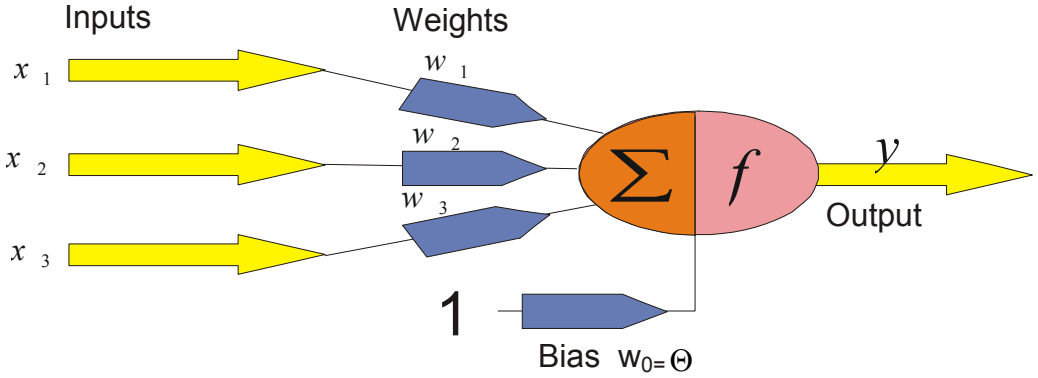
Q.1 a. What is a perceptron? Explain major features of single layer perceptron.

Answer:

One of the first models which made use of the McCulloch and Pitts (1943) model of a neuron was a neural network called the perceptron.

The neurons used in the perceptron have a simple summation input function and a hard-limited threshold activation function or linear threshold activation function. The input values are in general real numbers and the outputs are binary. The connection structure of the perceptron is feedforward and three-layered. The first layer is a buffer, where sensory data are stored. The elements of the first layer are connected either fully or arbitrarily to elements of a second layer, called the "feature layer." Each neuron from this layer combines information coming from different sensory elements and represents a possible feature. The neurons from this layer are connected fully to output neurons from an output layer called the "perceptron layer." The weights between the buffer and the feature layer are fixed; that is why usually only the two layers are presented graphically. This is also the reason why perceptrons are sometimes called "single-layer networks."

Figure is an illustration of a simple perceptron with two elements in the feature layer and one perceptron element in the output layer. The bias is also shown.



$$y = f\left(x_1 w_1 + x_2 w_2 + x_3 w_3 + W_0\right) = f\left(\sum x_i w_i + W_0\right)$$

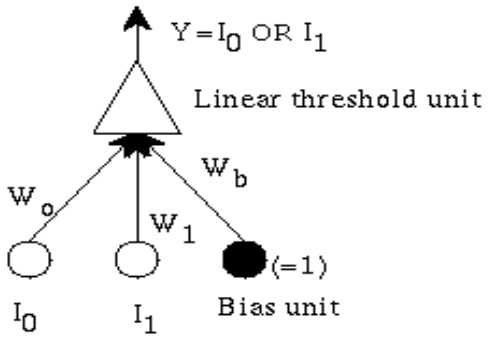
$$y = f\left(w_0 + \sum_{i=1}^{n-1} w_i x_i\right)$$

The neuron calculates a weighted sum of inputs and compares it to a **threshold Θ**. If the sum is higher than the **threshold Θ**, the output is set to ***1***, otherwise to ***0***.

The perceptron neuron produces a **1** if the net input into the transfer function **is equal to or greater than 0**, otherwise it produces a 0.

- It's a single-unit network
- *Change the weight by an amount proportional to the difference between the desired output and the actual output.*

  $\Delta W_i = \eta * (D-Y).X_i$ where η is the learning rate, D is desired output, Y is actual output , I_i is the input



b. Compare features of soft computing paradigm with hard computing paradigm.

Answer:

Soft computing is foundation of conceptual intelligence in machines. Main purpose of soft computing is to model cognitive behavior of human mind. Unlike hard computing , Soft computing is tolerant of imprecision, uncertainty, partial truth, and approximation with the help of technologies Fuzzy Logic (FL),Evolutionary Computation (EC) - based on the origin of the species s.a Genetic Algorithm, Swarm Intelligence, Ant Colony Optimizations, Neural Network (NN),Machine Learning (ML).

Soft Computing is an approach for constructing systems which have features

- computationally intelligent,
- possess human like expertise in particular domain,
- can adapt to the changing environment and can learn to do better
- can explain their decisions

Difference between Hard and Soft Computing

Hard computing

- Based on the concept of precise modeling and analyzing to yield accurate results.
- Works well for simple problems, but is bound by the NP-Complete set.
- Often requires lot of computation time
- Not suited for real world problems for which ideal model is not present.

Soft computing

- Aims to surmount NP-complete problems.
- Uses inexact methods to give useful but inexact answers to intractable problems.
- Represents a significant paradigm shift in the aims of computing - a shift which reflects the human mind.
- Tolerant to imprecision, uncertainty, partial truth, and approximation.
- Well suited for real world problems where ideal models are not available.

c. Describe Genetic Programming (GP). What is the advantage of GP over Genetic Algorithms?

Answer:

Genetic programming starts with randomly created computer programs and evolves programs progressively over a series of generations similar to genetic algorithm. Furthermore, genetic programming is useful in finding solutions where the variables are constantly changing.

A population of random *trees* representing programs is constructed. The genetic operators (crossover, reproduction, etc.) are performed on these trees. In order to create these individuals, two distinct sets are defined:

- the *terminal set* T which includes variables, as well as constants, and
- the *function set* F.

All the functions and terminals must be compatible (i.e. can pass information between each other). Random tree is generated until all the branches end in terminals. To generate a population of programs, just generate as many trees as needed.

Steps:

1. Initially generate a population of random compositions of the functions and terminals of the problem (computer programs).
2 Execute each program in the population and assign it a fitness value according to how well it solves the problem.
3 Create a new population of computer programs.
4 Copy the best existing programs.
5 Create new computer programs by mutation.

6 Create new computer programs by crossover.
7 The best computer program that appeared in any generation is designated as the result of genetic programming.

An advantage of genetic programming over genetic algorithm is that identical parents can yield different offspring, while in genetic algorithms identical parents would yield identical offspring.

d. Briefly describe the Takagi-Sugeno's fuzzy rules with example.

Answer:

Fuzzy rule introduced by Takagi and Sugeno and Kang is a systematic approach to generate fuzzy rules from given input-output data set.

A typical fuzzy rule in the model has the form :

Rule i: IF x is Ai and y is Bi, THEN z = fi(x, y)

Where A and B are fuzzy sets in the antecedent while z = fi(x, y) is the crisp function in the consequent part. Usually f(x, y) is the polynomial in input variables x and y but it can be any function as long as it can appropriately describe the output of the model within the fuzzy region specified by the antecedent of the rule. If the function is linear, the rule takes the following form:

Rule i: IF x1 is A1, i and x2 is A2, i and. . . xm is Am, i, THEN z = C0,i + C1,i·· x1 + ⋯ + Cm,i · xm

Example IF x is A and y is B, THEN z = 5x -2y + 3. These kind of fuzzy rules are very useful especially for function approximation.

e. What role membership function plays in fuzzy logic? Describe Trapezoidal membership function.

Answer:

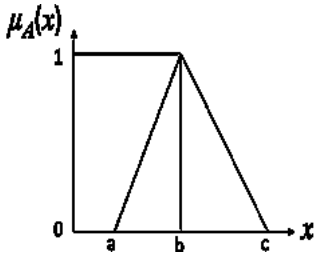
The membership function fully defines the fuzzy set. A membership function provides a measure of *the degree of similarity* of an element to a fuzzy set. There are different shapes of membership functions:

Triangular, Trapezoidal, Gaussian, etc.

Triangular membership function

A *triangular* membership function is specified by three parameters {a, b, c} where a, b and c represent the *x* coordinates of the three vertices of $\mu_A(x)$ in a fuzzy set A (a: lower boundary and c: upper boundary where membership degree is zero, b: the centre where membership degree is 1)

$$\mu_A(x) = \begin{cases} 0 & if\ x \le a \\ \dfrac{x-a}{b-a} & if\ a \le x \le b \\ \dfrac{c-x}{c-b} & if\ b \le x \le c \\ 0 & if\ x \ge c \end{cases}$$



Similarly one can define function from rest.

f. Describe applications of Genetic Algorithms.

Answer:

GAs have been used for many applications:

- Optimization (e.g. circuit layout, job shop scheduling)
- Prediction (Weather forecasting, Protein folding)
- Classification (fraud detection, Quality assessment)
- Economy (bidding strategies, market Evaluation)

- Ecology (Biological arm races, host-parasite coevolution)

In general they are best suited for

- Big search space, non unimodal, non smooth
- Noisy fitness function usually non-analytic
- We do not want to spend years looking for the global optimum, but we just want a good sub-optimum in reasonable time.

g. What is the purpose of using momentum term in error back-propagation algorithm in neural networks? (7×4)

Answer:

If the learning rate (h) is chosen too high (e.g., 0.9) the weights oscillate with a large amplitude; a small learning rate causes a very slow convergence. It has been found that the optimal learning rate is inversely proportional to the number of hidden nodes. One way to accelerate learning when h is chosen small and to prevent oscillation of weights when h is big is to introduce a parameter called momentum (also denoted in the literature by α). This parameter brings inertia to the next change Dwij (t + 1) of the weight wij depending on the direction of its previous change Dwij(t).

The formula used for calculating weight changes thus becomes:

Dwij (t + 1)= η . Err_j . o_j .(1-o_j).o_i+ α . Dwij (t)

where, Err_j is the error between the desired output value yj and the value oj produced by the neuron j which can be simply expressed as Err_j = |yj - oj|.

Q.2 a. What is the necessity of activation function in neural networks? What are various activations functions used in neural networks? Explain any two in detail. (10)

Answer:

In neural network each neuron has an activation function performs mathematical operations to specify output of the neuron to given input. Activation function Controls whether a neuron is “active” or “inactive”.

The most common activation functions are:

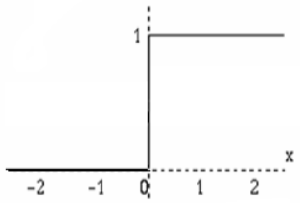
(a) hard-limited threshold;

(b) linear threshold: if the input is above a certain threshold, the output becomes saturated (to a value of 1); there are different variants of this function depending on the range of the neuronal output values.

(c) sigmoid function: logistic function ; bipolar logistic function, gaussian (bell shape) function.

The most used activation functions are shown in figure 4.3. They are:

1. The hard-limited threshold function. If net input value u to the neuron is above a certain threshold, the neuron becomes active (activation value of 1); otherwise it stays inactive (activation value of 0)

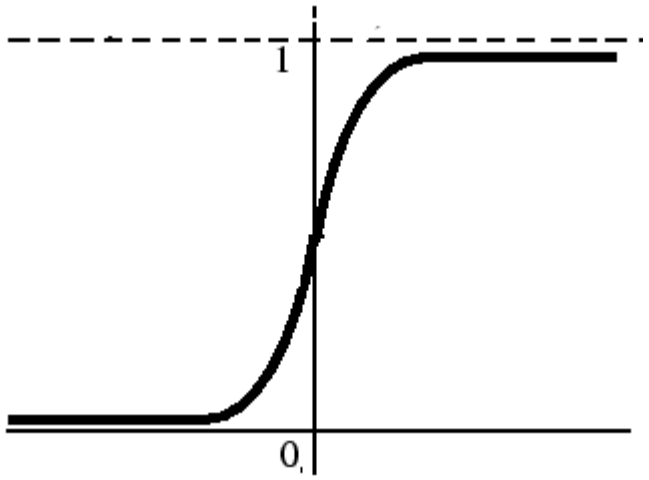


2. The linear threshold function. The activation value increases linearly with the increase of the net input signal u, but after a certain threshold, the output becomes saturated (to a value of 1); there are different variants of this function depending on the range of neuronal output values (figure 4.3[b-1] and [b-2]).

3. The sigmoid function. This is any S-shaped nonlinear transformation function g(u) that is characterized by the following:

a. Bounded, that is, its values are restricted between two boundaries, for example, [0, 1], [- 1, 1].

b. Monotonically increasing, that is, the value g(u) never decreases when u increases.
c. Continuous and smooth, therefore differentiable everywhere in its domain. Different types of sigmoid functions have been used in practice. Most of them are The logistic function: a = 1/(1 + e-u), where e is a constant, the base of natural logarithm (e, sometimes written as exp, is actually the limit of the n-square of (1 + l/n) when n approaches infinity). In a more general form, the logistic function can be written as:
a = 1/(1 + e-c·u), where c is a constant.



b. Two fuzzy sets are given as A={(x1,0.2), (x2,0.7), (x3,1), (x4,0)}, B={(x1,0.5), (x2,0.3), (x3,1), (x4,0.1)}.
Obtain the disjunctive sum of A and B defined as $A \oplus B = (A \cap \sim B) \cup (\sim A \cap B)$. (8)

Answer:

A= {(x1, 0.2), (x2, 0.7), (x3, 1), (x4, 0)}
B= {(x1, 0.5), (x2, 0.3), (x3, 1), (x4, 0.1)}
~A= {(x1, 0.8), (x2, 0.3), (x3, 0), (x4, 1)}
~B= {(x1, 0.5), (x2, 0.7), (x3, 0), (x4, 0.9)}
$(A \cap \sim B)$ = {(x1, 0.2), (x2, 0.7), (x3, 0), (x4, 0)}
$(\sim A \cap B)$ = {(x1, 0.5), (x2, 0.3), (x3, 0), (x4, 0.1)}
and as a consequence,
$A \oplus B = (A \cap \sim B) \cup (\sim A \cap B)$ = {(x1, 0.5), (x2, 0.7), (x3, 0), (x4, 0.1)}

Q.3 a. Explain the concept of binary Hopfield network. (9)

Answer:
Hopfield networks, named after their inventor John Hopfield, are fully connected feedback networks (figure 13.5). It is a recurrent (fully connected network) in which all neurons are connected to each other with the exception that no neuron has any connection to itself.

The Hopfield net normally develops a number of locally stable points in state space. Because the network's dynamics minimize *energy*, other points in state space drain into the stable points (called **attractors** or **wells**), which are (possibly local) energy minima.

The Hopfield net realizes the operation of a *content-addressable (auto-associative) memory* in the sense that newly presented input patterns (or arbitrary initial states) can be connected to the appropriate patterns stored in memories (i.e., attractors or

The Hopfield net normally develops a number of locally stable points in state space. Because the network's dynamics minimize energy, other points in state space drain into stable points (attractors or wells), which are (possibly local) energy minima.

The Hopfield net realizes the operation of a content-addressable (auto-associative) memory in the sense that newly presented input patterns can be connected to the appropriate patterns stored in memories (attractors). The neurons in Hopfield networks are characterized by the following: binary or bivalent input signals, binary or bivalent output signals, simple summation function, and hard-limited threshold activation function. There are alternative variants of realizations of a Hopfield network. Every neuron j, j = 1, 2, . . .,n in the network is connected back to every other one, except itself. Input patterns xj are supplied to the external inputs Ij and cause activation of the external outputs. The response of such a network, when an input vector is supplied during the recall procedure, is dynamic, that is, after supplying the new input pattern, the network calculates

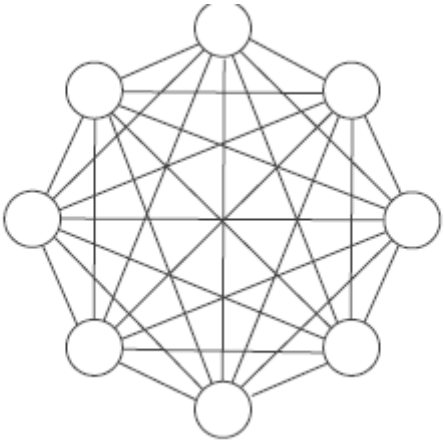


Fig. 13.5 The auto-associator network. All neurons are both input and output neurons, i.e., a pattern is clamped, the network iterates to a stable state, and the output of the network consists of the new activation values of the neurons.

the outputs and then feeds them back to the neurons; new output values are then calculated, and so on, until an equilibrium is reached. Equilibrium is considered to be the state of the system when the output signals do not change for two consecutive cycles, or change within a small constant. The weights in a Hopfield network are symmetrical for reasons of stability in reaching equilibrium, that is,
wij = wji. The network is of an additive type, that is,

$$u_j = \sum_{(i)} (w_{ij} \cdot o_i) + I_j, \qquad (i \text{ not equal to } j),$$

where oj = 1 if uj > Θ j (threshold for the jth neuron);
oj = 0 if uj < Θ j;
and oj is unchanged if uj = Θ j.

The training procedure for a Hopfield network is reduced to a simple calculation of the weights wij on the basis of the training examples with the use of the formula:

$$w_{ij} = \left(\sum_{p=1,m} (2 \cdot x_i^{(p)} - 1) \cdot (2 \cdot x_j^{(p)} - 1) \right)$$

where the summation is held for all the training patterns x(p),x_i(p) is the ith binary value of the input pattern p; and the expressions in parentheses can be only 1 or 0 according to the value of the input pattern. An interesting characteristic of the weights wij is that they measure the correlation between the frequencies of firing of neurons i and j over the full set of examples. It is a variant of the Hebbian learning law, that is, the connection weights increase if two adjacent nodes fire simultaneously.

b. What do you mean by defuzzification? State and define commonly used techniques needed for defuzzification. (9)

Answer:

Defuzzification refers to the way a crisp value is extracted from fuzzy set as a representative value. It is the process of calculating a single-output numerical value for a fuzzy output variable on the basis of the inferred resulting membership function for this variable.

Widely used methods for defuzzifying a fuzzy set of the universe of discourse Z are:

1. Centroid of Area method (COA): This method finds the geometrical centre y' in the universe Z of an output variable y, which center "balances" the inferred membership function B' as a fuzzy value for y. It is the most widely adopted defuzzification strategy.
2. The mean-of-maxima method (MOM). This method finds the value y' for the output variable y which has maximum membership degree according to the fuzzy membership function B'; if there is more than one value which has maximum degree, then the mean of them is taken.
3. Smallest-of-maxima method (SOM).This method finds minimum (in terms of magnitude) of the maximizing y.
4. Largest-of-maxima method (LOM). This method finds maximum (in terms of magnitude) of the maximizing y.

Q.4 a. Describe Kohonen Self Organizing Feature Maps and write its algorithm. Also, mention some of the applications of this network. (10)

Answer:

One of the most used neural network models, mainly for vector quantization and data analysis, is the self-organizing map introduced and developed by Teuvo Kohonen. It is a competition- based network paradigm for data clustering. A SOM consists of two layers, an input layer and an output layer; called a feature map, which represent the output vectors of the output space.The weights of the connections of an output neuron j to all the n input neurons, form a vector wj in an n-dimensional space. The input values may be continuous or discrete, but the output values are binary only.

The main ideas of SOM are as follows:

- Output neurons specialize during the training or recall procedure to react to input vectors of some groups (clusters) and to represent typical features shared by the input vectors. A SOM is able to extract abstract information from multidimensional primary signals and to represent it as a location, in one-, two-, three-, etc. dimensional space.
- The neurons in the output layer are competitive. Lateral interaction between neighboring neurons is introduced in such a way that a neuron has a strong excitatory connection to itself and fewer excitatory connections to its neighboring neurons within a certain radius; beyond this area, a neuron either inhibits the activation of the other neurons by inhibitory connections, or does not influence it. In general, this is "the winner-takes-all" scheme, where only one neuron is the winner after an input vector has been fed in and a competition between the output neurons has taken place. The winning neuron represents the class, the label, and the feature to which the input vector belongs.
- The SOM transforms or preserves the similarity between input vectors from the input space into topological closeness of neurons in the output space represented as a topological map. Similar input vectors are represented by near points (neurons) in the output space. The distance between the neurons in the output layer matters, as this is a significant property of the network.

The unsupervised algorithm for training a SOM, proposed by Teuvo Kohonen.

After each input pattern is presented, the winner is found and the connection weights in its neighbourhood area $NB_c(t)$ increase, while the connection weights outside the area are kept as they are. η is a learningrate parameter. The neighborhood size generally slowly decreases slowly with each iteration.

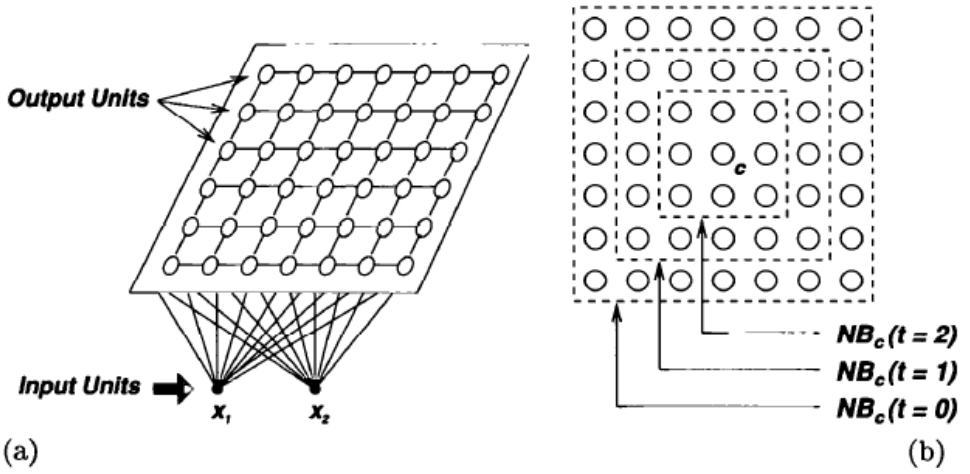


Figure 11.3. *(a) A Kohonen self-organizing network with 2 input and 49 output units; (b) the size of a neighborhood around a winning unit decreases gradually with each iteration.*

Step1: Select the winning output unit as the one with the largest similarity measure (or smallest dissimilarity measure) between all weight vectors $\mathbf{w}_i$ and the input vector $\mathbf{x}$. If the Euclidean distance is chosen as the dissimilarity measure, then the winning unit c satisfies the following equation:

$$\|\mathbf{x} - \mathbf{w}_c\| = \min_i \|\mathbf{x} - \mathbf{w}_i\|,$$

where the index c refers to the winning unit.

Step2: Let NB_c denote a set of index corresponding to a neighborhood around winner c. The weights of the winner and its neighboring units are then updated by

$$\Delta\mathbf{w}_i = \eta(\mathbf{x} - \mathbf{w}_i), \ i \in NB_c,$$

where η is a small positive learning rate. Instead of defining the neighborhood of a winning unit, we can use a **neighborhood function** $\Omega_c(i)$ around a winning unit c. For instance, the Gaussian function can be used as the neighborhood function:

$$\Omega_c(i) = \exp\left(\frac{-\|p_i - p_c\|^2}{2\sigma^2}\right),$$

where p_i and p_c are the positions of the output units i and c, respectively, and σ reflects the scope of the neighborhood. By using the neighborhood function, the update formula can be rewritten as

$$\Delta\mathbf{w}_i = \eta\Omega_c(i)(\mathbf{x} - \mathbf{w}_i), \text{ where } i \text{ is the index for all output units.}$$

Practical applications of the SOM neural networks occur in the following problem areas:

Speech recognition, industrial automatic design of digital systems, and optimization among others. Major known application is Neural phonetic type writer, to learn ballistic arm movements, Data compression in communication and image processing, graph partitioning and word perception models.

b. Define linguistic variable with the help of an example. What do you mean by fuzzy quantization? (8)

Answer:

: One of the most important steps toward using fuzzy logic and fuzzy systems for problem-solving is representing the problem in fuzzy terms. This process is called conceptualization in fuzzy terms. We often use linguistic terms in the process of identification and specification of a problem, or in the process of articulating heuristic rules. We use the term linguistic variable to denote a variable which takes fuzzy values and has a linguistic meaning.

A linguistic variable is "velocity" if it takes as values: "low," "moderate," or "high," for example. A linguistic variable is the variable "Score" if it takes the values "high" and "low." Linguistic values, also called fuzzy labels, fuzzy predicates or fuzzy concepts, have semantic meaning and can be expressed numerically by their membership functions. For example a fuzzy variable "Score" may have as a universe all the numbers between 150 and 200.

Linguistic variables can be Quantitative, for example, "temperature" (low, high); time (early, late); spatial location (around the corner); or Qualitative, for example, "truth," "certainty," "belief."

The process of representing a linguistic variable into a set of linguistic values is called fuzzy quantization. In many cases a linguistic variable can be quantized into some linguistic labels that can be represented by standard functional representations. Standard types of fuzzy membership functions are given:

- Single-valued, or singleton
- Triangular
- Trapezoidal
- S function (sigmoid function)

Two parameters must be defined for the quantization procedure: (1) the number of fuzzy labels, and (2) the form of the membership functions for each of the fuzzy labels.

Choosing the number and the form of all the fuzzy labels that represent a fuzzy variable is a crucial point for a fuzzy system. Are "Low" and "High" fuzzy labels sufficient to represent the fuzzy variable "Score," or do we need a third one, say "Medium," or are even more fuzzy labels needed?

If we have defined the fuzzy-quantizing labels, for example,"small," "medium," or "large," we can represent any particular data item as a set of membership degrees to the fuzzy labels. Usually fuzzy discretization does not lead to loss of information if the fuzzy labels are correctly chosen .

An attribute-value is uniquely represented by the membership coefficients obtained from the membership functions for the fuzzylabels. This process is called fuzzification, or fuzzifying. Fuzzy quantization is possible not only on numerical variables but also on qualitative variables like "truthfulness of events." Fuzzy qualifiers give a fuzzy evaluation of the truthfulness of an event. Typical fuzzy qualifiers are "very true," "more or less true," and "not true," Fuzzy qualifiers can be represented on the scale of truthfulness by fuzzy membership functions.

Q.5 a. Describe Particle Swarm Optimization technique comparing it with continuous GA. Also, write the advantages of PSO. (9)

Answer:

PSO was formulated by Edward and Kennedy in 1995. The thought process behind the algorithm was inspired by the social behavior of animals, such as bird flocking or fish schooling. PSO is similar to the continuous GA in that it begins with a random population matrix. Unlike the GA, PSO has no evolution operators such as crossover and mutation.

PSO concept:

PSO uses a number of agents (**particles**) that constitute a swarm moving around in the search space looking for the best solution. Each particle in search space adjusts its "flying" according to its own flying experience as well as the flying experience of other particles.
Each particle keeps track:

its best solution, personal best, *pbest*
the best value of any particle, global best, *gbest*

Particle update rule

$p = p + v$
with $v = v + c_1 * rand * (pBest - p) + c_2 * rand * (gBest - p)$

where

- p: particle's position
- v: path direction
- c_1: weight of local information
- c_2: weight of global information
- *pBest*: best position of the particle
- *gBest*: best position of the swarm
- *rand*: random variable

The particles update their velocities and positions based on the local and global best solutions:

The PSO algorithm updates the velocity vector for each particle then adds that velocity to the particle position or values. Velocity updates are influenced by both the best global solution associated with the lowest cost ever found by a particle and the best local solution associated with the lowest cost in the present population. If the best local solution has a cost less than the cost of the current global solution, then the best local solution replaces the best global solution.The particle velocity is reminiscent of local minimizers that use derivative information, because velocity is the derivative of position.
Algorithm

1. Create a 'population' of agents (particles) uniformly distributed over X
2. Evaluate each particle's position according to the objective function
3. If a particle's current position is better than its previous best position, update it
4. Determine the best particle (according to the particle's previous best positions)
5. Update particles' velocities:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{inertia}} + \underbrace{\mathbf{c}_1\mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{\text{personal influence}} + \underbrace{\mathbf{c}_2\mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{social influence}}$$

6. Move particles to their new positions:

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \mathbf{v}_i^{t+1}$$

7. Go to step 2 until stopping criteria are satisfied

The advantages of PSO are that it is easy to implement and there are few parameters to adjust.
The PSO is able to tackle tough cost functions with many local minima.

b. How rough set theory is different from fuzzy set theory? Explain rough sets approximations. (9)

Answer:
Rough set theory is another approach to handle vagueness. Imprecision in this approach is expressed by a **boundary region of a set**, and not by a partial membership, like in fuzzy set theory.

Rough set concept can be defined by topological operation ***interior*** and ***closure*** called ***approximations.***

Suppose we are given a set of objects U called the ***universe*** and an indiscernibility relation $R \subseteq U \times U$, representing our lack of knowledge about elements of U. For simplicity we assume that R is an equivalence relation. Let X be a subset of U.We want to characterize the set X with respect to R.

- The ***lower approximation*** of a set ***X*** with respect to ***R*** is the set of all objects, which can be for ***certain(sure)*** classified as X with respect to R (are *certainly* X with respect to R).
- The ***upper approximation*** of a set ***X*** with respect to ***R*** is the set of all objects which can be ***possibly(maybe)*** classified as X with respect to R (are *possibly* X in view of R).
- The ***boundary region*** of a set ***X*** with respect to ***R*** is the set of all objects, which can be classified neither as X nor as not-X with respect to R.

  *Set X is **crisp (Exact** with respect to R), if the boundary region of X is **empty.***

  *Set X is **rough (Inexact** with respect to R), if the boundary region of X is **nonempty.***

Formal definitions of approximations and the boundary region are as follows:

- ***The lower approximation*** of a set is union of all granules which are entirely included in the set.

  $R_*(x) = U \{R(x): R(x) \subseteq X\}$

- ***The upper approximation*** – is union of all granules which have non-empty intersection with the set.

  $R^*(x) = U \{R(x): R(x) \cap X \neq \phi\}$

- ***The boundary region*** of set is the difference between the upper and the lower approximation.

  $RN_R(X) = R^*(X) - R_*(X)$

Q.6 a. What do you mean by crossover in GA encoding? Describe various crossover operators with example. How mutation differs from crossover? (10)

Answer:

Crossover: Create new offspring program(s) for the new population by recombining randomly chosen parts from two selected programs. It operates on selected genes from parent chromosomes to create new offspring. The simplest way is to choose some crossover point randomly copy everything before this point from the first parent and then copy everything after the crossover point from the other parent.

Single point crossover:

- one crossover point is selected,
- binary string from the beginning of the chromosome to the crossover point is copied from the first parent,
- the rest is copied from the other parent

  11001011+11011**111** = **11001111**

Two point crossover:

- two crossover points are selected,
- binary string from the beginning of the chromosome to the first crossover point is copied from the first parent,
- the part from the first to the second crossover point is copied from the other parent and

- the rest is copied from the first parent again
  11001**01** + 11**011**11 = **11011101**

Uniform crossover: bits are randomly copied from the first and second parent
11101010 + 11010101 = 11010011

Arithmetic crossover: Arithmetic operation is performed to make a new offspring
11001011 + 11011111 = 11001001 (AND)

Mutation operation randomly changes the offspring resulted from crossover. Mutation is intended to prevent falling of all solutions in the population into a local optimum of the problem. In case of binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. The technique of mutation (as well as crossover) depends mainly on the encoding of chromosomes. Mutation can be then illustrated as follows:

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110100

b. What is genetic algorithm? Write down the basic genetic algorithm. (8)

Answer:

The genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function).The method was developed by John Holland.

Various applications of GA are in

- Control
- Design
- Scheduling
- Robotics
- Machine Learning
- Signal Processing
- Game Playing
- Combinatorial Optimization

Algorithm begins with a **set of initial solutions** (represented by set of **chromosomes**) called **population**.

- A **chromosome** is a string of elements called ***genes***.
- Solutions from one population are taken and are used to form a new population by generating offsprings.
- New population is formed using old population and offspring based on their fitness value.
- Promising candidates are kept and allowed to reproduce
- This is motivated by a hope, that the new population will be better than the old one.
- Genetic algorithms are broadly applicable and have the advantage that they require little knowledge encoded in the system.

Steps of Algorithm

1. [**Start**] Generate random population of *n* chromosomes (suitable solutions for the problem).
2. [**Fitness**] Evaluate the fitness *f(x)* of each chromosome *x* in the population.
3. Repeat until terminating condition is satisfied
   a. [**Selection**] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
   b. [**Crossover**] Crossover the parents to form new offsprings (children). If no crossover was performed, offspring is the exact copy of parents.
   c. [**Mutation**] Mutate new offspring at selected position(s) in chromosome).

d. [**Accepting**] Generate new population by placing new offsprings.

4. Return the best solution in current population

Q.7 a. Why there is a need for hybridization to build intelligent systems? What are the main characteristics of neuro-fuzzy systems? (10)

Answer:

NN, FL and GA are soft computing methods which are inspired by biological conceptual process. When these technologies are integrated to capture the best possible features of each one the process is known as hybridization. Thus, Hybrid systems employ more than one technology to solve a problem making them intelligent.

Each of these (NN, FL and GA) technology has provided efficient solution to wide range of problems. Many of these approaches use a combination of different knowledge representation schemes, decision making models and learning strategies to solve a computational task. Since each of these technologies suffers form advantages and disadvantages, therefore this integration aims at overcoming the limitations of individual techniques through hybridization or the fusion of various techniques. These ideas have led to the emergence of several different kinds of intelligent system architectures.

Characteristics of Neuro-fuzzy systems:

- Fuzzy logic and neural networks are natural complementary tools in building intelligent systems. Both represent different methodology to deal with uncertainity. Each of these has its own merits and demerits.
- Neural networks are low-level computational structures that perform well when dealing with raw data. It can model complex nonlinear relationship and are appropriately suited for classification phenomenon into predefined classes. Although neural networks can learn but its output precision is often limited to least square errors, training time required is quite large, training data has to be chosen over entire range where variables are expected to change.
- Whereas fuzzy logic deals with reasoning on a higher level, using linguistic information acquired from domain experts. However, fuzzy systems lack the ability to learn and cannot adjust themselves to a new environment.
- Integrated neuro-fuzzy systems can combine the parallel computation and learning abilities of neural networks with the human-like knowledge representation, and explanation abilities of fuzzy systems. As a result, neural networks become more transparent, while fuzzy systems become capable of learning.
- A neuro-fuzzy system is a neural network which is functionally equivalent to a fuzzy inference model.It can be trained to develop **IF-THEN fuzzy rules** and determine **membership functions** for input and output variables of the system. Expert knowledge can be incorporate into the structure of the neuro-fuzzy system.
- The structure of a neuro-fuzzy system is similar to a multi-layer neural network. In general, A Neuro-fuzzy system has input and output layers, and three hidden layers that represent membership functions and fuzzy rules.

There are two ways to do the hybridization:

- One is to provide NN with fuzzy capabilities, thereby increasing the network's expressiveness and flexibility to adapt to uncertain environment.
- Second is to apply neural learning capabilities to fuzzy systems so that fuzzy systems become more adaptive to changing environments. This method is called NN-driven fuzzy reasoning.

b. Write a note on decision table for rough set based data analysis. (8)

Answer:
Rough set based data analysis starts from a data table called a *decision table*, columns of which are labeled by *attributes*, rows – by *objects* of interest and entries of the table are *at- tribute values*. Attributes of the decision table are divided into two disjoint groups called *condition* and *decision* attributes, respectively. Each row of a decision table induces a *decision rule*, which specifies decision (action, results, outcome, etc.) if some conditions are satisfied. If a decision rule uniquely determines decision in terms of conditions – the decision rule is *certain*. Otherwise the decision rule is *uncertain*. Decision rules are closely connected with approximations. Roughly speaking, certain decision rules describe lower approximation of decisions in terms of conditions, whereas uncertain decision rules refer to the boundary region of decisions. With every decision rule two conditional probabilities, called the *certainty* and the *coverage* coefficient, are associated.

The certainty coefficient expresses the conditional probability that an object belongs to the decision class specified by the decision rule, given it satisfies conditions of the rule. The coverage coefficient gives the conditional probability of reasons for a given decision.It turns out that the certainty and coverage coefficients satisfy Bayes' theorem. That gives a new look into the interpretation of Bayes' theorem, and offers a new method data to draw conclusions from data.

If we distinguish in an information system two disjoint classes of attributes, called *condition* and *decision attributes*, respectively, then the system will be called a *decision table* and will be denoted by $S=(U;C;D)$, where C and D are disjoint sets of condition and decision attributes,
respectively.

TEXT BOOKS

I. S R Jang, C T Sun and E Mizutani, Neuro-Fuzzy and Soft Computing, Pearson Education 2004
II. Andries P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley and Sons, 2007