# An ARCHITECTURE FOR DESIGNING FUZZY LOGIC CONTROLLERS USING NEURAL NETWORKS

Hamid R. Berenji
Sterling Federal Systems
Artificial Intelligence Research Branch
NASA Ames Research Center
MS: 244-17, Moffett Field, CA 94035
e-mail: berenji@pluto.arc.nasa.gov

## Abstract

In this paper, we describe an architecture for designing fuzzy controllers through a hierarchical process of control rule acquisition and by using special classes of neural network learning techniques. Hierarchical development of the fuzzy control rules is a useful technique which has been used earlier in designing a fuzzy controller with interactive goals [5]. Also, we introduce a new method for learning to refine a fuzzy logic controller. A reinforcement learning technique is used in conjunction with a multi-layer neural network model of a fuzzy controller. The model learns by updating its prediction of the plant's behavior and is related to the Sutton's Temporal Difference (TD) method. The method proposed here has the advantage of using the control knowledge of an experienced operator and fine-tuning it through the process of learning. The approach is applied to a cart-pole balancing system.

# 1 Introduction

Fuzzy logic controllers have recently experienced a huge commercial success [12,6]. These controllers are usually developed based on the knowledge of human expert operators[4]. However, starting with the Self Organizing Control (SOC) techniques of Mamdani and his students (e.g., [9]), the need for research in developing fuzzy logic controllers which can learn from experience has been realized (e.g., [8]). The learning task may include the identification of the main control parameters (i.e., related to the system identification in conventional and modern control theory) or development and fine-tuning of the fuzzy memberships used in the control rules. In this paper, we concentrate on the latter learning task and develop a model which can learn to adjust the fuzzy memberships of the linguistic labels.

The organization of this paper is as follows. We first discuss the general model of our NeuroFuzzy Controller (NFC) and then we apply this model to the control of a cart-pole balancing system. Finally, we compare this model with other related research works such as the credit assignment in artificial intelligence [10], Barto et. al.'s AHC model [3], and Lee and Berenji's single layer model [8].

# 2 NFC: A Model for Intelligent Control

Figure 1 illustrates the general model of our intelligent controller. The two main elements in this model are the Action-state Evaluation Network (AEN), which acts as a critic and provides advice to the main controller, and the Action Selection Network (ASN) which includes a fuzzy controller.

## 2.1 Action-state Evaluation Network (AEN)

The only information received by the AEN is the state of the plant in terms of its state variables and whether a failure has occurred or not. Figure 2 illustrates the structure of an evaluation network including $m_h$ hidden units and $n$ input units from the environment (i.e., $x_0$, $x_1$,..., $x_n$). The triangles represent the calculation-center [1] of the units where the updating equations

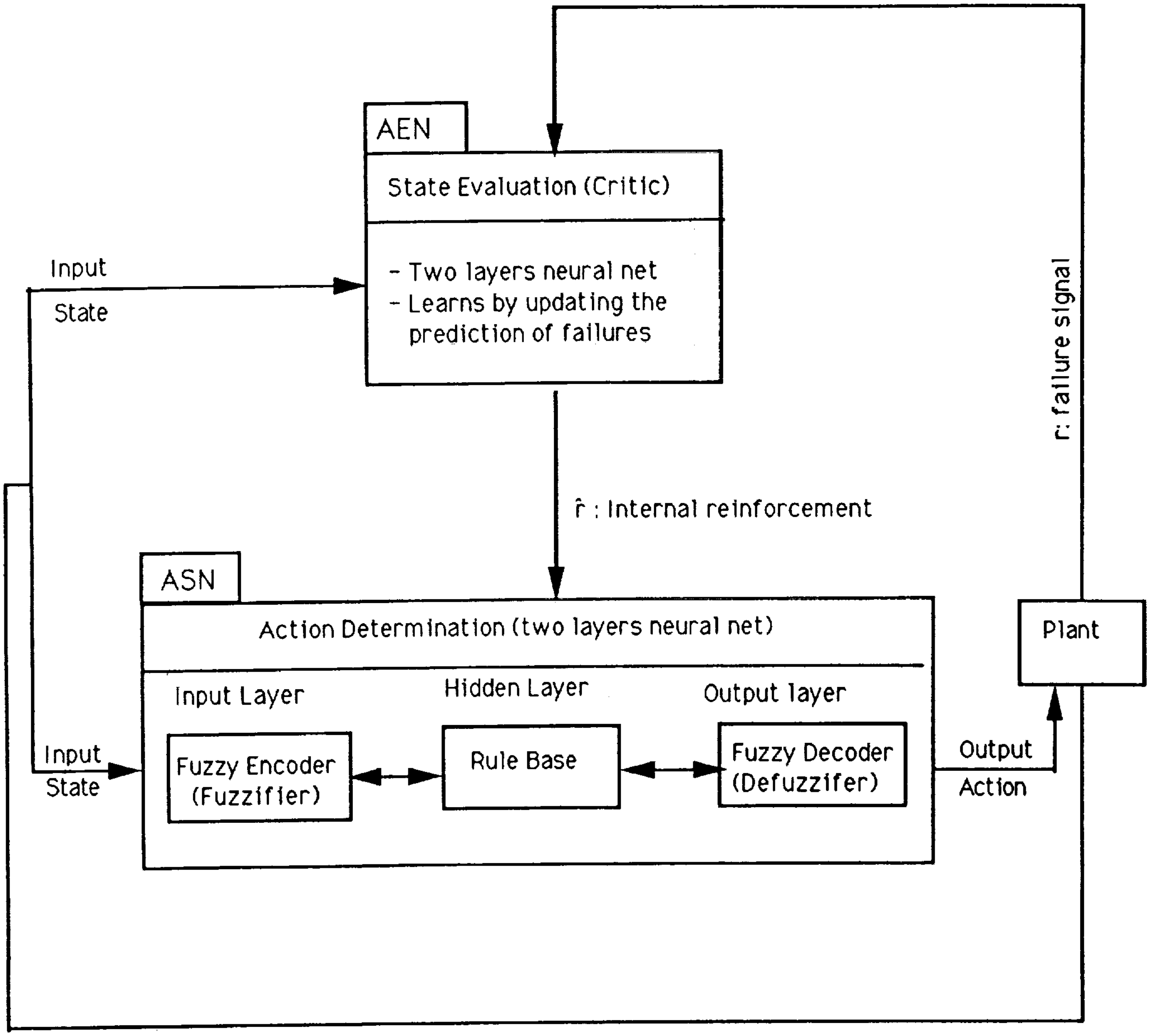
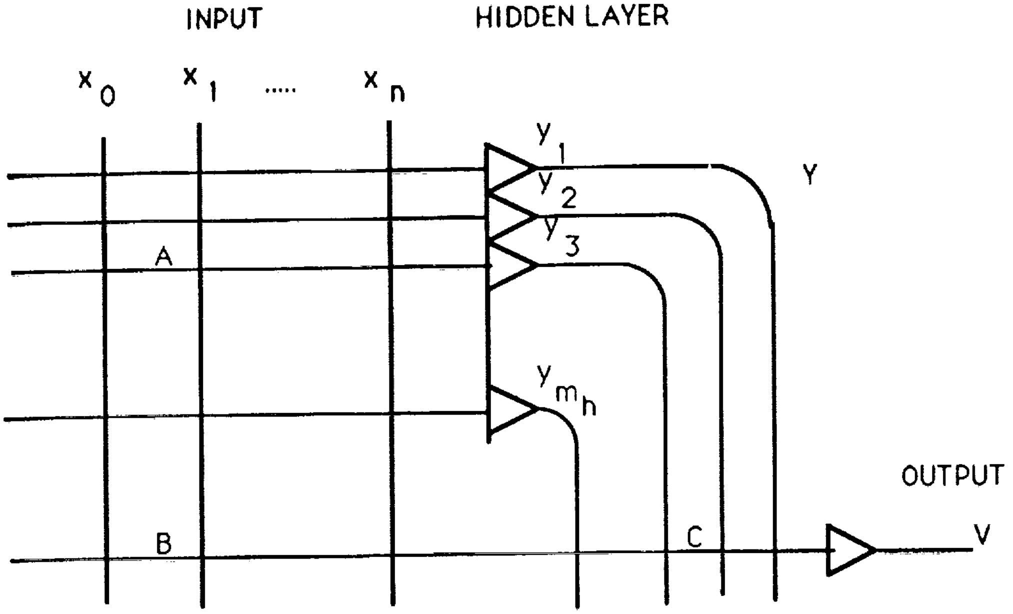

AEN
State Evaluation (Critic)
- Two layers neural net
- Learns by updating the prediction of failures
Input
State
r: failure signal
r̂ : Internal reinforcement
ASN
Action Determination (two layers neural net)
Input Layer
Hidden Layer
Output layer
Input
State
Fuzzy Encoder (Fuzzifier)
Rule Base
Fuzzy Decoder (Defuzzifer)
Output
Action
Plant

Figure 1: The NFC Model for Intelligent Control



A: Matrix of weights on the arcs connecting the input layer to the hidden layer
B: Matrix of weights on the arcs connecting the input layer to the output layer
C: Matrix of weights on the arcs connecting the hidden layer to the output layer

Figure 2: The Evaluation Network

(to be described bellow) are applied. The input from the environment is provided to all hidden units and output units while an interconnection weight exists at every intersection. Therefore in this network, hidden units receive $n+1$ inputs and have $n+1$ weights each while the output units receive $n+1+m_h$ inputs and have $n+1+m_h$ weights. If $A, B, C$ are the matrices of connection weights, then the output of the evaluation network is:

$$v[t_1,t_2] = \sum_{i=1}^{n} b_i[t_1]x_i[t_2] + \sum_{i=1}^{m_h} c_i[t_1]y_i[t_1,t_2] \tag{1}$$

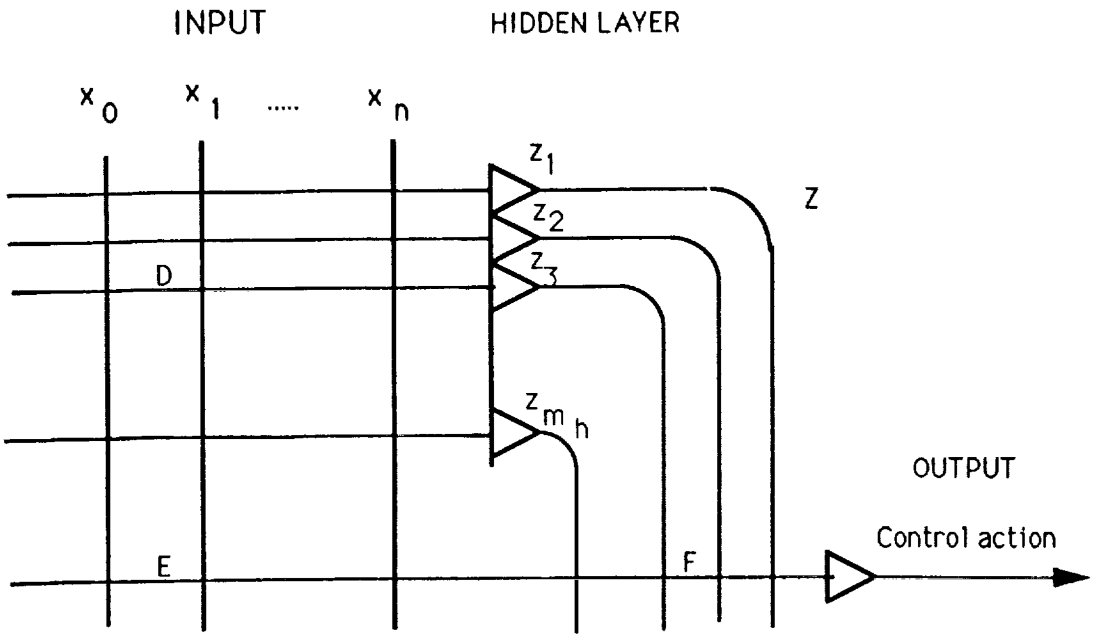
where

$$y_i[t_1,t_2] = g(\sum_{j=1}^{n} a_{ij}[t_1]x_j[t_2]) \tag{2}$$

and

$$g(s) = \frac{1}{1+e^{-s}} \tag{3}$$

In the above equations, double time dependencies are used to avoid instabilities in the updating of weights [2]. This network evaluates the action recommended by the action network as a function of the failure signal and



D: Matrix of weights on the arcs connecting the input layer to the hidden layer
E: Matrix of weights on the arcs connecting the input layer to the output layer
F: Matrix of weights on the arcs connecting the hidden layer to the output layer

Figure 3: The Action Selection Network

the change in state evaluation:

$$\hat{r}[t{+}1] = \begin{cases} 0 & \text{if state at time } t+1 \text{ is a start state;} \\ r[t+1] - v[t,t] & \text{if state at time t+1 is a failure state;} \\ r[t+1] + \gamma v[t,t+1] - v[t,t] & \text{otherwise} \end{cases} \tag{4}$$

The weights in this network are modified according to the followings:

$$b_i[t+1] = b_i[t] + \beta\hat{r}[t+1]x_i[t] \tag{5}$$

$$c_i[t+1] = c_i[t] + \beta\hat{r}[t+1]y_i[t,t] \tag{6}$$

$$a_{ij}[t+1] = a_{ij}[t] + \beta_h\hat{r}[t+1]y_i[t,t](1 - y_i[t,t])sgn(c_i[t])x_j[t] \tag{7}$$

where $0 < \gamma \leq 1$ and $\beta, \beta_h > 0$.

## 2.2 Action Selection Network (ASN)

The Action Selection Network (ASN) includes a fuzzy controller which consists of a fuzzifier, a rule base and decision making logic, and a defuzzifier all represented in a network. The design of the rule base for this fuzzy controller follows the algorithm developed in [5] which is based on a hierarchical process considering the interaction of multiple goals.

In this paper, the above fuzzy controller is modeled by a two layered neural network where the input layer includes the fuzzifier whose task is to match the values of the input variables against the labels used in the fuzzy control rules. The hidden layer in this network corresponds to the rules used in the controller and includes the decision making logic. The output layer includes the decoding (defuzzification) process. In the following, a brief explanation on fuzzy logic control is provided. However, for more detailed information, see [4]. The action selector is shown in Figure 3, where the matrices of connection weights are $D$, $E$, and $F$. The individual member of these matrices are labelled $d_{ij}$, $e_i$, and $f_i$. In this network, the hidden nodes represent a fuzzy control rule in the following manner. The inputs to the node are the preconditions of a rule and the output of the node is its conclusion. We assume a Multi Input Single Output (MISO) control system. The output layer combines the conclusion of the individual rules by using the Center Of Area (COA) method [4] which is described below. Let $w(i)$ represent the degree that rule $i$ is satisfied by the input state variables in $X$ which means

$$w(i) = Min\{d_{i1}\mu_{i1}(x_1), d_{i2}\mu_{i2}(x_2), \ldots, d_{in}\mu_{in}(x_n)\} \tag{8}$$

where $\mu_{i1}(x_1)$ represents the degree of membership of the input $x_1$ in a fuzzy set representing the label used in the first precondition of the rule $i$ and $n$ is the number of inputs. Then $m(i)$, which represents the result of applying the $w(i)$ on the conclusion of rule $i$, is calculated from

$$w(i) = \mu_{C_i}(m(i)) \tag{9}$$

where $\mu_{C_i}$ represents the monotonic membership function of the label used in the conclusion of rule $i$. The amount of the control action (i.e., $u$) is then calculated by using the Center Of Area (COA) method as the following. Assuming discretized membership functions, COA reveals

$$u(t) = \frac{\sum_{i=1}^{m_h} f_i \times m(i) \times w(i)}{\sum_{i=1}^{m_h} w(i) \times f_i} \tag{10}$$

where $m_h$ is the number of nodes in the hidden layer which is equivalent to the number of rules used in the model. We define two more functions here:

$$z_i[t] = g(\sum_{j=1}^{n} d_{ij}[t]x_j[t]) \tag{11}$$

$$p[t] = g(\sum_{i=1}^{n} e_i[t]x_i[t] + \sum_{i=1}^{m_h} f_i[t]z_i[t]) \tag{12}$$

and

$$q[t] = \begin{cases} 1, \text{ with probability } p[t]; \\ 0, \text{ with probability } 1 - p[t] \end{cases} \tag{13}$$

The connection weights are updated according to the followings:

$$e_i[t+1] = e_i[t] + \rho\hat{r}[t+1](q[t] - p[t])x_i[t] \tag{14}$$

$$f_i[t+1] = f_i[t] + \rho\hat{r}[t+1](q[t] - p[t])z_i[t] \tag{15}$$

$$d_{ij}[t+1] = d_ij[t] + \rho_h\hat{r}[t+1]z_i[t](1 - z_i[t])sgn(f_i[t])(q[t] - p[t])x_j[t] \tag{16}$$

where $\rho$ and $\rho_h > 0$.

# 3 Applying NFC to Cart-Pole Balancing

In this section, we describe the cart-pole balancing problem and apply the NFC model to its control.

## 3.1 The Cart-Pole balancing problem

In this system a pole is hinged to a motor-driven cart which moves on rail tracks to its right or its left. The pole has only one degree of freedom (rotation about the hinge point). The primary control tasks are to keep the pole vertically balanced and keep the cart within the rail tracks boundaries.

Four state variables are used to describe the system status, and one variable represents the force applied to the cart. These are:

$x$ : horizontal position of the cart on the rail<br>
$\dot{x}$ : velocity of the cart<br>
$\theta$ : angle of the pole with respect to the vertical line<br>
$\dot{\theta}$ : angular velocity of pole<br>
$u$ : force applied to the cart.

We assume that a failure happens when $|\theta| > 12$ degrees or $|x| > 2.4$ meters. Also, we assume that the equations of motion of the cart-pole system are not known to the controller and only a vector describing the cart-pole system's state at each time step is known. In other words, the cart-pole balancing system is treated as a black box by the learning system.

Figure 4 presents the model of NFC as it is applied to this problem. Among the components of this model, we only describe the Action Selection Network here.

## 3.2 The Action Selection Network

The action network was modeled by defining a multi-layered neural network which receives reinforcements from the evaluation network. This network, as shown in Figure 4, consists of 5 input nodes representing the four state variables and a bias unit, 13 nodes in the hidden layer, and an output node. The nodes in the hidden layer correspond to the fuzzy control rules. For example, node 1 corresponds to the rule:

IF $\theta$ is Positive and $\dot{\theta}$ is Positive Then Force is Positive-Large.

As mentioned earlier, the rule base of a fuzzy controller consists of rules which are described using *linguistic variables*. As shown in Figure 5(a) and Figure 5(b), three labels are used here to linguistically define the value of the state variables: Positive (P), Zero (Z), and Negative (N). Seven labels are used to linguistically define the value of force recommended by each control rule: Positive Large (PL), Positive Medium (PM), Positive Small (PS), Zero (ZE), Negative Small (NS), Negative Medium (NM), and Negative Large (NL). The forward calculations in this network is based on fuzzy logic control as described in [5], where nine fuzzy control rules were written for balancing the pole vertically and four control rules were used in positioning the cart at a specific location on the rail tracks. The presence of a connection between

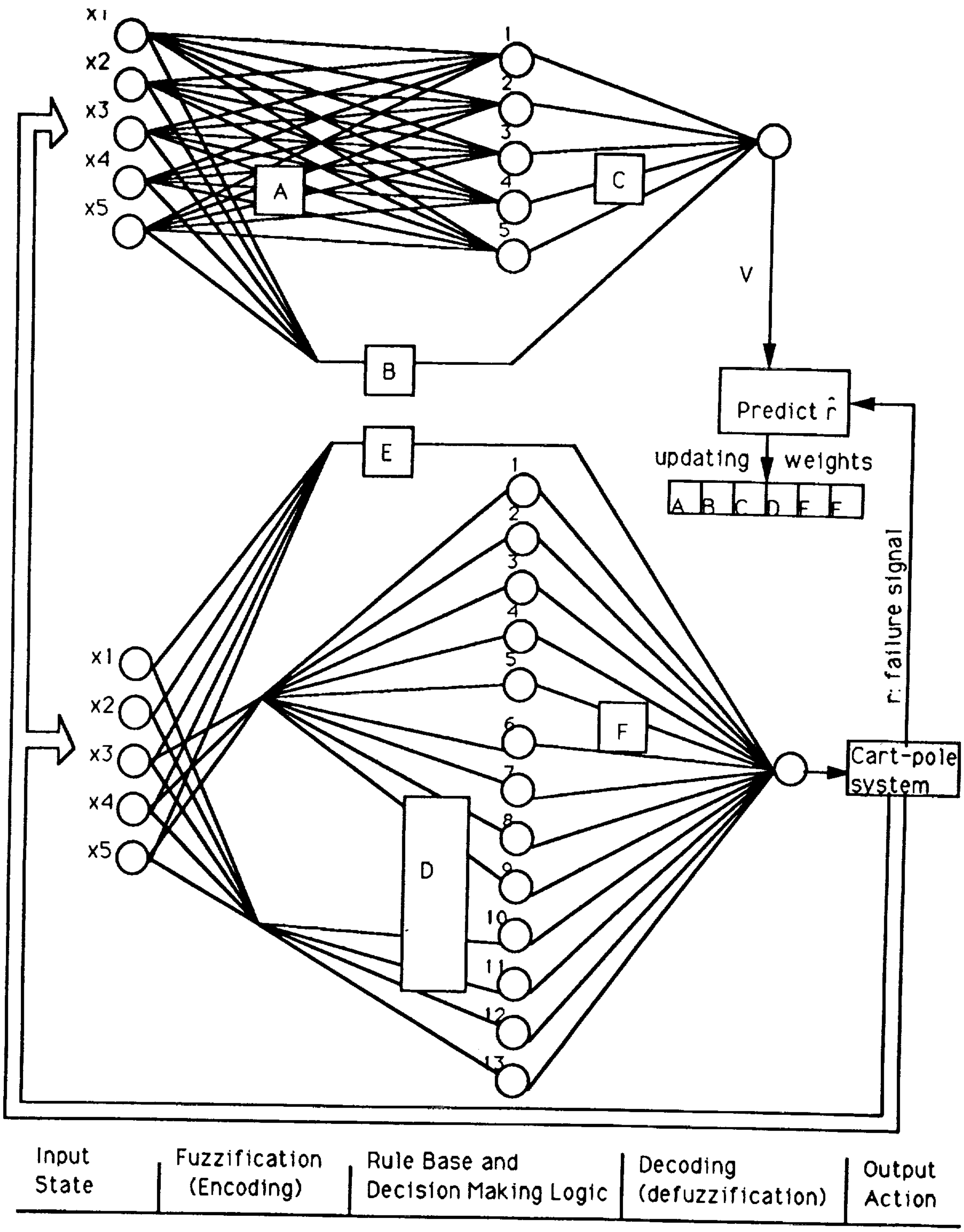


Figure 4: NFC applied to cart pole balancing

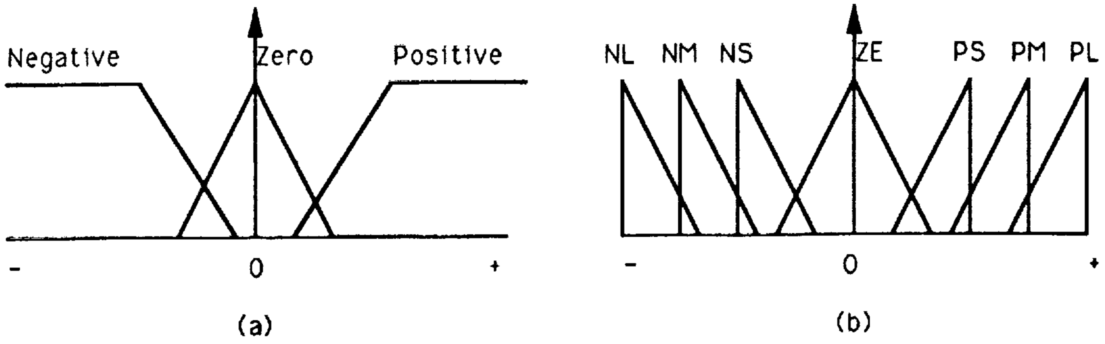


Figure 5: (a)- Three qualitative levels for $\theta, \dot{\theta}, x$, and $\dot{x}$, (b)- Seven qualitative levels for $F$

an input node $j$ and a node $i$ in the hidden layer indicates that the linguistic value of the input corresponding to node $i$ is used as a precondition in rule $i$. As shown in Figure 4, the first nine rules, corresponding to the hidden layer nodes 1 to 9, are rules with two preconditions (i.e., $\theta$, and $\dot{\theta}$). The rules 10 through 13 include four preconditions representing the linguistic values of $\theta$, $\dot{\theta}$, $x$, and $\dot{x}$. In this network, $D$ represents the matrix of connection weights between the input layer and the hidden layer, and $F$ represents a vector of connection weights between the hidden layer and the output node. The amount of force applied to the cart is calculated using the equations (8) to (10) as were given in the last section.

# 4 Relation to other research

**Credit Assignment** The evaluation network in our work is similar to the Samuel's early work on credit assignment [10]. The Adaptive Heuristic Critic (AHC) model of Barto et. al. [3] provides a more general approach to credit assignment which learns by updating the predictions of failures. If no failure signal is present, the internal reinforcement provided by AHC is just the difference between the successive predictions of failure. Recently, Sutton [11] has formalized this method as the Temporal Difference methods.

**Anderson's .Multi-layer networks** We use the same structure as proposed by Anderson [2], however, the action selection network in our model is based on fuzzy logic control. Using the structure of a fuzzy controller, Anderson's approach is extended here to provide for the following attributes in NFC.

- The continuous representation of the output value.
- The inclusion of the human expert operator's control rules in terms of hidden units in the action selection network.

It should be noted that Anderson's goal in [1] was to discover the interesting patterns and strategy learning schemes. Not much effort was spent on making the process learn faster. In our work, although we allow some of the strategy learning to happen automatically, we start from a knowledge base of fuzzy control rules and fine-tune them as learning happens in the neural network.

**Single Layer NeuroFuzzy Control** Lee and Berenji [8] and Lee [7] have used a single layer neural network which requires the identification of the trace functions for keeping track of the visited states and their evaluations. The generation of these trace function is a difficult task in larger control problems. However, the approach suggested in the current paper does not use trace functions. The neural network representation of the fuzzy control rules in NFC allows faster development and faster learning. Also, in the single layer model, only the generation of the output values were considered. The preconditions of the fuzzy control rules were left untouched. However, in NFC, based on reinforcements received from the environment, both the preconditions and the conclusions of rules can be modified (i.e., fine-tuned).

## 5 Conclusion

A new model based on the reinforcement learning technique and fuzzy logic control was proposed which is applicable to control problems for which the analytical models of the process are unknown. The NFC model presented here improves the previous models in neurofuzzy control by learning to fine-tune the performance of a fuzzy logic controller.

ACKNOWLEDGEMENT My thanks to Yung-Yaw Chen for many useful discussions. Also, thanks to Charles Anderson for providing me with his software for cart-pole balancing with multi-layer neural networks.

## References

[1] C. W. Anderson. *Learning and Problem Solving with Multilayer Connectionist Systems.* PhD thesis, University of Massachusetts, 1986.

[2] C. W. Anderson. *Strategy Learning with Multilayer Connectionist Representation.* Technical Report TR87-509.3, GTE Laboratories Inc., May 1988.

[3] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846, 1983.

[4] R.E. Bellman and L.A. Zadeh. Fuzzy logic controllers. In L.A. Zadeh Yager, R. R., editor, *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Publishers, (to appear).

[5] H.R. Berenji, Y.Y. Chen, C.C. Lee, J.S. Jang, and S. Murugesan. A hierarchical approach to designing approximate reasoning-based controllers for dynamic physical systems. In *Sixth Conference on Uncertainty in Artificial Intelligence*, pages 362–369, 1990.

[6] Y. Kasai and Y. Morimoto. Electronically controlled continuously variable transmission. In *Int. Congress on Transportation Electronics*, Dearborn, Michigan, 1988.

[7] C.C. Lee. Self-learning rule-based controller employing approximate-reasoning and neural-net concepts. *Int. Journal of Intelligent Systems*, 1990.

[8] C.C. Lee and H.R. Berenji. An intelligent controller based on approximate reasoning and reinforcement learning. In *Proc. of IEEE Int. Symposium on Intelligent Control*, Albany, NY, 1989.

[9] T. J. Procyk and E. H. Mamdani. A linguistic self-organizing process controller. *Automatica*, 15(1):15–30, 1979.

[10] A. L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers.* Journal of R & D, IBM, 1959.

[11] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[12] S. Yasunobu and S. Miyamoto. *Automatic Train operation by predictive fuzzy control*, pages 1–18. North-Holland, Amsterdam, 1985.

# An Architecture for Designing Fuzzy Controllers using Neural Networks

Hamid R. Berenji

Artificial Intelligence Research Branch
NASA Ames Research Center

# Outline

- Rule-based control by fuzzy logic
- Hierarchical fuzzy control
- A hybrid fuzzy logic-neural network controller model
    - Learning in neural networks
        - Supervised learning
            - error back-propagation
        - Unsupervised learning
            - Temporal Difference method
        - Evaluation network
        - Action network
- Conclusion

# Approximate Reasoning and Control

## Motivations

Human expert controllers perform well using approximate reasoning

Development of a mathematical physical model might not always be possible or might not be computationally feasible

Learning to control a physical system is regarded as one kind of <u>intelligence</u>

# Fuzzy Sets and Rule-Based Control

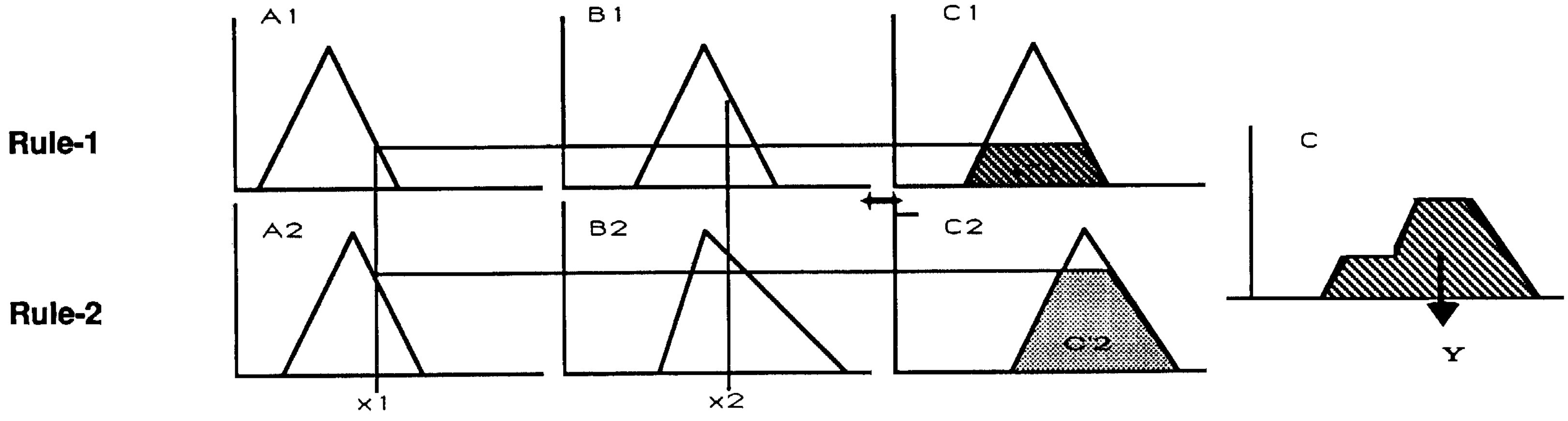
Labeling: Translation of a sensor reading to a label as done by a human expert controller

Example: IF Angular-position is ***Positive*** and Angular velocity is ***Positive*** THEN Force is ***Positive-Large***

Conflict resolution: is required when more than one rule are triggered by one or more sensor readings.

Example: RULE-1 IF x1 is A1 and x2 is B1 THEN Y is C1

RULE-2 IF x1 is A2 and x2 is B2 THEN Y is C2



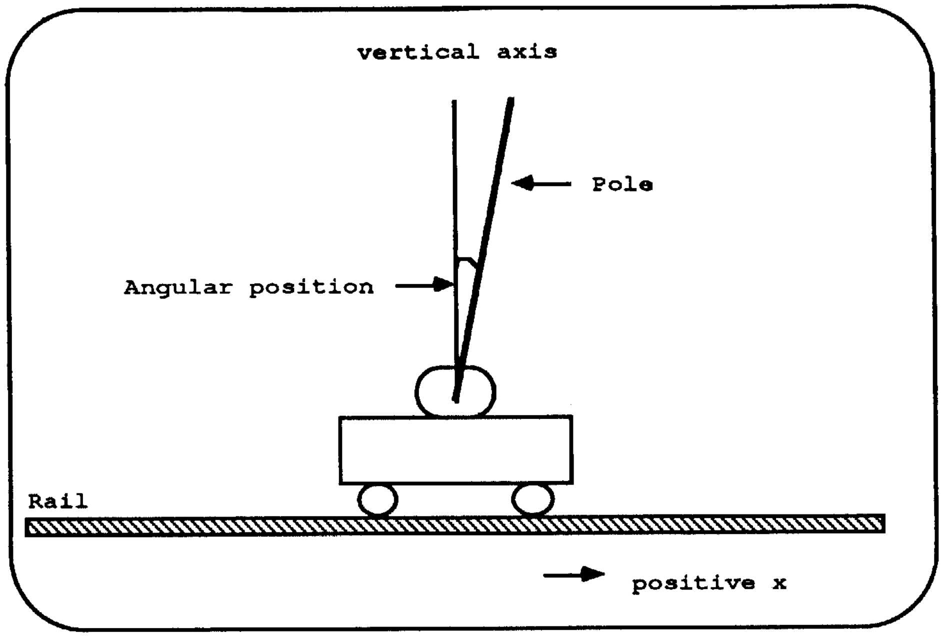
# Hierarchical Fuzzy Control

1. Identify the set of goals that the system should achieve and maintain.
2. Assign priorities among goals.
3. Identify the set of input control parameters.
4. Identify the set of linguistic values to describe the values of the input control parameters.
5. Identify the set of linguistic values to describe the values of the output.
6. Acquire the set of rules directly related to the highest priority goal.
7. Acquire the next set of control rules for a lower priority goal combining aspects of approximately achieving the higher level goal e.g.,

   IF goal (i-1) is approximately achieved and U(i) is A THEN Z(i) is C

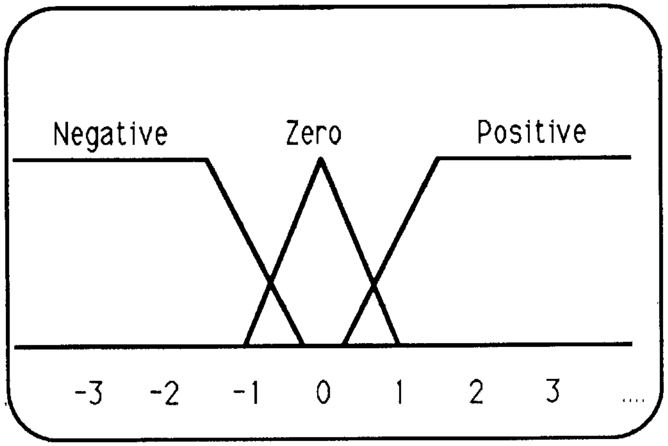
where Z(i) is the set of ouputs at level (i).

# The cart-pole balancing problem

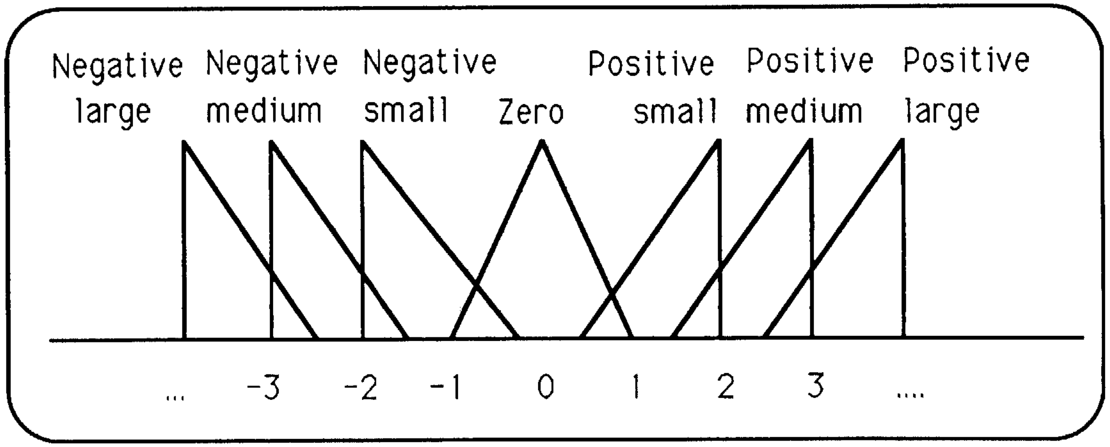


4 input and 1 output variables:

- Pole Angle
- Angular Velocity
- Cart Position
- Cart Velocity
- Force applied to cart



(a)



(b)

# Hiearchical Fuzzy Control (Example:cart-pole balancing)

1. Goals: {position the cart at location x on the track, keep the pole balanced}.
2. Goal priorities:

   Goal 1: Keep the pole balanced

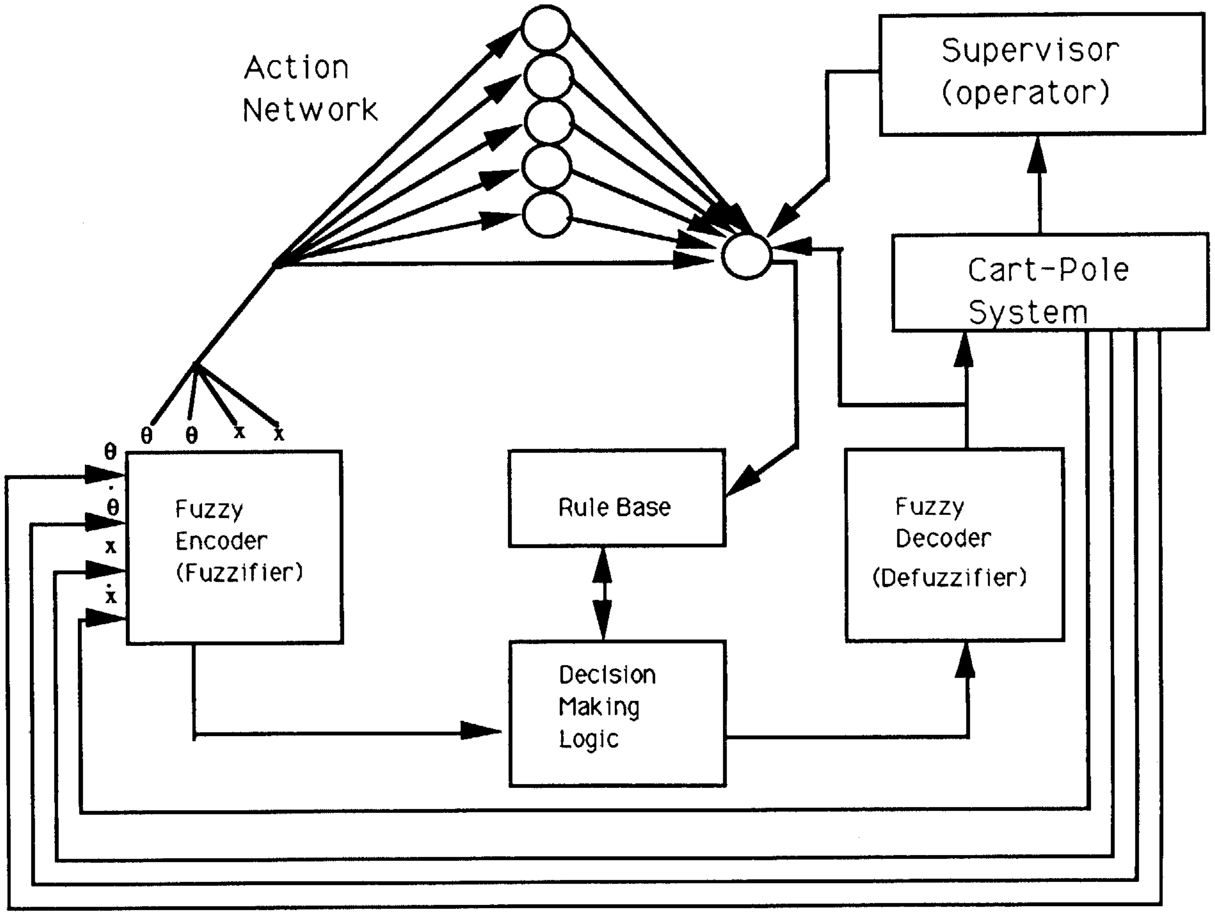
   Goal 2: position the cart at the location x
3. Control parameters: Pole angle, pole velocity, cart position, cart horizontal velocity}.
4. Linguistic values (input): {positive, zero,negative}.
5. Linguistic values (output):

   {positive-large, positive-medium,... negative-large}.
6. Rules for goal one, e.g.,

   If pole angle is positive and pole velocity is Zero then F is Positive-small.
7. Rules for lower level goals, e.g.,

   IF (pole angle is Very small and pole velocity is very small) and
   horizontal position is Positive and horizontal speed is Positive,
   THEN F is Positive-medium.

# Hybrid Model
# (Supervised Learning)



# Credit Assignment Problem

- A challenging topic of research in AI
- Given the performance (results) of a process, distribute awards or punishments to the individual elements
- In rule-based systems, this means assigning credit or blame to individual rules engaged in problem solving
- Examples: Samuel's checker program, and Michie and Chambers' BOXES system (which learned to balance a pole)
- Performance trace is crucially important in credit assignment

# Temporal Difference (TD) Methods

A class of incremental learning procedures specialized for prediction problems. Learning occurs whenever there is a change in prediction over time (e.g., a weatherman's prediction on each day of the week about rain next Saturday).

Related Research:

- Samuel's checker-playing program
- Backpropagation in connectionism
- Holland's bucket brigade

Advantages:

- TD methods appear to learn faster than supervised-learning methods
- Require less memory than supervised learning
- Have many similarities to animal learning internal models of the world (i.e., Pavlovian or classical conditioning)

Disadvantages:

- TD methods can fail!!

# TD Algorithms

Experience: Observation-outcome sequence $x_1, x_2, x_3, \ldots.\ x_m, z$

Prediction : $P_1, P_2, P_3, \ldots..\ P_m$ (estimates of z)

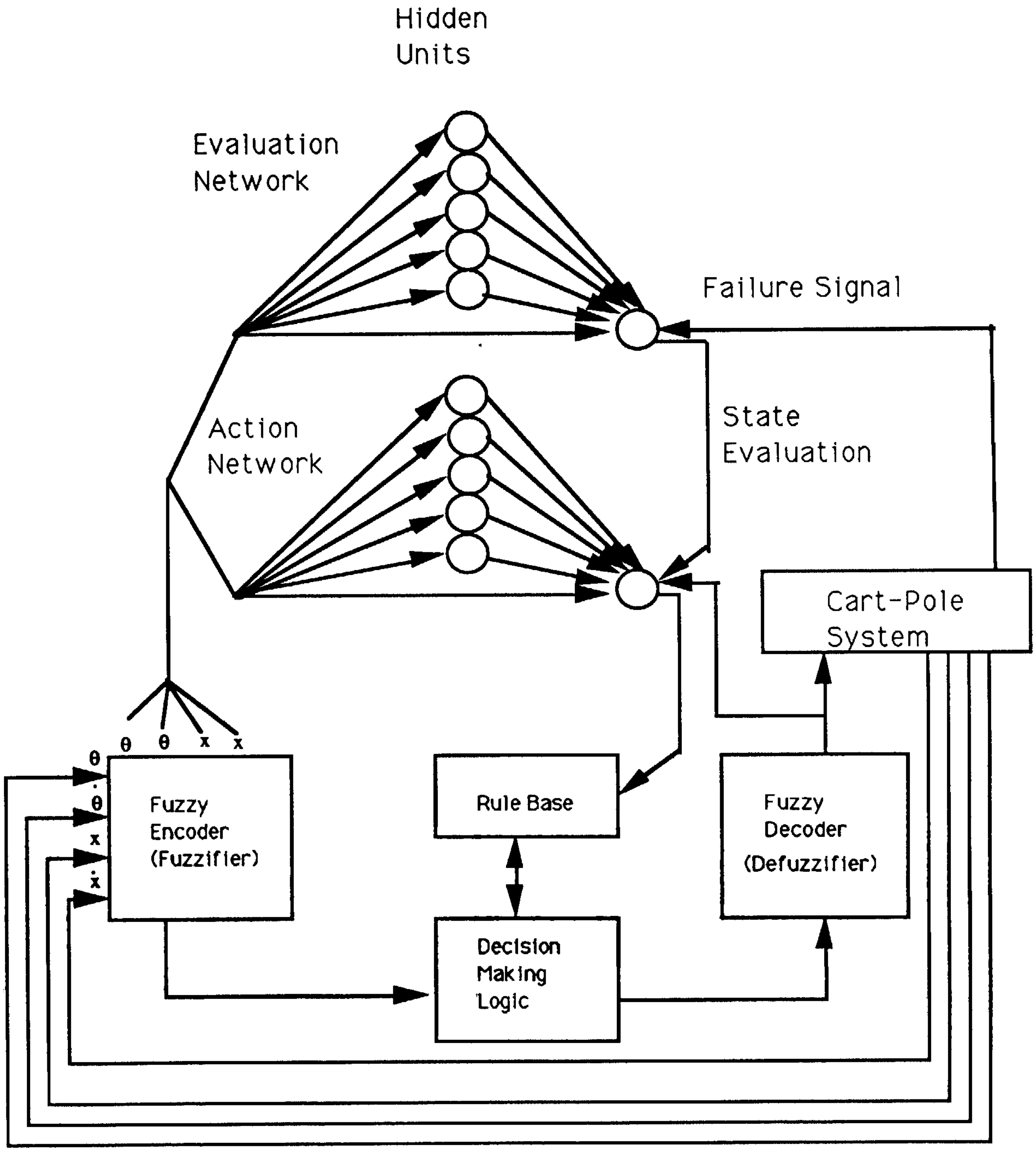
Assumption: $P_t$ is a function of $x_t$ (restrictive, but could be removed)

Weight w: Vector of modifiable parameters

After a complete sequence:

$$w \leftarrow w + \sum_{t=1}^{m} \Delta w_t$$

# Hybrid Model (Unsupervised Learning)



# Evaluation and Action networks

**Evaluation network:**

Apportions the blame for the failure among the actions in the sequence leading to a failure.

**Action network:**

Learns to select actions as a function of states.

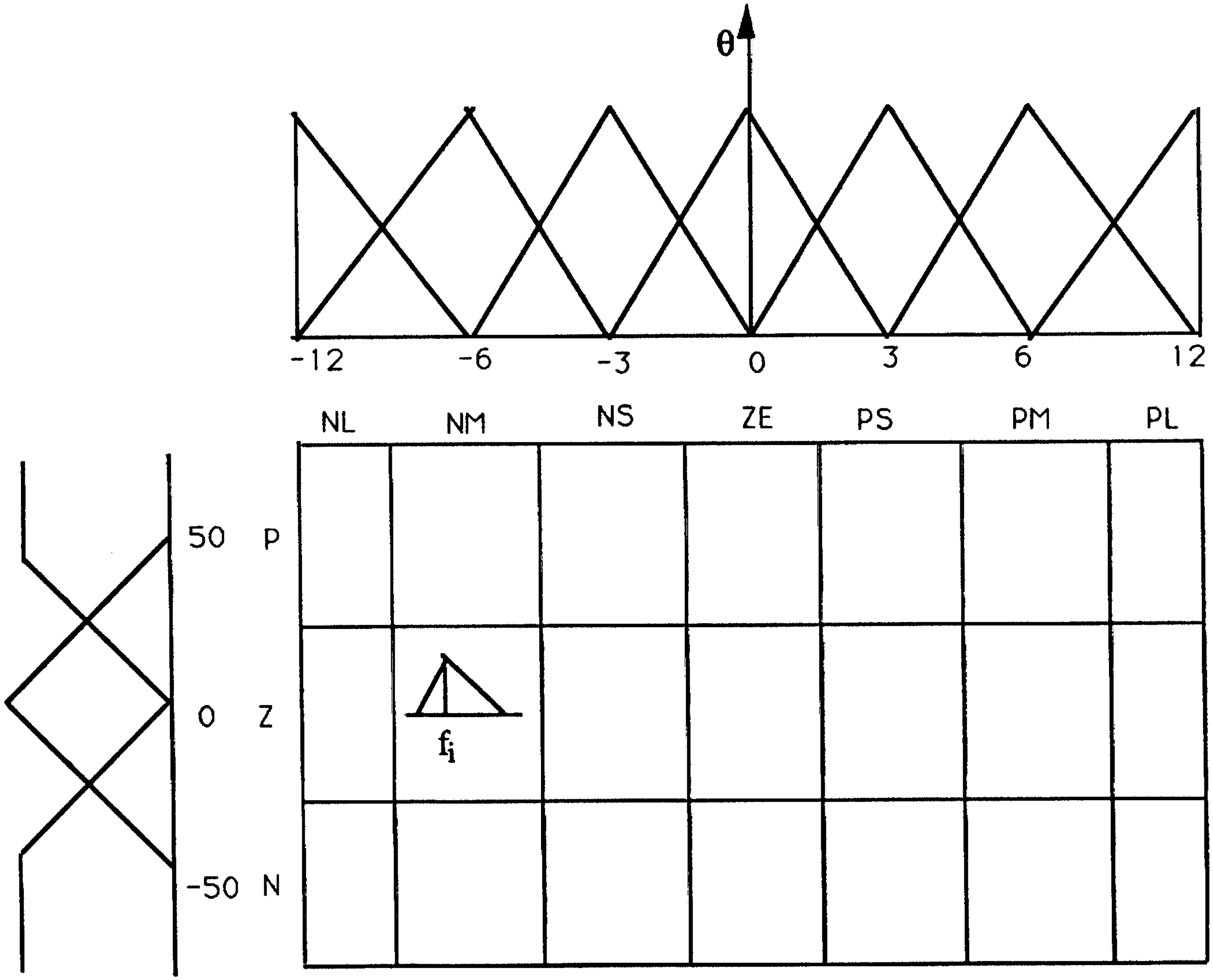
# Comparison:
# One-layer vs. Two-layer Networks

- With one-layer networks in the hybrid model, only the membership functions of the output control actions could be learned.

- With two-layer networks presented here, the membership functions of the input control parameters (used in the antecedent of rules) could also be learned.

- After forming helpful features, two-layer systems perform better.

# Linguistic state description



# Conclusion

- Extension to two-layer network allows the learning of the membership functions used in the antecedent of the rules.
- Approximate Reasoning based controllers could become able to learn from experience by using unsupervised learning algorithms such as the Temporal Difference Methods and supervised learning algorithms such as the error backpropagation.