Investigating the Adaptation and Forgetting in Fuzzy Neural Networks Through a Method of Training and Zeroing

Nikola Kasabov,
Department of Information Science
University of Otago, P.O.Box 56, Dunedin, New Zealand
Fax: + 64 3 479 8311, Phone: + 64 3 479 8319
email: nkasabov@otago.ac.nz

Abstract

Rules extraction, rules insertion and a method of alternative training and zeroing, based on zeroing of small connections in a fuzzy neural network, have been investigated in respect of adaptation and forgetting phenomena. The experiments show that updated (zeroed) fuzzy neural networks are more robust to forgetting, faster and better at adaptation and provide a good generalisation. The method may be used for building adaptive neuro-fuzzy systems. It may now be possible to control the level of adaptation and forgetting in practical systems for speech recognition, process control, decision making, time-series forecasting, etc. A specialised engineering environment which facilitates such experiments has been briefly described.

Introduction

There are several differences between rule-based and connectionist systems, among them the way knowledge is represented in them [7]. In the rule-based systems knowledge is structured. Every rule is a separate module and adding or deleting rules does not require re-compiling the whole rule base or changing the inference mechanism. In contrast to that, the connectionist systems have their knowledge distributed in the connectionist architecture and adding or deleting knowledge or data is not directly possible. In order to use connectionist methods for building intelligent adaptive systems one needs methods to add or delete knowledge. This objective has been tackled in the paper. A method called "Method of Training and Zeroing" (MTZ) has been experimented. This method differs from the method of structural learning with forgetting introduced by Ishikawa [4] in the way the connectionist architecture is initialised, rules extracted and inserted and the connection weights are processed.

1. Fuzzy Neural Networks - Rules Extraction and Adaptation

1.1. Different types of fuzzy neural networks

A fuzzy neural network (FNN) is a connectionist model for fuzzy rules implementation and inference. There is a big variety of architectures and functionalities of FNN. Adaptive network-based fuzzy inference systems have been presented and discussed in [1-8]. They differ mainly in the following parameters:

- *type of fuzzy rules* implemented; this reflects in the connectionist structure used;
- *type of inference* method implemented; this reflects in the selection of different neural network parameters and neuronal functions, such as summation (s), activation (a), output (o) function; it also influences the way the connection weights are initialized before training, and interpreted after training;
- *mode of operation*; we shall consider here three major modes of operation as suggested in [6]:
  - *Fixed mode* - "fixed membership functions-fixed set of rules", i.e. fixed set of rules is inserted in a network; the network performs inference, but does not change its weights. It cannot learn and adapt. It does not forget either.
  - *Learning mode, i.e.* a neural network is structurally defined to capture knowledge in a certain format, e.g. - some type of fuzzy rules. The network architecture is randomly initialized and trained with a set of data. Rules are then extracted from the structured network [5,7]. The rules can be interpreted either in the same network structure or by using other inference methods [7,8].
  - *Adaptation mode*- A neural network is either randomly initialised or structurally set according to a set of fuzzy rules, `hints', heuristics. The network is then trained with data and updated fuzzy rules are extracted from its structure

following some rule extraction algorithm [1,6,7]. The rules can either be interpreted in a fuzzy inference engine, or they can be inserted back to the fuzzy neural network structure in the same way initial set of rules have been inserted. The network is further trained with data and new updated rules are extracted afterwards, etc.

The FuNN model [5,7] facilitates *learning from data, fuzzy rules extraction, approximate reasoning, adaptation.* FuNN uses a MLP network and a backpropagation training algorithm. It is an adaptable FNN where the membership functions of the fuzzy predicates as well as the fuzzy rules inserted before training (adaptation) may adapt and change according to the training data. The general architecture of FuNN consists of five layers. Fig.1 depicts a FuNN for two exemplar fuzzy rules [5,6,7].

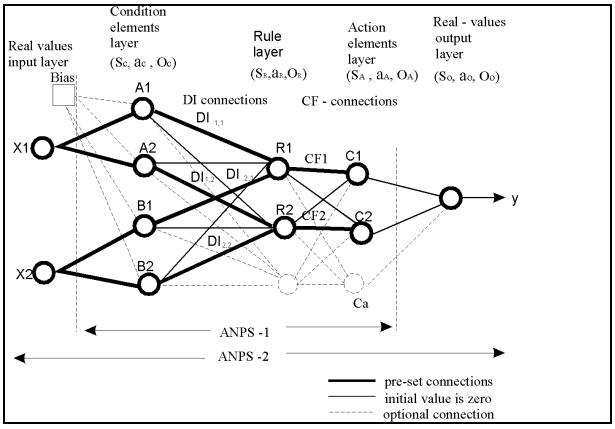

Figure 1. A FuNN structure for the two fuzzy rules: R1 IF x_1 is A_1 ($DI_{1,1}$) and x_2 is B_1 ($DI_{2,1}$) THEN y is C_1 (CF_1); R_2: IF x_1 is A_2 ($DI_{1,2}$) and x2 is B_2 ($DI_{2,2}$) THEN y is C_2 (CF_2), where DIs are degrees of importance attached to the condition elements and CFs are certainty factors attached to the consequent parts of the rules.

In the experiments below FuNN which consist of only the condition element layer, the rule layer and the action element layer will be considered. The membership functions are defined by the user. For the experiments in the next section the membership functions will be selected of a standard triangular type, uniformly distributed in the universe of discourse. Fuzzyfication and defuzzification are supposed to be done outside the structure.

1.2. Rules extraction from fuzzy neural networks

One of the advantages of fuzzy neural networks is that structured information (knowledge) can be inserted and extracted from them. A FNN, after having been trained, can be interpreted in linguistic terms . The way a FNN is structured also restricts the information (knowledge) representation and interpretation.

An algorithm called REFuNN for rules extraction from a trained FuNN is presented in [5,7]. The method is based on the following assumptions: simple operations are used and a low computational cost achieved; hidden nodes in a MLP can learn features, rules, groups in the training data; fuzzy quantization of the input and the output variables are done outside the algorithm; automatically extracted rules may need additional manipulation depending on the reasoning method applied afterwards. The algorithm uses thresholds above which connection weights are kept and represented in a linguistic form as fuzzy rules.

2. Adaptation and Forgetting in Fuzzy Neural Networks. The Method of Training and Zeroing

2.1. Adaptation vs Forgetting in Fuzzy Neural Networks

There are different ways to make a fuzzy neural network adapt to new data:

- to expand the structure if necessary by adding new nodes as new data comes;
- to make the system forget about old data thus making space for the new ones (forgetting)

The latter option is investigated here, but a combination of the two options above should be investigated in the future. Forgetting happens when an already trained neural network is further trained with new data without having been rehearsed with the old ones. This prevents connectionist models to be used as adaptive systems without retraining them

with both new and old data. It may be crucial to know how much a trained neural network would forget after having been trained with new data in both situations when forgetting is desirable, or it is not desirable. For example a neural network which predicts stock market price should be able to gradually forget the old values, say two years back, and keep only general `rules' for a future trading. On the other hand some connectionist models are required to adapt quickly to new data and not to forget the old ones. An example is a connectionist system trained on speech corpus data which is required to adapt in a real time to new speakers [7]. Controlling the level of forgetting and adaptation in a system might be as important as controlling the level of learning and generalisation. There are different ways to introduce forgetting in a neural network structure:

- use of decay parameter when calculating next activation value (t+1) based on the previous one (t):
$O(t+1) = a(decay.\ O(t) + s(Net))$
- decay of connection weights during training [4];
- pruning (cutting off) weak connections [4];
- *zeroing* weak connections before training the neural network with new data; this option is taken and investigated here.

The following steps form a *general methodology* for evaluating the adaptation and forgetting in connectionist models:

(i) A data set is divided into several sections (sets), say: A,B,C. The sections are either randomly chosen or they are consecutive time series of a same variable.

(ii) A connectionist model is initialised, trained on set A and tested for generalisation on set B. Corresponding errors E_{train} and E_{gener} are calculated. RMS error may be applicable depending on the type of the data set.

(iii) The model is further trained on the set B (error $E_{ad.train}$) and tested on the set A (error E_{forget}).

(iv) Coefficients of adaptation CA_d and forgetting CF_r are calculated as follows:

$CA_d = E_{gener}/E_{ad.train}$); $CF_r = E_{forget}/E_{train}$

(v) steps (iii) and (iv) are repeated for a next pair of data sections, e.g. B and C, and new coefficients of adaptation and forgetting are calculated . Average values between the ones calculated at step (iv) and (v) are calculated. This process continues over a next pair of data sections.

2.2. The Method of Training and Zeroing (MTZ) in Fuzzy Neural Networks

The following are the steps for updating a FuNN structure through zeroing in order to make the FuNN structure adaptable to new data and to be able to control the adaptation/forgetting phenomenon:

(1) A FuNN architecture is initialised (either randomly or according to an existing set of initial rules)

(2) The FuNN is trained with data. Backpropagation algorithms can be used. All the connections are subject to change during training.

(3) Fuzzy rules are extracted from the trained FuNN by using REFuNN algorithm (a threshold is set in advance, see [5,7])

(4) The FuNN architecture is then updated according to the extracted fuzzy rules by zeroing the connection weights which are less than a threshold (or their absolute values are less than the threshold if NOTs are used in the extracted fuzzy rules). A new updated FuNN is obtained, ZeFuNN1.

(5) The process of further training and updating (zeroing) the FuNN structures continues, thus new versions ZeFuNN2, ZeFuNN3 etc. are produced.

2.3. Rationale and Plausibility of the MTZ

There are several arguments in favour of the MTZ zeroing technique. Frequently zeroing of small connection weights may mean:

• removing the information which a neural network has learned from the noise in the training data; a better generalisation can be anticipated in this case; this may be one way to overcome the overfitting (overlearning) problem;
• removing low frequency components;
• keeping the underlying fuzzy rules of the previous data (previous experience) rather than keeping all the information; the threshold defines how much rough/precise the rules are;

Zeroing small connection weights is *biologically plausible* in the sense that synapses in the brain which have not been updated for a certain amount of time and are very small die out.

2.4. Experiments with the MTZ on Case Study Data Sets

Two case study data have been experimented with, the first one is a time series data on unemployment rate and the second one- the Fisher's Iris data set.

For the first case the problem is to predict the unemployment for the next quarter based on past data. The following attributes have been used: Time; Quarter; Private consumption expenditure; Government expenditure; Exports of Goods & Services; Number of people unemployed (in thousands) at the previous quarter; Number of people unemployed at the current quarter. The data set is divided into three sections (sets) A,B, and C.

The following experiments have been conducted:

. MLP networks are trained and tested on adaptation and forgetting; RMS error is used;
. FuNN networks are trained and tested; weighted simple fuzzy rules are extracted;
. ZeFuNN networks are produced, further trained and tested on adaptation and forgetting;

The results from the experiments are given in Table 1. CAd and CFr for the MLP2 model were calculated to be 16 and 4.2 correspondingly (see the first and the second lines in the table), while the values for the model ZeFuNN2 are 22 and 0.98 (lines 4 and 5). The results of a min-max compositional fuzzy inference on the extracted from ZeFuNN1 with the use the REFuNN algorithm and a threshold of 0.7 are shown in the last line of the table.

Table 1. Results from the experiment on adaptation and forgetting in a FuNN trained adaptively with a time series data set on unemployment prediction. Denotation: +(epochs) or ++(epochs) mean additional training started from the previous one, evaluated in the previous line.

Model	**Epochs**	**Data set A**	**Data set B**	**Data set C**
MLP1	10,000	E_{train}= 0.013	E_{gener}= 0.08	-
MLP2	+10,000	E_{forget}= 0.055	$E_{ad.train}$0.0055	E_{gener}= 0.038
MLP3	++10.000	E_{forget}= 0.032	E_{forget}= 0.037	$E_{ad.train}$= 0.02
ZeFuNN1, 0.7	-	E_{gener}= 0.22	E_{gener}= 0.55	-
ZeFuNN2	+500	E_{forget}= 0.2	$E_{ad.train}$= 0.025	E_{gener}= 0.11
ZeFuNN2	++500	E_{forget}= 0.2	$E_{ad.train}$= 0.016	E_{gener}= 0.12
ZeFuNN2	+++2,000	E_{forget}= 0.2	$E_{ad.train}$= 0.009	E_{gener}= 0.12
ZeFuNN2	++++400	E_{forget}= 0.21	E_{forget}= 0.09	$E_{ad.train}$= 0.05
fuzzy infer.	-	E_{gener}= 0.41	-	-

For the second case study problem the Iris data set is segmented into three sections: training section (90 instances); test section (30 instances); validation section (30 instances). Three triangular membership functions have been used to quantize the input and the class variables. A FuNN is trained on the training data section, fuzzy rules with NOT hedges have been extracted. A ZeFuNN structure is produced by using a threshold of 1.0. The learning, generalisation, adaptation and forgetting classification errors for different cases are shown in Table 2.

2.5. Using Genetic Algorithms to Optimise the MTZ

Genetic algorithms have been used so far for optimising parameters of fuzzy or neuro systems, e.g. optimising the number and the type of fuzzy rules, finding optimal number of hidden nodes and connection weights in neural networks. There are several points where genetic algorithms can be used for optimising FuNN structures and the MTZ: finding an optimal number of hidden nodes in a FuNN structure; building a dynamically growing and shrinking FuNN structure; finding the optimum values for the thresholds in a FuNN architecture for the purpose of rules extraction and zeroing connection weights afterwards.

Table 2. Experiments with FuNNs and ZeFuNNs on Iris data (three membership functions). TRN- training data segment (90 instances); TST - test segment (30); VAL - validation segment (30)

	Setosa	Versicolor	Virginica
Setosa	30		
Versicolor		28	2
Virginica		1	29

(a) A 12-9-9 FuNN trained on TRN data for 200 epochs and tested on the same data

	Setosa	Versicolor	Virginica
Setosa	10		
Versicolor		10	
Virginica			10

(b) The FuNN from (a) tested on the TST data

	Setosa	Versicolor	Virginica
Setosa	10		
Versicolor		6	4
Virginica		3	7

(c) The FuNN from (a) tested on the VAL data

	Setosa	Versicolor	Virginica
Setosa	30		
Versicolor		28	2
Virginica		1	29

(d) The FuNN from (a) is zeroed by keeping only the positive and negative connections which absolute values are above a chose threshold of 1.0. The new ZeFuNN is additionally trained for 50 epochs with VAL data and tested for forgetting on the TRN data

	Setosa	Versicolor	Virginica
Setosa	10		
Versicolor		10	
Virginica			10

(e) The ZeFuNN from (d) tested on the TST data

	Setosa	Versicolor	Virginica
Setosa	10		
Versicolor		10	
Virginica			10

(f) The ZeFuNN from (d) tested on the VAL data

These values may change dynamically over time with the progress of the training and the adaptation process. As a criterion for fitness a desired ratio adaptation/forgetting, set individually for every case, can be used.

3. Software Tools for Experimenting with the MTZ in a Hybrid Environment

As part of a hybrid environment FuzzyCOPE [8] a module FuNN has been developed to facilitate: initialisation of fuzzy neural networks with a set of fuzzy rules; training; rules extraction; zeroing of small connection weights when a threshold is set; test and training error evaluation; calculation of adaptation and forgetting coefficients as given in section 2. The FuNN module can be used for developing practical applications in different areas, such as signal processing and process control. The FuzzyCOPE environmnet is available free from the ftp site: eros.otago.ac.nz/pub/programs/ai/fuzzy-cope.

4. Conclusions

The paper discusses and illustrates several characteristics of fuzzy neural networks, mainly: fast training; good explanation on what has been learned by the network; means for rules extraction and rules refinement; means for adaptation; robustness; better control of the adaptation and forgetting phenomena; restricted (by the structure) way of capturing knowledge (it represents one view point only). The experiments show that updated (zeroed) fuzzy neural networks through the MTZ are more robust to forgetting, faster and better at adaptation and provide a good generalisation. They can be used for building practical real-time adaptable systems. It may be possible now to control the level of adaptation and forgetting through using different thresholds for fuzzy rules extraction. The use of specialised environments is crucial for the further development of these techniques.

Further research has been planned in the following directions: use of dynamically changing thresholds for rules extraction from fuzzy neural networks; use of genetic algorithms for finding the correlation between the forgetting/adaptation phenomenon and the threshold used for fuzzy rules extraction in a FuNN; finding analytical expressions for the rate of adaptation and forgetting in a fuzzy-neuro system; applications for real-time adaptable spoken language recognition systems; applications for decision making systems based on time series data; applications for data trawling and data mining; applications for modelling brain damage, forgetting and recovery phenomena.

References

[1] T. Furuhashi, Hasegawa, T., Horikawa S., Uchikawa, Y., An Adaptive Fuzzy Controller Using Fuzzy Neural Networks, in: *Proceedings of Fifth IFSA World Congress* (1993) 769- 772.

[2] M.M. Gupta, D.H. Rao, On the principles of fuzzy neural networks, *Fuzzy Sets and Systems*, 61 (1) (1994) 1-18.

[3] M.Brown and C.Harris, Neurofuzzy Adaptive Modelling and Control, Prentice Hall, 1994

[4] M. Ishikawa, Neural Network Approach to Rule Extraction, in: N.Kasabov and G.Coghill (eds) Proc. of the Second New Zealand Int. Conf. on Artificial Neural Networks and Expert Systems ANNES'95, IEEE Computer Society Press, pp.6-9, 1995

[5] N. Kasabov, Learning and approximate reasoning in fuzzy neural networks and hybrid systems, in *Fuzzy Sets and Systems*, special issue, 1995

[6] N. Kasabov, Adaptable Neuro Production Systems, to appear in *Neurocomputing*, Elsevier Science Publ. B.V, [7] N. Kasabov, *Neural Networks, Fuzzy Systems and Knowledge Engineering*, MIT Pres, 1996

[8] N.Kasabov, Hybrid Connectionist Fuzzy Production Systems - Towards Building Comprehensive AI, *Intelligent Automation and Soft Computing* vol.1, No.4, 351-360, 1995