Computational Intelligence

Unit # 8

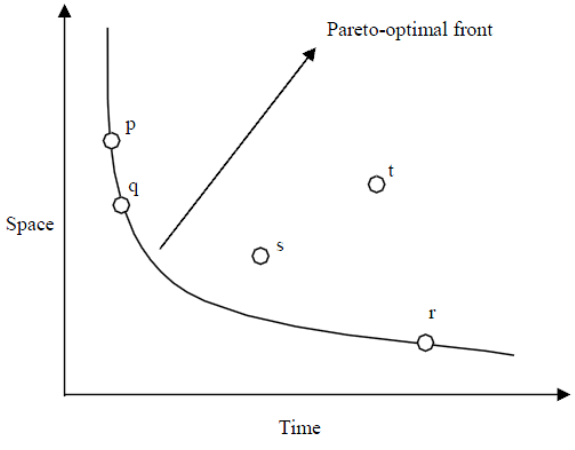
Multi-Criteria Optimization

- In a multi-criterion optimization problem, there is more than one objective function, each of which may have a different individual optimal solution.
- If there is a sufficient difference in the optimal solutions corresponding to different objectives then we say that the objective functions are conflicting to each other.
- Multi-criterion optimization with such conflicting objective functions give rise to a set of optimal solutions, instead of one optimal solution.

Pareto Optimal Solutions

- The reason for the optimality of many solutions is that no one can be considered to be better than any other with respect to all other objective functions.
- These optimal solutions have a special name called Pareto-optimal solutions following the name of an economist Vilfredo Pareto.

Pareto-Optimal Front



Dominant and Non-dominant Solutions

Let us consider a problem having m objectives (say $f_i, i = 1,2,3,.....,m$ and $m>1$). Any two solutions $u^{(1)}$ and $u^{(2)}$ (having 't' decision variables each) can have one of two possibilities-one dominates the other or none dominates the other. A solution $u^{(1)}$ is said to *dominate* the other solution $u^{(2)}$, if the following conditions are true:

1. The solution $u^{(1)}$ is no worse (say the operator $\prec$ denotes worse and $\succ$ denotes better) than $u^{(2)}$ in all objectives, or $f_i(u^{(1)}) \geq f_i(u^{(2)}), \forall i = 1,2,3.....,m.$
2. The solution $u^{(1)}$ is strictly better than $u^{(2)}$ in at least one objective, or $f_i(u^{(1)}) \succ f_i(u^{(2)})$ for at least one, $i \in \{1,2,3,...,m\}$.

Two Important Goals

- There are primarily two goals that a multi-criterion optimization algorithm must try to achieve:
  - Guide the search towards the global Pareto-optimal region, and
  - Maintain population diversity in the Pareto-optimal front.

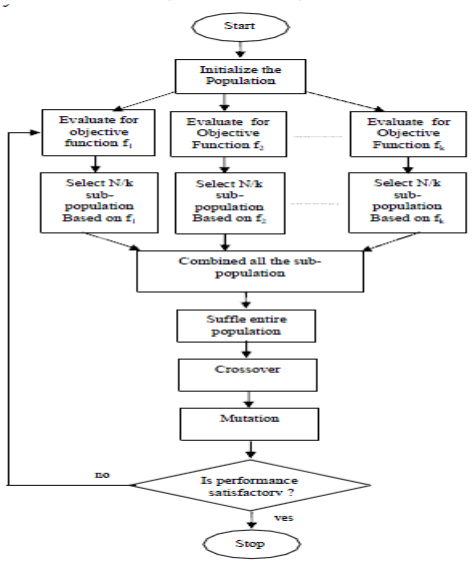
Weighted Sum Approach

- Generate the initial population randomly.
- Compute the value of the m objectives for each individual in the population.
  - Place dominant solution separately (external list)
  - The remaining solutions are stored in 'current'.
- Apply crossover/mutation to solutions in 'current'.
- Merge the external list with the new solutions and repeat the process.

Vector Evaluation Genetic Algorithm (VEGA)

- Generate the initial population randomly.
- Compute the value of the m objectives for each individual in the population.
  - Select k best solutions based on each objective.
  - Combine the k-best solutions (for each objective).
- Apply crossover/mutation to solutions in 'the combined list.
- Repeat the process.

Vector Evaluation Genetic Algorithm (VEGA)



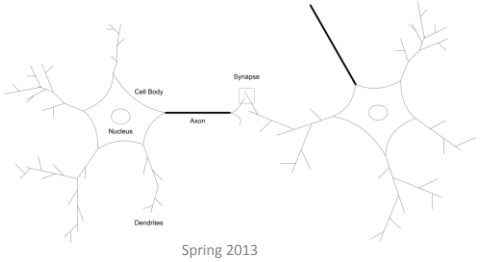
Strength Pareto Approach

- Initialize the population of size n.
- Determine all the dominant solutions from current population and store them in a separate population called EXTERNAL.
- Assign fitness to individuals according to how many solutions it dominates.
- After assigning the fitness to all the individuals, select n individuals for next generation.
- Apply crossover and mutation to get the new population of size n.
- Update EXTERNAL population.

Artificial Neural Networks

Human Brain

- The human brain has an estimated 10^{11} tiny units called neurons (100 of billions).
- These neurons are connected with an estimated 10^{15} links.
- Although more research needs to be done, the neural network of the brain is considered to be the fundamental functional source of intelligence, which includes perception, cognition, and learning for humans as well as other living creatures.



How Our Brain Works?

- While neural networks are modeled after our understanding of the way in which our brain works, surprisingly little is known about how our brains actually function.
- Through various types of inspection, we can see our brain in operation, but because of the massive number of neurons and interconnections between these neurons, how it works remains a mystery (though many theories exist).

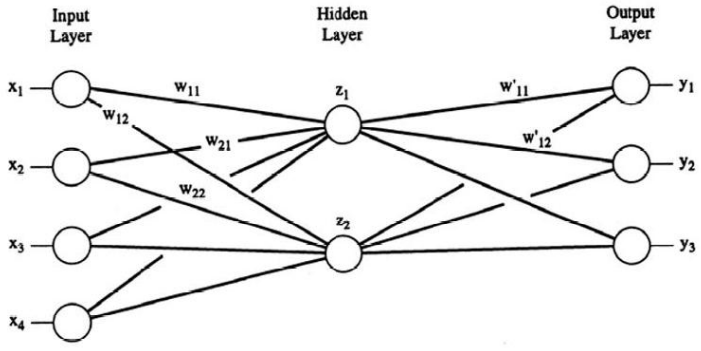
Artificial Neural Networks

- Neural networks are biologically motivated computing structures that are conceptually modeled after the brain.
- Similar to the brain, a neural network is composed of artificial neurons (or units) and interconnections.
- When we view such a network as a graph, neurons can be represented as nodes (or vertices), and interconnections as edges.
- The basic element of the brain is a natural neuron; similarly, the basic element of every neural network is an artificial neuron, or simply **neuron.** That is, a neuron is the basic building block for all types of neural networks.

Multilayer Feed-Forward Networks

- Multilayer feed-forward networks are one of the most important and most popular classes of ANNs in real-world applications.
- They are commonly referred to as multilayer perceptrons, which represent a generalization of the simple perceptron.

Architecture



Characteristics of MLP

- A multilayer perceptron has three distinctive characteristics
  - The model of each neuron in the network includes usually a nonlinear activation function, sigmoid or hyperbolic.
  - The network contains one or more layer of hidden neurons that are not a part of the input or output of the network. These hidden nodes enable the network to learn complex and highly nonlinear tasks by extracting progressively more meaningful features from the input patterns.
  - The network exhibits a high degree of connectivity from one layer to the next one.

Weights on the Link

- All the neurons in the input layer are connected to all the neurons in the hidden layer through the edges.
- Similarly, all the neurons in the hidden layer are connected to all the neurons in the output layer through the edges.
- A **weight** is associated with each edge.

Backpropagation

- Although many neural network models have been proposed, the backpropagation is the most widely used model in terms of practical applications.
- No statistical surveys have been conducted, but probably over 90% of commercial and industrial applications of neural networks use backpropagation or its derivatives [*Source: Fundamental of the New Artificial Intelligence, 2008*].

Input Format

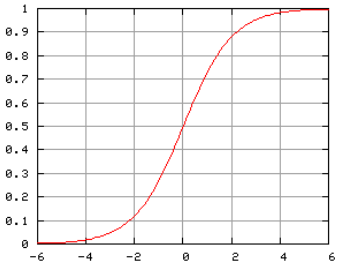
- The input to individual neural network nodes should be numeric and fall in the closed interval range [0,1].
- We need a way to numerically represent categorical data.
  - Attribute Color: {Red, Green, Blue, Yellow}
- We also need a conversion method for numerical data falling outside the [0,1] range.
  - Values: 100, 200, 300, 400

Output Format

- The nodes of the input layer pass input attribute values to the hidden layer unchanged.
- A hidden or output layer node takes input from the connected nodes of the previous layer, combines the previous layer node values into a single value, and uses the new value as input to an evaluation function.
- The output of the evaluation function is a number in the closed interval [0, 1].

Sigmoid Function

- The first criterion of an evaluation function is that the function must output values in the [0, 1] interval range.
- A second criterion is that the function should output a value close to 1 when sufficiently excited.
- The sigmoid function meets both criterion and is often used for node evaluation.
  - $f(x) = 1 / (1 + e^{-x})$

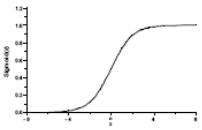


Transfer Functions

Sigmoid Functions These are smooth (differentiable) and monotonically increasing.

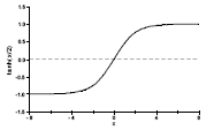
The logistic function

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$$

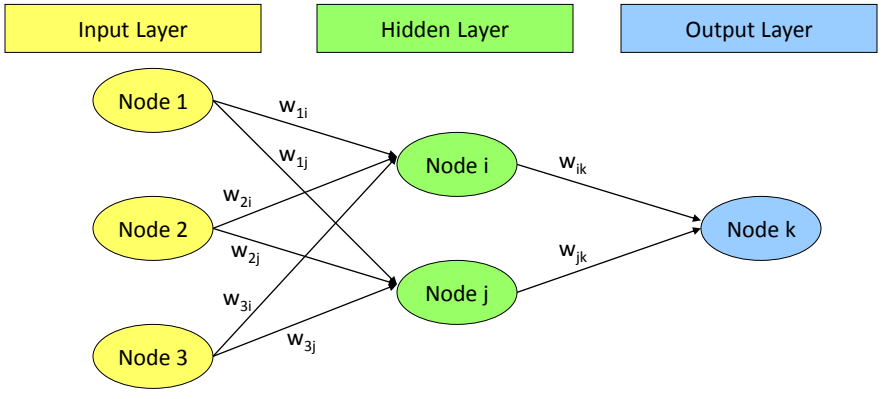


Hyperbolic tangent

$$\tanh\left(\frac{x}{2}\right) = \frac{1-e^{-x}}{1+e^{-x}}$$



A Fully Connected Feed-Forward Network



Explanation of the Backpropagation Algorithm

w_{1i}=0.20, w_{1j}=0.10, w_{2i}=0.30, w_{2j}=-0.10, w_{3i}=-0.10, w_{3j}=0.20, w_{ik}=0.10, w_{jk}=0.50, T=0.95

- Input = {1.0, 0.4, 0.7}
- Input to node i = 0.2x1.0 + 0.3x0.4 - 0.1x0.7 = 0.25
- Now apply the sigmoid function: f(0.25) = 0.562
- Input to node j = ?
- Input to node k = ?
- Error(k) = $(T - O_k)\ O_k\ (1 - O_k)$
  - T = the target output
  - O_k = the computed output at node k
- Error(k) = ?

Explanation of the Backpropagation Algorithm

w_{1i}=0.20, w_{1j}=0.10, w_{2i}=0.30, w_{2j}=-0.10, w_{3i}=-0.10, w_{3j}=0.20, w_{ik}=0.10, w_{jk}=0.50

- Error(i) = Error(k) $w_{ik}\ O_i\ (1 - O_i)$
  = ?
- Error(j) = ?
- The next step is to update the weights associated with the individual node connections.
- Weight adjustments are made using the delta rule
  - To minimize the sum of the square errors, where error is defined as the distance between computed and actual output

Explanation of the Backpropagation Algorithm

w_{1i}=0.20, w_{1j}=0.10, w_{2i}=0.30, w_{2j}=-0.10, w_{3i}=-0.10, w_{3j}=0.20, w_{ik}=0.10, w_{jk}=0.50

- $w_{ik} = w_{ik}$ (current) + Δw_{ik}
- Δw_{ik} = r x Error(k) x O_i
  - where r is learning rate parameter, 0 < r < 1
- Compute: Δw_{jk} Δw_{1i} Δw_{2i} Δw_{3i}

Supervised Learning

- When we view the neural network macroscopically as a black box, it learns mapping from the input vectors to the target vectors.
- Microscopically it learns by adjusting its weights.
- We assume that there is a teacher who knows and tells the neural network what are correct input-to-output
- mapping.
- The backpropagation model is called a **supervised learning method** for this reason, i.e., it learns under supervision.
- It cannot learn without being given correct sample patterns.

Outline of the Learning Process

- Outer loop.
  - Repeat the following until the neural network can consecutively map all patterns correctly.
- Inner loop.
  - For each pattern, repeat the following Steps 1 to 3 until the output vector y is equal (or close enough) to the target vector t for the given input vector x.

Outline of the Learning Process (Cont'd)

- Inner Loop (Cont'd)
  - Step 1. Input x to the neural network.
  - Step 2. Feedforward. Go through the neural network, from the input to hidden layers, then from the hidden to output layers, and get output vector y.
  - Step 3. Backward propagation of error corrections. Compare y with t. If y is equal or close enough to t, then go back to the beginning of the Outer loop. Otherwise, backpropagate through the neural network and adjust the weights so that the next y is closer to t, then go back to the beginning of the Inner loop.
- In the above, each Outer loop iteration is called an **epoch**. An epoch is one cycle through the entire set of patterns under consideration.

Number of Neurons in Each Layer

- The number of neurons in the input and output layers are usually determined from a specific application problem.
- For example, for a written character recognition problem, each character is plotted on a two-dimensional grid of 100 points.
- The number of input neurons would then be 100.
- For the hidden layer(s), there are no definite numbers to be computed from a problem. Often, the trial-and-error method is used to find a good number.

Neural Network Learning

- A neural network learns patterns by adjusting its weights.
- Note that "patterns“ here should be interpreted in a very broad sense. They can be visual patterns such as two-dimensional characters and pictures, as well as other patterns which may represent information in physical, chemical, biological, or management problems.
- To predict the performance of a particular stock in the stock market, the abscissa may represent various parameters of the stock (such as the price of the stock the day before, and so on), and the ordinate, values of these parameters.

NN Learning Process

- A neural network is given correct pairs of (input pattern, target pattern). That is, (input pattern 1, target pattern 1), (input pattern 2, target pattern 2), and so forth, are given.
- The learning task of the neural network is to adjust the weights so that it can output the target pattern for each input pattern.

Example of Learning Process

- Suppose, for example, that we have two sample patterns to train the neural network.
- We repeat the inner loop for Sample 1, and the neural network is then able to map the correct t after, say, 10 iterations.
- We then repeat the inner loop for Sample 2, and the neural network is then able to map the correct t after, say, 8 iterations.
- This is the end of the first epoch.
- The end of the first epoch is not usually the end of the algorithm or outer loop. After the training session for Sample 2, the neural network "forgets" part of what it learned for Sample 1.
- Therefore, the neural network has to be trained again for Sample 1.

Example of Learning Process (Cont'd)

- But, the second round (epoch) training for Sample 1 should be shorter than the first round, since the neural network has not completely forgotten Sample 1.
- It may take only 4 iterations for the second epoch.
- We can then go to Sample 2 of the second epoch, which may take 3 iterations, and so forth.
- When the neural network gives correct outputs for both patterns with 0 iterations, we are done. This is why we say "consecutively map all patterns" in the first part of the algorithm.
- Typically, many epochs are required to train a neural network for a set of patterns.

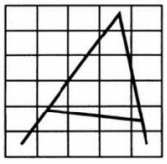
Alternate Ways of Learning

- There are alternate ways of performing iterations. One variation is to train Pattern 1 until it converges, then store its w_{ij}s in temporary storage without actually updating the weights.
- Repeat this process for Patterns 2, 3, and so on, for either several or the entire set of patterns.
- Then take the average of these weights for different patterns for updating.
- Another variation is that instead of performing the inner loop iterations until one pattern is learned, the patterns are given in a row, one iteration for each pattern.
- For example, one iteration of Steps 1, 2, and 3 are performed for Sample 1, then the next iteration is immediately performed for Sample 2, and so on.

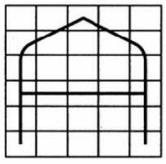
Pattern Recognition of Hand-Written Characters

- When people hand-write characters, often the characters are off from the standard ideal pattern.
- The objective is to make the neural network learn and recognize these characters even if they are slightly deviated from the ideal pattern.

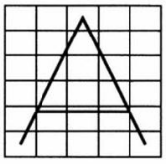
Pattern Recognition of Hand-Written Characters (Cont'd)



(a)



(b)



(c)

0	0	0	1	0
0	0	1	1	0
0	0	1	1	1
0	1	0	0	1
1	1	1	1	1
1	0	0	0	1

(d)

x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{26}	x_{27}	x_{28}	x_{29}	x_{30}

(e)

- (a) and (b): two sample input patterns;
- (c): a target pattern;
- (d) input vector x for
- Pattern (a);
- (e) layout for input vector x, output vector y (for y, replace *x with y), and target* vector t (for t, replace *x with t);*

Problem Representation

- Each pattern in this example is represented by a two-dimensional grid of 6 rows and 5 columns.
- We convert this two-dimensional representation to one-dimensional by assigning the top row squares to x1 to x5, the second row squares to x6 to x10, etc., as shown in Fig. (e).
- Since xi ranges from i = 1 to 30, we have 30 input layer neurons.
- Similarly, since yi also ranges from i = 1 to 30, we have 30 output layer neurons.
- In this example, the number of neurons in the input and output layers is the same, but generally their numbers can be different.
- We may arbitrarily choose the number of hidden layer neurons as 15.

Weights Initialization

- After designing the architecture, we initialize all the weights associated with edges randomly, say, between -0.5 and 0.5.
- Then we perform the training algorithm until both patterns are correctly recognized.
- In this example, each yi may have a value between 0 and 1. 0 means a complete blank square, 1 means a complete black square, and a value between 0 and 1 means a "between" value: gray.
- Normally we set up a threshold value, and a value within this threshold value is considered to be close enough. For example, a value of yi anywhere between 0.95 and 1.0 may be considered to be close enough to 1; a value of yi anywhere between 0.0 and 0.05 may be considered to be close enough to 0.

Learned Neural Network

- After completing the training sessions for the two sample patterns, we might have a surprise.
- The trained neural network gives correct answers not only for the sample data, but also it may give correct answers for totally new similar patterns.
- In other words, the neural network has robustness for identifying data.
- This is indeed a major goal of the training - a neural network can generalize the characteristics associated with the training examples and recognize similar patterns it has never been given before.

Learned Neural Network (Cont'd)

- We can further extend this example to include more samples of character "A," as well as to include additional characters such as "B," "C," and so on.
- However, a word of caution for such extensions in general: training of a neural network for many patterns is not a trivial matter, and it may take a long time before completion. Even worse, it may never converge to completion.
- It is not uncommon that training a neural network for a practical application requires hours, days, or even weeks of continuous running of a computer.
- Once it is successful, even if it takes a month of continuous training, it can be copied to other systems easily and the benefit can be significant.

Number of Neurons in Hidden Layer

- The number of neurons in the input and output layers is determined from the problem description.
- However, there is no formula for the number of hidden layers, so we have to find it from our experiment. As a gross approximation, we may start with about half as many hidden layer neurons as we have input layer neurons.
- When we choose too few hidden layer neurons, the computation time for each iteration will be short since the number of weights to be adjusted is small.
- On the other hand, the number of iterations before convergence may take a long time since there is not much freedom or flexibility for adjusting the weights to learn the required mapping. Even worse, the neural network may not be able to learn certain patterns at all.
- On the other hand, if we choose too many hidden layer neurons, the situation would be the opposite of too few. That is, the potential learning capability is high, but each iteration as well as the entire algorithm may be time-consuming since there are many weights to be adjusted.

Order of Training Patterns

- training patterns and subsequent weight modifications can be arranged in different sequences.
- For example, we can give the network a single pattern at a time until it is learned, update the weights, then go to the next pattern.
- Or we can temporarily store the new weights for several or all patterns for an epoch, then take the average for the weight adjustment.
- These approaches and others often result in different learning speeds.
- We can experiment with different approaches until we find a good one.
- Another approach is to temporarily drop input patterns that yield small errors, i.e., easy patterns for the neural network for learning.
- Concentrate on hard patterns first, then come back to the easy ones after the hard patterns have been learned.