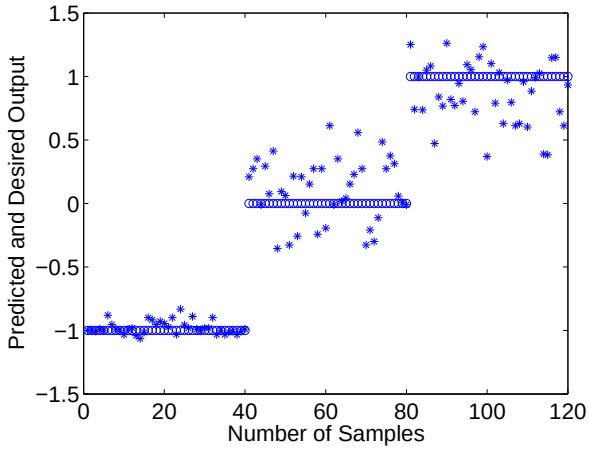

Fig. 1.17. The desired and predicted results of the second simplest Pareto-optimal neural network on the training data.

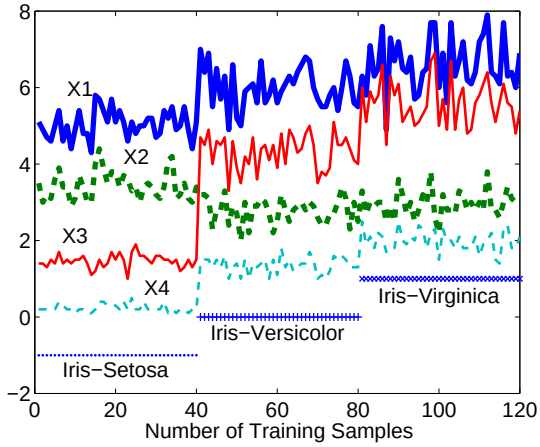

Fig. 1.18. The input-output distribution of the Iris data.

consisting of a number of solutions, which is able to reveal much deeper insights into the system than a single neural network.

References

1. 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

2. Y. Jin and B. Sendhoff. Extracting interpretable fuzzy rules from RBF networks. *Neural Processing Letters*, 17(2):149–164, 2003
3. Y. Jin. *Advanced Fuzzy Systems Design and Applications*. Springer, Heidelberg, 2003
4. W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for rule-based data understanding. *Proceedings of the IEEE*, 92(5):771–805, 2004
5. E. Kolman and M. Margaliot. Are artificial neural networks white boxes? *IEEE Transactions on Neural Networks*, 16(4):844–852, 2005
6. T.A. Etchell and P.J.G. Lisboa. Orthogonal search-based rule extraction (OSRE) for trained neural networks: a practical and efficient approach. *IEEE Transactions on Neural Networks*, 17(2):374–38, 2006
7. S. Mitra, S.K. Pal, and P. Mitra. Data mining in soft computing franework: A survey. *IEEE Transactions on Neural Networks*, 13(1):3–14, 2002
8. Y. Jin, W. v. Seelen, B. Sendhoff. Generating FC3 fuzzy rule systems from data using evolution strategies. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 29(6):829–845, 1999
9. Y. Jin. Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement. *IEEE Transactions on Fuzzy Systems*, 8(2):212-221, 2000.
10. Y. Jin, B. Sendhoff, and E. Körner. Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations. *The Third International Conference on Evolutionary Multi-Objective Optimization*, pages 752–766, 2005
11. Y. Jin, B. Sendhoff, and E. Körner. Simultaneous generation of accurate and interpretable neural network classifiers. In: *Multi-objective Machine Learning*, Y. Jin(ed.), Chapter 14, pages 291–312, 2006
12. Y. Jin (editor). Multi-Objective Machine Learning. Springer, Berlin, 2006.
13. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.
14. K.P. Burnham and D.R. Anderson. *Model Selection and Multimodel Inference*. Springer, New York, second edition, 2002.
15. C. Coello Coello, D. Veldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic, New York, 2002.
16. K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, 2001.
17. 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.
18. W. Duch, R. Adamczak, and K. Grabczewski. Extraction of logical rules from backpropagation networks. *Neural Processing Letters*, 7:1–9, 1998.
19. J.A. Fodor and Z.W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(3):3–71, 1988.
20. J. Gabrieli, R. Poldrack, and J. Desmond. The role of left prefrontal cortex in langrange and memory. *Proceedings of the National Academy of Sciences*, 95:906–913, 1998.
21. 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.

22. 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.
23. 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.
24. M. Ishikawa. Rule extraction by successive regularization. *Neural Networks*, 13:1171–1183, 2000.
25. Y. Jin, T. Okabe, and B. Sendhoff. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation*, pages 1–8. IEEE, 2004.
26. D.A. Miller and J.M. Zurada. A dynamical system perspective of structural learning with forgetting. *IEEE Transactions on Neural Networks*, 9(3):508–515, 1998.
27. L. Prechelt. PROBEN1 - a set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.
28. R.D. Reed and R.J. Marks II. *Neural Smithing*. The MIT Press, 1999.
29. M. Riedmiller and 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.
30. R. Setiono. Generating concise and accurate classification rules for breast cancer disgnosis. *Artificial Intelligence in Medicine*, 18:205–219, 2000.
31. R. Setiono and H. Liu. Symbolic representation of neural networks. *IEEE Computer*, 29(3):71–77, 1996.
32. I. Taha and J. Ghosh. Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):448–463, 1999.
33. E. Anderson. The Irises of the gaspe peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935