Rule Extraction from Linguistic Rule Networks and from Fuzzy Neural Networks: Propositional versus Fuzzy Rules

Nikola Kasabov[1], Robert Kozma[1] and Włodzisław Duch[2]

[1] Department of Information Science, University of Otago, P.O Box 56,
Dunedin, New Zealand, nkasabov,rkozma@otago.ac.nz; http://divcom.otago.ac.nz
[2] Department of Computer Methods, Nicholas Copernicus University
Grudziądzka 5, 87-100 Toruń, Poland, duch@phys.uni.torun.pl;
http://www.phys.uni.torun.pl/~duch

Abstract.

This paper explores different techniques for extracting propositional rules from linguistic rule neural networks and fuzzy rules from fuzzy neural networks. The applicability and suitability of different types of rules to different problems is analyzed. Hierarchical rule structures are considered where the higher the level is the smaller the number of rules which become more vague and more approximate. The issue of quality of the rules extracted and ways to improve it is discussed. The paper takes for case study several benchmark datasets from the UCI Machine Learning Repository. It compares results produced by propositional and fuzzy rules extracted from the two types of neural networks.

1. Introduction to the rule extraction problem: a rule is worth a thousand of data examples

Extracting rules from data has been explored through different techniques (Andrews *et al.* 1995, Mitchell 1997, Fayyad, U. *et al.* 1996), some of them listed below:

- case-based methods
- decision tree methods
- clustering methods
- incremental inductive learning
- neural networks
- fuzzy neural networks
- others

Fuzzy neural networks, which combine both connectionist and fuzzy logic principles, have proved to be efficient when used for rule extraction and rule adaptation (Hashiyama *et al*, 92; Jag, 93; Hauptmann and Heesche, 95; Kasabov, 96). It is interesting to consider the crisp limit of the rule extraction from fuzzy neural networks and compare the quality of propositional and fuzzy rules. Two neural architectures, FuNN and LR-net, designed for rule extraction, are described below and the crisp propositional rules obtained with LR-net are compared with the fuzzy rules obtained with the FuNN network. We discuss the weak and the strong points of both approaches.

2. Fuzzy Neural Network (FuNN) – architecture, principles, applications.

Fuzzy neural networks are neural networks that realise a set of fuzzy rules and a fuzzy inference machine in a connectionist way. FuNN is a fuzzy neural network introduced first in Kasabov (1996) and then developed as FuNN/2 in Kasabov *et al.* (1997). It is a connectionist feed-forward architecture with five layers of neurons and four layers of connections. The first layer of neurons receives the input information. The second layer calculates the fuzzy membership degrees to which the input values belong to predefined fuzzy membership functions, e.g. small, medium, large. The third layer of neurons represents associations between the input and the output variables, fuzzy rules. The forth layer calculates the degrees to which output membership functions are matched by the input data and the fifth layer does defuzzification and calculates values for the output variables. A FuNN has both the features of a neural network and a fuzzy inference machine. A simple FuNN structure is shown in Fig.1. The number of neurons in each of the layers can potentially change during operation through growing or shrinking. The number of connections is also modifiable through learning with forgetting, zeroing, pruning and other operations. The membership functions, used in FuNN to represent fuzzy values, are usually of triangular type, the centres of the triangles being attached as weights to the corresponding connections. The membership functions can be modified through learning. Several training algorithms have been developed for FuNN (Kasabov *et al.* 1997):

(a) A modified back-propagation (BP) algorithm that does not change the input and the output connections representing the membership functions.

(b) A modified BP algorithm that utilises structural learning with forgetting, i.e. a small forgetting ingredient, e.g. 10^{-3}, is used when the connection weights are updated (cf. Ishikawa 1996).

(c) A modified BP algorithm that updates both the inner connection layers and the membership layers. This is possible when the derivatives are calculated separately for the two parts of the triangular membership functions. These are also the non-monotonic activation functions of the neurons in the condition element layer.

(d) A genetic algorithm for training

(e) A combination of any of the methods above used in different time intervals as part of a single training procedure.

Several algorithms for rule extraction from FuNN have been developed for extracting different types of rules:

(a) Aggregated fuzzy rules, i.e. each rule node of a trained FuNN is represented as a fuzzy IF-THEN rule;

(b) weighted fuzzy rules

(c) simple fuzzy rules

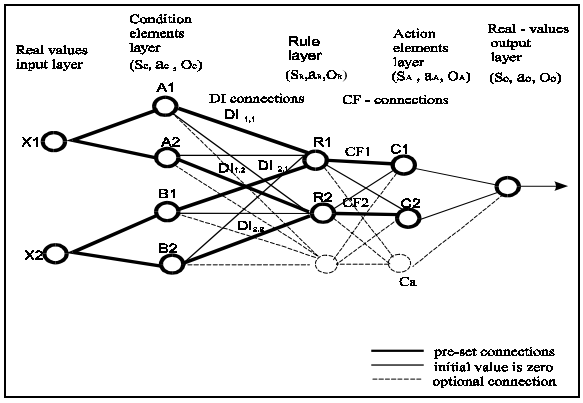


Fig.1. A FuNN structure for two initial fuzzy rules: R1: IF x1 is A1 ($DI_{1,1}$) and x2 is B1 ($DI_{2,1}$) THEN y is C1 (CF1); R2: IF x1 is A2 ($DI_{1,2}$) and x2 is B2 ($DI_{2,2}$) THEN y is C2 (CF2), where DIs are degrees of importance attached to the condition elements and CFs are confidence factors attached to the consequent parts of the rules (adopted from [11]). The triplets (s,a,o) represent summation, activation, and output functions specific for the layer.

FuNNs have several advantages when compared with the traditional connectionist systems or with the fuzzy systems:

(a) They are both statistical and knowledge engineering tools.

(b) They are robust to catastrophic forgetting, i.e. when further trained only on new data, they keep a reasonable memory of the old data.

(c) They interpolate and extrapolate well in regions where data is sparse.

(d) They can be used as replicators (autoassociative memory), where the same input data is used as an output data during training; in this case the rule nodes perform an optimal encoding of the input space.

(e) They accept both real input data and fuzzy input data represented as singletons (centres of gravity of the input membership functions)

(e) They are appropriate tools to build multi-modular Inteligent Information systems (IIS) as explained below.

The FuNN fuzzy neural networks have been used so far for different tasks: speech recognition; time series modelling and prediction; decision making; classification (Kasabov 1996; Kasabov *et al.* 1997; Kozma and Kasabov 1998).

3. The LR networks

The architecture of the LR networks (Duch *et al.* 1996) is roughly similar to the FuNN networks: there are also five layers, including the input and the output layer. The first three layers are used to define linguistic variables (Figures 2 and 3) or membership functions that in general have trapezoidal shapes:

$$L(x;b,b') = \sigma(x+b) - \sigma(x+b')$$

The difference between the two sigmoidal functions (realized by the neurons of the second layer) defines a smooth trapezoidal membership functions. In the high slope limit this function changes into a rectangular function and crisp linguistic variables are obtained. If the difference between the two biases, *b* and *b'* is small the membership function has a bell shape and if the sigmoidal neurons are replaced by semi-linear neurons it has triangular shape. Since the weights between the first (input) and second layer are all +1 and the second-third layer are +1 or -1 pairs of the second layer nodes are combined with one third layer node into a single „linguistic variable" unit with two adaptive parameters (plus variable slopes), *L(x;b,b').* The output from the L-units is combined by the rule nodes (R-units), hence the name LR-network. Linear combination of the rule unit outputs gives in turn the final output. To make transition to the crisp logic straightforward a penalty term using two regularization hyperparameters, λ_1 and λ_2 is added to the error function, enforcing the zero or ±1 value of weights.

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

where $E_0(W)$ is the standard error function. The term scaled by λ_1 is used at the beginning of the training to enforce large number of zero weights and the term scaled by λ_2 is slowly increased towards the end of training. The alternative form with Laplace regularization is:

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

The modification of the backpropagation procedure is quite trivial. Since our goal is to obtain small networks that are easy to interpret simple regularization using the λ_1 term (Ishikawa 1996) is not sufficient. As Neal (1996) has pointed out large number of small weights may lead to good generalization (small weights mean that only the linear part of the sigmoid, around the threshold value, is used), but in compact neural networks some weights should stay large and this is ensured by the λ_2 term in our penalty function. The crisp logic limit is very interesting because propositional rules are obtained.

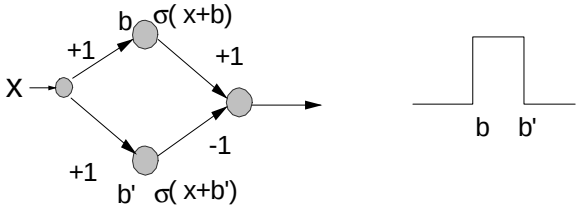


Fig.2. L-unit in the limit of large slope values realizes a window type crisp membership function.

In practice we have found a constructive version of the training algorithm much easier to use than the version starting with the larger network. One rule neuron per output class is created and the network trained on all the data until convergence is reached. Second rule neuron is added for those classes for which errors remain and the training repeated. This training procedure is called MLP2LN (Duch *et al.* 1996a) because it attempts to change the MLP network into a network performing logical functions (LN).

Since the first R-unit is trained on all data until the error will decrease as much as possible it codes the rules covering the largest number of cases, with suubsequently added R-units coding more and more specific rules. The optimal number of R-units is easily determined by examining the rules that are derived from each new added unit: if the number of derived rules exceeds the number of vectors classified by these rules the network becomes too specialized and such a unit should not be added. Some constructive algorithms, such as the cascade correlation, have been criticized because they may lead to overtraining and thus poor generalization, but this problem is easily avoided here. The optimal number of L-units may also be found by starting from a single unit per each input, covering the whole data range, and adding more units until no significant reduction of the error occurs. It should be stressed that finding good linguistic variables leads to a hard minimization problem that local descent algorithms are rarely able to solve. Therefore a good initialization, based for example on analysis of histograms, iterative optimization of linguistic variables (using the final values as a start for creation of a new LR network), or on initialization of MLP networks by cluster information (Duch *et al.* 1997c) is highly desirable.

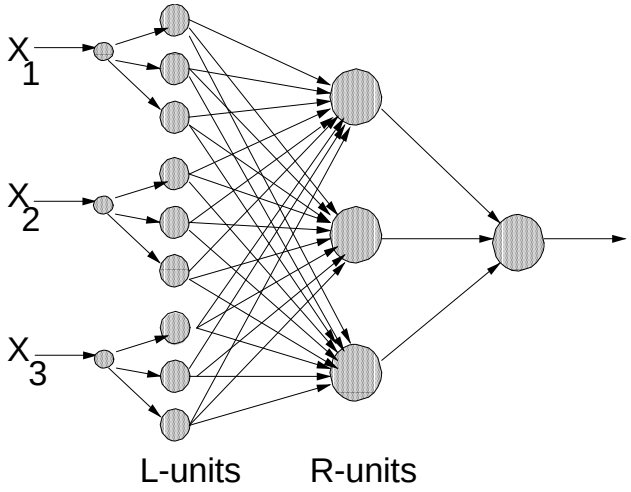


Fig.3 Architecture of L-R network for a two-class problem (single output unit is sufficient).

4. Extracting Propositional Rules from LR- networks on Case Study Data Sets

The **Iris data** (Mertz and Murphy 1997), serving as a well-known benchmark, has 150 vectors evenly distributed in three classes, called iris-setosa, iris-versicolor and iris-virginica. Each vector has four features: sepal length *x1* and width *x2*, and petal length *x3* and width *x4* (all in cm). Analysis of histograms of the individual features for each class may provid initial linguistic variables. For example, Iris-virginica class is more frequent for $x3 > 4.9$ and Iris-versicolor are more frequent below this value. Since the number of vectors per class is rather small discretization should be based on smoothed histograms. Initial discretization is further improved by changing the cutoffs of linguistic variables to minimize the classification error of the extracted set of rules - the whole rule extraction procedure may then be repeated with new linguistic variables.

Several solutions may be generated by LR-networks, depending on the strength of regularization terms enforcing network skeletonization. The simplest result involves one attribute, petal length *x3*, and leaves two rules giving 95.3% accuracy (7 errors) on the whole dataset:

IF $x3 < 2.5$ THEN iris-setosa;
IF $x3 > 4.9$ THEN iris-virginica;
else iris-versicolor.

This is the simplest description of the Iris dataset. Decreasing the regularization parameters we can

generate more accurate rules, involving two attributes, for example:

IF $x3=s$ THEN iris-setosa
IF $x3=l \vee x4=l$ THEN iris-virginica;
ELSE iris-versicolor.

where $x3=s$ (small) means $x3 < 2.5$, $x3=l$ means that $x3 > 4.93$ and $x4=l$ means $x4 > 1.7$. These simple rules make only 3 errors on the whole data set. One can argue that more accurate solutions do not offer a better generalization because decreasing the regularization hyperparameters further leads to three new rules per one correctly classified vector, which clearly indicates the overfitting of the data. It would be hard to improve upon the accuracy and explanation power of the rules given above (Duch *et al.* 1997a,b).

Thus in the case of such simple classification problem crisp propositional rules may be more useful than fuzzy rules and one does not gain much on leaving fuzzy membership functions instead of going to the crisp logic limit. One can draw similar conclusions from investigation of a number of medical datasets. We have obtained propositional rules for the following datasets:

Hepatitis dataset (Mertz and Murphy 1997) contains 155 samples belonging to two different classes (32 „die" cases, 123 „live" cases). There are 19 attributes, 13 binary and 6 attributes with 6 to 8 discrete values. The propositional rules for the first of these classes are:

age > 52 ∧ bilirubin > 3.5 OR
age ∈ [30,51] ∧ histology=yes ∧ ascites=no

These rules give 11.6% error on the whole dataset. Further efforts to add new neurons to classify the remaining data lead to a very large number of rules which is a clear indication of overfitting the data. For example the next neuron classified properly 6 new vectors, but yielded more that 50 rules. For comparison LDA (linear discriminant analysis) gives 13.6%, k-NN gives about 15%, MLP about 18% and CART decision tree about 17% of errors for this case.

The cancer dataset (Mertz and Murphy 1997) contains 286 cases, with 9 attributes and 2 classes. Usually 30% of the cases are selected randomly as a test data, and the best classification results give only about 23% of error on the training and similar error on the test set. One rule:

deg_malig=3 ∧ ~ inv_nodes ∈ [0-2]

gives 22.9% accuracy on the test data with 10-fold cross-validation, and 22.6% on the training part. It may be obtained by optimization of two slightly more complex rules (coming from one neuron)

(deg_malig=3 ∨ breast=left) ∧ ~ age ∈ [50-59] ∧ ~ inv_nodes ∈ [0-2]

deg_malig =3 ∧ breast=left ∧ node_caps=yes

Further training of the network leads to one new rule per each data vector, decreasing accuracy on the test set. These rules give 22.0% errors on cancer data and give the best classification results, although similar accuracy has been achieved by the CART decision tree and PVM method (Weiss and Kapouleas 1990). Results obtained using machine learning methods (AQ15 and weighted networks), quoted in the UCI repository that contains this dataset, are within 26.5-35% error range, for example k-NN gives 34.7% and MLP 28.5% errors on the test data.

Heart disease dataset (Mertz and Murphy 1997) contains 303 cases collected by the Cleveland Clinic Foundation. While the database has 76 raw attributes, only 13 of them are actually used, 6 of which are continuous. The network with only one hidden neuron is equivalent to the three crisp logical rules:

~cp=2 ∧ slope=3 OR
ca=0.0 ∧ (~ cp=2 ∨ slope=3) OR
~ thal=2 ∧ (ca=0.0 ∨ ~ cp=2 ∨ slope=3)

These rules give 17.5% of error on the whole dataset. This is the only medical data tested so far for which linear discriminant analysis gives slightly better result (15.5% of error) than propositional rules. k-NN, MLP and CART give about 19% of errors on this dataset (Mertz and Murphy 1997).

The hypothyroid data (Mertz and Murphy 1997) is somewhat larger, with 3772 cases for training, 3428 cases for testing, 22 attributes, and 3 classes: primary hypothyroid, compensated hypothyroid and normal (no hypothyroid). For the first two classes after some optimization the following rules are obtained (Duch *et al.* 1997):

1: TSH > 6.1 ∧ FTI < 64.4
2: TSH > 6 ∧ TT4 < 149 ∧ FTI ≥ 64.4
∧ on_thyroxine = no ∧ surgery=no

These rules are very accurate, giving only 0.67% on the test and 0.21% on the training set.

LR-network gave more complex rules:

1: (TSH∈[6.1,29) ∧ FTI < 63 ∧ T3 < 20)
∨ (TSH ≥ 29 ∧ FTI < 63)

2: TSH ≥ 6.1 ∧ FTI ∈ [64.4,180] ∧ on_thyroxine = no ∧ (surgery=no ∨ TSH∉[6.1,29])

which make 0.3% error on the training set and 0.9% on the test set. Comparing to other classification systems these rules are not only easy to understand but also highly accurate. The best MLP with genetic optimization (Schiffman *et al*. 1993) gives 1.5% error on the training set and k-NN makes 4.7% error.

5. Extracting Fuzzy Rules from FuNN – A Comparative Analysis

The FuNN structure is designed to facilitate rule extraction and rule insertion in addition to adaptive learning with fixed or adaptable membership functions. If a FuNN structure is trained different sets of aggregated rules can be extracted depending on the threshold used which removes rule condition and conclusion elements with importance less than the threshold. In this way a hierarchical knowledge representation of the information learned in the FuNN structure can be created. On the top of this structure are simple general rules.

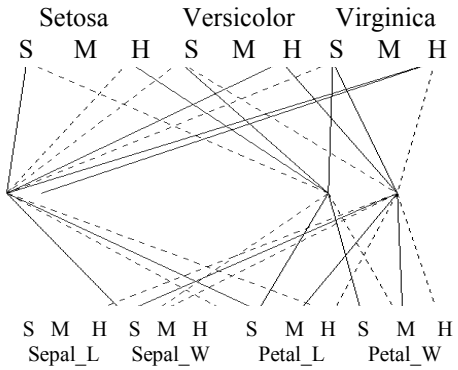
A set of rules extracted from FuNN trained on the Iris data when the threshold = 2 is shown below:

IF *x3*= Sm (4) AND *x4*=Sm (4) THEN setosa (8);
IF *x2*=Med (7) AND *x4*=Med(37)
THEN versicolour (16);
IF *x1*=Sm (12) AND *x3*=L (38) AND *x4*=Med (6)
THEN virginica (17),

where: Sm, Med and L denote small, medium and large fuzzy values represented as triangular membership functions having centers of 0, 0.5 and 1 respectively in a normalised domain for each of the input variables. Rules in this form, assigning importance factors to different conditions and conclusions, carry more information than simple propositional rules. The rules above can be presented in a more general, simplified way which will make a new level of rule representation in a knowledge representation hierarchical scheme as shown below:

IF *x3*= Sm AND *x4*=Sm THEN setosa ;
IF *x4*=Med THEN versicolour;
IF *x3*=L THEN virginica

The FuNN architecture allows for training with adaptive membership functions when the centers 'move' if necessary in order to approximate better the data set. The FuNN structure above had initially three rule nodes and three rules were extracted with different level of accuracy. When using the FuNN model, a FuNN is initialized with more rule nodes, in the given example we start with 40 rule nodes, and learning with forgetting is applied. After sufficient training with a consecutive pruning, similar rules as shown above can be extracted, while the classification performance is 100%. The connection weights which are not contributing to the solution 'fade away' and a skeleton network is obtained which is a structural representation of the fuzzy system with 3 aggregated fuzzy rules. This is illustrated in fig.4.



(R1) IF *(PL Small 2.42) THEN (setosa 1.70)* ELSE
(R2) IF *(PL Large 3.17)* AND *(PW Large 4.24) THEN (virginica 1.67)* ELSE
(R3) IF *(PL Medium 2.98)* AND *(PL non-Large 3.17) AND (PW Large 4.24)* THEN *(versicolor 1.70)*

Fig.4. The skeleton a FuNN structure trained on Iris data with forgetting FuNN and pruning; and a set of rules extracted from it; S, M, L denote small, medium, large, and PL and PW are petal length and petal width.

6. Conclusions and Directions for Further Research

Several conclusions may be drawn from our study:

1. Many network parameters leading to the same classification may be found, but the parameters may have no significance for the classification decisions. Rule extraction requires feature selection which is easily done using regularization terms in the penalty function. In addition small networks should use large weights, therefore it is not sufficient to make all weights small. The learning with forgetting implemented in FuNN allows for such feature extraction.

2. Finding the simplest set of rules requires global minimization procedures for determination of the network parameters. Multiple random starts of the gradient procedure do not always find the best solution. After the rules have been extracted they may be simplified and optimized by checking all conditions (how many vectors are misclassified if a condition is dropped) and performing additional optimization of linguistic variables.

4. The rule insertion mode in FuNN allows to start the training procedure not only with small or big connection weights randomly generated, but to insert existing heuristic rules in the FuNN structure. The same may be done using cluster-based initialization methods for MLPs (Duch 1997).

5. Crisp logical rules may be highly accurate in classification problems because it is easy to recognize when overfitting occurs. It may not be so simple with the fuzzy rules. Fuzzy rules carry more information but their interpretation is more difficult. Smooth transition between the fuzzy and crisp rules may be realized either by using constrained MLPs or density estimation networks (Duch and Diercksen 1995, Duch *et al*. in print) and is always worth doing. If some accuracy is lost in this transition propositional rules may still be used for approximate justification of the results.

6. It is not clear how to approach the time series or signal analysis using propositional rules. One approach could rely on basis functions, such as wavelets, and form the rules not in terms of input features but expansion coefficients. In this case the use of fuzzy rules is more appropriate.

Acknowledgements

W.D. is grateful for support by the Polish Committee for Scientific Research, grant 8T11F 00308. N.K. and R.K. acknowledge grant UOO606 from the FRST of New Zealand.

References

Amari, S. and Kasabov, N. eds (1997) Brain-like Computing and Intelligent Information Systems, Springer Verlag

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

Duch W and Diercksen GHF (1995) Feature Space Mapping as a universal adaptive system, Computer Phys. Communic. 87: 341-371

Duch W, Adamczak R, Grąbczewski K (1996) Constrained backpropagation for feature selection and extraction of logical rules, in: First Polish Conference on Theory and Applications of Artificial Intelligence CAI'96, Łódź, pp. 163-170; Extraction of logical rules from training data using backpropagation networks; *ibid*, pp. 171-178

Duch W, Adamczak R, Grąbczewski K (1996a) Extraction of logical rules from training data using backpropagation networks, The First Online Workshop on Soft Computing, pp. 25-30; available at: http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/

Duch W, Adamczak R, Grąbczewski K (1997) Extraction of logical rules from backpropagation networks, Neural Processing Letters (in print)

Duch W, Adamczak R, Grąbczewski K (1997a) Extraction of crisp logical rules using constrained backpropagation networks. Int. Conf. on Artificial Neural Networks (ICANN'97), Houston, pp. 2384-2389

Duch W, Adamczak R., Grąbczewski K, Ishikawa M, Ueda H (1997b) Extraction of crisp logical rules using constrained backpropagation networks - comparison of two new approaches, Proc. of the European Symposium on Artificial Neural Networks (ESANN'97), Bruge, pp. 109-114

Duch W, Adamczak R, Jankowski N (1997c) Initialization and optimization of multilayered perceptrons, Third Conference on Neural Networks and Their Applications, Kule, Poland, pp. 371-377

Fayyad, U.M, Piatetsky-Shapiro G, smyth P, Uthurusamy R (eds), Advances in Knowledge Discovery and Data Mining, The MIT Press, 1996

Hashiyama, T., Furuhashi, T., Uchikawa, Y.(1992) A Decision Making Model Using a Fuzzy Neural Network, in: Proceedings of the 2nd International Conference on Fuzzy Logic & Neural Networks, Iizuka, Japan, pp. 1057-1060.

Hauptmann, W., Heesche, K. (1995) A Neural Net Topology for Bidirectional Fuzzy-Neuro Transformation, in: Proceedings of the FUZZ-IEEE/IFES, Yokohama, Japan, pp. 1511-1518.

Ishikawa, M. (1996) Structural Learning with Forgetting, Neural Networks 9: 501-521.

Jang, R. (1993) ANFIS: adaptive network-based fuzzy inference system, IEEE Trans. on Syst.,Man, Cybernetics, 23: 665-685

Kasabov, N (1996) Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, The MIT Press, CA, MA.

Kasabov, N., Kim J S, Watts, M., Gray, A (1997) FuNN/2- A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition, Information Sciences - Applications

Kasabov, N. and Kozma, R. eds. (1998) Neuro-Fuzzy Tools and Techniques for Information Processing, Physica Verlag, Heidelberg, in print

Kasabov, N. (1996) Connectionist methods for fuzzy rules extraction, reasoning and adaptation. In: Proc. of the Int. Conf. on Fuzzy Systems, Neural Networks and Soft Computing, Iizuka, Japan, World Scientific, pp. 74-77

Kozma, R. and Kasabov, N. (1998) Rules of chaotic behaviour extracted from the fuzzy neural network FuNN, Proc. of the WCCI'98 FUZZ-IEEE International Conference on Fuzzy Systems, Anchorage, May 1998

Mertz C.J, Murphy P.M (1997), UCI repository of machine learning databases, available at: www.ics.uci.edu/pub/machine-learning-databases The breast cancer data was provided by M. Zwitter and M. Soklic from the University Medical Centre, Institute of Oncology, Ljubljana.

Mitchell T.M. (1997) Machine Learning, MacGraw-Hill

Neal R. (1996) Bayesian Learning in Neural Networks, Lecture Notes in Statistics vol. 118, Springer Verlag

Schiffman W, Joost M, Werner R (1993) Proc. of the European Symposium on Artificial Neural Networks, Brussels, pp. 97-104

Weiss S.M, Kapouleas I. (1990) in: J.W. Shavlik and T.G. Dietterich, Readings in Machine Learning, Morgan Kauffman Publ, CA