Construction Features and Data Analysis by BP-SOM Modular Neural Network

Vladimir Gridin and Vladimir Solodovnikov

Design information technologies Center Russian Academy of Sciences,
Odintsovo, Moscow region, Russia
info@ditc.ras.ru
http://ditc.ras.ru

Abstract. Data are a valuable resource that keeps a great potential for recovery of the useful analytical information. One of the most promising toolkits to solve the problems of data mining could be the usage of neural network technology. The problem of initial parameters values and ways of neural network construction on example of a multilayer perceptron are considered. Also information about the task and available raw data are taken into account. The modular BP-SOM network, which combines the multi-layered feed-forward network with the Back-Propagation (BP) learning algorithm and Kohonens self-organising maps (SOM), is suggested for visualization of the internal information representation and the resulting architecture assessment. The features of BP-SOM functioning, methods of rule extraction from trained neural networks and the ways of the result interpretation are presented.

Keywords: neural network, multilayer feedforward network, Kohonen self-organizing maps, modular network BP-SOM, rules extraction.

1 Introduction

Data analysis processes are often related to the tasks which are characterized by the lack of information about the sample structure, dependencies and distributions of analyzed indicators. One of the closest correspondences to this condition could be the usage of an approach based on neural network technology. The ability of neural networks to train, simulate nonlinear processes, deal with noisy data, extract and generalize the essential features from the incoming information makes them one of the most promising toolkits in solving data mining problems. However, there are several difficulties in using of this approach. In particular there is the problem of choosing an optimal network topology, parameter values and structural features that would best meet the problem being solved on available raw data. This is due to the fact that different neural networks can show very similar results on the samples from training set and significantly different when working with new, never shown data. Designing the optimal topology of the neural network can be represented as a search of architecture that provides the best (relative to the chosen criterion) solution of a particular problem. Usually the particular architecture and structural features could be selected on the

results of the assessment based on knowledge of the problem and the available source data. After that, the training and testing processes are taking place. Their results are used in the decision-making process, that the network meets all the requirements.

Another complication in using neural network approach could be related with the results interpretation and its preconditions. Especially clearly, this problem appears for the multilayer perceptron (MLP) [1]. In fact, neural network acts as a "black box", where the source data are sent to the input neurons and the result is got from the output, but the explanation about the reasons of such a solution is not provided. The rules are contained in the weight coefficients, activation functions and connections between neurons, but usually their structure is too complex for understanding. Moreover, in the multilayer network, these parameters may represent non-linear, non-monotonic relationship between input and target values. So, generally, it is not possible to distinguish the influence of a certain characteristic to the target value, because the effect is mediated by the values of other parameters. Also, you may experience some difficulties in using the learning algorithm of back-propagation BP, both with local extremes of the error function, and with the solutions of a certain class of problems.

2 The architecture and initial values choice for the neural network

The choice in favor of neural network architecture can be based on knowledge of the problem being solved and the available source data, their dimension and the samples scope. There are different approaches for choosing the initial values of the neural network characteristics. For example, the "Network Advisor" of the ST Neural Networks package offers by default one intermediate layer with the number of elements equals to the half of the sum of the quantity of inputs and outputs for the multilayer perceptron. In general, the problem of choosing the number of hidden elements for the multilayer perceptron should account two opposite properties, on the one hand, the number of elements should be adequate for the task, and on the other, should not be too large to provide the necessary generalization capability and avoid overfitting. In addition, the number of hidden units is dependent on the complexity of the function, which should be reproduced by the neural network, but this function is not known in advance. It should be noted that while the number of elements increases, the required number of observations also increases. As an estimate, it is possible to use the principle of joint optimization of the empirical error and the complexity of the model, which takes the following form [1]:

$$\min\{description\_of\_the\_error + description\_of\_the\_model\}. \qquad (1)$$

The first part of this expression is responsible for the accuracy of the approximation and the observed learning error. The less it is - the less bits it is needed to correct the model predictions. If the model predicts the data accurately, then the

length of the error description equals to zero. The second part makes sense to the amount of information needed to select a specific model from the set of all possible. Its accounting allows to apply the necessary constraints on the complexity of the model by suppressing an excessive amount of tuning parameters.

The accuracy of the neural network function approximation increases with the number of neurons in the hidden layer.

When there are h neurons the error could be estimated as $O\left(\frac{1}{h}\right)$. Since the number of outputs in the network does not exceed, and typically much smaller than the number of inputs, so the main number of weights in the two-layer network would be concentrated in the first layer, i.e. $w = i \cdot h$, where i - is the input dimension. In this case, the average approximation error is expressed by the total number of weights in the network as follows: $O\left(\frac{i}{w}\right)$.

The network description is associated with the models complexity and is basically comes down to the consideration of the amount of information in the transmission of its weights values through some communication channel. If we accept the hypothesis ψ about the network settings, its weights and the number of neurons, the amount of information (in the absence of noise) while transferring the weights will be $-\log(Prob)$, where $Prob$ is the probability of this event before the message arrives at the receiver input. For a given accuracy this description requires about $-\log\ P(\psi) \sim w$ bit. Therefore, a specific error for one pattern associated with the complexity of the model could be estimated as: $\sim \frac{w}{p}$, where p is the number of patterns in the training set. The error decreases monotonically with increasing the number of patterns. So Haykin, using the results from the work of Baum and Hessler, gives recommendations about the volume of a training sample relative to the number of weighting coefficients and taking into account the proportion of errors allowed during the test, which can be expressed by the inequality: $p \geq \frac{w}{\varepsilon}$, where ε is the proportion of errors which allowed during testing. Thus, when 10% of errors are acceptable then the number of training patterns must be 10 times greater than the number of available weighting coefficients in the network.

Thus, both components of the network generalization error from expression (1) were considered. It is important that these components are differently depend on the network size (number of weights), which implies the possibility of choosing the optimal size that minimizes the total error:

$$Expression\ (1) \sim \frac{i}{w} + \frac{w}{p} \geq 2 \cdot \sqrt{\frac{i}{p}}. \tag{2}$$

Minimum error (equal sign) is achieved with the optimal number of the weights in the network: $w \sim \sqrt{p \cdot i}$ which corresponds to the number of neurons in the hidden layer:

$$h = \frac{w}{i} \sim \sqrt{\frac{p}{i}}, \tag{3}$$

where i, h are the quantity of neurons in input and hidden layers, w is the number of weights and p is the amount of patterns in the training sample.

3 Evaluation of the resulting architecture and parameters

After the network architecture was selected the learning process is carried out. In particular, for a multi-layer perceptron this may be the error back-propagation (BP) algorithm. One of the most serious problems is that the network is trained to minimize the error on the training set, rather than an error that can be expected from the network while it will process completely new patterns. Thus, the training error will differ from the generalization error for the previously unknown model of the phenomenon in the absence of the ideal and the infinitely large training sample [2].

Since the generalization error is defined for data, which are not included in the training set, the solution may consist of separation of all the available data into two sets: a training set, which is used to match the specific values to the weights, and validation set, which evaluates the predictive ability of the network and selects the models optimal complexity. The training process is commonly stops with consideration of "learning curves" which track dependencies of learning and generalization errors according to the neural network size [3, 4]. The optimum matches to local minima and points, where the graphs meet asymptote. Figure 1 shows the stop point, which corresponds to the validation error minimum (dash-dotted line), while the training error (solid line) keeps going down.

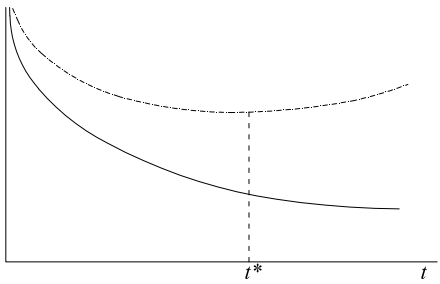

Fig. 1. The stop training criterion at time t*.

Another class of learning curves uses the dependencies of the neural network internal properties to its size, and then mapped to an error of generalization. For example, in [3] the analysis of the internal representation of the problem being solved, relationship of the training error and the maximum sum of the synapse weights modules attributable to the neuron of network are carried out. Also there are variants of generalized curves, which are based on dependence of the wave criterion from the neural network size [5] or perform a comparison of the average module values of synapse weights [6].

In simplified form, the following criteria could be formulated to assess the already constructed neural network model:

- if the training error is small, and testing error is large, it means that the network includes too much synapses;
- if the training error and testing error is large, then the number of synapses is too small;
- if all the synapse weights are too large, it means that there are too few synapses.

After evaluation of the neural network model, the decision-making process is taking place about the necessity of changing the number of hidden elements in one or another direction, and the learning process should be repeated. It is worth to mention that modified decision trees, which are based on the first-order predicate logic, could be applied as a means of decision making support and neural network structure construction automation.

4 Rules extraction and results interpretation

Generally speaking, there are two approaches to extract rules from the multilayer neural networks. The first approach is based on extraction of global rules that characterize the output classes directly through the input parameter values. An alternative is in extraction of local rules, separating the multilayer net on a set of single-layer networks. Each extracted local rule characterizes a separate hidden or output neuron based on weighted connections with other elements. Then rules are combined into a set, which determines the behavior of the whole network.

The NeuroRule algorithm is applicable for rules extraction from the trained multilayer neural networks, such as perceptron. This algorithm performs the network pruning and identifies the most important features. However, it sets quite strict limitations on the architecture, the number of elements, connections and type of activation functions. As an alternative approach may be highlighted TREPAN type algorithms which extract structured knowledge not only of the extremely simplified neural networks, but also arbitrary classifiers in the form of a decision tree [7]. However, this approach does not take into account structural features that can introduce additional information.

The decision of such kind of problems could be based on the usage of the modular neural network BP-SOM[8].

5 Modular neural network BP-SOM

5.1 Network architecture

The main idea is to increase the reactions similarity of the hidden elements while processing the patterns from the sample, which belong to the same class. The traditional architecture of the direct distribution network [1], in particular the multi-layer perceptron with the back-propogation learning algoritm (BP), combines with the Kohonen self-organizing maps (SOM) [9], where each hidden layer of the perceptron network is associated with a certain self-organizing map. The structure of such a network is shown in Figure 2.

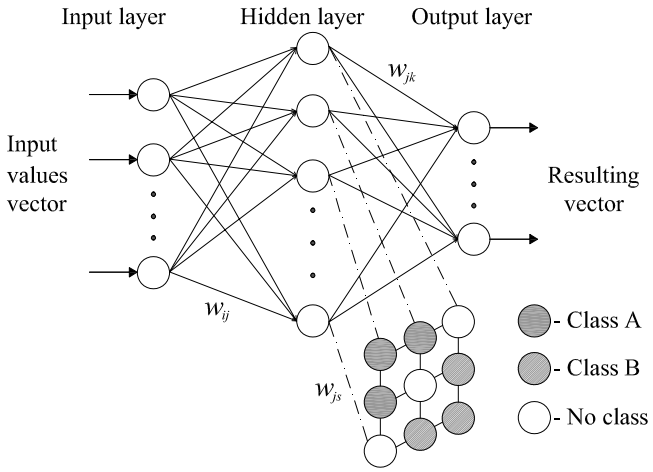

Fig. 2. The modular neural network (BP-SIM) with one hidden layer.

5.2 Learning algorithm

BP-SOM learning algorithm largely corresponds a combination of algorithms which are specific to the learning rules of its component parts [8]. First, the initial vector from the training sample is fed to the input of the network and its direct passage is carried out. At the same time, the result of neurons activation in each hidden layer is used as a vector of input values for the corresponding SOM network. Training of SOM components is carried out in the usual way and ensures their self-organization. In further, this self-organization is used to account classes, which tags are assigned to each element of the Kohonen maps. For this purpose, a counting is taking place, which purpose is to get the number of times the SOM-neuron became the winner and determine what class the initial vector of training sample belongs to. The winner is chosen from the SOM-neuron, whose weights vector is the closest in terms of the Euclidean distance measure to the output values vector of the hidden layer neurons. The most common class is taken as the mark. Reliability is calculated from the ratio of the number of class mark occurrences to the total number of victories of the neuron, i.e. for example, if the SOM-neuron became 4 times winner of class A and 2 times for class B, class A label with certainty 4/6 is selected. The total accuracy of the self-organizing map is equal to the average reliability of all elements of the card. Also, SOM allows data visualizing and displays areas for the various classes (Fig. 2). Learning rule of the multilayer perceptron component part is carried out by the similar to BackPropagation (BP) algorithm, minimizing aggregate square error:

$$BP_{Error} = \sum_i (d_i - y_i)^2, \tag{4}$$

where index i runs through all outputs of the multi-layer network, d_i is the desired output of the neuron i, y_i is the current output of the neuron i from

the last layer. This error is transferred over the network in the opposite direction from the output to the hidden layers. Also, an additional error component SOM_{Error} for neurons from hidden layers is introduced, which is based on information about the particular class of input vector and taking into account the self-organizing map data. Thus, in the SOM, which corresponds to the current hidden layer, the searches for an special element V_{SOM} is taking place. This element should be closest, in terms of Euclidean distance, to the output vector of the hidden layer V_{hidden}, and be the same class label as the input vector. The distance between the detected vector V_{SOM} and the vector V_{hidden} is taken as the error value SOM_{Error} and accounted for all of the hidden layer neurons. If the item V_{SOM} is not found, then the error value SOM_{Error} is assumed to be 0. Thus, the total error for the neurons of the hidden layer takes the form:

$$BP\_SOM_{Error} = (1 - \alpha) \cdot BP_{Error} + r \cdot \alpha \cdot SOM_{Error}, \tag{5}$$

where BP_{Error} is the error of perceptron (from BackPropagation algorithm); SOM_{Error} is the error of the winner neuron from Kohonen network; r is the reliability factor of the winner neuron from Kohonen network; α is the influence coefficient of the Kohonen network errors (if the value is equal to 0, then it will become original BP).

The results of the Kohonen maps self-organization are used for changing of the weight coefficients in the process of network training. This provides an effect, in which the activation of neurons in the hidden layer will become more similar to all other cases processing vectors of the same class [10]. The SOM with the dimension 7 per 7 is shown in Figure 3.

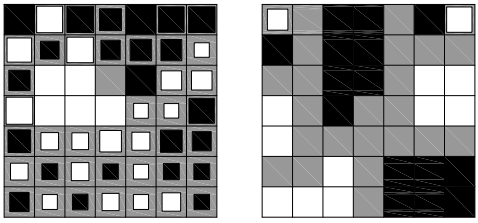

Fig. 3. Coloring of the hidden layer for the Kohonen self-organizing maps, depending on the learning algorithm.

It characterizes the reaction of the hidden layer neurons of the BP-SOM network, which was trained to solve the problem of classifying for two classes. Map on the left corresponds to the base algorithm of back-propagation (BP), the right - with the influence of the Kohonen map, i.e. BP-SOM training. Here white cells correspond to class A, and black cells to class B. In turn, the size of the shaded region of the cell determines the accuracy of the result. So, completely white or completely black cell is characterized by the accuracy of 100% .

This could ensure structuring and visualization of information extracted from data, improve the perception of the under study phenomenon and help in the network architecture selection process. For example, if it is impossible to isolate the areas at the SOM for the individual classes, then there are not enough neurons and their number should be increased. Moreover, this approach can simplify the rules extraction from already trained neural network and provide the result in a hierarchical structure of the consistent rules such as "if-then".

5.3 Rules extraction

As an example, let's consider a small test BP-SOM network that will be trained to solve the classification problem which is defined by the next logical function [11]:

$$F(x_0, x_1, x_2) = (x_0 \wedge \bar{x_1} \wedge \bar{x_2}) \vee (\bar{x_0} \wedge x_1 \wedge \bar{x_2}) \vee (\bar{x_0} \wedge \bar{x_1} \wedge x_2)$$

This function is set to True (Class 1) only in case where one of the arguments is True, otherwise the function value is False (Class 0). Two-layer neural network could be used for implementation, which consists of three input elements, three neurons in the hidden layer, and two neurons in the resulting output layer. The dimension for the Kohonen map for the intermediate layer is 3×3 (Figure 4).

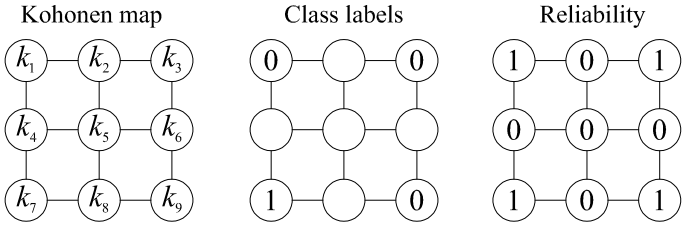

Fig. 4. Kohonen map, class labels and reliability.

Four elements of the Kohonen map acquired class labels with certainty 1 and 5 elements were left without a label, and their reliability is equal to 0 after training (Figure 4).

One of the methods for rules extraction from such a neural network could be the algorithm, which was designed for classification problems with digital inputs. It consists of the following two steps [10,11]:

1. Searching for such groups of patterns in the training set, which are potentially could be combined into individual subsets, each of which is connected to one element of the Kohonen map (Table 1).
2. Then, each of the subsets is examined to identify the values of the inputs that have constant value in the subset. For example, in the subgroup associated with the element k_1 all the attributes x_0, x_1 and x_2 have a constant value 0, and for the element k_3 value 1 respectively.

SOM element	Class	x_0	x_1	x_2
k_1	0	0	0	0
k_3	0	1	1	1
k_7	1	0	0	1
		0	1	0
		1	0	0
k_9	0	0	1	1
		1	0	1
		1	1	0

Table 1. Splitting the training set into groups with the reference to a specific element of the SOM.

Thus, it is possible to get the following two rules from the Table 1:

- $IF(x_0 = 0 \wedge x_1 = 0 \wedge x_2 = 0)THEN(Class = 0)$;
- $IF(x_0 = 1 \wedge x_1 = 1 \wedge x_2 = 1)THEN(Class = 0)$;

However, it is rather problematic to use this way to distinguish rules for elements k_7 and k_9. Of course, it is possible to compose a disjunction of all the possible options for each SOM-element, but these rules would be complicated in perception.

Additional available information consists of the attributes sum for each patterns from the training sample. It is easy to notice that each SOM-element is responsible for a certain obtained sum value (Table 2).

	k_1	k_3	k_7	k_9
Sum	0	3	1	2
Class	0	0	1	0

Table 2. The sum of the attribute values for each patterns from the training sample concerning the SOM-elements.

These considerations could be summarized. We need to find constraints on the attribute values of the input vectors and their weight coefficients for extraction rules, which corresponds to Kohonen map elements. This may be done by back-propagation of minimum and maximum values of the neuron activation back to the previous layer, i.e. we have to apply the function $f^{-1}(V_i^{Cur})$ (inverse to activation function) to the output neuron value [12].

$$f^{-1}(V_i^{Cur}) = f^{-1}(f(\sum_j w_{ji} V_j^{Prev} + bias_i)) = \sum_j w_{ji} V_j^{Prev} + bias_i, \qquad (6)$$

where V_i^{Cur} is the i-th neuron output from the current layer, V_j^{Prev} is the output of the j-th neuron of the previous layer. Assuming that sigmoid was used

as the neurons activation function, then in this case we will get:

$$f^{-1}(V_i^{Cur}) = -\ln(\frac{1}{V_i^{Cur}} - 1). \quad (7)$$

Additionally, it is known that self-organizing map elements, which are connected to the elements of the first hidden layer of the perceptron, will respond to proximity of the weight vectors and outputs of the hidden layer neurons. Therefore, it is proposed to replace the back-propagation neural activation of the hidden layer to the weight vector values of the self-organization map element during the rules construction for each SOM-element. For example, if the hidden layer contains neuron A, and the weight between this neuron and SOM-element k was denoted by w_A^k, then the restriction takes the form:

$$f^{-1}(w_A^k) - bias_A \approx \sum_j w_{jA}x_j \text{ or } 1 \approx (\sum_j w_{jA}x_j)/(f^{-1}(w_A^k) - bias_A). \quad (8)$$

Such restrictions, which were obtained for all the neurons of the hidden layer, could be used for construction of following rules:

$$IF\ (\wedge_i(f^{-1}(w_A^k) - bias_A \approx \sum_j w_{jA}x_j))\ THEN\ (Class = Class(k)). \quad (9)$$

Similar rules are required for all SOM-components, the reliability of which exceeds a certain threshold.

If we apply the considered method for the initial example, then four sets of restrictions would be obtained. Each set includes three restrictions, which corresponds to the number of neurons in the hidden layer of the perceptron.

For SOM-element k_1 :

- $1 \approx 687 * x_0 + 687 * x_1 + 687 * x_2$;
- $1 \approx 738 * x_0 + 738 * x_1 + 738 * x_2$;
- $1 \approx 1062 * x_0 + 1062 * x_1 + 1062 * x_2$.

Taking into account that $x_0, x_1, x_2 \in \{0, 1\}$ then the best results may be achieved if $x_0 = 0$, $x_1 = 0$ and $x_2 = 0$.

For element k_3 all restrictions coincide and have the form:

- $1 \approx 0.33 * x_0 + 0.33 * x_1 + 0.33 * x_2$;

Thus, the values are as follows: $x_0 = 1$, $x_1 = 1$, $x_2 = 1$.

For element k_7 restrictions coincide and take the form:

- $1 \approx x_0 + x_1 + x_2$;

This corresponds to the case when only one of the attributes is equal to 1.

Restrictions for element k_9:

- $1 \approx 0.5 * x_0 + 0.5 * x_1 + 0.5 * x_2$;

This condition characterizes the case when two of three attributes are 1.

Thus, it is obtained the following set of rules, if all restrictions would be generalized:

- $IF\ (x_0 + x_1 + x_2 \approx 0)\ THEN\ (Class = 0)$
- $IF\ (x_0 + x_1 + x_2 \approx 3)\ THEN\ (Class = 0)$
- $IF\ (x_0 + x_1 + x_2 \approx 1)\ THEN\ (Class = 1)$
- $IF\ (x_0 + x_1 + x_2 \approx 2)\ THEN\ (Class = 0)$

It is easy to notice that this set correctly describes all the elements of a self-organizing network.

6 Conclusion

Approaches for the selection of the initial values of the neural network parameters were considered. The analysis of the training process, its stopping criteria and evaluation of the received architecture were carried out. Combining different neural network architectures, such as multi-layer perceptron with the back-propagation learning algorithm, and Kohonen's self-organizing maps, could bring additional possibilities in the learning process and in the rules extraction from trained neural network. Self-organizing maps are used both for information visualization, and for influencing the weights changes during the network training process. That provides an effect, in which the neurons activation in the hidden layer will become increasingly similar to all the cases processing vectors of the same class. This ensures the extracted information structuring and has the main purpose to improve the perception of the studied phenomena, assist in the process of selecting the network architecture and simplify the extraction rules. The results could be used for data processing and hidden patterns identification in the information storage, which could become the basis for prognostic and design solutions.

Acknowledgeents. Work is carried out with the financial support of the RFBR (grant 15-07-01117-a).

References

1. Ezhov A. A., Shumskiy S. A. Neyrokomp'yuting i ego primeneniya v ekonomike i biznese. Moscow, MEPhI Publ., 1998. (in Russian)
2. Bishop C.M. Neural Networks for Pattern Recognition. Oxford Press. 1995.
3. Watanabe E., Shimizu H. Relationships between internal representation and generalization ability in multi layered neural network for binary pattern classification problem /Proc. IJCNN 1993, Nagoya, Japan, 1993. Vol.2. pp. 1736-1739.
4. Cortes C., Jackel L. D., Solla S. A., Vapnik V., Denker J. S. Learning curves: asymptotic values and rate of convergence / Advances in Neural Information Processing Systems 7 (1994). MIT Press, 1995. pp. 327-334.

5. Lar'ko, A. A. Optimizaciya razmera nejroseti obratnogo rasprostraneniya. [Electronic resource]. http://www.sciteclibrary.ru/rus/catalog/pages/8621.html.
6. Caregorodcev V.G. Opredelenie optimal'nogo razmera nejroseti obratnogo rasprostraneniya cherez sopostavlenie srednih znachenij modulej vesov sinapsov. /Materialy 14 mezhdunarodnoj konferencii po nejrokibernetike, Rostov-na-Donu, 2005. T.2. S.60-64. (in Russian)
7. Gridin V.N., Solodovnikov V.I., Evdokimov I.A., Filippkov S.V. Postroenie derev'ev reshenij i izvlechenie pravil iz obuchennyh nejronnyh setej / Iskusstvennyj intellekt i prinyatie reshenij 2013. 4 Str. 26-33. (in Russian)
8. Weijters, A. The BP-SOM architecture and learning rule. Neural Process-ing Letters, 2, 13-16, 1995.
9. T. Kohonen. Self-Organization and Associative Memory, Berlin: Springer Verlag, 1989.
10. Ton Weijters, Antal van den Bosch, Jaap van den Herik Interpretable neural networks with BP-SOM, Machine Learning: ECML-98, Lecture Notes in Computer Science Volume 1398, 1998, pp 406-411.
11. J.Eggermont, Rule-extraction and learning in the BP-SOM architecture, Thesis 1998.
12. Sebastian B. Thrun. Extracting provably correct rules from artificial neural networks. Technical Report IAI-TR-93-5, University of Bonn, Department of Computer Science, 1993.