INTERPRETING CHANGES IN THE FUZZY SETS OF A SELF-ADAPTIVE NEURAL FUZZY CONTROLLER

Detlef Nauck Rudolf Kruse
Department of Computer Science
Technical University of Braunschweig
W-3300 Braunschweig, Germany

ABSTRACT

We describe a procedure for the adaptation of membership functions in a fuzzy control environment by using neural network learning principles. The changes in the fuzzy sets can be easily interpreted. By using a fuzzy error that is propagated back through the architecture of our fuzzy controller, we receive an unsupervised learning technique, where each rule tunes the membership functions of its antecedent and its consequence.

INTRODUCTION

Classical control theory is based on mathematical models that describe the behaviour of the plant under consideration. The main idea of fuzzy control [9, 10], which has proved to be a very successful method [5], is to build a model of a human control expert who is capable of controlling the plant without thinking in a mathematical model. The control expert specifies his control actions in the form of linguistic rules. These control rules are translated into the framework of fuzzy set theory providing a calculus which can simulate the behaviour of the control expert.

The specification of good linguistic rules depends on the knowledge of the control expert. But the translation of these into fuzzy set theory is not formalized and arbitrary choices concerning for example the shape of membership functions have to be made. The quality of the fuzzy controller can be drastically influenced by changing shapes of membership functions. Thus methods for tuning fuzzy controllers are necessary.

Neural networks offer the possibility to the solve this tuning problem. The combination of neural networks and fuzzy controllers assembles the advantages of both approaches and avoids the drawbacks of them. Although a neural networks is a able to learn from given data, the trained neural network is generally understood as a black box. Neither is it possible to extract structural knowledge from the trained neural network, nor can we integrate special information about the problem into the neural network in order to simplify the learning procedure. On the other hand, a fuzzy controller is designed to work with knowledge in the form of rules. But there exists no formal framework for the choice of parameters and the optimization of parameters has to be done by hand.

We solve one of the design problems, the choice and the tuning of a priori membership functions in order to improve the performance of the fuzzy controller, by defininig a fuzzy error that is propagated back through the neural-like architecture of our fuzzy controller. According to this fuzzy error, the strength of its antecedent, the current state of the controlled system and the control action derived from the consequence, each fuzzy rule determines its amount of error and tunes the membership functions of its antecedent and its consequence. This paper is an extension to our work in [3], and the fuzzy error propagation algorithm is described at length in [11].

In: Proc. 2nd Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems IFIS'92, College Station, Texas, Dec. 2-4 (1992), 146-152

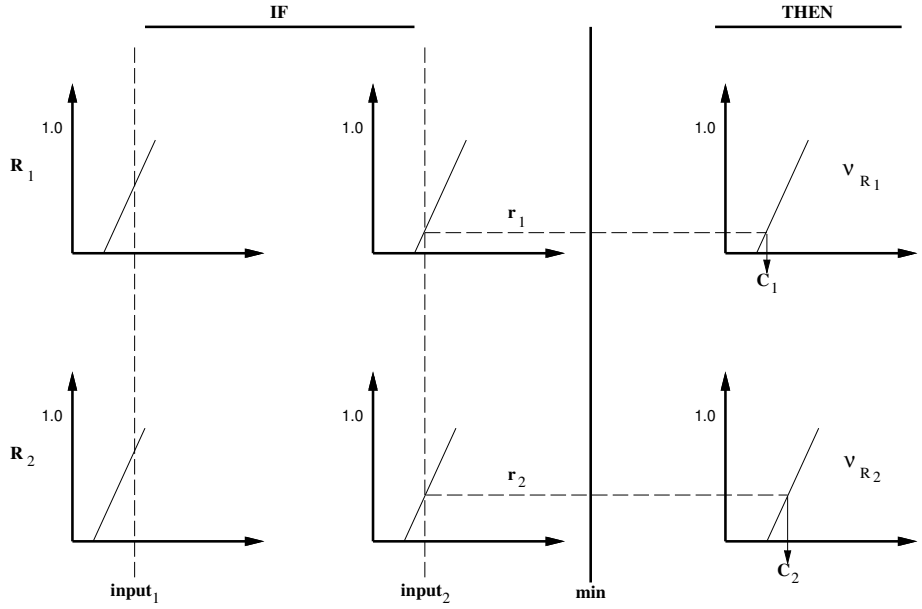

Figure 1: Defuzzification using Tsukamoto's monotonic membership functions

We refrained from just integrating neural nets in certain parts of the architecture as black boxes, or from adding an extra module to the architecture taking care of the correction of errors for example by weighting the rules according to the errors [8, 15]. There exist approaches that use neural networks to create or tune fuzzy sets from training data [4, 6, 13, 16, 17], and the extraction of fuzzy rules from training data is presented e.g. in [2, 8, 14]. Our approach that is described in this paper has to be compared to hybrid approaches like the ARIC architecture of Berenji presented in [1].

Our main concern is to keep the structure of the fuzzy controller that is determined by the fuzzy rules. We think of these rules as a piece of structural knowledge that gives us a roughly correct representation of the system to be controlled. We view the membership functions used in the antecedents and consequents of the rules as sub-optimal representations of the linguistic values attributed to the variables of the plant to be controlled. The tuning of the membership functions gives us a better representation of theses linguistic values under the given circumstances, and increases the performance of the controller.

We consider it a natural approach to use a fuzzy error for our system, which, according to its structure, we may call neural fuzzy controller. This enables us to define a non-supervised learning algorithm that does not need a comparison between the actual output value and a desired optimal value (which is not known in general) but learns from reinforcement. Training a fuzzy controller with such a learning procedure allows us to keep track of the changes and to interpret the modified rules.

A NEURAL FUZZY CONTROLLER

We consider a dynamical system S that can be controlled by one variable C and whose state can be described by n variables $X_1, \ldots, X_n$, i.e. we have a multiple input – single output system. The linguistic values of the variables are modelled by membership functions, and the control action that drives the system S to a desired state is described by the well-known concept of fuzzy if-then rules [18].

To overcome the problem of defuzzification, and to determine the individual part that each rule contributes to the final output value, we use Tsukamoto's monotonic membership functions (see figure 1), where the defuzzification is reduced to an application of the inverse function [1, 10]. Such a membership function μ is

characterized by two points a, b with $\mu(a) = 0$ and $\mu(b) = 1$, and it is defined as

$$\mu(x) = \begin{cases} \frac{-x+a}{a-b} & \text{if } (x \in [a,b] \wedge a \leq b) \vee (x \in [b,a] \wedge a > b) \\ 0 & \text{otherwise} \end{cases}$$

The defuzzification is carried out by

$$x = \mu^{-1}(y) = -y(a-b) + a$$

with $y \in [0,1]$.

Consider the following two rules

R_1: **IF** θ is PM **AND** $\dot{\theta}$ is PS **THEN** F is PS,
R_2: **IF** θ is PS **AND** $\dot{\theta}$ is PZ **THEN** F is PZ,

where PM, PS and PZ represent the usual linguistic expressions *positive medium, positive small* and *positive zero.* The evaluation of these kind of rules is presented in figure 1.

For our purposes we only need to restrict ourselves to monotonic membership functions to represent the linguistic values of the output variable. For the input variables the usual triangular, trapezoidal etc. membership functions can be chosen.

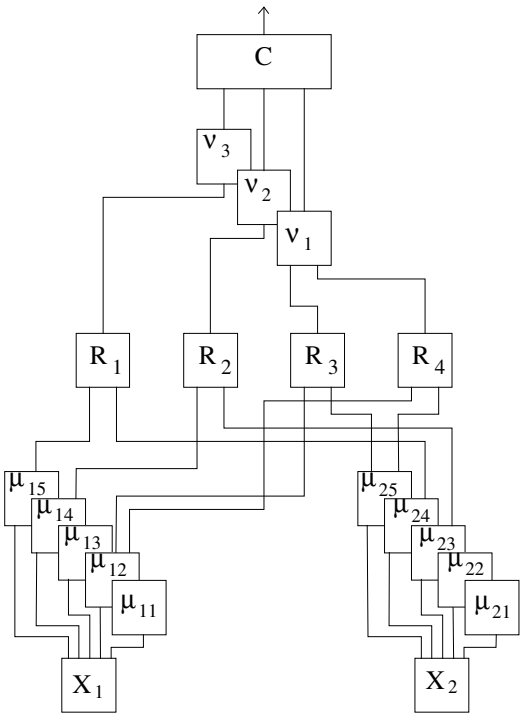

Figure 2: The structure of the neural fuzzy controller

An example for the structure of our neural fuzzy controller is depicted in figure 2. The modules X_1 and X_2 represent the input variables, and deliver their crisp values to their μ-modules which contain the respective membership functions. The μ-modules are connected to the R-modules which represent the fuzzy if-then

rules. Each μ-module gives to all of its connected R-modules, the membership value $\mu_{ij}(x_i)$ of its input variable X_i. The R-modules use a t-norm (min-operation in this case) to calculate the conjunction of their inputs and pass this value forward to one of the ν-modules, which contain the membership functions representing the output variable. By passing through the ν-modules these values are changed to the conclusion of the respective rule. This means the implication (min-implication in this case) is carried out to obtain the value of the consequence, and because monotonic membership functions are used, the ν-modules pass pairs $(r_l, \nu_k^{-1}(r_l))$ to the C-module, where the final output value is calculated by

$$c = \frac{\sum_{i=1}^{n} r_i \nu_{R_i}^{-1}(r_i)}{\sum_{i=1}^{n} r_i},$$

where n is the number of rules, and r_i is the degree to which rule R_i has fired.

As one can easily see, the system in figure 2 resembles a feedforward neural network. The X-, R-, and C-modules can be viewed as the neurons and the μ- and ν-units as the adaptable weights of the network. The fact that one μ-module can be connected to more than one R-module is equivalent to connections in a neural network that share a common weight. This is very important, because we want each linguistic value to be represented by only one membership function that is valid for all rules. By this restriction we retain the structural knowledge that we put into the system by defining the rules. In other neural fuzzy systems this fact is not recognized [1, 7] and it is possible that one linguistic value is represented by different membership functions.

FUZZY ERROR PROPAGATION

Our goal is to tune the membership functions of the controller by a learning algorithm according to a measure that adequately describes the state of the plant under consideration. The desired optimal state of the plant can be described by a vector of state variable values. But usually we are content with the current state if the variables have roughly taken these values. And so it is natural to define the goodness of the current state by a membership function from which we can derive a fuzzy error [11] that characterizes the performance of our neural fuzzy controller.

Consider a system with n state variables $X_1, \ldots, X_n$. We define the fuzzy-goodness G_1 as

$$G_1 = \min(\mu_{X_1}^{optimal}, \ldots, \mu_{X_n}^{optimal}),$$

where the membership functions $\mu_{X_i}^{optimal}$ have to be defined according to the requirements of the plant under consideration.

In addition of a near optimal state we also consider states as good, where the incorrect values of the state variables compensate each other in a way, that the plant is driven towards its optimal state. We define the fuzzy-goodness G_2 as

$$G_2 = \min(\mu^{compensate_1}(X_1, \ldots, X_n), \ldots, \mu^{compensate_k}(X_1, \ldots, X_n))$$

where the membership functions $\mu^{compensate_j}$ again have to be defined according to the requirements of the plant. There may be more than one $\mu^{compensate_j}$ and they may depend on two or more of the state variables.

The overall fuzzy-goodness is defined as

$$G = g(G_1, G_2),$$

where the operation g has to be specified according to the actual application [11]. The fuzzy-error that is made by our neural fuzzy controller is defined as

$$E = 1 - G.$$

The learning algorithm depending on this fuzzy error works for each fuzzy rule in parallel. Each rule R_i knows the value r_i of the conjunction of its antecedents and the (crisp) value c_i of its consequence. After the control action has been determined by the controller and the new state of the plant is known, we propagate the fuzzy error E and the current values of the state variables to each R-module. If the rule has contributed to the control output, i.e. $r_i \neq 0$, it has to evaluate its own consequence. According to the current state of the plant the rule can decide, whether its conclusion would drive the system to a better or to a worse state. The actual optimal control value c_{opt} cannot be determined, but its direction, i.e. $\text{sgn}(c_{\text{opt}})$, is known. For the case $\text{sgn}(c_i) = \text{sgn}(c_{\text{opt}})$ the rule has to be made more sensitive and has to produce a conclusion that increases the current control action, i.e. makes it more positive or negative respectively. For the second case the opposite action has to be taken.

Consider that we are using Tsukamoto's monotonic membership functions. Each membership function can be characterized by a pair (a,b) such that $\mu(a) = 0$ and $\mu(b) = 1$ hold. A rule is made more sensitive by increasing the difference between these two values in each of its antecedents. That is done by keeping the value of b and changing a. That means the membership functions are keeping their positions determined by their b-values, and their ranges determined by $|a - b|$ are made wider. To make a rule less sensitive the ranges have to be made smaller. In addition to the changes in its antecedents, each firing rule has to change the membership function of its consequence. If a rule has produced a good control value, this value is made better by making the range of the membership function smaller, and a bad control value is made less worse by making the range wider.

The rules change the membership functions by propagating their own rule-error

$$e_{R_i} = \begin{cases} -r_i \cdot E & \text{if } \text{sgn}(c_i) = \text{sgn}(c_{\text{opt}}) \\ r_i \cdot E & \text{if } \text{sgn}(c_i) \neq \text{sgn}(c_{\text{opt}}) \end{cases}$$

to the connected μ- and ν-modules. The changes in the membership functions of the consequences (ν-modules) are calculated according to

$$a_k^{\text{new}} = \begin{cases} a_k - \sigma \cdot e_{R_i} \cdot |a_k - b_k| & \text{if } (a_k < b_k) \\ a_k + \sigma \cdot e_{R_i} \cdot |a_k - b_k| & \text{otherwise,} \end{cases}$$

where σ is a learning factor and R-module R_i is connected through ν_k to the C-module. If ν_k is shared, it is changed by as much R-modules as are connected to the C-module through this membership function. For the membership functions of the antecedents (μ-modules) the follwing calculation is carried out:

$$a_{jk_j}^{\text{new}} = \begin{cases} a_{jk_j} + \sigma \cdot e_{R_i} \cdot |a_{jk_j} - b_{jk_j}| & \text{if } (a_{jk_j} < b_{jk_j}) \\ a_{jk_j} - \sigma \cdot e_{R_i} \cdot |a_{jk_j} - b_{jk_j}| & \text{otherwise,} \end{cases}$$

where the X-module X_j is connected to the R-module R_i through the membership function μ_{jk_j}, with $k_j \in \{1, \ldots, s_j\}$, and s_j is the number of linguistic values of X_j. If μ_{jk_j} is shared, it is changed by as much R-modules as X_j is connected to through this μ-module.

The neural fuzzy controller and the fuzzy error propagation algorithm have been implemented and have been applied to the simulation of an inverted pendulum [11]. For reasons of simplicity Tsukamoto's monotonic membership functions are use in the antecedents and consequents of the rules, although non-monotonic functions can be chosen for the antecedents. For each variable eight linguistic variables are defined.

The fuzzy error E has been defined by

$$E = \begin{cases} 1 - \min(1 - \frac{|\theta|}{3}, 1 - \frac{|\dot{\theta}|}{0.3}) & \text{if } \text{sgn}(\theta) = \text{sgn}(\dot{\theta}),\ -3 \leq \theta \leq 3,\ -0.3 \leq \dot{\theta} \leq 0.3 \\ \frac{|\theta + 10\,\dot{\theta}|}{3} & \text{if } \text{sgn}(\theta) \neq \text{sgn}(\dot{\theta}),\ -3 \leq \theta + 10\,\dot{\theta} \leq 3 \\ 1 & \text{otherwise.} \end{cases}$$

That means the fuzzy error is defined by one two-dimensional and two one-dimensional membership functions. The learning rate σ has been set to 0.01, and θ is the angle, and $\dot{\theta}$ the angle velocity of the pendulum.

It was found that the controller was already able to balance the pendulum with the defined rule base and the initial membership functions, but it was not able to cope with extreme starting positions (e.g. angle > 20,

and angle velocity > 2). After the learning took place the controller was able to handle those situations with the help of the adapted membership functions. The simulation environment is currently under development and different kinds of membership functions and learning parameters are tried out.

An extension of the learning algorithms that can be used on an extended structure of the controller, which can be viewed as a fuzzy neural network, is able to learn the fuzzy rules, too [12].

INTERPRETATION OF THE LEARNING RESULTS

Our fuzzy error propagation algorithm is an unsupervised learning algorithm where each R-module learns according to the extent it has fired under the given circumstances and according to an error describing the performance of the control system. The neural fuzzy controller does not have to learn from scratch, but knowledge in the form of fuzzy if-then rules is coded into the system. The learning procedure does not change this structural knowledge. It tunes the membership functions in an obvious way, and the semantics of the rules are not blurred by any semantically suspicious factors or weights attached to the rules.

The algorithm finds the most appropriate membership functions for the current control situation, while the meaning of each rule is not changed. If the learning is successful, this means that the rule base is suitable for the control task, and the changes in the fuzzy sets are due to an unsuitable modelling of the linguistic values by the a priori membership functions. The adapted membership functions represent the linguistic values of the variables in way that is best suited for the given circumstances and reduce the fuzzy error made by the controller. Because it is not possible that two rules use different fuzzy sets describing the same linguistic value, we can actually interpret the new membership functions to give us the best fuzzy sets for the current task.

If the learning fails, this can be due to an unsuitable rule base, where either important rules are missing, or some rules are contradictory. In this case the structural knowledge is incomplete, and the rule base has to be refined by other means, and the learning process has to be started over, or a rule learning algorithm has to be used [12].

CONCLUSIONS

We have presented a learning algorithm for a neural fuzzy controller based on a fuzzy error measure. The structure of the controller resembles a neural network and the fuzzy error propagation can be compared to non-supervised learning procedures as they exist for certain kinds of connectionistic systems. Simulations of the controller used to balance an inverted pendulum have shown that the learning procedure improves the behaviour of the fuzzy controller and is able to handle extreme situations where the non-learning controller fails [11].

The introduced fuzzy error measure is suitable for describing the performance of the controller and allows each rule to determine changes for the membership functions of its antecedent and its consequence. The learning algorithm starts from a predefined rule base, and it does not change the structural knowledge encoded in these rules. It leaves the semantics of each rule intended by the user unchanged, but removes the errors caused by an inaccurate modelling by changing the fuzzy sets. The results of the learning procedure can be easily interpreted.

REFERENCES

[1] Berenji, Hamid R. (1992). A Reinforcement Learning-Based Architecture for Fuzzy Logic Control. *Int. J. Approximate Reasoning.* Vol. 6, No. 2, 267-292.

[2] Bezdek, J.C., Tsao, E.C., Pal, N.K. (1992).Fuzzy Kohonen Clustering. *Networks. Proc. IEEE International Conference on Fuzzy Systems 1992, San Diego*, 1035–1043.

[3] Eklund, P.,Klawonn, F., Nauck, D. (1992). Distributing Errors in Neural Fuzzy Control. *Proc. Int. Conf. on Fuzzy Logic and Neural Networks (IIZUKA '92)*, 1139-1142.

[4] Hayashi, I., Nomura, H., Yamasaki, H., Wakami, N. (1992). Construction of Fuzzy Inference Rules by NFD and NDFL. *Int. J. Approximate Reasoning.* Vol. 6, 241–266.

[5] Hirota, K. (1992). Survey of Industrial Applications of Fuzzy Control in Japan. *Proc. IJCAI–91 Workshop on Fuzzy Control, Sydney,* 18–20.

[6] Ichihashi, H. (1992). Iterative Fuzzy Modelling and a Hierarchical Network. In: R. Lowen, M. Roubens (eds.) (1992). *Proc. 4th IFSA Congress, Engineering.* Brussels, 49–52.

[7] Jang, Jyh-Shing R. (1991). Fuzzy Modelling Using Generalized Neural Networks and Kalman Filter Algorithm. *Proc. AAAI-91.* Vol. 2, 762-767.

[8] Kosko, B. (1992). *Neural Networks and Fuzzy Systems.* Prentice–Hall, Englewood Cliffs.

[9] Lee, C.C. (1990). Fuzzy Logic Control Systems: Fuzzy Logic Controller – Part I. *IEEE Trans. Syst. Man Cybern.* Vol. 20, No. 2, 404-418.

[10] Lee, C.C. (1990). Fuzzy Logic Control Systems: Fuzzy Logic Controller – Part II. *IEEE Trans. Syst. Man Cybern.* Vol. 20, No. 2, 419-435.

[11] Nauck, D., Kruse, R. (1992). A Neural Fuzzy Controller Learning by Fuzzy Error Propagation. To be presented at *NAFIPS '92, Dec. 14–17, Puerto Vallarta, Mexico.*

[12] Nauck, D., Kruse, R. (1992). A Fuzzy Neural Network learning Fuzzy Control Rules and Membership Functions by Fuzzy Error Backpropagation. *Submitted for publication.*

[13] Nomura, H., Hayashi, I., Wakami, N. (1992). A Learning Method of Fuzzy Inference Rules by Descent Method. *Proc. IEEE International Conference on Fuzzy Systems 1992, San Diego,* 203–210.

[14] Pedrycz, W., Card, H.C. (1992). Linguistic Interpretation of Self–Organizing Maps. *Proc. IEEE International Conference on Fuzzy Systems 1992, San Diego,* 371–378.

[15] Shao, S. (1988). Fuzzy Self-Organizing Controller and its Application for Dynamic Processes. *Fuzzy Sets and Systems.* Vol. 26, 151-164.

[16] Takagi, H., Hayashi, I. (1991). NN–Driven Fuzzy Reasoning. *Int. J. Approximate Reasoning.* Vol. 5, 191–212.

[17] Takagi, H., Kouda, T., Kojima, Y. (1991). Neural Networks Based on Approximate Reasoning Architecture. *Japanese Journal of Fuzzy Theory and Systems.* Vol. 3, 63-74.

[18] Zadeh, L.A. (1973). Outline of a New Approach to the Ananlysis of Complex Systems and Decision Processes. *IEEE Trans. Syst. Man Cybern.* Vol. SMC–3, 28–44.