DECOMPOSITIONAL RULES EXTRACTION METHODS FROM NEURAL NETWORKS

Andrey Bondarenko, Tatjana Zmanovska, Arkady Borisov

Riga Technical University
Desicion Support Systems Group
Kalku str. 1, Riga, LV-1658
Latvia
andrejs.bondarenko@gmail.com, zm@itl.rtu.lv, arkadijs.borisovs@cs.rtu.lv

Abstract: *Given paper is a review on existing decompositional rules extraction methods from artificial neural networks of several types: feed-forward network, radial basis functions network, second order reccurent network, generalized relevance learning vector quantization and finally support vector machine. Descriptions of all rules extraction methods are containing details on method itself, type of rules extracted, applicable problems and some test results.*

Keywords: *rule extraction, neural network, pruning, support vector machine, reccurent network, RBF network.*

1 Introduction

In recent years many authors came up with publications devoted to symbolic rules extraction methods [17] from neural networks and other "black-box" classification methods. It was shown [1], [2], [3] that many of these methods perform classification tasks better than more formal decision trees based methods like C4.5 algorithm [26] and other similar methods [27]. The main drawback of "black-box" algorithms is lack of formality and expressive power which limits their application field. Absence of explanations on how these methods are classifying input vector or make approximation of function, as well as what input vector parameters are dominating among others prevents systematization and creation of new knowledge and present little help to domain experts. Because of mentioned problems, model can not be validated by domain expert, and therefore cannot be used in healthcare, nuclear industry, financial sector and other mission-critical areas. This paper gives an overview on current situation in this area covering approaches on rules extraction from feed-forward artificial neural networks, generalized relevance learning vector quantization classifier (GRLVQ), recurrent neural network and from linear support vector machine (SVM).

Although this paper doesn't present any new algorithm in neural rules extraction area it gives an overview on rules extraction methods. It describes main goals behind these algorithms, application areas for such algorithms comprehensiveness of extracted rules and some evaluation. The rest of this paper is organized as follows. **Section 2** briefly introduces rule extraction from neural networks. **Section 3** provides description of algorithms. **Section 4** concludes.

2 Overview

2.1 Definition

We are using definition for 'rule extraction' term proposed by [7]. As it is more broad than other definitions like [4] and underlines fact that extracted rules can take different forms, not only lexical.

"Given an opaque predictive model and the data on which it was trained, produce a description of the predictive models hypothesis that is understandable yet closely approximates the predictive models behavior"

Another important point here to note is duality between terms *comprehensibility* and *accuracy* of extracted rules. While first assumes extracted knowledge is tackling correlations between parameters of model in simplest and clear way, second assumes these rules are as accurate as possible. What point is more desirable depends on problem context, often rule extraction algorithms can be tuned by applying some parameters to produce either more comprehensible, and hence compact and understandable, or more accurate rules.

2.2 Motivation

Driving force which moves research in this area is fact that intermediate "black-box" models are performing much better than "white-box" models. As a consequence rules extracted from these algorithms have higher quality than those extracted from "white-box" methods. But on the other hand usage of "black-box" models are prohibited in some areas like medicine and financial sector, where domain expert verification is often a requirement, as well as in data mining applications where aim of the whole process is to acquire new knowledge about input data. All this is impossible without extraction of rules from "black-box" models.

2.3 Rules Types

It's worth to note that extracted rules can significantly differ in their nature and hence in their readability for domain expert depending on extraction method used. Each type of rules is describing significance and inner correlations of input parameters in different manner. Often there is tradeoff between readability and high classification rate. Below we will briefly describe each type of logical rules as per [7]:

Propositional If-Then / If-Then-Else rules Rules of this type consist of *If* conditional clause and *Then* classification clause. Optionaly rule can contain *Else* classification clause. The *If* condition part of a propositional rule is a boolean combination of logical conditions on the input variables. Condition part can contain conjunctions, disjunctions and negations. An example of such a rule is: "$If X = x$ and $Y = y$ then $Class = A$", with X, Y input variables and x, y possible values of these variables. For continuous input variables, the conditions are usually specified in form of range restrictions on the values, e.g "$X \in [c_1, c_2]$ and $Y > c_3$ with $c_1, c_2, c_3 \in \mathbb{R}$".

M-of-N rules This type of rules is closely related to propositional rules. They are expressions of the form "If (at least/exactly/at most) M of the N conditions $C_1, C_2, \ldots, C_N$ are satisfied then $Class = A$". These rules usually can be easily transformed into *propositional rules.* For example, the M-of-N rule "if exactly 2-of-$X = a_1$, $Y = a_2$, $Z = a_3$ then $Class = A$ is logically equivalent to "if $((X = a_1$ *and* $Y = a_2)$ or $(X = a_1$ *and* $Z = a_3)$ or $(Y = a_2$ *and* $Z = a_3))$ then $Class = A$". Observe that for $M = 1$ all rules can be written as collection of disjunctions, while in case of $M = N$ expression will be conjunction of conditions.

Oblique rules While previously mentioned types of rules are denoting regions of hyperspace by defining hyperplanes parallel to axises it sometimes might require large amount of conditions to define classification with satisfactory quality. On the other hand oblique rules are capable of defining picewise hyperplanes not parallel to axises of the input space, thus smaller amount of rules can satisfactory divide space into subspaces. These rules take form $If(a_1X_1 + a_2X_2 + a_3 > a_4)$ and $(a_4X_1 + a_5X_3 > c_6)$ then $Class = A$.

Equation rules This type of rules is very similar to oblique rules, but defines boundaries using equations of slightly more complex hyperplanes, like ellipsoids : $If(a_1X_1^2 + a_2X_2^2 + a_3X_1X_2 + a_4X_1 + a_5X_2 + a_6) \leq a_7$ then $Class = B$. Drawback of these rules is that they are more difficult to understand.

Fuzzy rules Rules of this type are very similar to propositional *If-Then-Else* rules, the only difference is that instead of boolean logic, fuzzy rules are multi-valued. In fuzzy rules the universe of discourse of the variables is a fuzzy set. Example of fuzzy classification rule is: "$If\ X\ is\ high\ and\ Y\ is\ medium\ then\ Class = A$". Here *high*, *medium* and *low* are members of fuzzy set. In fuzzy sets each element is associated with corresponding grade of membership. As well as propositional rules fuzzy rules are generally good understandable as they operating with linguistic concepts.

As well we should note that knowledge can be extracted from network in different forms, like visual representation or state machines. Main factor of evaluation of extracted rules is their quality. More specifically quality is equal to comprehensibility which in turn is more or less subjective measure in comparison to accuracy, often maximizing one leads to some degradation of other.

Taxonomy According to taxonomy proposed in [5] and [6] we are covering no pedagogical or eclectic algorithms - only decompositional ones. Decompositional - means that covered algorithms are trying to extract knowledge from internal structure of neural network or support vectors of SVM. Thus internals of "black-box" are essential to such algorithms.

Pedagogical rules are not interested in inner structure of classificator as they are trying to build rules by comparing input and output of "black-box" using it as a classificator and in some cases as generator of aditional synthetic data.

Eclectic rules are mix of both decompositional and pedagogical methods. They are using knowledge about inner structure to complement pedagogical method.

3 Survey

3.1 Feed-forward network

Training

Many of decompositional rules extraction algorithms are trying to extract knowledge from feed-foward network interconnections structure. Usually interconnections are mapped into lexical rules. Several approaches exist here, one of them is to start with minimal neurons count and gradually add additional neurons to get statisfactory error rate [8], [9] the others are removing extra neurons or weights until needed precision is reached. Methods described below are using adjusted learning algorithm which prevents significant weights from growing too big and ensures nonsignificant weights are staying small. In other words network is trained to minimize error rate with minimal count of weights. This is essential for getting compact set of rules. Both covered methods NeuroRule and NeuroLinear are using same training-prunning algorithm described as follows.

Algorithm trains standard feed-forward network by minimizing augmented cross-entropy error function

$$\theta(w,v) = P(w,v) - \sum_{i=1}^{C}\sum_{p=1}^{P}[t_{ip}\log S_{ip} + (1-t_{ip})\log 1 - S_{ip})]$$

Here $P(W,v)$ is a penalty term which is introduced to prune network weights of both layers

$$P(w,v) = \epsilon_1 \sum_{j=1}^{J}(\sum_{i=1}^{C}\frac{\beta v_{ij}^2}{1+\beta v_{ij}^2} + \sum_{k=1}^{K}\frac{\beta w_{jk}^2}{1+\beta w_{jk}^2}) + \epsilon_2 \sum_{j=1}^{J}(\sum_{i=1}^{C} v_{ij}^2 + \sum_{k=1}^{K} w_{jk}^2)$$

In the above equations parameters ϵ_1, ϵ_2 and β are positive parameters, the greater they are the less significant weights will remain during training process. By adjusting these parameters it is possible to variate values of weights.

Prunning

Pruning phase is needed to remove unnecesary connections which are not significant, and thus can be removed without increase of error function. Less interconnections will result in less complex rules. Prunning algorithm for both NeuroRule and NeuroLinear are generally the same.

Main goal of prunning is to minimize connections count while preserving desired classification rate. For these special criterias training and prunning algorithm was developed [10], [12].

It should be noted that different neural networks with different active (non-pruned) weights count and accuracy rate are usually generated and feed into rule generation algorithm. This gives researcher a choice between more comprehesive and probably more compact rule set from one side and more accurate rules on the other side.

NeuroRule rules extraction

This apporach is proposed in [10]. First assumption of this algorithm is that input data are discreet. In case of continuous data prior discretization is done either by expert or using discretization algorithms or combining outputs of both approaches.

Before rules extraction phase can start it is necesary to descretize activation values for all neurons. To achieve that clusterization can be used as it is described in [17] and [10]. Once clusterization is done and boundaries of clusters are defined (in sence of ranges of activation values for all neurons) rules extraction algorithm can extract rules.

Rules extraction is done in 2 steps for one hidden layer network.

Step 1 Generate rules that describe hidden layer activation values (clusters) in sence of input layer input values. For this data set is extracted from training data such, that input vector will produce activation values in hidden neurons that will match all of the defined clusters for each hidden neuron. Thus all clusters will be covered with rules.

Step 2 Similar procedure is repeated to generate rules that will describe output layer activation values in respect to hidden layer clustered values.

Step 3 Optionally C4.5 [26] algorithm can be used to optimize generated rules.

Algorithm X2R [13] was developed to automate rules extraction. It extracts rules in general propositional *If-Then* form.

NeuroLinear rules extraction

This decompositional algorithm is capable of extracting oblique classification rules from multi-layered perceptron with one hidden neurons layer. This method was proposed in [12]. It is very similar to previously described NeuroRule method but shows slightly better results in sence of compactness of extracted rules.

First distinction between these two methods is that NeuroLinear works with continuous data and prior dsicretization is not needed as in case with NeuroRule. As a result simbolic rules are presenting piece-wise discriminant functions. These functions are transformed to oblique rules of form:

$$\sum_{x} c_i x_i < \eta$$

where c_i is a real coefficient x_i is the value of the attribute i, and η is a threshold.

It was observed that neural networks trained on continuous data (in case of NeuroLinear) have fewer connections compared to network trained on discretized input data and thus we can expect fewer rules to be extracted from former. According to authors [15] accuracy of both methods is almost the same.

Chi2 [11] is used for discretization as a part of rules extraction algorithm. Rules extraction algorithm is generating set of linear regression rules from a pruned network once the network hidden unit activation function $\tanh(\xi)$ has been approximated by the 3-piece linear function [14] (pp. 7-9):

Evaluation of NeuroRule and NeuroLinear Comparison of these two methods along with others is made in [15]. Generated rules was used for clasification of medical data - diagnosis of hepatobility disorders. The data set consisted of nine continuous medical measurments taken from patients. In table 3.1 it can be seen that both NeuroRule and NeuroLinear are performing much better than competitive Linear DA method and Fuzzy Neural Network.

	Linear DA (%)	Fuzzy NN (%)	NeuroRule (%)	NeuroLinear (%)
Alcoholic Liver Damage (ALD)	57.6	69.7	87.9	97.0
Primary Hepatoma (PH)	64.7	82.4	92.2	98.0
Liver Cirrhosis (LC)	65.7	71.4	80.0	74.3
Cholelithiasis (C)	63.6	81.8	90.9	88.6
Total	63.2	77.3	88.3	90.2

Table 1: NeuroRule and NeuroLinear Comparison

3.2 RBF network

RBF neural network used here is a general feed-forward neural network proposed by [18], [19] with single input layer, a hidden layer of RBF neurons and an output layer of linear units. The responce of the output units is calculated using equation: $\sum_{j=l}^{J} W_{lj} Z_j(x)$ where: W = weight matrix, Z = hidden units acivation values and x = input vector.

The input layer simply transfers input values to RBF neurons which form localized responce to the input pattern. The activation levels of the output units provide an indication of nearness of the input vector to the classes. Learning is normally takes two steps. An unsupervised clustering methods are applied to hidden layer units, while supervised method is applied to the output layer units. Gaussian fucntion used in hidden units is $Z_j(x) = exp(-\frac{||x-\mu^2||}{\sigma_j^2})$ where: μ = n-dimensional parameter vector, σ = width of receptive field and x = input vector.

The adjustable parameters that will affect classification accuracy and that can be used in rules extraction are: Number of basis functions used, location of center of the basis function, width of the basis function and the wights connection hidden RBF units to the linear output units.

Rule extraction algorithm proposed in [16] uses clusterisation of meaningful (large enough) hidden weights between RBF neurons and output classes. After clusterisation phase for each class, it's domain space boundaries between clusters for all factors are defined. After that rules are directly extracted using these boundaries for all dimensions.

Rules are produced in the prpositional *If-Then* form containing conjunctions of boundaries definitions across all meaningful dimensions. Experimental results given by authors ([16]) on Iris and Vibration data sets. The Iris data set is classical one, while Vibration dataset consist of input features that are continuous values representing the spectral interpretation of the running speed (rotations per minute (RPM)) of the motor or fan and the various harmonics that occur at twice, three times running speed etc. This data set contains linearly non-separable datasets. It was noted, that algorithm has prunning abilities and was able to produce rules that reduced role of non-significant parameters. As can be seen from table 3.2 extracted rules have quite low classification rates. Classification results for NeuroLinear is taken from [17] But it should be noted that proposed algorithm (RBF rules extraction)produces rules without default rule and secondly it is assumed that if more than one rule is fired classification is unsuccessfull. Moreover authors claimed that local character of RBF networks is particulary useful for rules extraction, thus they were able to create RBF rules extraction algorithm which produces rules of accuracy equal to the original RBF network.

	Iris	Vibration
RBF network	88 %	75 %
RBF extracted rules	40 %	25 %
Extracted rules count	3 (without default rule)	7 (without default rule)
NeuroRule rules	98 %	–
NeuroRule rules count	2 (with default rule)	–

Table 2: RBF extracted rules accuracy

3.3 Reccurent Neural Network

Rule extraction method described in [20] works on binary recurrent network and uses quantization output values from output nodes . Algorithm proposes hypothesis that if we will get all two-dimensional projections of all possible pairs of output neurons output values, we will get clustered network responses. These clusters correspond to the states of finite-state automata the network has learned. Furthermore these two-dimensional planes must be divided accordingly into equal areas according to quantization level. For example if $q = 3$, then two dimensional plane will have nine equal squares - they will represent possible network states. Although it is not necessary that all of them will be visited during extraction process of deterministic finite automata (DFA). Later network is presented with descreet inputs "0" and "1" and it is analyzed which quantized areas are getting response, so that automata gets constructed.

By supplying different input data patterns we will see different hidden neurons outputs, thus network will be localized in different states. Thus built deterministic finite automata represents underlying model of reccurent neural network. This automata will present knowledge hidden in reccurent neural network.

Table 3.3 shows generalization performance of reccurent neural network. Neural network and generated DFA was classifying strings from alphabet {'0', '1'}, where one class is strings with three or more consecutive 0s and other class is all other strings. Authors specifically note that along with rules extraction there exist possibility of rules insertion into the reccurent neural network.

3.4 Generalized relevance learning vector quantization

Generalized relevance learning vector quantization (GRLVQ) [22] is further improvement of GLVQ method [23] which in turn is based on LVQ algorithm [24].

Tree extraction method [21] is using properties of GRLVQ algorithm. In order to extract so called BB-tree of a trained GRLVQ-network, it is using information gain across different dimensions of input vector (α_i values) and prototype vectors to define boundaries for classication. Extraction starts with separation of input data set starting with dimension which have highest α_i value. Interval boundaries are selected as midpoints between prototypes. All dimensions that have α_i smaller than defined threshold can be ommited as they do

Test Set	Training Set Size	Average Network μ	Average smallest DFA μ
Strings of length 1-15	1000	0.84%	0.01%
	700	1.63%	0.10%
	300	9.74%	3.21%
	100	38.09%	38.19%
	50	57.96%	39.53%
1000 random strings of length 100	1000	4.61%	0.02%
	700	7.07%	1.35%
	300	26.23%	12.00%
	100	48.50%	38.52%
	50	61.17%	42.91%
1000 random strings of length 1000	1000	4.59%	0.02%
	700	7.68%	0.92%
	300	25.89%	12.39%
	100	47.47%	37.38%
	50	60.24%	41.70%

Table 3: Reccurent neural network and extracted DFA comparison

not significantly contribute to classification. As well it is assumed that all idle prototypes, i.e. prototypes with empty receptive fields are deleted.

Method can be presented as recursive tree-extraction procedure which extracts tree nodes N denoted by indexes I^N and intervals. Each leaf L is labeled with a class number $C_L \leq C$:

```
**BB-Tree**$(X, W, \Lambda)$:
    delete all idle prototypes from W,
    if STOP: output a leaf with class $argmax_c|x^i|y^i = c, (x^i, y^i) \in X|$
    else: output an interior node $N$ with $|W|$ children,
        choose $I^N := first(\Lambda)$,
        denote by $[a_1, \ldots, a_{|W|}]$ a sorted list of $x^i_{I^N}|(x^i, y^i) \in X$
        choose $W^N_i := (a_i + a_{i+1})/2, i = 1, \ldots, |W| - 1$
        choose the i-th child of $N, i = 1, \ldots, |W|$, as the output of
            **BB-Tree**$(x \in X|x_{I^N} \in [W^N_{i-1}, W^N_i], W, rest(\Lambda) \cdot [first(\Lambda)])$
```

Here X - training set, W - set of prototypes, Λ is the list of dimensions, indexes i sorted according to the magnitude of the weighting factors λ_i. $first(\Lambda)$ denotes the first entry, $rest(\Lambda)$ the rest of the list. $\cdot$ denotes concatenation of the lists, $W^N_0 := -\infty$ and $W^N_{|W|} := +\infty$.

Extracted rules are presented in form of decision tree but can be easily converted to propositional *If-Then* form. Its worth to note that sometimes extracted rules BB-tree (Babsi-Baumchen tree) is performing better than GRLVQ algorithm itself on well known datasets. See for comparison table 3.4.

	Iris	Monks1	Wisconsin
Desicion tree (ID3/C4.5)	96 %	98.6 %	96 %
Sigmoidal Neural Network	98 %	100 %	96.7 %
Rules from neural network	95.7-100 %	100 %	96-99 %
GRLVQ	98 %	100 %	96.5 %
BB-Tree	95-98 %	96-100 %	95-97 %

Table 4: GRLVQ and extracted BB-tree comparison

3.5 SVM+Prototypes

Algortihm proposed in [25] is designed specially for linear support vector machines. It's output can have two forms: either propositional *If-Then* rules definining intervals on axises of input space. These intervals are defining hypercubes with edges parallel to original input space axises. Or extracted rules can take *equational* form, these equations are defining ellipsoids with centers placed on prototype vectors.

The SVM+Prototypes algorithm is an iterative process which can be described as follows:

- Step 1 Train a Support Vector Machine The SVM's decision boundary will divide the training data into two subsets S^+ and S^-. These sets will contain data samples for which predicted dclass is respectively positive or negative. Create variable $i = 1$ for each class, it will denote numbers of clusters used for classification of specific class.
- Step 2 Use clustering algorithm for each subset to find i clusters for ech subset and calculate centroids or prototype vectors for each cluster. For all clusters find farthest support vector belonging to same data subset as cluster. To build hypercube use found support vector as one of its vertices and centroid as center. To build ellipsoid use known support vector and centroid. These two points will form one of the axises of the ellipsoid. For both hypercube and ellipsoid other axises and asociating vertices can be found by means of simple geometry, for example by finding point farthest to the prototype.
- Step 3 Make a partition test to the regions defined by hypercubes or ellipsoids. Partition test is performed to minimize overlapping of regions.
- Step 4 For each region check if partition test is negative , if yes, then the region is transformed into a rule. If test is positive then check if variable i is reached maximum defined threshold, if yes convert region into rule otherwise increase variable i by one and go to STEP 2 to generate more clusters.

Experiments conducted by authors on Iris data set was giving 3% error rate for SVM, 4.6% for equation rules and 4.7% for interval rules.

Rules generated in described manner are initially containing whole set of parameters which limits comprehensibility. In case of large amount of input data, as well as in case of large dimensions count algorithm will generate more regions, which will reduce understnadability of extracted rules.

4 Conclusion

All covered algorithms are specific to classifier models they are designed to work with - this somehow limits their application to other classificators, but from the other side as it was already shown in [3] they are performing better than pedagogical methods. Thus decompositional methods or eclectic ones are seemed more perspective in sence of further research area.

According to authors of mentioned algorithms quality of generated rules is more or less acceptable, usually it is hard to get compact rules of high accuracy. This is due to fact that most common *If-Then* rules are producing input space cuts parallel to axises, and in some cases this is not the best knowledge presentation form compared to equation rules form, which can have higher quality and compactness. This point shows that one of the main problems is mapping between input data space and rules space, in this area *equational* rules are more promising, while less understandable. As well generated rules form can be restricted by application area. For some cases knowledge can be extracted in forms other than rules. Thus other possible ways of expressing knowledge can be searched for.

As we saw high classification quality of "black-box" model itself is a prerequirement for extracion of high quality rule set. Thus improvement of classifiers is another topic for further research. Although it should be noted that usually changes in classification method leads to changes in rule extraction algorithm.

Although rule extraction research field is showoing stable growth it should be noted that most of the developed algorithms are oriented on extraction of rules from classic feed-forward neural networks. Thus SVM rules extraction lacks diversity of extraction methods, therefore we can see rise of new methods that will fill this gap.

We covered several quite diverse in their nature decompoisitional algorithms for simbolic rules extraction. Showed main rules types and problems that are arising when such methods are constrcuted. And finally we have pointed some possible areas for further research, which can help in future engeneering of rule extraction algorithms. We believe such algorithms will gain more attention as good data mining tool, which can help in gaining and generating new knowledge.

References

[1] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens and J. Van-thienen, "Benchmarking state of the art classification algorithms for credit scoring", Journal of the Operational Research Society, Vol. 6, Num. 54, 2003, pp. 627635.

[2] S. Thrun, "The MONKs problems: A performance comparison of different learning algorithms", Technical Report CS-91-197, Pittsburgh, PA, 1991.

[3] J.R. Quinlan, "Comparing connectionist and symbolic learning methods", In: S.J. Hanson, G.A. Drastall, and R.L. Rivest (eds.) Comuptational Learning Theory and Natural Learning Systems, Vol. 1, 1994, pp.445-456.

[4] F. Chen. "Learning accurate and understandable rules from SVM classifiers", Masters thesis, Simon Fraser University, 2004.

[5] R. Andrews, J. Diederich, and A.B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks", Knowledge-Based Systems, Elsevier, Vol. 8, Num. 6, 1995, pp. 373-389.

[6] H. Jacobsson, "Rule Extraction from Recurrent Neural Networks: A Taxonomy and Review", Neural Computation, Vol. 17, No. 6, 2005, pp. 1223-1263.

[7] J. Huysmans, B. Baesens and J. Vanthienen, "Using rule extraction to improve the comprehensibility of predictive models", Katholieke Universiteit Leuven, Department of Decision Sciences and Information Management Naamsestraat 69, 3000 Leuven, Belgium.

[8] T. Ash, "Dynamic node creation in backpropagation networks", Connection Science, Vol. 1, No. 1, 1989, pp. 365-375.

[9] Y. Hirose, K. Yamashita, and S Hijiya, "Back-propagation algorithm which varies the number of hidden units", Neural Networks, Vol. 4, 1991, pp. 61-66.

[10] H. Lu, R. Setiono, H. Liu, "NeuroRule: A Connectionist Approach to Data Mining", Proceedings of the 21st VLDB Conference Zurich, Switzerland, 1995.

[11] H. Liu, R. Setiono, "Chi2: Feature selection and discretization of numeric attributes". Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence, 1995, pp. 388-391.

[12] R. Setiono, H. Liu, "NeuroLinear: From neural networks to oblique decision rules", Neurocomputing, 1997.

[13] H. Liu, S.T. Tan, "X2R: A fast rule generator". Proceedings of IEEE International Conference on Systems, Man and Cybernetics, IEEE Press 1995.

[14] R. Setiono, J. Y. L. Thong, "An Approach to Generate Rules from Neural Networks for Regression Problems ", European Journal of Operational Research. 2001.

[15] Y. Hayashi, R. Setiono, K. Yoshida, "A Comparison between two neural network rule extraction techniques for the diagnosis of hepotibiliary disorders".

[16] Kenneth J.McGarry, J. Tait, S. Wermter and J. MacIntyre, "Rule-Extraction from Radial Basis Function Networks", International Conference on Artificial Neural Networks, Edinburgh, 7-10th Sept, 1999.

[17] R. Setiono, H. Liu, "Symbolic Representation of Neural Networks", IEEEC, Volume 29 , Issue 3, (March 1996), pp. 71-77.

[18] J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units", Neural Computation, 1, 281-294 (1989)

[19] Martin D. Buhmann, "Radial Basis Functions: Theory and Implementations". Cambridge University. ISBN 0-521-63338-9.

[20] Christian W. Omlin, and C. Lee Giles, "Extraction of Rules from Descreet-time Reccurent Neural Networks", Neural Networks, Vol.9, No. 1, 1996, pp. 41-52.

[21] Barbara Hammer, Andreas Rechtien, Marc Strickert, and Thomas Villimann, "Rule Extraction from Self-Organizing Networks".

[22] Barbara Hammer, Thomas Villimann, "Estimating relevant input dimensions for self-organizing algorithms". In: N. Allison, H. Yin, L. Allison, J. Slack (eds.), Springer, 2001, Advances in Self-Organizing Maps, pp. 173-180.

[23] A.S. Sato, K. Yamada, "Generalized Learning Vector Quantization". In: G. Tesauro, D. Touretzsky, and T.Leen (eds.), Advances in NIPS, volume 7, pp.423-429, MIT Press, 1995.

[24] T. Konohen, "Self-Organizing Maps", Springer 1997.

[25] H. Nunez, C. Angulo, A. Catala, "Rule extraction from support vector machines", ESANN'2002 proceeedings, ISBN 2-930307-02-1, pp. 107-112.

[26] R. Quinlan, "C4.5: Programs for machine learning", 1993, San Mateo, CA: Morgan Kaufman.

[27] M. W. Craven, "Extracting comprehensible models from trained neural networks", Ph.D. Thesis, University of Wisconsin, Madison, 1996, available at http://www.cs.wisc.edu/ shavlik/ abstracts/craven.thesis.abstract.html.