Extraction of logical rules from backpropagation networks

Włodzisław Duch, Rafał Adamczak and Krzysztof Grąbczewski

***Abstract*—Three neural-based methods for extraction of logical rules from data are presented. These methods facilitate conversion of graded response neural networks into networks performing logical functions. MLP2LN method tries to convert a standard MLP into a network performing logical operations (LN). C-MLP2LN is a constructive algorithm creating such MLP networks. Logical interpretation is assured by adding constraints to the cost function, forcing the weights to ± 1 or 0. Skeletal networks emerge ensuring that a minimal number of logical rules are found. In both methods rules covering many training examples are generated before more specific rules covering exceptions. The third method, FSM2LN, is based on the probability density estimation. Several examples of performance of these methods are presented.**

I. Introduction

Classification using crisp logical rules is preferable to humans over other methods because it exposes the inherent logical structure of the problem. Although the class of problems with logical structure simple enough to be manageable by humans may be rather limited nevertheless it covers some important applications, such as the decision support systems in financial institutions. One way to obtain logical description of the data is to analyze neural networks trained on these data. Many methods for extraction of logical rules from neural networks exist (for a review and extensive references see [**?**]).

In this paper several new methods of logical rule extraction and feature selection are presented. Although we concentrate on crisp logical rules these methods can easily obtain also fuzzy rules. In contrast with the existing neural rule extraction algorithms based on analysis of small connection weights, analysis of sensitivity to changes in input or analysis of the total network function [**?**] our goal is to create a simplified network with nodes performing logical functions. This goal is achieved either by constraining the multi-layered perceptron (MLP) network that has already been trained, or by constructing the network during training, adding new neurons and immediately simplifying their connections. Third possibility is based on density estimation. Cuboidal density regions are easily interpreted as crisp logic rules, therefore one should select appropriate transfer functions and either convert smooth complicated densities into cuboids or create cuboidal densities directly. These three algorithms are briefly described in the next section. Pedagogical example and a few illustrative applications are presented in the third and the fourth section. The paper is finished with a short discussion.

Authors are with the Department of Computer Methods, Nicholas Copernicus University, Grudziądzka 5, 87-100 Toruń, Poland. E-mail: wduch,raad,kgrabcze@is.umk.pl

II. Three rule extraction algorithms

Preliminary: linguistic variables. Logical rules require symbolic inputs (linguistic variables). If the input data components x_i are real numbers finding optimal linguistic variables is a part of the classification problem. Density estimation networks provide linguistic variables from analysis of response of network nodes. The inputs from features giving strong response along the whole range of data are irrelevant and should be removed. The range of values taken by continuous inputs may also be partitioned into distinct (for crisp logic) sets by analysis of class-membership histograms. The initial linguistic variable boundary points are optimized after the rules are found and the whole rule extraction process is repeated with new linguistic variables. MLPs may find linguistic variables using a combination of two neurons, called here an L-unit, performing for each continuous input a "window-type" function:

$$\begin{aligned} s(x;b,b') &= \sigma(x-b)(1-\sigma(x-b')) \\ s(x;b,b') &= \sigma(x-b)-\sigma(x-b') \end{aligned} \qquad (1)$$

where the gain of the sigmoidal functions $\sigma(x)$ reaches a very high value during learning, changing $s(x;b,b')$ into a step-like logical function. This function has two biases which may be treated as boundaries defining linguistic variable $(l = \text{true}) \equiv (x \in [b,b'])$.

MLP2LN method. Logical rules giving classification results corresponding to the trained, multi-layered network, are required. MLP network is modified by retraining it while the slope of sigmoidal functions is gradually increased and the weights simplified. Integer weight values are enforced: 0 for irrelevant inputs, $+1$ for features that must be present and -1 for features that must be absent. This is achieved by modifying the error function:

$$E(W) = E_0(W) + \frac{\lambda_1}{2}\sum_{i,j} W_{ij}^2 + \frac{\lambda_2}{2}\sum_{i,j} W_{ij}^2 (W_{ij}-1)^2 (W_{ij}+1)^2 \qquad (2)$$

$E_0(W)$ is the standard quadratic error measure, the second term with λ_1 leads to a large number of zero weights, i.e. elimination of irrelevant features, and the third term vanishes for weights equal 0 or ± 1. Similarly as in the case of weight pruning techniques in the backpropagation algorithm these terms lead to the additional change of weights:

$$\Delta W_{ij} = \lambda_1 W_{ij} + \lambda_2 W_{ij}(W_{ij}^2 - 1)(3W_{ij}^2 - 1) \qquad (3)$$

where λ_1 and λ_2 scale the relative importance of auxiliary conditions. This form of error function has two advantages: independent parameters control enforcing of 0 and ± 1 weights, and an interpretation of this function from the Bayesian point of view [**?**] is straightforward. It defines our prior knowledge about the probability distribution $P(W|M)$ of the weights in our model M. A network trained on classification tasks should give crisp logical decision "yes", "no" or "irrelevant", therefore *a priori* conditional probability [**?**] is:

$$P(W|M) = Z(\alpha)^{-1} e^{-\alpha E_a(W|M)} \propto \prod_{ij} e^{-\alpha_1 W_{ij}^2} \prod_{kl} e^{-\alpha_2 (W_{kl}-1)^2} \prod_{mn} e^{-\alpha_2 (W_{mn}+1)^2} \qquad (4)$$

To find logical rules giving classifications equivalent to those of trained MLP network new error function Eq. (**??**) with relatively large values of regularization parameters is used for further training. Typical MLP network has many irrelevant connections and large non-integer weights, therefore the training error will initially be large. After such retraining a skeletal network emerges, with a few connections left. The network is capable of rough classification performing simple logical operations. This part of the network is inserted into the original network and kept frozen. Modified MLP is now retrained with smaller regularization parameters. After several cycles of retraining the original network is changed into a simplified MLP performing logical operations that approximate original classifications.

C-MLP2LN method. In this approach a single hidden layer skeleton MLP is constructed. One hidden neuron per output class is created and the modified error function minimized during training on all data until convergence is reached. The remaining non-zero weights and the thresholds obtained are then analyzed and the first group of logical rules is found, covering the most common input-output relations. The input data correctly handled by the first group of neurons does not contribute to the error, therefore the weights of these neurons are kept frozen during further training. A second group of neurons is added and trained on the remaining data. This process is repeated until all data are correctly classified and a set of rules $R_1 \vee R_2 ... \vee R_n$ for each class is found, or until the number of cases correctly classified by a given new neuron drops below certain minimum. Neurons handling only few specific training cases model noise, rather than regularities in the data, and thus should be deleted. The output neuron for each class is connected to the hidden neurons created for that class, performing a simple summation of the incoming signals. Logical rules are sometimes simpler if, in a small number of cases, wrong classifications by a newly added neuron are allowed. These cases are treated as exceptions to the rules.

The set of extracted logical rules has then *a hierarchical order*. Rules handling exceptional cases, placed at the top of hierarchy, should be applied first.

Extraction of rules from skeleton network with small number of integer connections is quite simple. For each neuron non-zero inputs define levels of a search tree, and values of linguistic variables define possible branches of this tree. To each node of the search tree a contribution to the activation of neuron is assigned. Since at each level maximum activation that lower levels may contribute is known, nodes that do not lead to activations larger than threshold are quickly deleted.

Rules obtained by both MLP2LN algorithms are ordered, starting with rules that are used most often and ending with rules that handle only a few cases. Quality of a set of rules is checked using test data – an optimal balance between the number of rules and the generalization error is usually obtained when only the rules that classify larger number of cases are retained. The final solution may be presented as a set of rules or as a network of nodes performing logical functions, with hidden neurons realizing the rules and the hidden-output neuron weights all set to $+1$.

FSM2LN method. It is well known that RBF networks with Gaussian functions are equivalent to the fuzzy logic systems with Gaussian membership functions [**?**]. To obtain crisp logic one can either use rectangular basis functions [**?**] or use transfer functions that may be smoothly changed into functions with cuboidal contour surfaces, for example products of L-units defined in Eq. (**??**). Such functions are separable, but not radial, therefore we use the Feature Space Mapping (FSM) density estimation constructive network [**?**] instead of RBF. The slopes of sigmoidal functions are gradually increased during the training process, allowing for smooth transition from fuzzy to crisp rules. Feature selection is performed by adding penalty term for small dispersions to the error function:

$$E(V) = E_0(V) + \lambda \sum_i^N 1/(1+\sigma_i^2) \qquad (5)$$

where V represents all adaptive parameters, such as positions and dispersions $\sigma_i = |b_i - b_i'|$ of localized units, and the sum runs over all active inputs for the node that is the most active upon presentation of a given training vector. The penalty term encourages dispersions to grow – if σ_i includes the whole range of input data the connection to x_i is deleted. After the training each node has class label and represents a cuboid in relevant dimensions of the input subspace, easily interpreted as logical conditions for a given class. An alternative approach is to test how much the nodes may expand without changing the classification error.

III. Pedagogical example

Because of the lack of space only one pedagogical example of C-MLP2LN application to the rule extraction from the classical iris dataset is presented here. The data has 150 vectors evenly distributed in three classes, called iris setosa, iris versicolor and iris virginica. Each vector has four features: sepal length x_1 and width x_2, and petal length x_3 and width x_4 (all in cm). The input values (length) for each of these features were initially obtained by analysis of histograms of the individual features for each class, cutting the histograms into the regions where values of features are most frequently found in a given class. Discretization was made by smoothing the histogram (assuming small Gaussian width for each sample). The following linguistic variables were obtained:

TABLE I
Linguistic variables obtained by analysis of histograms.

	s	m	l
x_1	[4.3,5.5]	(5.5,6.1]	(6.1,7.9]
x_2	[2.0,2,75]	(2.75,3.2]	(3.2,4.4]
x_3	[1.0,2.0]	(2.0,4.93]	(4.93,6.9]
x_4	[0.1,0.6]	(0.6,1.7]	(1.7,2.5]

Thus x_1 is called small if it is in $[4.3, 5.5]$ range etc. Instead of four inputs x_i a network with 12 linguistic inputs equal to ± 1 is constructed. For example, the medium value of a single feature is coded by $(-1, +1, -1)$. With this discretization of the input features two vectors of the iris versicolor class (coded as (m, m, l, l) and (m, l, m, l)) become identical with a number of iris virginica vectors and cannot be classified correctly. These vectors were removed from the training sequence.

A single neuron per class was sufficient to train the network, therefore the final network structure is 12 input nodes and 3 output nodes (hidden nodes are only needed when more than one neuron is necessary to cover all rules for a given class).

The scaling parameter was increased from $\lambda = 0.001$ at the beginning of the training to $\lambda = 0.01 - 0.1$ near the end. The network needed about 1000 epochs on average and the final weights were within 0.05 from the desired ± 1 or 0 values. The following weights and thresholds are obtained (only the signs of the weights are written):

Iris setosa: $(0, 0, 0; 0, 0, 0; +, 0, 0; +, 0, 0), \theta = 1$

Iris versicolor: $(0, 0, 0; 0, 0, 0; 0, +, -; 0, +, -), \theta = 3$

Iris virginica: $(0, 0, 0; 0, 0, 0; -, -, +; -, -, +), \theta = 1$

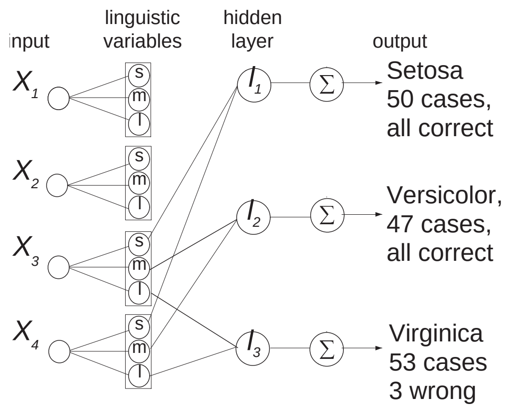
input
linguistic variables
hidden layer
output
X_1
X_2
X_3
X_4
s
m
l
l_1
l_2
l_3
Σ
Setosa 50 cases, all correct
Versicolor, 47 cases, all correct
Virginica 53 cases 3 wrong

Fig. 1. Final structure of the network for the iris dataset.

To find logical rules contributions from various inputs to the activation of the output neuron are considered. For the iris setosa vectors the weights for x_3 and x_4 are $(+, 0, 0)$, therefore to obtain activation larger than $\theta = 1$ both $x_3 = x_4 = (+, -, -) = s$ are necessary. Therefore iris setosa class is obtained for $x_3 = s \wedge x_4 = s$. For iris versicolor $x_3 = m \wedge x_4 = m$ is needed to exceed the threshold $\theta = 3$ and for iris virginica $x_3 = l \vee x_4 = l$ is sufficient.

$$\begin{array}{lll} \text{IF} & (x_3 = s \wedge x_4 = s) \text{ THEN} & \text{iris setosa} \\ \text{IF} & (x_3 = m \wedge x_4 = m) \text{ THEN} & \text{iris versicolor} \\ \text{IF} & (x_3 = l) \vee (x_4 = l) \text{ THEN} & \text{iris virginica} \end{array} \quad (6)$$

These rules allow for correct classification of the 147 vectors, achieving 98% of accuracy or 3 errors. Using smaller constraint parameters λ additional logical rules are obtained, but since clasiffication of one vector requires 10 additional logical conditions these rules are clearly overfitting the data. Increasing constraint hyperparameters further selects only one attribute, petal length x_3, and leaves two rules: iris setosa if $x_3 < 2.5$, iris virginica if $x_3 > 4.9$, else iris versicolor, giving 95.3% accuracy (7 errors).

The accuracy of classification using logical rules critically depends on selection of features. L-units allowing neural network to find linguistic variables were also used. Three linguistic variables per each input were assumed. In the trained network only two of the twelve L-units were relevant, one distinguishing the iris setosa, the other iris virginica class. If none of the two L-units is activated iris versicolor class is assumed. Two rules with two attributes were obtained: iris setosa for $x_3 \leq 2.56$, iris virginica for $x_4 > 1.63$, else iris versicolor. These rules allow for correct classification of all but six vectors, achieving 96% of accuracy. Finally FSM2LN method has also been applied to the iris data. After training with rectangular basis function or with functions Rq. (**??**) four rules gave 98% accuracy. The remaining 3 vectors need 8 additional rules for correct classification – again a clear case of overfitting.

IV. Illustrative results

The rule extraction methods presented in this paper have been applied to a number of datasets obtained from the UCI repository [**?**], producing small number of crisp logic rules and giving frequently better classification accuracy than the original neural classifiers.

The mushroom dataset: 8124 training cases, each case has 22 discrete attributes, with 51.8% of the cases representing edible and the rest nonedible mushrooms. For poisonous mushrooms odor is the single most important feature for distinguishing poisonous and edible mushrooms – a single rule odor=not(almond.or.anise.or.none) obtained with C-MLP2LN method gives 120 errors, or 98.5% accuracy on the whole dataset. Second rule with one additional attribute leaves 48 errors, third rule with two attributes leaves only 8 errors, and fourth rule with two attributes allows for perfect classification. Only 6 of the 22 attributes are important for classification. Same rules are obtained by training on a small subset (10%) of all the cases as well as on the whole dataset.

The three monk problems: this is an artificial data used frequently to test machine learning algorithms. In each of the three monk problems one should determine whether an object described by six features is a monk or not [**?**]. "Being a monk" is defined by the following formulae in the three problems:
Monk 1: head shape = body shape $\vee$ jacket color = *red*
Monk 2: exactly 2 of 6 features have their first values
Monk 3: $\neg$ (body shape = *octagon* $\vee$ jacket color = *blue*) $\vee$ (holding = *sward* $\wedge$ jacket color = *green*)

The data for the last problem is corrupted by adding 5% noise. Rules allowing for perfect classification of all three monk problems have been found, although it was necessary to create exceptions to the rules and use them in a hierarchical way. The first problem requires 4 explicit rules (head and body shapes have 3 values). C-MLP2LN procedure has generated 4 rules and one exception while FSM2LN has generated rules in their original form using either sigmoidal products or rectangular functions. Monk 2 problem requires 15 explicit rules; C-MLP2LN generated 16 rules and 8 exceptions, while FSM2LN was again able to find 15 rules. For the third monk problem using C-MLP2LN approach perfect classification accuracy was obtained with 3 rules and 4 exceptions, despite the noise in the data – for small datasets it is possible to find rules classifying all examples. Using FSM2LN 8 rules were generated. None of the methods tested in [**?**] found rules giving 100% accuracy for all three monk problems.

The cancer dataset contains 286 cases, with 9 attributes and two classes. This is rather difficult noisy data. With one hidden neuron two rules were made by C-MLP2LN giving 22.03% error on all cancer data. Best MLP results give about 2% larger errors [**?**].

The appendictis dataset contains only 106 cases, with 8 attributes (results of medical tests), and 2 classes: 88 cases with acute appendicitis and 18 cases with other problems. Two simple rules (MNEA $>$ 6650 or MBAP $>$ 12) were found by C-MLP2LN method giving 8.5% error; decreasing constraints leads to two new complex rules that already overfit the data. Similar results were found by FSM2LN. The best MLP classification results give 9.8% error [**?**].

The hypothyroid dataset has 3772 cases for training, 3428 cases for testing, 22 discrete and continuous attributes, and 3 classes: primary hypothyroid, compensated hypothyroid and normal (no hypothyroid). About 10% of values are missing. For the first class two rules found by the C-MLP2LN are sufficient to get 0.48% accuracy on the training set and 1.02% accuracy on the test set. For comparison, MLP and Cascade Correlation errors on the test set are slightly larger than 1.5%.

V. Discussion and summary

New methods for extraction of logical rules with the help of neural classifiers have been presented. Results obtained for several artificial and real-world datasets are very encouraging. Crisp and fuzzy rules may be obtained, rules are ordered from the most common to more specialized, hierarchy of rules and exceptions is established, overfitting is avoided by dropping more specialized rules and the accuracy is frequently better than given by other (neural and statistical) approaches. These methods require careful analysis and sometimes simplifications of the rules extracted from weight analysis. The MLP2LN method is similar to the "successive regularization method" recently published by Ishikawa [**?**], although he does not use integer weights while we do not use his "hidden unit clarification" process. His method gives comparable results for the mushroom problem but we believe that our C-MLP2LN constructive approach is simpler and more reliable. The rectangular basis function method [**?**] and the Rulex method [**?**] are comparable to the FSM2LN, although quite different in actual implementation.

Hierarchy of rules and exceptions gives natural flexibility to the rules derived in our approach, uncommon in other approaches. One concept that is worth more attention is the stability of rules (and other classification systems). Rules are brittle if a small change in the input data leads to a different set of rules or to large errors. The choice of linguistic variables is most important for stability of rules. This is true in the density networks as well as in MLPs – the training data places the classification borders very close to the class boundaries, therefore the test data may not fit in such tight compartments. In FSM2LN the penalty Eq. (**??**) used to eliminate input features is also useful to stabilize the rules. In MLP2LN linguistic variables are optimized using extracted rules and the rule extraction process is iteratively repeated with new, optimized variables.

References

[1] R. Andrews, J. Diederich, A.B. Tickle, A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks, *Knowledge-Based Systems* vol. 8, pp. 373-389, 1995.

[2] D.J. MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computations* Vol. 4, pp. 448-472, 1992.

[3] J-S. R. Jang, C.T. Sun, Functional Equivalence Between Radial Basis Function Neural Networks and Fuzzy Inference Systems, *IEEE Trans. on Neural Networks*, vol. 4, pp. 156-158, 1993.

[4] M. Berthold, K. Huber, Building Precise Classifiers with Automatic rule Extraction, in: Proc. of the IEEE ICNN, Perth, Vol. 3, pp. 1263-1268, 1995.

[5] W. Duch, G.H.F. Diercksen, Feature Space Mapping as a universal adaptive system, *Computer Physics Communications,* vol. 87, pp. 341–371, 1995.

[6] C.J. Mertz, P.M. Murphy, UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-databases.

[7] S. Thrun et al., The MONK's Problems. A Performance Comparison of Different Learning Algorithms. Carnegi Mellon University Technical Report CMU-CS-91-197, 1991

[8] M. Ishikawa, Rule extraction by successive regularization, in: *Proc. of the 1996 IEEE ICNN*, Washington, June 1996, pp. 1139–1143.