Learning in the context of neural networks is defined as a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. A prescribed set of well-defined rules for the solution of a learning problem is called a learning algorithm. Different learning algorithms differ from each other in the way in which the adjustment to a synaptic weight of a neuron is formulated, and the manner in which the network relates to its environment. The two important learning paradigms are: 1) supervised learning and 2) unsupervised learning.

A.1 Supervised Learning

It is also referred to as learning with a teacher, who has the knowledge of the environment. The knowledge is represented by a set of input output examples. The environment is, however unknown to the neural network of interest. When the teacher and the neural network are both exposed to a training vector drawn from the environment, the teacher by virtue of built-in knowledge, is able to provide the neural network with a desired response for that training vector. The desired response represents the optimum action to be performed by the neural network. The network parameters are then adjusted under the combined influence of the training vector and the error signal. The error signal is defined as the difference between the desired response and the actual
response of the network. This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the neural network emulate the teacher. Thus the knowledge of the environment available to the teacher is transferred to the neural network through training as fully as possible. When this condition is reached, the teacher may be dispensed with and the neural network can deal with the environment completely by itself.

When the supervised learning system is used to perform a task such as pattern classification, the net is presented with an input pattern together with the target output for that pattern. The target output usually constitutes the correct answer or correct classification for the input pattern. In response to these paired examples, the net adjusts the values of its internal weights. If training is successful the internal parameters are then adjusted to the point where the network can produce the correct answers in response to each input pattern. Thus the class membership information used by the supervised learning algorithm enables it to detect pattern mis-classifications and compute an error vector that reinforces the learning process.

A.2 Unsupervised Learning

In the unsupervised or self-organized learning there is no external teacher or critic to oversee the learning process. The learning must rely on guidance obtained heuristically by the system upon examining the different sample data or the environment. Here, provision is made for a task-independent measure of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure. Once the network has become tuned to the statistical regularities of the input data, it develops the ability to form the internal representations for encoding features of the input and thereby to create new classes automatically. Thus the unsupervised learning approach does not use class-membership information, but uses unlabelled pattern samples. It blindly processes each sample X but does not know that X belongs to class Di and not to class Dj. They have less computational complexity and less accuracy as compared to supervised learning algorithms. They learn rapidly often on a
single pass of noisy data. Thus they are suitable in many high-speed real-time environments where we may not have enough time, information or computational precision to use supervised techniques. It is very much similar to the way the biological synapses of the human brain learn.
APPENDIX B
Fuzzy Algorithm

The fuzzy algorithm and the manner in which it works are explained in this appendix. The overall algorithm consists of five steps. The working mechanism of each block can be programmed in different ways. Here we present one possible algorithmic flow that can be used as a reference for development. The fuzzy algorithm is a generalized work flow through which fuzzy IF-THEN computations take place in a sequence, as in many pattern recognition, control, and industrial applications. The algorithm includes elements subject to design in fuzzy logic as well as input-output processing.

B.1 The Fuzzy Algorithm

The algorithmic steps are as follows:

1. Input Data Processing
2. Evaluating Antecedent Fuzzy Variables
3. LHS Computations
4. RHS Computations
5. Output Data Processing

B.1.1. Input Data Processing

During input data processing the following events occur:

1. Receiving an input data set is one of the two physical boundaries between the external world and the fuzzy system. Everything else after this point assumes a digital computing domain even though fuzzy systems, in general, are not restricted to this domain only. A very important point is that the data set received from the external world represents one initiation step.
2. Data transformation, if necessary, is accomplished in this step. Depending on the design, two or higher dimensional numerical data is converted into one dimensional feature vector.

3. A loop starts here by selecting one input data element per cycle. This loop will continue until all the elements of the input data set are processed.

4. A trivial checking operation takes place at this level to identify the type of data.

5. For numerical data, the universe of discourse of the corresponding antecedent variable is used to check if data is between the upper and lower boundaries of the universe. The boundaries of the universe of discourse for each variable are in the memory of the fuzzy system. If data is nonsingular, that is, if there is more than one data point, the same operation is employed. For symbolic data, a recognition step is taken in which the symbol, which is often a linguistic statement, is translated into a numerical form. This is only possible if the translation of the symbol was embedded in the memory of the fuzzy system during the design. In other words, if High is the linguistic input data of the antecedent variable temperature, it must be known by the fuzzy system what High means in terms of temperature values. This is analogous to speaking in daily language to someone who must understand the words. Thus, an inference engine, in general, has a language memory. If the symbol is recognized, its numerical translation will automatically be within the universe of discourse by design.

6. In the case of detecting inappropriate data, a failure record is kept at this step. Failure record keeping, which represents a general input processing strategy, allows the option of continuing fuzzy computations in case of a partially unrecognized input data set.

7. Loop ending condition checks to see if all the input data are analyzed.

8. The last step before declaring the validity of the input data set is somewhat more involved than the trivial operations explained above. When all the input data elements are found appropriate, the input data set is acceptable. However, when a fraction of the input data set is inappropriate, an acceptability criterion is applied. When the input data set is rejected, the algorithmic flow returns to collecting a new data set.
B.1.2. Evaluating Antecedent Fuzzy Variables

It includes the following events:

1. An input data set, which is validated through the previous step, is obtained from the external world for the initiation time t. An input data element is picked.

2. The properties of the corresponding antecedent fuzzy variables are retrieved from the memory of the fuzzy system. These properties include the number of membership functions, the shape of each membership function, their layout on the universe of discourse, and the threshold properties.

3. A simple test is applied to identify whether the picked input element contains a single data point or a distribution of points (fuzzy set).

4. Operations in this level involve fuzzy computing and they are subject to design. The evaluation of antecedent variables for a single data point or for a distribution of data points is performed in this level. There are two types of design issues: 1) design of membership functions, and 2) method of evaluating them given the input data.

5. To avoid undesired residual truth (very small possibility values), a threshold filtering is applied at this level.

6. A membership value vector obtained in the last step is stored in the memory of the fuzzy system for future use.

7. Loop ending condition.

B.1.3. LHS Computations

To conduct the LHS computations, the design information in the memory must be organized in a certain way. For this purpose four matrices are considered. They are, the membership value matrix, design index matrix, logic operator matrix, and auxiliary parameter matrix.
Antecedent Variable List

The antecedent variable list includes all fuzzy variables that are on the left hand side of each rule with respect to the THEN operator. This list, which may be defined as a vector of character fields in the memory, is only important for labeling purposes. It serves as a road map to which all other index definitions are referenced. Let $X$ be a character (linguistic) variable of the form,

$$X_A = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix}$$

The index $N$ indicates the number of antecedent fuzzy variables in the memory.

Membership Value Matrix

If there are $N$ antecedent fuzzy variables, then there are $N$ vectors, each with different vector size depending on the number of membership functions defined. The membership values are obtained through the input absorption process for a given initiation step. The numerical values will be different at the next initiation step depending on the input data set from the external world. Each row of the membership value matrix denoted by $\mu$ corresponds to a different fuzzy antecedent variable. Each column corresponds to different membership functions. Note that the matrix may have empty locations based on the number of membership functions defined for each variable.

$$\mu = \begin{pmatrix} \mu_{1,1} & \mu_{1,2} & \ldots & \mu_{1,N} \\ \mu_{2,1} & \mu_{2,2} & \ldots & \mu_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{i,1} & \mu_{i,2} & \ldots & \mu_{i,N} \end{pmatrix}$$
Antecedent Predicate Matrix

Another important illustration to facilitate the computational aspects of the fuzzy algorithm is the antecedent predicate matrix. A predicate, which can be a fuzzy qualifier or quantifier is denoted by $P$ and takes linguistic values. Unlike the membership value matrix, the predicate matrix is fixed by design and it also serves as an address to memory where the data of its corresponding membership functions are stored.

$$P = \begin{pmatrix}
   P_{1,1} & P_{1,1} & \ldots & P_{1,j} \\
   P_{2,1} & P_{2,2} & \ldots & P_{2,m} \\
   \vdots & \vdots & \ddots & \vdots \\
   P_{N,1} & P_{N,2} & \ldots & P_{N,n}
\end{pmatrix}$$

The predicate matrix has the same dimensions as the membership value matrix because each predicate corresponds to one membership function at the same matrix location.

LHS Design Index Matrix

The LHS design matrix contains the index references to the membership value matrix.

$$I_{LHS} = \begin{pmatrix}
   I_{1,1} & I_{1,1} & \ldots & I_{1,1} \\
   I_{2,1} & I_{2,2} & \ldots & I_{2,m} \\
   \vdots & \vdots & \ddots & \vdots \\
   I_{Z,1} & I_{Z,2} & \ldots & I_{Z,n}
\end{pmatrix}$$

This matrix corresponds to the rule layout. The elements of $I_{LHS}$ can have a negative sign indicating a negatively constructed fuzzy statement (i.e. using IS NOT). Every row is a new rule. Columns correspond to fuzzy statements. Because the length of a rule can be anything, the column indices $l$, $m$ and $n$ are not the same. All the elements are the statements before the THEN operator.
Logic Operator Matrix

This matrix contains composition operator indices such as AND and OR. Each operator index contains two kinds of information: 1) type of operator and 2) order of computation. The order of computation is related to the grouping of different operators. Grouping of different operators, which is normally indicated by parentheses requires a particular sequence of computations. The sequence is determined by the innermost statements.

LHS Auxiliary Parameter Matrix

When fuzzy statements are subject to modification by linguistic hedges or by adaptive algorithms, the type of modification is stored in the memory of the fuzzy system using the LHS auxiliary parameter matrix.

B.1.4. RHS Computations

All consequent fuzzy variables are kept in the memory of the fuzzy system along with their properties, such as membership function shapes and threshold levels. The order of listing these variables in the memory by means of a vector constitutes a road map which all other index definitions are referenced to.

\[ X_c = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} \]

The index N indicates the number of consequent fuzzy variables in the memory. N does not have to be equal to the number of rules.
Consequent Predicate Matrix And Membership Functions

The consequent predicate matrix is fixed by design in the same way as the consequent variable vector and it resides in the memory of the fuzzy system. This definition is useful both during the RHS computations and during the output processing stage. The consequent predicate vector $P_C$ is expressed as:

$$P_{RHS} = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,x} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,y} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N,1} & P_{N,2} & \cdots & P_{N,w} \end{bmatrix} \otimes \begin{bmatrix} \mu_{1,1} & \mu_{1,1} & \cdots & \mu_{1,x} \\ \mu_{2,1} & \mu_{2,1} & \cdots & \mu_{2,y} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{N,1} & \mu_{N,1} & \cdots & \mu_{N,w} \end{bmatrix}$$

Where $N$ is the number of consequent variables and the indices $x,y,w$ are the number of predicates (membership functions) of each variable. As illustrated above, the consequent predicate matrix corresponds to the consequent membership function residing in the memory.

RHS Design Index Matrix

This matrix reflects the rule composition structure and is based on the two previous definitions. The elements of this matrix are the row numbers of the LHS result vector.

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,x} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,y} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N,1} & A_{N,2} & \cdots & A_{N,w} \end{bmatrix}_{k \leq Z}$$

Here each row corresponds to one consequent fuzzy variable in the variable vector, and each column corresponds to one of the consequent membership functions. Because the length of the LHS vector is limited by the number of rules $Z$, the width of
the matrix cannot be larger than \( Z \). This matrix is the road map for implication computations just like the logic operator matrix is a road map for the LHS computations.

**Aggregation Vector**

This connector determines the type of aggregation for each consequent fuzzy variable. The size of this vector is equal to the number of consequent variables \( N \). All the elements are designed to be the same in most of the applications.

**RHS Auxiliary Parameter Vector**

Because the RHS part of the rule base is assumed to be reduced to one statement per rule, the RHS auxiliary parameters are in a vector form. The dimension of this vector is equal to the number of rules \( Z \). Note that this generalized representation does not impose any restrictions on how to compute the auxiliary parameter effects in a fuzzy inference engine.

**Weight Vector**

In certain applications, some rules may be considered to be more important than others. The basic fuzzy inference algorithm is its most generalized form, allows weight assignments during the design (or during the implementation in adaptive fuzzy systems). Weights are not restricted to probabilities or possibilities.

**RHS Computations**

The RHS of the entire rule base is symbolically represented by seven entities: 1) consequent variable vector; 2) RHS consequent matrix; 3) RHS design index matrix; 4) aggregation vector; 5) RHS auxiliary parameter vector; 6) weight vector; and 7) LHS result vector. The first six entities are fixed by design, and the seventh entity, the LHS result vector, is dynamically updated (at each initiation time) by the input absorption.
process. Besides these seven entities, the properties of consequent membership functions are also stored in the memory.

The important information embedded in these definitions from the algorithmic solution point of view are the indices that indicate the layout of design data, and the distinction between the static data (determined by design) and dynamic data (driven by external inputs). The RHS design index matrix is the road map in the basic fuzzy inference algorithm. If we denote the algorithmic implication function by $\Phi$, then the result of the implication computations is given in a vector form as illustrated below.

**Implication Computation Algorithm:**

1. Design data including the RHS design index matrix, predicate matrix, LHS auxiliary parameter vector, weight vector, threshold information for each consequent variable, consequent membership functions, and aggregation operator vector are retrieved from the memory of the fuzzy system.

2. The variable loop starts here. These are the consequent fuzzy variables as defined by the consequent variable vector. Each consequent variable constitutes one output of the inference engine. The outer loop in the LHS algorithm counts the rules whereas the outer loop in the RHS algorithm counts the consequent variables.

3. Each consequent fuzzy variable has its corresponding aggregation operator defined in the memory. At this step, the type of the aggregation operator is determined for this variable.

4. The column counter loop operates on the RHS design index matrix, in which every column corresponds to one predicate (and one membership function) in the RHS predicate matrix.

5. The location of each element in the column counter indicates which predicate (which membership function), will be used in the implication computation along with the LHS result, weight value, and auxiliary parameter.
6. Using all the information gathered above, the implication computation takes place at this step for each predicate. This step involves fuzzy computations and is subject to design.

7. The result from the previous step is a new fuzzy set and is stored in the memory to be used later in the aggregation process.

8. The loop termination condition is employed at this step.

9. This step involves fuzzy computations and is subject to design. A threshold is applied to all individual implication results to avoid residual truth. Note that the thresholding of consequent fuzzy variables is different from that of the antecedent fuzzy variables because they produce different effects.

10. This step also involves fuzzy computations and is subject to design. All individual results (fuzzy sets) belonging to this consequent variable are aggregated in this step using the aggregation operator determined previously. Weight factors are also incorporated by obtaining numerical values from the weight vector. Which weight element to use from the weight vector is also determined by the RHS design index matrix. The result is a new fuzzy set to be used next in the output-processing step.

11. The result from this consequent variable is stored in the memory.

12. Loop terminates when all the consequent fuzzy variables are processed.

B.1.5. Output Data Processing

The output processing is the final step in the fuzzy inference algorithm. It includes the interpretation of the aggregated implication result. The defuzzification method is one of the standard practices to obtain a scalar from the implication result (a fuzzy set).
Algorithm:

1. If weights are chosen to be computed during defuzzification, then the weight vector is retrieved from the memory at this stage. Note that aggregation matrix is also needed along with the weight vector to determine which implication result gets which weight value.

2. Consequent loop counter.

3. Defuzzification is implemented at this stage. If a linguistic output is also expected from this inference engine, then an appropriate matching method is employed. The two most widely used methods are the center-of-gravity defuzzification and maximum possibility linguistic matching.

4. Results obtained through the previous steps are processed and presented.

5. Loop termination condition.

B.2. Fuzzy Membership Functions

The shape of the fuzzy membership function describes the transition between the two prototypes on a feature space. The overlap of membership functions for antecedent and consequent fuzzy variables produces different effects. In antecedent domain, the amount of overlap determines the degree of co-operation among the rules contributing to the same consequent. In the consequent domain, overlap is not significantly effective on the output behavior of the fuzzy inference algorithm when all membership functions are symmetrically expanded or shrunk. Changing the overlap of a single consequent membership function symmetrically, on the other hand, changes the strength of its contribution to the aggregated result, thus the effects are easily traceable and significant.
Appendix C

Representative set of samples from the generated database

(a) Sample data for training

(b) Sample data for testing