3. Fuzzy neural networks

3.1 Integration of fuzzy logic and neural networks

*Hybrid systems* combining fuzzy logic, neural networks, genetic algorithms, and expert systems are proving their effectiveness in a wide variety of real-world problems.

Every intelligent technique has particular computational properties (e.g., ability to learn, explanation of decisions) that make them suited for particular problems and not for others. For example, while neural networks are good at recognizing patterns, they are not good at explaining how they reach their decisions. Fuzzy logic systems, which can reason with imprecise information, are good at explaining their decisions but they cannot automatically acquire the rules they use to make those decisions. These limitations have been a central driving force behind the creation of intelligent hybrid systems where two or more techniques are combined in a manner that overcomes the limitations of individual techniques. Hybrid systems are also important when considering the varied nature of application domains. Many complex domains have many different component problems, each of which may require different types of processing. If there is a complex application which has two distinct sub-problems, say a signal processing task and a serial reasoning task, then a neural network and an expert system respectively can be used for solving these separate tasks. The use of intelligent hybrid systems is growing rapidly with successful applications in many areas including process control, engineering design, financial trading, credit evaluation, medical diagnosis, and cognitive simulation.

While fuzzy logic provides an inference mechanism under cognitive uncertainty, computational neural networks offer exciting advantages, such as learning, adaptation, fault-tolerance, parallelism and generalization. To enable a system to deal with cognitive uncertainties in a manner more like humans, one may incorporate the concept of fuzzy logic into the neural networks.

The computational process envisioned for fuzzy neural systems is as follows. It starts with the development of a "fuzzy neuron" based on the understanding of biological neuronal morphologies, followed by learning mecha-
nisms. This leads to the following three steps in a fuzzy neural computational process

- development of fuzzy neural models motivated by biological neurons,
- models of synaptic connections which incorporates *fuzziness* into neural network,
- development of learning algorithms (that is the method of adjusting the synaptic weights)

Two possible models of fuzzy neural systems are

- In response to linguistic statements, the fuzzy interface block provides an input vector to a multi-layer neural network. The neural network can be adapted (trained) to yield desired command outputs or decisions.

![Fig. 3.1. The first model of fuzzy neural system.](image)

- A multi-layered neural network drives the fuzzy inference mechanism.

Neural networks are used to *tune* membership functions of fuzzy systems that are employed as decision-making systems for controlling equipment. Although fuzzy logic can encode expert knowledge directly using rules with linguistic labels, it usually takes a lot of time to design and tune the membership functions which quantitatively define these linguistic labels. Neural network learning techniques can automate this process and substantially reduce development time and cost while improving performance.

In theory, neural networks, and fuzzy systems are equivalent in that they are convertible, yet in practice each has its own advantages and disadvantages. For neural networks, the knowledge is automatically acquired by the backpropagation algorithm, but the learning process is relatively slow and analysis of the trained network is difficult (black box). Neither is it possible to extract structural knowledge (rules) 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.

Fuzzy systems are more favorable in that their behavior can be explained based on fuzzy rules and thus their performance can be adjusted by tuning the rules. But since, in general, knowledge acquisition is difficult and also the universe of discourse of each input variable needs to be divided into several intervals, applications of fuzzy systems are restricted to the fields where expert knowledge is available and the number of input variables is small.

To overcome the problem of knowledge acquisition, neural networks are extended to automatically extract fuzzy rules from numerical data. Cooperative approaches use neural networks to optimize certain parameters of an ordinary fuzzy system, or to preprocess data and extract fuzzy (control) rules from data.

Based upon the computational process involved in a fuzzy-neuro system, one may broadly classify the fuzzy neural structure as feedforward (static) and feedback (dynamic).

A typical fuzzy-neuro system is Berenji's ARIC (Approximate Reasoning Based Intelligent Control) architecture [11]. It is a neural network model of a fuzzy controller and learns by updating its prediction of the physical system's behavior and fine tunes a predefined control knowledge base.

This kind of architecture allows to combine the advantages of neural networks and fuzzy controllers. The system is able to learn, and the knowledge used within the system has the form of fuzzy IF-THEN rules. By predefining these rules the system has not to learn from scratch, so it learns faster than a standard neural control system.

ARIC consists of two coupled feed-forward neural networks, the Action-state Evaluation Network (AEN) and the Action Selection Network (ASN). The ASN is a multilayer neural network representation of a fuzzy controller.
In fact, it consists of two separated nets, where the first one is the fuzzy inference part and the second one is a neural network that calculates $p[t, t+1]$, a measure of confidence associated with the fuzzy inference value $u(t+1)$, using the weights of time $t$ and the system state of time $t+1$. A stochastic modifier combines the recommended control value $u(t)$ of the fuzzy inference part and the so called "probability" value $p$ and determines the final output value

$$u'(t) = o(u(t), p[t, t + 1])$$

of the ASN. The hidden units $z_i$ of the fuzzy inference network represent the fuzzy rules, the input units $x_j$ the rule antecedents, and the output unit $u$ represents the control action, that is the defuzzified combination of the conclusions of all rules (output of hidden units). In the input layer the system state variables are fuzzified. Only monotonic membership functions are used in ARIC, and the fuzzy labels used in the control rules are adjusted locally.
within each rule. The membership values of the antecedents of a rule are then multiplied by weights attached to the connection of the input unit to the hidden unit. The minimum of those values is its final input. In each hidden unit a special monotonic membership function representing the conclusion of the rule is stored. Because of the monotonicity of this function the crisp output value belonging to the minimum membership value can be easily calculated by the inverse function. This value is multiplied with the weight of the connection from the hidden unit to the output unit. The output value is then calculated as a weighted average of all rule conclusions.

The AEN tries to predict the system behavior. It is a feed-forward neural network with one hidden layer, that receives the system state as its input and an error signal $r$ from the physical system as additional information. The output $v[t, t']$ of the network is viewed as a prediction of future reinforcement, that depends of the weights of time $t$ and the system state of time $t'$, where $t'$ may be $t$ or $t + 1$. Better states are characterized by higher reinforcements. The weight changes are determined by a reinforcement procedure that uses the output of the ASN and the AEN. The ARIC architecture was applied to cart-pole balancing and it was shown that the system is able to solve this task [11].

3.2 Fuzzy neurons

Consider a simple neural net in Fig. 3.4. All signals and weights are real numbers. The two input neurons do not change the input signals so their output is the same as their input. The signal $x_i$ interacts with the weight $w_i$ to produce the product

$$p_i = w_i x_i, \; i = 1, 2.$$  

The input information $p_i$ is aggregated, by addition, to produce the input

$$\text{net} = p_1 + p_2 = w_1 x_1 + w_2 x_2,$$

to the neuron. The neuron uses its transfer function $f$, which could be a sigmoidal function,

$$f(x) = \frac{1}{1 + e^{-x}},$$

to compute the output

$$y = f(\text{net}) = f(w_1 x_1 + w_2 x_2).$$

This simple neural net, which employs multiplication, addition, and sigmoidal $f$, will be called as regular (or standard) neural net.

If we employ other operations like a t-norm, or a t-conorm, to combine the incoming data to a neuron we obtain what we call a hybrid neural net.
These modifications lead to a fuzzy neural architecture based on fuzzy arithmetic operations. Let us express the inputs (which are usually membership degrees of a fuzzy concept) $x_1, x_2$ and the weights $w_1, w_2$ over the unit interval $[0,1]$.

A hybrid neural net may not use multiplication, addition, or a sigmoidal function (because the results of these operations are not necessarily in the unit interval).

**Definition 3.2.1** A hybrid neural net is a neural net with crisp signals and weights and crisp transfer function. However,

- we can combine $x_i$ and $w_i$ using a t-norm, t-conorm, or some other continuous operation,
- we can aggregate $p_1$ and $p_2$ with a t-norm, t-conorm, or any other continuous function
- $f$ can be any continuous function from input to output

We emphasize here that all inputs, outputs and the weights of a hybrid neural net are real numbers taken from the unit interval $[0,1]$. A processing element of a hybrid neural net is called fuzzy neuron. In the following we present some fuzzy neurons.

**Definition 3.2.2** *(AND fuzzy neuron [81, 82])*

The signal $x_i$ and $w_i$ are combined by a triangular conorm $S$ to produce

$$p_i = S(w_i, x_i), \ i = 1, 2.$$  

The input information $p_i$ is aggregated by a triangular norm $T$ to produce the output

$$y = AND(p_1, p_2) = T(p_1, p_2) = T(S(w_1, x_1), S(w_2, x_2))$$

of the neuron.

So, if $T = \min$ and $S = \max$ then the AND neuron realizes the min-max composition

$$y = \min \{w_1 \lor x_1, w_2 \lor x_2\}.$$
Definition 3.2.3 (OR fuzzy neuron [81, 82])

The signal $x_i$ and $w_i$ are combined by a triangular norm $T$ to produce

$$p_i = T(w_i, x_i), \ i = 1, 2.$$  

The input information $p_i$ is aggregated by a triangular conorm $S$ to produce the output

$$y = OR(p_1, p_2) = S(p_1, p_2) = S(T(w_1, x_1), T(w_2, x_2))$$

of the neuron.

So, if $T = \min$ and $S = \max$ then the AND neuron realizes the max-min composition

$$y = \max\{w_1 \land x_1, w_2 \land x_2\}.$$  

The AND and OR fuzzy neurons realize pure logic operations on the membership values. The role of the connections is to differentiate between particular levels of impact that the individual inputs might have on the result of aggregation. We note that (i) the higher the value $w_i$ the stronger the impact of $x_i$ on the output $y$ of an OR neuron, (ii) the lower the value $w_i$ the stronger the impact of $x_i$ on the output $y$ of an AND neuron.
The range of the output value $y$ for the AND neuron is computed by letting all $x_i$ equal to zero or one. In virtue of the monotonicity property of triangular norms, we obtain

$$y \in [T(w_1, w_2), 1]$$

and for the OR neuron one derives the boundaries

$$y \in [0, S(w_1, w_2)].$$

**Definition 3.2.4 (Implication-OR fuzzy neuron [44, 46])**

The signal $x_i$ and $w_i$ are combined by a fuzzy implication operator $I$ to produce

$$p_i = I(w_i, x_i) = w_i \leftarrow x_i, \; i = 1, 2.$$  

The input information $p_i$ is aggregated by a triangular conorm $S$ to produce the output

$$y = I(p_1, p_2) = S(p_1, p_2) = S(w_1 \leftarrow x_1, w_2 \leftarrow x_2)$$

of the neuron.

![Implication-OR fuzzy neuron](image)

**Definition 3.2.5 (Kwan and Cai's fuzzy neuron [126])**

The signal $x_i$ interacts with the weight $w_i$ to produce the product

$$p_i = w_i x_i, \; i = 1, \ldots, n$$

The input information $p_i$ is aggregated by an aggregation function $h$ to produce the input of the neuron

$$z = h(w_1 x_1, w_2 x_2, \ldots, w_n x_n)$$

the state of the neuron is computed by

$$s = f(z - \theta)$$
where $f$ is an activation function and $\theta$ is the activating threshold. And the $m$ outputs of the neuron are computed by

$$y_j = g_j(s), \ j = 1, \ldots, m$$

where $g_j, \ j = 1, \ldots, m$ are the $m$ output functions of the neuron which represent the membership functions of the input pattern $x_1, x_2, \ldots, x_n$ in all the $m$ fuzzy sets.

**Definition 3.2.6 (Kwan and Cai’s max fuzzy neuron [126])**

The signal $x_i$ interacts with the weight $w_i$ to produce the product

$$p_i = w_i x_i, \ i = 1, 2.$$  

The input information $p_i$ is aggregated by the maximum conorm

$$z = \max\{p_1, p_2\} = \max\{w_1 x_1, w_2 x_2\}$$

and the $j$-th output of the neuron is computed by

$$y_j = g_j(f(z - \theta)) = g_j(f(\max\{w_1 x_1, w_2 x_2\} - \theta))$$

where $f$ is an activation function.
Definition 3.2.7 (Kwan and Cai's min fuzzy neurons [126])

The signal $x_i$ interacts with the weight $w_i$ to produce the product

$$p_i = w_i x_i, \ i = 1, 2.$$  

The input information $p_i$ is aggregated by the minimum norm

$$y = \min\{p_1, p_2\} = \min\{w_1 x_1, w_2 x_2\}$$  

and the $j$-th output of the neuron is computed by

$$y_j = g_j(f(z - \theta)) = g_j(f(\min\{w_1 x_1, w_2 x_2\} - \theta))$$  

where $f$ is an activation function.

It is well-known that regular nets are universal approximators, i.e. they can approximate any continuous function on a compact set to arbitrary accuracy. In a discrete fuzzy expert system one inputs a discrete approximation to the fuzzy sets and obtains a discrete approximation to the output fuzzy set. Usually discrete fuzzy expert systems and fuzzy controllers are continuous mappings. Thus we can conclude that given a continuous fuzzy expert system, or continuous fuzzy controller; there is a regular net that can uniformly approximate it to any degree of accuracy on compact sets. The problem with this result that it is non-constructive and only approximative. The main problem is that the theorems are existence types and do not tell you how to build the net.

Hybrid neural nets can be used to implement fuzzy IF-THEN rules in a constructive way. Following Buckley & Hayashi [33], and, Keller, Yager & Tahani [112] we will show how to construct hybrid neural nets which are computationally equivalent to fuzzy expert systems and fuzzy controllers. It should be noted that these hybrid nets are for computation and they do not have to learn anything.

Though hybrid neural nets can not use directly the standard error backpropagation algorithm for learning, they can be trained by steepest descent


3. Fuzzy neural networks


78. Y. Hayashi, J.J. Buckley and E. Czogala, Direct fuzzification of neural network and fuzzified delta rule, *Proc. 2nd Internat. Conf. on Fuzzy logic and Neural Networks*, Iizuka, Japan, 1992 73-76.


Index

$C_\infty$ distance, 39
$R$-implication, 48
$S$-implication, 48
$\alpha$-cut, 4, 212
$\varepsilon$-completeness, 87, 194
Łukasiewicz, 1
Łukasiewicz $t$-conorm, 14
Łukasiewicz $t$-norm, 13

activation function, 135, 175
activation hyperbox, 226
additive generator, 14, 279
AND fuzzy neuron, 176
andlike OWA operator, 102
ANFIS architecture, 193
Archimedean $t$-norm, 14
ARIC architecture, 173
arithmetic mean, 99
artificial neuron, 134
averaging operator, 98

bag, 96
ball and beam system, 91
basic property of GMP, 58
binary fuzzy relation, 19
bipolar activation function, 150

Cartesian product, 22
center-of-gravity defuzzification, 78
centroid defuzzification, 93
clause checking layer, 196
complement of a fuzzy set, 12
composition of fuzzy relations, 22
compositional rule of inference, 56
conjunction rule, 56
corvex fuzzy set, 5
crisp control action, 81
crisp relation, 18
cumulative error, 140, 152, 216

data base strategy, 87
defuzzification, 78
degree of membership, 2
degree of necessity, 42
degree of possibility, 42
delta learning rule, 149
descent method, 145
discrete Hamming distance, 40
disjunction rule, 87
downhill direction, 145

discretization, 87
empty fuzzy set, 9
entailment rule, 55
equality of fuzzy sets, 9
error correction learning, 138
error function, 193, 214, 218, 221, 235
error signal, 148
Euclidean norm, 137
expected value, 44
extended addition, 30
extended division, 31
extended multiplication, 31
extended subtraction, 30
extension principle, 26

feature space, 229
firing strength, 81, 190, 219, 271
FNN of Type 1, 181
FNN of Type 2, 208
FNN of Type 5, 181
FNN of Type 6, 181
FNN of Type 3, 212
Frank t-norm, 13
Funahashi's theorem, 158
fuzzification operator, 74
fuzzy classification, 229
fuzzy control rule, 72
fuzzy implication, 47
fuzzy logic controller, 72
fuzzy mapping, 32
fuzzy max, 35
fuzzy min, 36
fuzzy neuron, 176
fuzzy number, 5
fuzzy number of type LR, 7
fuzzy partition, 49, 88, 229, 262
fuzzy point, 10
fuzzy quantity, 3
fuzzy relation, 19, 264
fuzzy rule extraction, 226
fuzzy screening system, 109
fuzzy set, 2
fuzzy subseshet, 9
fuzzy training set, 209

Gaines implication, 48
Gaussian membership function, 93
generalized p-mean, 99, 234
generalized delta rule, 156
Generalized Modus Ponens, 57
Generalized Modus Tollens, 57
generalized Nguyen's theorem, 37
genetic algorithms, 217
geometric mean, 99
gradient vector, 146
Gödel implication, 48

Hamacher t-conorm, 15
Hamacher t-norm, 13
Hamacher's t-norm, 263
Hamming distance, 40
harmonic mean, 99
Hausdorff distance, 39
height defuzzification, 79
hidden layer, 155
hybrid fuzzy neural network, 183
hybrid neural net, 176, 272

identity quantifier, 107
implication-OR fuzzy neuron, 178
individual rule output, 81, 95, 191
inference mechanism, 81
inhibition hyperbox, 226
intersection of fuzzy sets, 11

Kleene-Dienes implication, 48
Kleene-Dienes-Lukasiewicz, 48
Kohonen's learning algorithm, 162
Kwan and Cai's fuzzy neuron, 178

Larsen implication, 48
Larsen's inference mechanism, 85
learning of membership functions, 217
learning rate, 139, 221
linear activation function, 147
linear combination of fuzzy numbers, 33

linear threshold unit, 135
linguistic modifiers, 50
linguistic quantifiers, 104
linguistic variable, 49

Mamdani implication, 48
Mamdani's inference mechanism, 81
Mamdani-type FLC, 73
maximum t-conorm, 14
measure of andness, 102
measure of dispersion, 103
measure of orness, 102
MICA operator, 97
middle-of-maxima method, 79
minimum t-norm, 13
Modus Ponens, 56
Modus Tollens, 57

negation rule, 56
Nguyen's theorem, 34
normal fuzzy set, 4

OR fuzzy neuron, 177
orlike OWA operator, 102
overall system output, 81, 95
OWA operator, 100

parity function, 143
partial derivative, 146
perceptron learning, 140
probabilistic t-conorm, 14
product t-norm, 13
projection of a fuzzy relation, 21
projection rule, 56

quasi-arithmetic mean, 99

regular fuzzy neural net, 181
regular neural net, 175
removal of the threshold, 136
rule tuning, 259

scalar product, 136
simplified fuzzy rules, 218
single-layer feedforward net, 138
singleton fuzzifier, 93
slope, 221
steepest descent method, 218, 220
Stone-Weierstrass theorem, 158
subset property of GMP, 59
Sugeno's inference mechanism, 84
sup-T composition, 23
sup-T compositional rule, 57
superset property of GMP, 59
supervised learning, 155
support, 4

t-conorm-based union, 16
t-norm implication, 48
t-norm-based intersection, 16
threshold-level, 135
total indeterminance, 58
trade-offs, 98
training set, 138, 220
trapezoidal fuzzy number, 7
triangular conorm, 14
triangular fuzzy number, 6
triangular norm, 13
Truth, 50
Tsukamoto's inference mechanism, 82

Uehara and Fujise's method, 205
Umano and Ezawa's method, 202

union of fuzzy sets, 11
unipolar activation function, 150
unit score, 111
universal approximators, 93
universal fuzzy set, 9
unsupervised learning, 160

variance, 44

week t-norm, 13
weight vector, 135
window type OWA operator, 101

XOR problem, 143

Yager t-conorm, 15
Yager t-norm, 13
Yager's t-norm, 234