

Symbolic Rule Representation in Neural Network Models

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

Symbolic knowledge extraction from mapping/extrapolating neural networks is presented in the paper. An algorithm to obtain crisp rules in the form of logical implications which roughly 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 of the soft quantization of the inputs. A fuzzy decision system for a medication dosage problem has been developed and tested to demonstrate this approach.

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 KBNNs require 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 interpretations from network classifier models [3, 4]. 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 $\boldsymbol{\rho}$ 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 problem of predicting an appropriate dose of gentamycin (a medicine used to treat kidney disease) based on a few parameters known by inspection or by measurement, such as: a person's weight, height, body surface area, sex, and age. The history of treatment is also important as are factors like the dosage time interval, recent gentamycin dose levels, gentamycin peak and trough concentration levels, serum creatinine, and creatinine clearance. The appropriate dosage of gentamycin is of great importance in order to achieve the desired peak and trough levels in a patient's body. Simplifying, given an instance of relevant parameters, the problem is to predict the amount of gentamycin needed to produce the desired peak and trough concentration levels.

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 dose prediction problem vector \mathbf{x} would be n -dimensional ($X = \mathbb{R}^n$) with entries being the observed parameters. 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} .

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.

Assume that each entry y_i of the network output state can be interpreted as one of a few classes $\rho_i \in P_i$. This can be done by a classification function $\boldsymbol{\rho} = \mathbf{h}(\mathbf{y})$.

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 :

$$D_{\varrho_i} = \{\mathbf{x} : h_i(y_i) = \varrho_i \wedge \mathbf{y} = \mathbf{f}(\mathbf{x})\}. \quad (1)$$

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_{ϱ} for some particular output class ϱ is a set product $D_{\varrho} = \bigcap_i D_{\varrho_i}$. 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 a 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 refer to discrete classes possibly with barely defined and overlapping boundaries. Therefore, a fuzzy representation of analog quantities appears 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 ϱ can be obtained by evaluating $\varrho = \mathbf{h}[\mathbf{f}(\mathbf{x})]$ 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 $\mu_{\pi_i}(x_i)$ [5]. Value of the membership function corresponds to the distance between entry x_i and the center of gravity of a fuzzy class π_i . Fuzzification of x_i may be thought of as 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 the output \mathbf{y} has to be evaluated with respect to the rule set $\{r_k\}$. These rules enable finding memberships $\mu_{\varrho_i}(y_i)$ describing linguistic variables \tilde{y}_i on the basis of $\tilde{\mathbf{x}}$. For each rule $r_k = \boldsymbol{\pi}_k \Rightarrow \varrho_k$ and the given input \mathbf{x} a t-norm $T^{\boldsymbol{\pi}_k}$ is defined as:

$$T^{\boldsymbol{\pi}_k}(\mathbf{x}) = \min \{\mu_{\pi_1}(x_1), \mu_{\pi_2}(x_2), \dots, \mu_{\pi_n}(x_n)\}; \quad \boldsymbol{\pi}_k = \langle \pi_1, \pi_2, \dots, \pi_n \rangle. \quad (2)$$

If some entry π_i is skipped in instance $\boldsymbol{\pi}_k$ the membership value $\mu_{\pi_i}(x_i)$ in equation (2) 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 class set P_i an s-norm is defined as:

$$S_{\varrho_i}(\mathbf{x}) = \max_{\boldsymbol{\pi}_k} \{T^{\boldsymbol{\pi}_k}(\mathbf{x}) : r_k = \boldsymbol{\pi}_k \Rightarrow \varrho_k \wedge \varrho_{k_i} = \varrho_i\}. \quad (3)$$

Thus $S_{\varrho_i}(\mathbf{x})$ is a maximum of the t-norms which are obtained for those rules r_k in which the i th entry in ϱ_k equals ϱ_i . If the i th entry is skipped in a rule's output instance ϱ_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:

$$\mu_{\varrho_i}(y_i) = S_{\varrho_i}(\mathbf{x}). \quad (4)$$

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 $\varrho_i \in P_i$:

$$h_i(y_i) = \begin{cases} \hat{\varrho}_i & \text{if } (\forall \varrho_i \in P_i \wedge \varrho_i \neq \hat{\varrho}_i) \mu_{\hat{\varrho}_i}(y_i) > \mu_{\varrho_i}(y_i), \\ - & \text{otherwise.} \end{cases} \quad (5)$$

Note that the output classifier defined by equation (5) is undecidable if for two different output classes the membership function for y_i has the same value. To avoid a 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:

$$h_i^\varepsilon(y_i) = \begin{cases} \hat{\varrho}_i & \text{if } (\forall \varrho_i \in P_i \wedge \varrho_i \neq \hat{\varrho}_i) \mu_{\hat{\varrho}_i}(y_i) - \mu_{\varrho_i}(y_i) > \varepsilon, \\ - & \text{otherwise.} \end{cases} \quad (6)$$

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 $\mathbf{h}(\mathbf{y})$. Each output y_i is classified as $\varrho_i \in P_i$. 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 elements of training set Q , is given by equation (1). Every point $\mathbf{x} \in D_{\varrho_i}$ can be represented as linguistic variable $\tilde{x} = \langle \tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n \rangle$. Hence instances $\pi_k \in \Pi_1 \times \Pi_2 \times \dots \times \Pi_n$ 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 π_k :

$$T_{\varrho_i}^{\pi_k} = \{T^{\pi_k}(\mathbf{x}) : \mathbf{x} \in D_{\varrho_i}\}. \quad (7)$$

According to equation (2) $T_{\varrho_i}^{\pi_k}$ is a set of minimum input memberships taken among entries x_i for instance π_k created for all inputs \mathbf{x} belonging to the decision region corresponding to the output class ϱ_i . The cardinality of the set $T_{\varrho_i}^{\pi_k}$ is the same as that of D_{ϱ_i} . Now, for each set $T_{\varrho_i}^{\pi_k}$ its maximum element is found as

$$S_{\varrho_i}^{\pi_k} = \max T_{\varrho_i}^{\pi_k}. \quad (8)$$

Note that $S_{\varrho_i}^{\pi_k}$ is a maximum of t-norms chosen for one instance π_k (to create potential rule r_k) and various input points \mathbf{x} , whereas $S_{\varrho_i}(\mathbf{x})$, defined by (3), was a maximum of t-norms chosen for one input \mathbf{x} and various rules $r_k = \pi_k \Rightarrow \varrho_k$.

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

$$\varrho_{ki} = \begin{cases} \hat{\varrho}_i & \text{if } (\forall \varrho_i \in P_i \wedge \varrho_i \neq \hat{\varrho}_i) S_{\hat{\varrho}_i}^{\pi_k} - S_{\varrho_i}^{\pi_k} > \varepsilon, \\ - & \text{otherwise.} \end{cases} \quad (9)$$

Here ε is the uncertainty margin introduced in (6). 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 π_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 dosage prediction problem. A data set Q of 150 examples, each consisting of three patient characteristics and routine dosage levels, was used as a training set for a neural network classifier with three inputs and one output. For the sake of fuzzy rule extraction the input fuzzifiers have been formed using the standard triangular membership function shapes. Each input has been quantized into three classes with centers of gravity located in the middle and at both ends of the range of changes of the input. Various rule sets composed of 27 rules have then been created using the introduced algorithm with different values of the parameter ε . Totally undecidable rules were subsequently pruned from the rule sets. An example of a rule set minimized to a disjunctive normal form [2] is shown in Fig. 1a. The rules can be represented graphically on a cube whose corners and sides correspond to input instances while the output classification is indicated in three grey-levels (see Fig. 1). On the figure variables x_1 , x_2 , and x_3 are the vertical, horizontal and axial dimensions, respectively. Roughly, this draft shows a monotonic relationship between the dose and a linear combination of the inputs. Even though increasing the number of fuzzy classes at the inputs and the output would improve the approximation of the neural network mapping, the small set of rules provides an indication of the dosage problem solution.

3. Conclusions

This paper presents a method of extracting crisp rules from a trained neural network. 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.

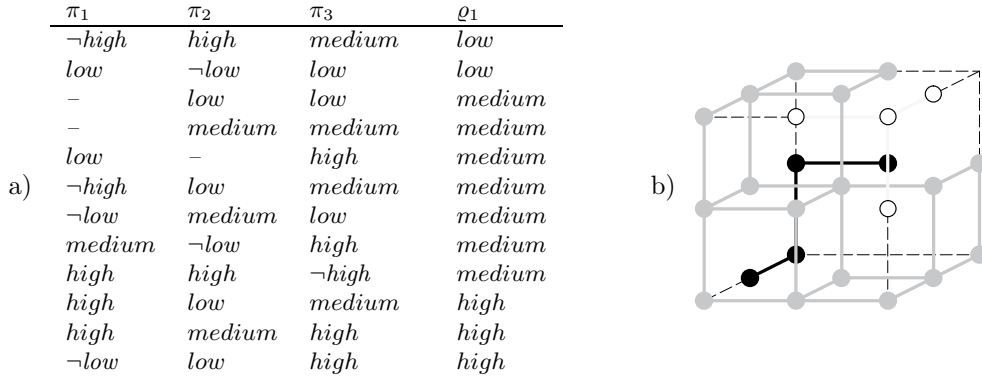


Fig. 1: (a) Rules created for the gentamycin dosage problem with $\varepsilon = 0.01$. (b) The decidable rules graphic representation.

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 created rules. The aim is to extract knowledge from the NN, not to replicate the mapping, since it is more important that the rules 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 presented algorithm is relatively insensitive to the size of the training data and 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] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.