

Crisp Rule Extraction from Perceptron Network Classifiers

Andrzej Lozowski, Tomasz J. Cholewo, and Jacek M. Zurada
Department of Electrical Engineering, University of Louisville
Louisville, Kentucky 40292
e-mail: jmzura02@starbase.spd.louisville.edu

Abstract

A method of extracting intuitive knowledge from neural network classifiers is presented in the paper. An algorithm which obtains crisp rules in the form of logical implications which approximately describe the neural network mapping is introduced. The number of extracted rules can be selected using an uncertainty margin parameter as well as by changing the precision level of the soft quantization of inputs. A fuzzy decision system based on the IRIS database has been developed using this approach to produce linguistic rules for flower classification.

1. Introduction

The input-output mapping capability of multilayer perceptron networks plays a fundamental role in neurocomputing, leading to versatile data-driven interpolations derived from examples. This modeling capability includes a complete range of continuous or binary input-output relationships. System models, pattern classifiers, and expert systems have successfully been built based on these premises.

One of the drawbacks of such systems is the lack of adequate rule extraction or explanation facilities. This shortcoming becomes especially serious when a neurocomputing system is supposed to replace a human expert. Human experts can typically formulate rules describing underlying causal relationships involved in the decision-making process. Neural network-based decision aids, however, do not inherently possess such a feature.

To satisfy the pressing need for rules, many attempts have been made to develop a methodology for their extraction. Mappings produced by neural networks, however, even for the case of binary outputs, are very complex. Therefore, neither weights, activation values, nor hidden layer responses can typically be meaningfully interpreted [1], since internal mappings emerge as superimposed intertwined relationships. Also, decision regions themselves are intertwined, producing additional difficulties when extracting rules. An especially difficult case of rule extraction faced by a modeler is when Boolean-type logic rules must be obtained. A number of reports indicate that knowledge-based neural networks are necessary to extract such knowledge [2]. The drawback of this approach is that it requires initial knowledge insertion in the form of logic rules and they operate only in binary environments.

To circumvent these obstacles, fuzzy techniques are being employed for extracting linguistic interpolations from network classifier models [1, 3, 4, 5, 6, 7]. Here, crisp linguistic terminology and input partitioning can be used to provide finer resolution of input variables. However, a fuzzy methodology for handling various degrees of membership of objects in classes now becomes necessary. Our approach as presented below provides a closed-form algorithm for rule extraction.

2. Fuzzy rule extraction algorithm

Consider an expert who has some knowledge concerning a given problem. He is able to answer questions of the form: what is the judgment \mathbf{q} for a given instance $\boldsymbol{\pi} = \langle \pi_1, \pi_2, \dots, \pi_n \rangle$. Each instance entry π_j is a variable representing the value of a feature taken from a set of feature classes which together appropriately describe a problem. Consider the well known IRIS classification problem: given an instance of four measurements of a plant's leaves, determine its species from three possible classes.

For example, if the instance is found to be $\langle \textit{short}, \textit{thin}, \textit{medium}, \textit{medium} \rangle$, the conclusion is that the species is C, whereas another instance $\langle \textit{medium}, -, \textit{long}, \textit{wide} \rangle$ (where "-" denotes no information) would

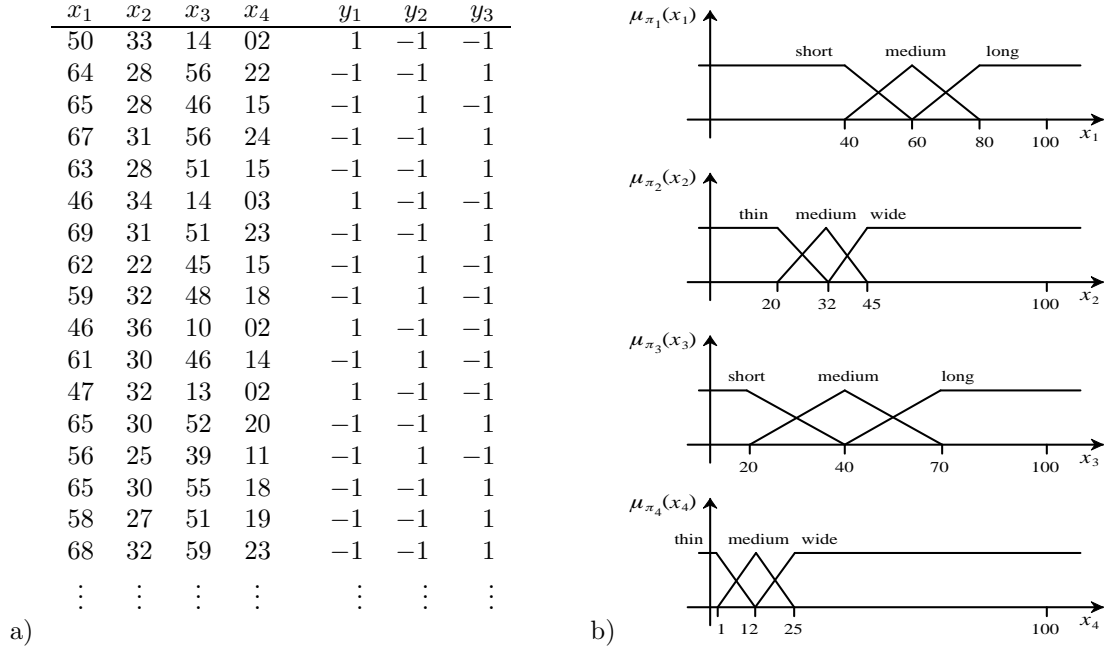


Fig. 1: IRIS problem: (a) NN training set; (b) fuzzification of the input variables. Input $\mathbf{x} = \langle x_1, x_2, x_3, x_4 \rangle$ is represented by membership functions $\mu_{\pi_1}(x_1)$, $\mu_{\pi_2}(x_2)$, $\mu_{\pi_3}(x_3)$, and $\mu_{\pi_4}(x_4)$ ($n = 4$).

result in a different conclusion: not species A. These examples can be represented by means of rules of the form $\boldsymbol{\pi} \Rightarrow \boldsymbol{\varrho}$, such as:

$$r_1 : \langle \text{short}, \text{thin}, \text{medium}, \text{medium} \rangle \Rightarrow \langle \text{no}, \text{no}, \text{yes} \rangle \quad (1)$$

$$r_2 : \langle \text{medium}, -, \text{long}, \text{wide} \rangle \Rightarrow \langle \text{no}, -, - \rangle \quad (2)$$

$$r_3 : \langle \text{long}, \text{medium}, \text{long}, \text{medium} \rangle \Rightarrow \langle \text{yes}, \text{no}, \text{no} \rangle \quad (3)$$

Thus the judgment $\boldsymbol{\varrho} = \langle \varrho_1, \varrho_2, \dots, \varrho_n \rangle$ can be precise as in rule (1), or can only exclude some possibilities, as in rule (2). An expert's knowledge can be described by a set of rules $\{r_k\}$. In most cases the knowledge base is incomplete, i.e., the rule set does not provide conclusions for all possible instances $\boldsymbol{\pi}$. Some information can still be derived, however, particularly when fuzzy reasoning is used instead of inflexible Boolean rules.

The rule extraction problem is inverse to the one introduced above. Here the goal is to create the set of rules $\{r_k\}$ based on some existing decision system. Assume that the system is a neural network classifier. An input instance $\mathbf{x} = \langle x_1, x_2, \dots, x_n \rangle$ is a vector $\mathbf{x} \in X$ whose entries express parameters specified in a given problem. In the IRIS problem vector \mathbf{x} would be 4-dimensional ($X = \mathbb{R}^4$) with entries being the four measurements of plant's leaves, as shown in Fig. 1a. The neural network is trained with a training set $Q = \{(\mathbf{x}, \mathbf{y}_d)\}$ which is a set of input instances \mathbf{x} and desired classifications \mathbf{y}_d . Each input instance \mathbf{x} should be classified to one of m classes. The desired output instances $\mathbf{y}_d = \langle y_{d1}, y_{d2}, \dots, y_{dm} \rangle$ are composed of entries $y_{di} \in \{-1, 1\}$ of which only one, corresponding to the chosen class, equals 1, while the remaining $(m - 1)$ entries are -1 .

The purpose of using a neural network in categorization tasks is to obtain proper classification even if the network is trained with noisy, uncertain, or incomplete training data Q . The neural network calculates a vector function $\mathbf{y} = \mathbf{f}(\mathbf{x})$, $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Given a set of training pairs $(\mathbf{x}, \mathbf{y}_d)$, the network learning process is a minimization of a measure $\sigma = \sum_k \|\mathbf{y}_k - \mathbf{y}_{d_k}\|$ until a criterion $\sigma < \delta$ is reached. For the frequently used norm $\|\cdot\|_2$, the criterion value δ is the final mean-square-error of the learning process.

Since the network should serve as a classifier, the output classifier $\mathbf{h}(\mathbf{y})$ must be employed. Assume that each entry y_i of the network output state can be interpreted as a positive or negative answer

$\varrho_i \in \{no, yes\}$. This can be done by a function $\boldsymbol{\varrho} = \mathbf{h}(\mathbf{y})$ such that

$$\varrho_i = h_i(y_i) = \begin{cases} yes & \text{if } y_i \geq 0, \\ no & \text{if } y_i < 0. \end{cases} \quad (4)$$

Thus the network output is considered to belong to one of 2^m classes created with the output classifier. For the training set S , however, only m classes (those with only one positive entry) are likely to occur as the network responses.

Define decision regions D_{ϱ_i} independently for each entry ϱ_i in the output classes as sets of all possible inputs \mathbf{x} such that the i th entry in the output's classified state is ϱ_i :

$$D_{\varrho_i} = \{\mathbf{x} : h_i(y_i) = \varrho_i \wedge \mathbf{y} = \mathbf{f}(\mathbf{x})\}. \quad (5)$$

Note that decision regions can overlap since D_{ϱ_i} is defined according to a single output entry ϱ_i only. Hence, the decision region $D_{\boldsymbol{\varrho}}$ for some particular output class $\boldsymbol{\varrho}$ is a set product $D_{\boldsymbol{\varrho}} = \bigcap_i D_{\varrho_i}$. Note also that D_{ϱ_i} and D_{ϱ_j} for two different entries ϱ_i and ϱ_j may overlap as well.

The decision system built with a neural network classifier learns from training data and contains some knowledge concerning a given problem. The rule extraction problem can be defined as obtaining an intuitive knowledge representation from neural network in the form of crisp rules. Intuitive judgment of some quantity usually involves expressions like *small*, *large*, or *moderate*, depending on how precise the judgment needs to be. These terms still refer to discrete classes possibly with barely defined and overlapping boundaries. Therefore, a fuzzy representation of analog quantities seems to be a good model for expert reasoning.

Crisp rule extraction from a neural network classifier is an appropriate method for finding a relation between input and output classes if the network input was represented in a fuzzy set form. Prior to the derivation of fuzzy rule extraction from a neural network, consider a fuzzy reasoning algorithm. Suppose the set of fuzzy rules $\{r_k\}$ is already known and represents knowledge embedded in a neural network classifier. In this case for some input vector \mathbf{x} the appropriate classification $\boldsymbol{\varrho}$ can be obtained by evaluating $\boldsymbol{\varrho} = \mathbf{h}[\mathbf{f}(\mathbf{x})]$ using either a neural network or fuzzy reasoning for a fuzzified representation of input \mathbf{x} referred to as $\tilde{\mathbf{x}}$. If both $\tilde{\mathbf{x}}$ and $\{r_k\}$ are sufficiently precise, answers given by the fuzzy system and the neural network should be identical. In order to obtain an answer from the fuzzy system the following steps must be followed:

First, the input \mathbf{x} needs to be fuzzified. Each entry x_i of the input can be represented by a linguistic variable \tilde{x}_i which is a vector of memberships $\mu_{\pi_i}(x_i)$ as in [7]. Value of the membership function corresponds to the distance between entry x_i and the center of gravity of a fuzzy class π_i . As shown by example in Fig. 1b, fuzzification of x_i is a soft quantization of a real value x_i resulting in a spread of memberships \tilde{x}_i in a set of fuzzy classes π_i .

Second, a fuzzy representation of output \mathbf{y} has to be evaluated with respect to the rule set $\{r_k\}$. The rules in the form of (1-3) allow for finding memberships $\mu_{\varrho_i}(y_i)$ describing linguistic variables \tilde{y}_i on the basis of $\tilde{\mathbf{x}}$. For each rule $r_k = \boldsymbol{\pi}_k \Rightarrow \boldsymbol{\varrho}_k$ and the given input \mathbf{x} a t-norm $T^{\boldsymbol{\pi}_k}$ is defined as:

$$T^{\boldsymbol{\pi}_k}(\mathbf{x}) = \min \{\mu_{\pi_1}(x_1), \mu_{\pi_2}(x_2), \dots, \mu_{\pi_n}(x_n)\}; \quad \boldsymbol{\pi}_k = \langle \pi_1, \pi_2, \dots, \pi_n \rangle. \quad (6)$$

If some entry π_i is skipped in instance $\boldsymbol{\pi}_k$ (as in (2)) the membership value $\mu_{\pi_i}(x_i)$ in equation (6) is considered to be 1. Thus the number of created t-norms is equal to the number of rules in the set $\{r_k\}$. Independently for each output \tilde{y}_i and class ϱ_i from the output classes set P_i an s-norm is defined as:

$$S_{\varrho_i}(\mathbf{x}) = \max_{\boldsymbol{\pi}_k} \{T^{\boldsymbol{\pi}_k}(\mathbf{x}) : r_k = \boldsymbol{\pi}_k \Rightarrow \boldsymbol{\varrho}_k \wedge \varrho_{k_i} = \varrho_i\}. \quad (7)$$

Thus $S_{\varrho_i}(\mathbf{x})$ is a maximum of the t-norms which are obtained for those of rules r_k in which the i th entry in $\boldsymbol{\varrho}_k$ equals ϱ_i . If the i th entry is skipped in a rule's output instance $\boldsymbol{\varrho}_k$, then the t-norm corresponding to this rule is not taken into consideration. Memberships $\mu_{\varrho_i}(y_i)$ for the output linguistic variables y_i are equal to the s-norms:

$$\mu_{\varrho_i}(y_i) = S_{\varrho_i}(\mathbf{x}). \quad (8)$$

Finally, each entry in the output \mathbf{y} can be classified based on memberships $\mu_{\varrho_i}(y_i)$. Entry y_i is found to belong to class $\hat{\varrho}_i$ if the membership $\mu_{\hat{\varrho}_i}(y_i)$ is the largest among all the output classes $\varrho_i \in P_i$:

$$h_i(y_i) = \begin{cases} \hat{\varrho}_i & \text{if } (\forall \varrho_i \in P_i \wedge \varrho_i \neq \hat{\varrho}_i) \mu_{\hat{\varrho}_i}(y_i) > \mu_{\varrho_i}(y_i), \\ - & \text{otherwise.} \end{cases} \quad (9)$$

Note that the output classifier defined by equation (9) is undecidable if for two different output classes membership function for y_i has the same value. To avoid decidable classification in a case where the membership functions are not equal but very close, an uncertainty margin ε ($0 \leq \varepsilon < 1$) can be employed. The modified classifier \mathbf{h}^ε evaluates the following criterion:

$$h_i^\varepsilon(y_i) = \begin{cases} \hat{\varrho}_i & \text{if } (\forall \varrho_i \in P_i \wedge \varrho_i \neq \hat{\varrho}_i) \mu_{\hat{\varrho}_i}(y_i) - \mu_{\varrho_i}(y_i) > \varepsilon, \\ - & \text{otherwise.} \end{cases} \quad (10)$$

Now a difference of at least ε between the largest and next to largest membership values is required to classify the output y_i . Based on the introduced fuzzy classification algorithm the reverse order of steps leads to fuzzy rule extraction from a decision system with known answers to inputs \mathbf{x} . Consider the neural network $\mathbf{y} = \mathbf{f}(\mathbf{x})$ trained on the training data Q and the classifier function $\mathbf{h}(\mathbf{y})$ defined by equation (4). Each output y_i is classified as $\varrho_i \in P_i$ where $P_i = \{no, yes\}$ for all outputs i . Thus the network returns outputs $\boldsymbol{\varrho} = \langle \varrho_1, \varrho_2, \dots, \varrho_m \rangle$ to every input \mathbf{x} from data Q . Assume that the network inputs x_i are fuzzified in such a way that values of membership functions $\mu_{\pi_i}(x_i)$ are known for arbitrarily assigned input class sets π_i . Concentrate on output y_i and assume that it was classified as ϱ_i . The decision region, in terms of the elements of training set Q , is given by equation (5). Every point $\mathbf{x} \in D_{\varrho_i}$ can be represented as a linguistic variable $\tilde{x} = \langle \tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n \rangle$. Hence instances $\boldsymbol{\pi}_k \in \Pi_1 \times \Pi_2 \times \dots \times \Pi_n$ can be generated for each point \mathbf{x} and evaluated in terms of membership functions $\mu_{\pi_1}(x_1), \mu_{\pi_2}(x_2), \dots, \mu_{\pi_n}(x_n)$. Define a set of t-norms for the decision region D_{ϱ_i} and one of the instances $\boldsymbol{\pi}_k$:

$$T_{\varrho_i}^{\boldsymbol{\pi}_k} = \{T^{\boldsymbol{\pi}_k}(\mathbf{x}) : \mathbf{x} \in D_{\varrho_i}\}. \quad (11)$$

According to equation (6) $T_{\varrho_i}^{\boldsymbol{\pi}_k}$ is a set of minimum input memberships taken among entries x_i for instance $\boldsymbol{\pi}_k$ created for all inputs \mathbf{x} belonging to the decision region corresponding to the output class ϱ_i . The cardinality of set $T_{\varrho_i}^{\boldsymbol{\pi}_k}$ is the same as that of D_{ϱ_i} . For each set $T_{\varrho_i}^{\boldsymbol{\pi}_k}$ its maximum element is found as

$$S_{\varrho_i}^{\boldsymbol{\pi}_k} = \max T_{\varrho_i}^{\boldsymbol{\pi}_k}. \quad (12)$$

Note that $S_{\varrho_i}^{\boldsymbol{\pi}_k}$ is a maximum of t-norms chosen for one instance $\boldsymbol{\pi}_k$ (to create potential rule r_k) and various input points \mathbf{x} , whereas $S_{\varrho_i}(\mathbf{x})$, defined by (7), was a maximum of t-norms chosen for one input \mathbf{x} and various rules $r_k = \boldsymbol{\pi}_k \Rightarrow \boldsymbol{\varrho}_k$.

At this stage instance $\boldsymbol{\pi}_k$ is validated for the output class ϱ_i by the number $S_{\varrho_i}^{\boldsymbol{\pi}_k}$. For all other classes $\varrho_i \in P_i$ and all other outputs i the number $S_{\varrho_i}^{\boldsymbol{\pi}_k}$ can be evaluated in the same way. Rule $r_k = \boldsymbol{\pi}_k \Rightarrow \boldsymbol{\varrho}_k$ is then created, where output instance is $\boldsymbol{\varrho} = \langle \varrho_{k1}, \varrho_{k2}, \dots, \varrho_{km} \rangle$. Its entries ϱ_{k_i} are found from:

$$\varrho_{k_i} = \begin{cases} \hat{\varrho}_i & \text{if } (\forall \varrho_i \in P_i \wedge \varrho_i \neq \hat{\varrho}_i) S_{\hat{\varrho}_i}^{\boldsymbol{\pi}_k} - S_{\varrho_i}^{\boldsymbol{\pi}_k} > \varepsilon, \\ - & \text{otherwise.} \end{cases} \quad (13)$$

Here ε is the uncertainty margin introduced in (10). The larger the ε , the more undecidable entries are included in the output instance of rule r_k . Rule r_k becomes totally undecidable as ε approaches 1 since the membership function value belongs to the range $[0, 1]$. Following the introduced algorithm, a rule for each input instance $\boldsymbol{\pi}_k$ can be created and thus the whole rule set $\{r_k\}$ is generated.

The algorithm of fuzzy rule extraction has been tested on the IRIS problem. A data set Q of 150 examples, each consisting of four leaf characteristics and an expert classification, were used as a training set for a neural network classifier with four inputs and three outputs. A fragment of this data is shown in Fig. 1a. For fuzzy rule extraction the input fuzzifiers have been formed using the standard triangular membership function shapes (see Fig. 1b). Each input has been quantized into three classes with a center

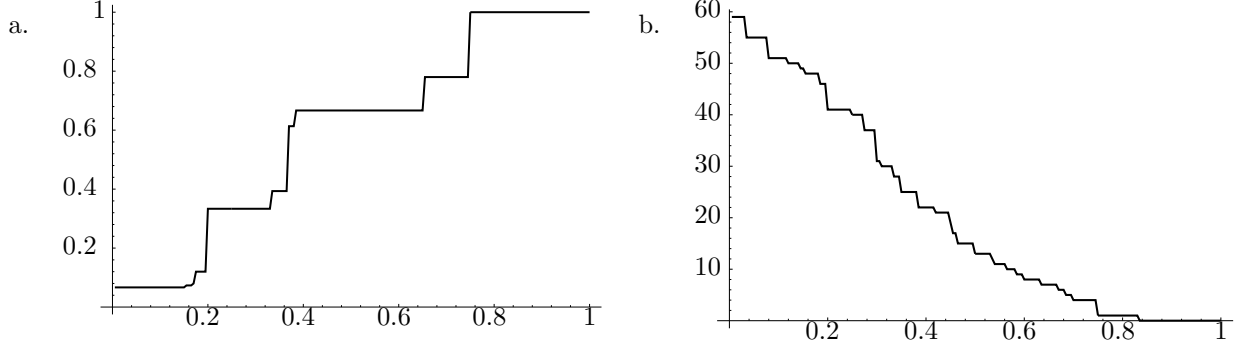


Fig. 2: Iris: (a) classification RMS error; (b) number of generated rules vs. the parameter ε .

of gravity located in the middle and both ends of the range of changes of the input. Various rule sets composed of 81 rules were created using the introduced algorithm with different values of the parameter ε . Totally undecidable rules were subsequently pruned from the sets. Figure 2b presents the number of the remaining rules as a function of ε . Performance of the resulting fuzzy rule sets were compared to the answers given by the neural network by calculating the RMS error. The resulting RMS error versus ε is indicated in Figure 2a. An example of a rule set minimized to a disjunctive normal form [2] is shown in Fig. 3a. The rules can be represented graphically on a four-dimensional hypercube whose corners and sides correspond to input instances while the classification is indicated in three grey-levels (see Fig. 3). On the figure variables x_1 , x_2 and x_3 are the vertical, horizontal and axial dimensions, respectively, and variable x_4 is represented as three different cubes for three different input classes π_4 .

Example B: Rule extraction in IRIS problem

Now consider the inverse problem: given three data points \mathbf{x}_1 , \mathbf{x}_2 , and \mathbf{x}_3 and the neural network classifier responses ϱ_1 , ϱ_2 , and ϱ_3 (see the first three rows of data in Fig. 1) create a set of rules $\{r_k\}$ describing the network performance with $\varepsilon = 0.2$.

First, find decision regions D_{ϱ_i} for all possible output classes ϱ_i : $D_{\varrho_1=no} = \{\mathbf{x}_2, \mathbf{x}_3\}$, $D_{\varrho_1=yes} = \{\mathbf{x}_1\}$, $D_{\varrho_2=no} = \{\mathbf{x}_1, \mathbf{x}_2\}$, $D_{\varrho_2=yes} = \{\mathbf{x}_3\}$, $D_{\varrho_3=no} = \{\mathbf{x}_1, \mathbf{x}_3\}$, and $D_{\varrho_3=yes} = \{\mathbf{x}_2\}$. According to (11) evaluate sets of t-norms $T_{\varrho_i}^{\pi_k}$ independently for all possible instances π_k ($\pi_1 = \langle short, thin, short, thin \rangle$, $\pi_2 = \langle short, thin, short, medium \rangle$, \dots , $\pi_{81} = \langle long, wide, long, wide \rangle$). The sets are: $T_{\varrho_1=no}^{\pi_1} = \{T_{\varrho_1=no}^{\pi_1}(\mathbf{x}_2), T_{\varrho_1=no}^{\pi_1}(\mathbf{x}_3)\} = \{\min\{0, 0.33, 0, 0\}, \min\{0, 0.33, 0, 0\}\} = \{0, 0\}$, $T_{\varrho_1=no}^{\pi_2} = \dots$, $T_{\varrho_3=yes}^{\pi_{81}} = \{T_{\varrho_3=yes}^{\pi_{81}}(\mathbf{x}_2)\} = \{0\}$.

In this manner 81×6 sets are created—most of these contain only zeroes. Find the minimum element in each set $T_{\varrho_i}^{\pi_k}$ and assign this value to $S_{\varrho_i}^{\pi_k}$. Create 81 rules of the form $r_k = \pi_k \Rightarrow \varrho_k$ as follows: $r_1 = \langle short, thin, short, thin \rangle \Rightarrow \langle \hat{\varrho}_1, \hat{\varrho}_2, \hat{\varrho}_3 \rangle$ where, according to (13), $\hat{\varrho}_1 = no$ if $S_{\varrho_1=no}^{\pi_1} - S_{\varrho_1=yes}^{\pi_1} > 0.2$, or $\hat{\varrho}_1 = yes$ if $S_{\varrho_1=yes}^{\pi_1} - S_{\varrho_1=no}^{\pi_1} > 0.2$; otherwise place a “-” in the position of $\hat{\varrho}_1$. Analogically assign $\hat{\varrho}_2$ and $\hat{\varrho}_3$ and then create the remaining rules r_2, r_3, \dots, r_{81} .

3. Conclusions

This paper presents a method of extracting crisp rules from a trained neural network classifier. The rules have a form of those used in fuzzy reasoning. Furthermore, the neural network is a necessary element if the training data is noisy, since the neural network provides filtration of the training data and thus the number of conclusive rules becomes reasonable. Even one noisy data point would cause an enormous increase in the number of created rules if the rules were obtained based on the training data instead of the network outputs. Here we assume that the network is trained with a sufficiently large final MSE δ to allow smooth filtration of data.

Moreover, by choosing the parameter ε , the number of rules (after pruning the totally undecidable rules) can be adjusted; the number of rules is related to the accuracy of a fuzzy classifier using the

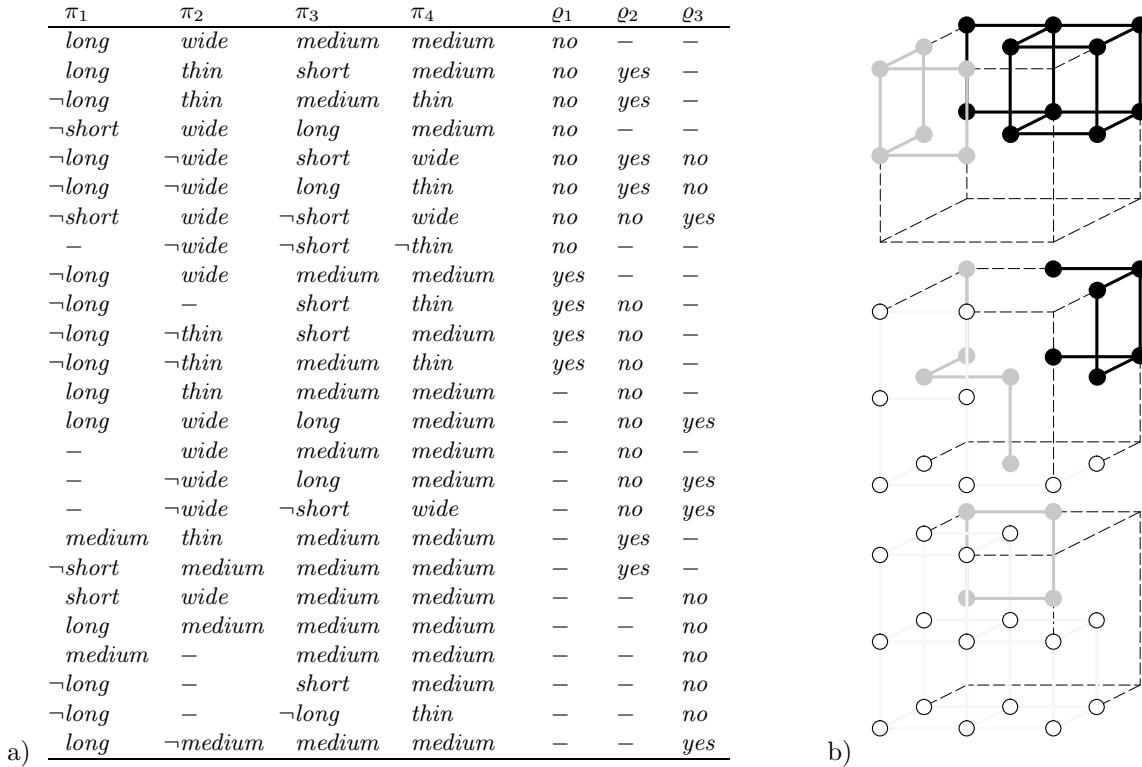


Fig. 3: (a) Rules created for the IRIS problem with $\varepsilon = 0.01$; (b) graphic representation of the rules providing classification.

created rules. The aim is to extract knowledge from the NN, not to replicate the classifier, since it is more important for the rules to be as compact as possible even if classification with such rules is less than optimum. In addition, the numeric complexity of the rule extraction algorithm increases with the number of fuzzy classes for each input and with the number of inputs with a factor larger than 1. However, the algorithm presented is relatively insensitive to the size of the training data, and even large data sets can be handled efficiently.

References

- [1] J. M. Zurada, *Introduction to Artificial Neural Systems*. Boston: PWS, 1992.
- [2] G. G. Towell, J. W. Shavlik, and M. O. Noordewier, “Refinement of approximately correct domain theories by knowledge-based neural networks,” *Proc. of the 8th Nat. Conf. on Artificial Intelligence*, pp. 861–866, 1990.
- [3] C. T. Sun, “Rule–base structure identification in an adaptive–network–based fuzzy inference system,” *IEEE Trans. on Fuzzy Systems*, vol. 2, no. 1, pp. 64–73, 1994.
- [4] I. Jagielska and C. Matthews, “Fuzzy rule extraction from a trained multilayered neural network,” in *Proc. of the IEEE International Conference on Neural Networks*, (Perth, Australia), pp. 744–748, Nov. 27–Dec. 1, 1995.
- [5] S. K. Tang, T. S. Dillon, and R. Khosla, “Fuzzy logic and knowledge representation in a symbolic–subsymbolic architecture,” in *Proc. of the IEEE International Conference on Neural Networks*, (Perth, Australia), pp. 349–353, Nov. 27–Dec. 1, 1995.
- [6] P. A. Egri and P. F. Underwood, “Hilda: Knowledge extraction from neural networks in legal rule based and case based reasoning,” in *Proc. of the IEEE International Conference on Neural Networks*, (Perth, Australia), pp. 1800–1804, Nov. 27–Dec. 1, 1995.
- [7] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, pp. 338–353, 1965.