5 Machine Learning in Bioinformatics

Supawan Prompramote[1], Yan Chen[1] and Yi-Ping Phoebe Chen[1,2]

[1] School of Information Technology
Faculty of Science and Technology
Deakin University, 221 Burwood Highway, VIC3125, Australia
[2] ARC Centre in Bioinformatics

5.1 Introduction

Given the complexity and gigantic volume of biological data, the traditional computer science techniques and algorithms fail to solve complex biological problems of the real world. However, there are modern computational approaches called machine learning that can address the limitations of the traditional techniques. Machine learning is an adaptive process that enables computers to learn from experience, learn by example, and learn by analogy. Learning capabilities are essential for automatically improving the performance of a computational system over time on the basis of previous results. A basic learning model typically consists of the following four components:

- learning element, responsible for improving its performance,
- performance element, which decides the choice of actions to be taken,
- critical element, which tells learning element how the algorithm performs, and
- problem generator, responsible for suggesting actions that could lead to new or informative experiences (Adeli, 1995; Finlay and Dix 1996;

Kuonen, 2003; Narayanan et al., 2002; Negnevitsky, 2002; Nilsson, 1996; Baldi and Brunak, 2001; and Westhead et al., 2002).

Machine learning typically can be divided into three phases, as follows:

1. analysis of a training set of examples and generation of a set of rules from training set,
2. verification of the rules by human experts or automatic knowledge based components and
3. use of the validated rules in responding to some new testing datasets (Finlay and Dix 1996).

There are a number of reasons why machine learning approaches are widely used in practice, especially in bioinformatics (Narayanan et al., 2002; Nilsson, 1996; Baldi and Brunak, 2001; and Westhead et al., 2002)

- Traditionally, a human being builds such an expert system by collecting knowledge from specific experts. The experts can always explain what factors they use to assess a situation; however, it is often difficult for the experts to say what rules they use, for example, for disease analysis and control. This problem can be resolved by machine learning mechanisms. Machine learning can extract the description of the hidden situation in terms of those factors and then fire rules that match the expert's behavior.
- Systems often produce results different from the desired ones. This may be caused by unknown properties or functions of inputs during the design of systems. This situation always occurs in the biological world because of the complexities and mysteries of life sciences. However, with its capability of dynamic improvement, machine learning can cope with this problem.
- In molecular biology research, new data and concepts are generated every day, and those new data and concepts update or replace the old ones. Machine learning can be easily adapted to a changing environment. This benefits system designers, as they do not need to redesign systems whenever the environment changes.
- Missing and noisy data is one characteristic of biological data. The conventional computer techniques fail to handle this. Machine learning techniques are able to deal with missing and noisy data.
- With advances in biotechnology, huge volumes of biological data are generated. In addition, it is possible that important hidden relationships and correlations exist in the data. Machine learning methods are de-

signed to handle very large data sets, and can be used to extract such relationships.

Table 5.1. The existing research on bioinformatics that has applied machine learning techniques.

Research Area	Application	Reference
Sequence alignment	BLAST	http://www.ncbi.nlm.nih.gov/BLAST/
	FASTA	http://www.ebi.ac.uk/fasta33/
Multiple sequence alignment	ClustalW	http://www.ebi.ac.uk/clustalw/
	MultAlin	http://prodes.toulouse.inra.fr/multalin/multalin.html
	DiAlign	http://www.genomatix.de/cgi-bin/dialign/dialign.pl
Gene finding	Genscan	http://genes.mit.edu/GENSCAN.html
	GenomeScan	http://genes.mit.edu/genomescan/
	GeneMark	http://www.ebi.ac.uk/genemark/
Protein domain analysis and identification	Pfam	http://www.sanger.ac.uk/Software/Pfam/
	BLOCKS	http://www.blocks.fhcrc.org/
	ProDom	http://prodes.toulouse.inra.fr/prodom/current/html/ home.php
Pattern identification	Gibbs Sampler	http://bayesweb.wadsworth.org/gibbs/gibbs.html
	AlignACE	http://atlas.med.harvard.edu/cgi-bin/alignace.pl
	MEME	http://meme.sdsc.edu/meme/website/intro.html
Protein folding prediction	PredictProtein	http://www.embl-heidelberg.de/predictprotein/ predictprotein.html
	SwissModle	http://www.expasy.org/swissmod/SWISS-MODEL.html

- There are some biological problems in which experts can specify only input/output pairs, but not the relationships between inputs and outputs, such as the prediction of protein structure and structural and functional sequences. This limitation can be addressed by machine learning methods. They are able to adjust their internal structure to produce approximate results for the given problems.

Machine learning mechanisms form the basis of adaptive systems. In bioinformatics research, a number of machine learning approaches are applied to discover new meaningful knowledge from the biological databases, to analyze and predict diseases, to group similar genetic elements, and to find relationships or associations in biological data. Examples of machine learning approaches in bioinformatics research are shown in Table 5.1.

In this chapter, the most popular of machine learning approaches, namely, artificial neural networks, genetic algorithms, and fuzzy expert systems, are elaborated. The basic background, definition, and models of each method are presented. Further, a survey of tools for using the learning techniques used in bioinformatics is included.

5.2 Artificial Neural Network

The process of learning is a complex phenomenon. Many puzzling questions arise from of it. How can one recognize the faces of others? How can one identify complex patterns from the faces? How does one discriminate images and backgrounds? How does one learn a shortcut to go to his or her university? In order to answer these questions, one needs to know how the brain works.

The human brain has been studied since the late Middle Ages; however, its detailed structure began to be unraveled only in the nineteenth century. Neuronists claim that the brain is a collection of about 10 billion densely interconnected cellular units called neurons. The structure of a neuron and its network is shown in Fig. 5.1.

Each neuron consists of a cell body called soma, a number of root-like extensions connected to a thousand adjacent neurons called dendrites, and a single transmission line extending out from the soma called axon. The two specialized extensions of a soma are responsible for carrying information from/to a cell body. Dendrites bring information to a cell body and axons take information away from a cell body. The connection between

two neurons, in particular, between an axon terminal and another neuron, is called synapse.

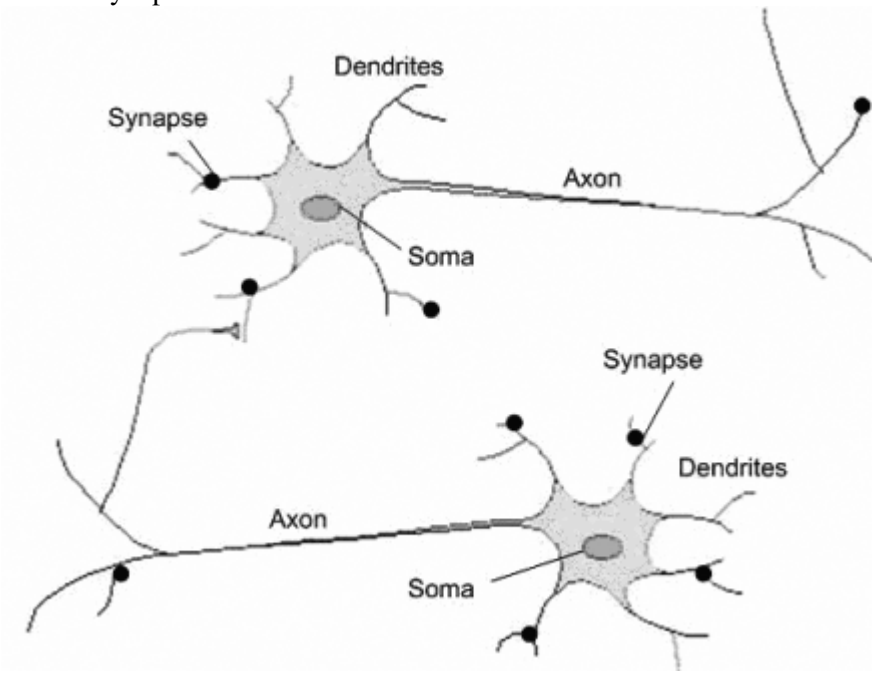

Fig. 5.1. Biological neural network (Adapted from: http://ffden2.phys.uaf.edu/-212_fall2003.web.dir/Keith_Palchikoff/Intro_page.html)

Each neuron uses biochemical reactions to receive processes and transmit information. Neurons communicate with each other through an electrochemical process. This means that chemicals create an electrical signal. When a neuron does not send a signal, it is in a resting state. The inside of the neuron has a negative electric potential. When a neuron sends a signal, it causes a change in the electrical potential of the cell body. The change occurs due to the release of chemical substances from the synaptic cell, called neurotransmitter. When the potential exceeds a certain threshold, an action potential occurs. Consequently, the neuron will fire the electrical signal down the axon. The occurrence of action potential can be increased or decreased by changing the constitution of various neurotransmitters.

An essential characteristic of biological neural networks is plasticity, an ability of the brain to reorganize with learning, based on experience or sensory stimulation. Scientists believe that there are two types of modifications that form the basis of learning in the brain, namely, 1) a change in the internal structure of the synapses and 2) an increase in the number of synapses between neurons.

The natural and power of a biological neural network, in particular, the potential of learning process, motivated computer scientists to design and develop a new network platform that worked in a way similar to that of the biological neurons (Adeli, 1995; Freeman and Skapura, 1991; Haykin, 1994; Müller and Reinhardt, 1990; and Negnevitsky, 2002). This leads to the introduction of Artificial Neural Networks (ANNs).

An Artificial Neural Network (ANN) is an information processing model that is able to capture and represent complex input-output relationships. The motivation the development of the ANN technique came from a desire for an intelligent artificial system that could process information in the same way the human brain. Its novel structure is represented as multiple layers of simple processing elements, operating in parallel to solve specific problems. An architecture of a typical artificial neural network is illustrated in Fig. 5.2. ANNs resemble human brain in two respects: learning process and storing experiential knowledge. An artificial neural network learns and classifies a problem through repeated adjustments of the connecting weights between the elements. In other words, an ANN learns from examples and generalizes the learning beyond the examples supplied. For instance, human beings learn to recognize faces from examples of faces.

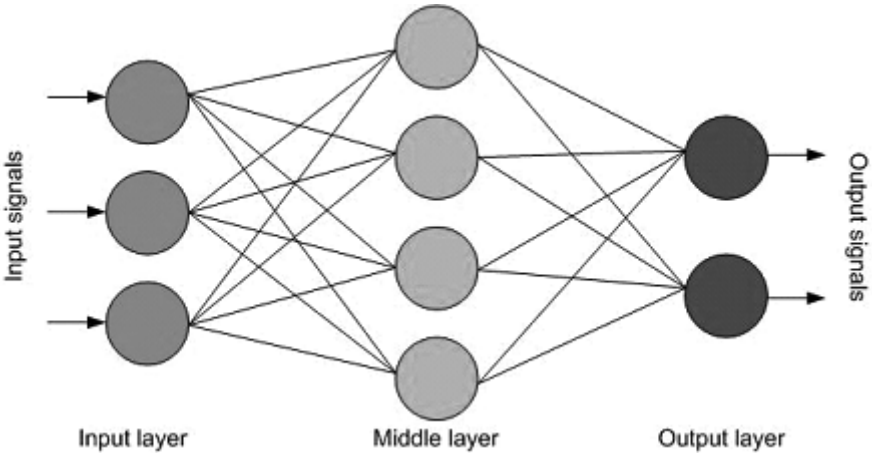

Fig. 5.2. Schematic representation of a generic ANN

Each element (analogous to a neuron) in the network is connected to its neighbors with weights (analogous to synapses) that represent the strengths of the connections. Typically, a single processing element receives a number of inputs (analogous to dendrites) through its connection, combines them, performs a (non-)linear operation on the result, and then produces

the final result (analogous to an axon). The input can be information from external environments or outputs of other neurons. The output can be either a final solution to the problem or an input to other neurons. Figure 5.3 illustrates a neuron model, and Table 5.2 shows that the artificial neural network concepts are similar to those of the biological brain.

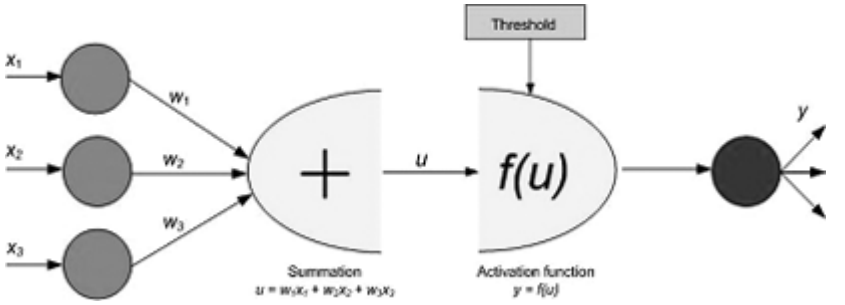

Fig. 5.3. A neuron model

The neuron determines its output on the basis of the weighted sum of the inputs, a threshold value (θ), and an activation function. An activation function of a neuron can be any mathematical function. In practice, four functions are commonly used. They are step function, sign function, sigmoid function, and linear function. If one chooses a sign function as an activation function and the net input is less than θ, the neuron output is 1; otherwise, it is -1.

Table 5.2. An analogy between the biological and artificial neural networks and the functions of their components

Function of each component	Biological neural networks	Artificial neural networks
Accept Inputs	Dendrite	Input
Process the inputs	Soma	Neuron
Turn the processed inputs into outputs	Axon	Output
Involve learning process	Synapse	Weight

To build an artificial network, one must decide which network architecture and learning algorithm should be used. Network architecture tells how the neurons are used, and how they are connected in a network. The aim of the learning function is to modify the weights of the inputs to achieve the desired outputs.

Based on the arrangement of the internal nodes in the network layer, the neural network architecture can be classified into different types, namely, perceptron, feedforward networks, and feedback networks. The simplest type of neural network is a perceptron (Rosenblatt, 1958). It consists of a single layer wherein weights are trained to produce a correct output when presented with inputs. The perceptron is typically used for class classification, where the classes are linearly separable, regardless of the type of activation function. If the classification problem is not linearly separable, the perceptron cannot perform classification correctly. Therefore, perceptrons are suitable only for simple problems in pattern classification. The limitations of the single layered perceptron were mathematically analyzed. The outcome of this analysis was the multilayer perceptron (Minsky and Papert, 1969).

The multilayer perceptron expands the basic single layer network by having one or more *hidden layers*. In the multilayer structure, the input layer accepts information from the external environment and passes the information to all units in the first hidden layer. The outputs from the first hidden layer are redistributed to the next hidden layer, and so on. The output layer accepts output from the last hidden layer and generates the final output of the entire network.

A feedforward network is a network of neurons that have signals traveling from input layer to output layer only. In contrast, feedback networks allow signals traveling in both directions (from input layer to output layer and vice versa). A type of feedback network is a recurrent neural network.

One important function of the human brain is to collect down and recall the memories. This is done with short and long term memories. The human memory is associative, that is, people recognize an input pattern by comparing it with patterns stored in their memories. If the input pattern is noisy, the associative memory returns the closest stored pattern. In other words, if a corrupted image is given to a network, the network will automatically reconstruct a perfect image. A *recurrent neural network,* a variation of the multilayer perceptron, is able to emulate the associative characteristics. It is a modification to the multilayer neural networks, trained with the backpropagation algorithm; that is, a recurrent neural network has feedback loops from its outputs to its inputs. As in backpropagation learning, the feedbacks are used to adjust the weights of inputs. Then the output is computed again. The algorithm is repeatedly iterated until output becomes convergent.

Learning in neural networks can be divided into two types: supervised and unsupervised learning. In supervised learning, an artificial neural network is trained by an external teacher who presents inputs, weights, and

desired outputs to a network. Weights are randomly initialized to the inputs of the network to compute the actual outputs. The actual outputs are compared to the desired outputs. The weights are then adjusted by the network to produce actual outputs that are close to the desired outputs. The input weights are continuously modified until acceptable actual outputs are achieved. In contrast, unsupervised learning, also known as self-supervised learning, does not require an external teacher. During the training phase, a neural network receives a number of inputs, discovers regularities in the inputs, and learns how to organize itself.

With remarkable abilities such as nonlinearity, adaptive learning, self organization, real-time operation, very large-scale integrated implementation, and fault tolerance via redundant information coding, neural networks are able to solve complex problems that human and other computer techniques cannot do. For example, neural networks outperform the decision tree approach on the same data. However, neural networks have some limitations. For instance, complex neural network models lack explanations to interpret the decisions of each node in the network as rules; as testing and verification. This problem comes from adaptive learning capability, in which a network learns how to solve problem by itself, and its operations cannot therefore be interpreted.

The neural network is one of several machine learning approaches that have been successfully applied to solving a wide variety of bioinformatics problems. In sequence analysis, ANNs have been applied or integrated with other methods or systems. For example, a knowledge-based neural network system was applied to analyzing DNA sequence (Fu, 1999). An artificial neural network was trained to predict the sequence of the human TP53 tumor suppressor gene based on a p53 GeneChip (Spicker et al., 2002). A multilayered feed-forward ANN was developed as a tool to predict a mycobacterial promoter sequence in a nucleotide sequence (Kalate et al., 2003).

There are two popular gene finder tools that accommodated ANNs. GRAIL (Uberbacher and Mural, 1991) is the first gene finder program, which was designed to identify genes, exons, and various features in DNA sequences. It uses a neural network that combines a series of coding prediction algorithms to recognize coding potential in fixed length windows without looking for additional features. Figure 5.4 shows a snapshot of the GRAIL tool screen.

Another gene finder system is GeneParser (Synder and Stormo, 1993, 1997). It was designed to identify and determine the fine structure of protein genes in genomic DNA sequences. It comprises two variations of a single layer network, namely, 1) one fully connected and one partially connected with an activation bias added to some inputs, and 2) a partially

connected two-layer network. Dynamic programming has been used as the learning algorithm to train the system for protein sequencing.

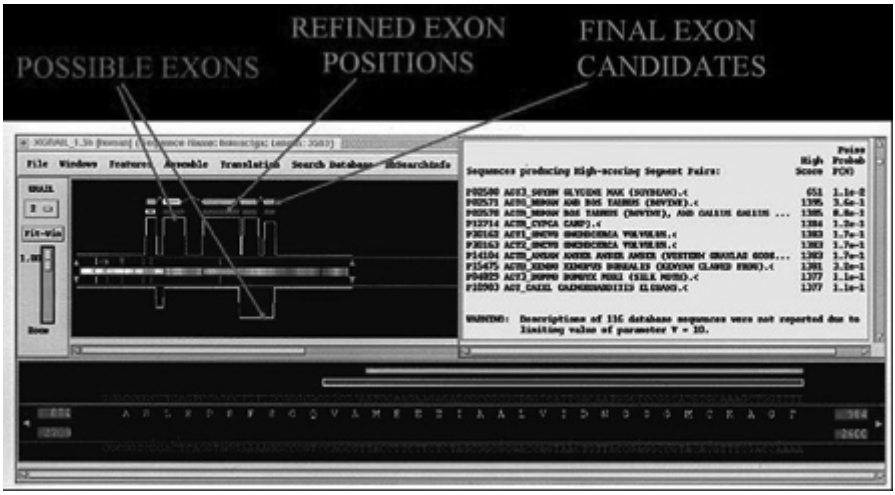

Fig. 5.4. GRAIL gene finding tool (Source from: Presentation slides appeared in Bioinformatics Forum, Thailand. 2002)

ANN has been widely used in protein structural and functional prediction. The prediction of protein secondary structure using neural networks was formerly carried out in 1988 (Bohr et al., 1990, and Qian and Sejnowski, 1988). However, this has requirements of training several neural networks and of adding an extra layer. Much work has been done to improve the effective methods (Baldi, 2000; Fairchild et al., 1995; Riis and Krogh, 1996; and Rost and Sander, 1993). Most of the recent methods use ensembles of neural networks.

ANN has also been used to carry out expression analysis. An artificial neural system for gene classification called GenCANS was developed to analyze and manage a large volume of molecular sequencing data from the Human Genome Project (Wu, 1993, 1996; and Wu et al., 1992). GenCANS is based on a three-layered feed-forward backpropagation network. GenCANS was initially designed to classify unknown sequences into known classes. There are two extensive works of GenCANS – GenCANS-RDP (Wu and Shivakumar, 1994) and GenCANS-PIR (Wu, 1995). GenCANS-RDP is the RNA classification system which groups a number of small subunit ribosomal RNAs together based on RDP (Ribosomal Database Project) phylogenetic classes. GenCANS-PIR is the protein classification system which currently classifies protein sequences into more than 3,300 PIR (Protein Identification Resource) superfamilies.

Unsupervised learning neural networks can be generally categorized into the following types:

- self-organizing map (SOM) (Golub, 1999; Tamayo et al., 1999; Toronen et al., 1999),
- self-organizing tree (SOTA) (Herrero et al., 2001), and
- adaptive resonance theory (ART) (Azuaje, 2003, and Tomida et al., 2001).

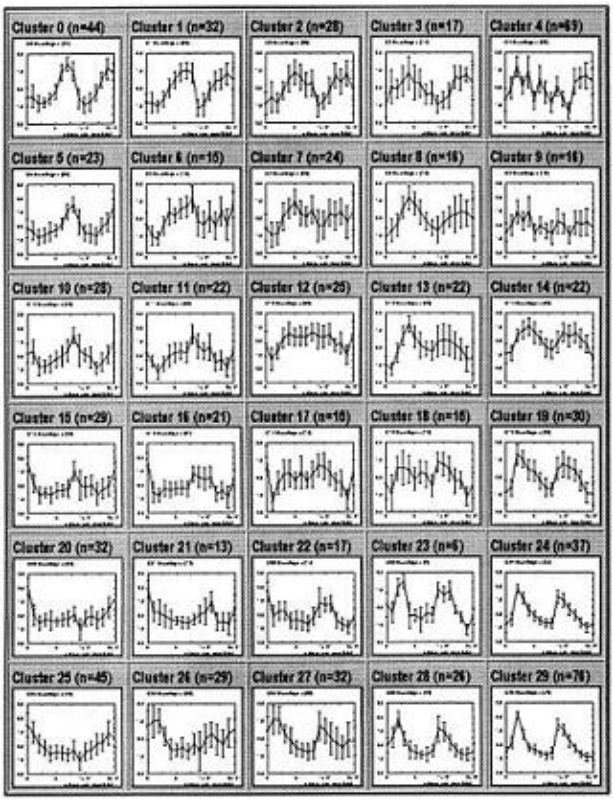

Fig. 5.5. The 828 genes of yeast cell cycle were grouped into 30 clusters (source: Tamayo et al., 1999)

They have been used to analyze gene expression data. ART was used to show that unsupervised learning neural network tools outperform for the analysis and visualization of gene expression profiles. Figure 5.5 shows an example result of applying SOM to analyze gene expression data.

5.3 Neural Network Architectures and Applications

Neural networks are parallel and distributed information processing systems that are inspired by and derived from biological learning systems such as the human brain. The architecture of neural networks consists of a network of nonlinear information processing elements that are normally arranged in layers and executed in parallel. This layered arrangement is referred to as the topology of a neural network. The nonlinear information processing elements in the network are called neurons, and the interconnections between these neurons in the network are called synapses or weights. A learning algorithm must be used to train a neural network so that it can process information in a useful and meaningful way. Neural networks are used in a wide variety of applications in pattern classification, language processing, complex systems modeling, control, optimization, and prediction (Lippman, 1987). Neural networks have also been actively used in many bioinformatics applications such as DNA sequence prediction, protein secondary structure prediction, gene expression profile classification, and analysis of gene expression patterns (Wu and McLarty, 2000). In this section, we provide a review of neural network applications in bioinformatics that accommodates the most recent advances.

A review of neural network architectures and learning algorithms is briefly presented in the next section. This is followed by a review of applications of neural networks in bioinformatics. The reviewed applications are then compared and categorized based on the areas of application.

5.3.1 Neural Network Architecture

Feed-Forward Neural Networks

As discussed in section 5.2, a perceptron is the most basic and the simplest feed-forward neural network model. It consists of an input layer and a single output layer of processing units called nodes. Input values presented to neurons in the input layer are mapped directly to neurons in the output layer. There are no intermediate processing steps. Each input is associated with a weight to reflect the significance of the input to the output. Given a set of training patterns that consist of exemplar "input" and "desired output" pairs, the perceptron is trained by feeding the input patterns to it and minimizing the error between its outputs and the desired outputs. Since the perceptron performs a direct mapping of input to output, it is a linear classifier, because only its weights define a hyperplane that divides input space into regions of pattern classes. The perceptron, therefore is, incapable of

performing tasks that require nonlinear mappings between input and output.

For more complicated problems, a linear hyperplane is not good enough as a separator. A nonlinear surface that separates the classes is used instead. This can be achieved by the multi-layer perceptron (MLP), or the feed-forward network that consists of three layers of nodes, or neurons. Besides having an input layer and output layer, MLP has one (or several) hidden layer(s) in the middle. All artificial neural networks have a similar structure or topology, as shown in Fig. 5.6.

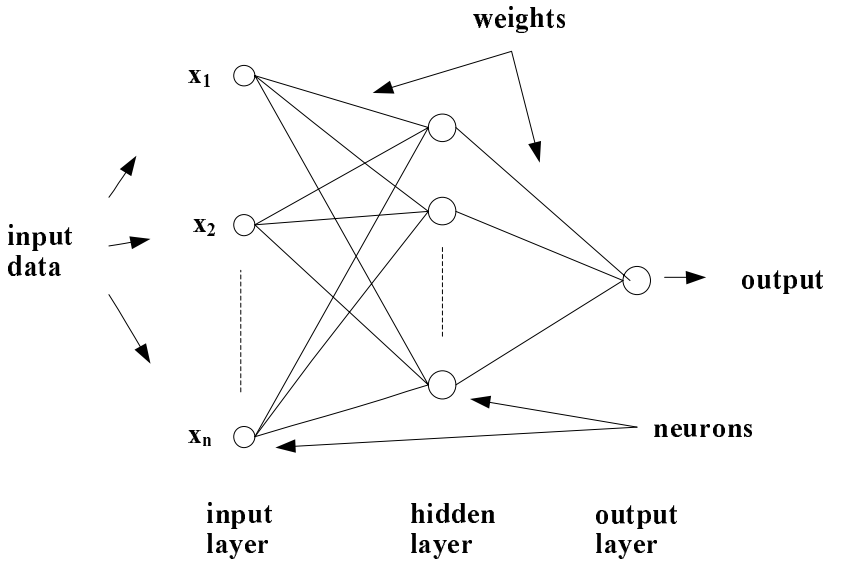

Fig. 5.6. The architecture of a multi-layer perceptron

Input data is a long continuous-valued vector that contains n elements, $x = (x_1, x_2,\ldots,x_n)$. The n elements can be considered as the lengths of the inputs, and are determined by the problem specification. Each hidden neuron ($i = 1, 2,\ldots,m$) stores an exemplar training sample faithfully as its weight vector $w = (w_{i,1}, w_{i,2},\ldots,w_{i,n})$. A hidden neuron i is computed from the inputs

$$h_i = F(\sum_n w_{i,n} x_n) \tag{5.1}$$

where x_n denotes the nth input and $w_{i,n}$ denotes the weights between the input and hidden layers.

The hidden neurons are then used as inputs for the output y

$$y_i = G(\sum_n v_{i,n} h_n) \tag{5.2}$$

where $v_{i,n}$ denotes the weights between the hidden and output layers. The activation function F or G is a sigmoid or logistic function which is usually differentiable and contributes to stability in neural network learning (Narayanan et al., 2003a).

Despite the simplicity of neural network, the summation functions can be more complex than just the simple sum of the products of inputs and their weights. The specific algorithm to combine neural inputs is determined by the chosen network architecture and hypothesis.

Training of Feed-Forward Neural Networks

Once a network has been structured for a problem specification, training of the network is the next step to be followed. The training of the network is nothing but finding the weights to minimize possible error. The initial weights are allocated randomly. Then, the training, or learning, begins. The commonly used algorithm for error is defined by

$$E = \frac{1}{2}\sqrt{\sum_i (t_i - O_i)^2} \tag{5.3}$$

,

where t_i is the target output and O_i is the actual output. The steps used to find the weights for minimizing error are:

- choose the initial weights randomly for a sample input values,
- compare the actual output value with the target output value,
- calculate the error, and
- modify the weights so that the actual output is closer to the target output next time, with smaller error.

This process is repeated for all samples in the dataset and results, and then repeated until the output error for all the samples achieves an acceptable low value, which indicates the end point of the training. Once the training is finished, testing can be done using the rest of the data set, not used during the training phase, to test the trained neural network. If the testing is not satisfactory, further modification of the weights has to be

done. Otherwise, the output value of the tested data is preserved for any decision making.

5.3.2 Neural Network Learning Algorithms

There are many different types of neural networks. Based on the type of learning, they can be categorized into supervised and unsupervised neural networks.

Supervised Learning Neural Networks

Most neural networks are trained with supervised training algorithms. This means that the desired output must be provided for each input used in the training. In other words, both the inputs and the outputs are already known. In supervised training, a network processes the inputs and compares its actual outputs against the expected outputs. Errors are then propagated back through the network, and the weights that control the network are changed. This process is repeated until the errors are minimized. This means that the same dataset is processed many times while the weights between the layers of the network are being refined during the training of the network. Figure 5.7 demonstrate the architecture for a supervised neural network that includes three layers, namely, input layer, output layer and, a hidden layer in the middle.

Support vector machines (SVMs) are considered supervised computer learning methods. Since the support vector machine (SVM) is well known as a training algorithm for learning classification from data, SVMs, as one of the major supervised neural networks, are widely used for the applications of classification and pattern recognition problems in bioinformatics (Vapnik, 1995, and Cristianini and Shawe-Taylor, 2000).

The theory of SVMs can be applied to the clustering of yeast microarray expression data. When the misclassification rates of SVMs are compared with those of other machine learning approaches, SVMs are found to be the best performing methods (Brown et al., 2000). In addition to their use for evaluating microarray expression data, SVMs have been shown to perform well in multiple areas of biological analysis, including detecting remote protein homologies (Jaakkola, 1999) and recognizing translation initiation sites. SVMs can also be used to analyze expression data (Furey et al., 2000). Gene expression data is usually high dimensional data that constitutes a serious problem in several machine learning methods. Dimensionality reduction can be used, but it leads often to information loss and

performance degradation. Fortunately, SVMs can overcome this problem as they can generalize high dimensional data well (Valentini, 2002).

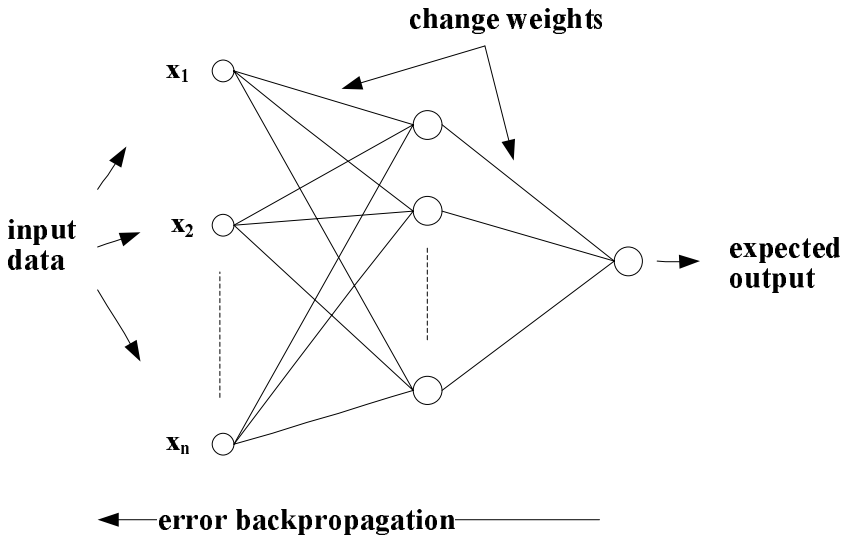

Fig. 5.7. A sample structure of supervised neural network

Unsupervised Learning Neural Networks

The learning algorithm used in unsupervised neural networks is an unsupervised learning algorithm. In unsupervised training, the network is provided only with inputs, while the expected output is unknown. The neural network must itself choose features to group the input data without being trained (Agatonovic-Kustrin and Beresford, 2000). Once an unsupervised neural network has been trained, it must be tested to show that the network really represents the data; the data is expected to be well represented in clusters.

A self-organizing network known as self-organizing map (SOM), or Kohonen network, is the most common algorithm used in unsupervised neural networks (Kohonen, 1982). It is different from the supervised learning described earlier. The neighborhood of a neuron is used to find and group the data that has the similarity. The grouped neurons are arranged in a matrix pattern called a map. Every input neuron is connected to other neurons in this map. Finally, these neurons form the output of the neural network, as shown Fig. 5.8.

The SOM consists of an input layer and a competitive output layer. The output layer is normally organized into a two-dimensional grid of fully connected neurons, as illustrated in Fig. 5.8. The input vectors are fed into input layer and mapped with competitive neurons in the output layer. The competition learning algorithm in the output layer ensures that similar input vectors are mapped with competitive neurons that are closer to each other in the grid than dissimilar ones. In SOM, input vectors in high dimensional space are, therefore, projected on to two-dimensional output space based on their spatial similarities. Similar input patterns are clustered into one small region in the grid of the output layer.

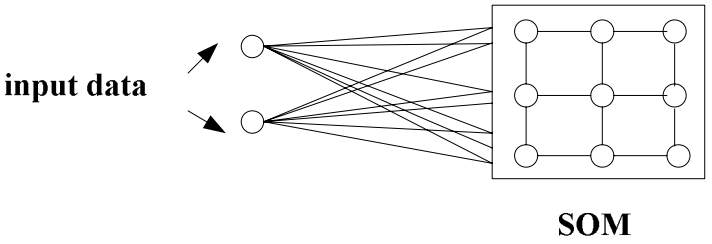

Fig. 5.8. Self-organizing map (adapted from Narayanan et al., 2003a)

The SOM is widely used as a data mining and visualization method in bioinformatics. It is a more robust and accurate method for the clustering of large amounts of noisy data than hierarchical clustering methods are for analyzing the gene expression data. In the analysis of the Stanford yeast gene expression dataset using SOMs, the best performance of gene expression analysis was the result of combining clustering and visualization methods (Torkkola et al., 2001). SOMs, can be used to reduce the amount of data through clustering, and to construct a nonlinear projection of the data onto a low dimensional display simultaneously. Therefore, SOMs can be used to combine aspects of gene analysis, namely, clustering and visualization.

Nevertheless, this approach presents several problems (Fritzke, 1994). They are as follows:

- As the SOM is a topology-preserving neural network, the number of clusters is randomly fixed from the beginning. Therefore, the clustering obtained is not proportionate.
- The lack of a tree structure makes it impossible to detect higher order relationships between clusters.

The hierarchical clustering and the SOM can be combined to surmount the problems faced by these methods in analysing the gene expression profiles and the gene expression data from DNA array experiments (Herrero et al., 2001, and Dopazo and Carazo, 1997).

The advantages of SOTA are as follows:

- the clustering obtained is proportional to the heterogeneity of the data
- the binary topology produces a nested structure in which nodes at each level are averages of the items below them.

An alternative way to avoid the problems is to use Fuzzy Kohonen Neural Networks that combines a Kohonen network and a fuzzy c-means algorithm to keep the advantages and overcome the shortcomings of both techniques (Granzow et al., 2001).

The advantages of the SOM can be attributed to its ability to map high dimensional data onto more comprehensible lower dimensional space and to its fast execution. It is potentially very useful for dealing with high dimensionality and large-scale databases to extract information from gene expression data. However, the effectiveness of its combining with database queries warrants further investigation. SOM also has limitations, namely, 1) no convergence guarantee and 2) the nondeterministic results that depend on learning rates.

5.3.3 Neural Network Applications in Bioinformatics

Neural networks have been widely used in biology since the early 1980s (Brusic and Zeleznikow, 1999). They can be used to

- predict the translation initiation sites in DNA sequences (Stormo et al., 1982).
- explain the theory of neural networks using applications in biology (Baldi and Brunak, 1998).
- predict immunologically interesting peptides by) combining an evolutionary algorithm (Brusic et al. 1998a)
- study human TAP transporter (Brusic et al., 1998b).
- carry out pattern classification and signal processing successfully in bioinformatics; in fact, a large number of applications of neural network can be found in this area
- perform protein sequence classification; neural networks are applied to protein sequence classification by extracting features from protein data

and using them in combination with the Bayesian neural network (BNN) (Wu et al., 1993, 1995, 1997, 2000).
- predict protein secondary structure prediction (Qian and Sejnowski, 1988).
- analyze the gene expression patterns as an alternative to hierarchical clusters (Toronen, Kolehmainen, et al., 1999; Wang, Ma, et al., 2000; Bicciato, Pandin, et al., 2001; and Torkkola, Gardner, et al., 2001); gene expression can even be analyzed using a single layer neural network (Narayanan, Keedwell, et al., 2003b).

In summary, a neural network is presented with a pattern on its input nodes, and the network produces an output pattern based on its learning algorithm during the training phase. Once trained, the neural network can be applied to classify new input patterns. This makes neural networks suitable for the analysis of gene expression patterns, prediction of protein structure, and other related processes in bioinformatics..

5.4 Genetic Algorithm

The genetic algorithm is an artificial system based on biological evolutionary mechanisms (Holland, 1975). A modern biological evolutionary theory came into existence by incorporating genetics and population biology theory into the classical evolution theory of Charles Darwin (Darwin, 1859). It can be defined as the inheritable changes, via genetic materials in a population of chromosomes, from one generation to the next generation. The main goal of evolution is to produce a population of chromosomes with increasing fitness. The fitness is a quantitative measure of the success of a chromosome in survival and reproduction. The main processes of natural evolution are reproduction of some chromosomes within a population, mutation in the DNA sequence within a gene or chromosome of an organism to create a new character not found in the parental type, and competition and fitness selection to limit expanding populations of different species in finite space.

The recombination (or crossover) first occurs during reproduction, resulting in the combination of genes from parents to form a new chromosome. The new chromosome, which consists of genes or blocks of DNA, can be mutated. The mutation can be caused, for example, by errors in copying genes from parents. These errors change the gene's position (or locus) in the chromosome, and this change is called genetic variation. But one might question which parents should be chosen to form a new chro-

mosome. Naturally, parental chromosomes are selected to produce new chromosome according to the fitness of the genotypes. A chromosome with higher fitness has a higher chance than other chromosomes of being selected to reproduce. Natural evolution is a gradual, continuous, and never ending process.

The biological evolutionary theory inspired computer scientists to develop an intelligent system that is capable of imitating the principles of natural evolution.

An automatic mechanism to adapt and learn is desirable for producing good solutions. This is the starting point of a genetic algorithm.

The genetic algorithm is a search algorithm that operates on pieces of information. It is similar to a natural evolutionary process that operates on the information stored in genes. In the genetic algorithm, chromosomes are represented as binary strings; these strings are modified in the same way that populations of chromosomes evolve in nature. The population of strings improves its fitness over interactions, and after a number of generations the population finally evolves to the best solution for a given problem. In each generation, all strings are evaluated by a fitness function for their performance. Based on these evaluations, a new population of strings, with well adapted effectiveness, is formed by using genetic operators such as selection, crossover, and mutation.

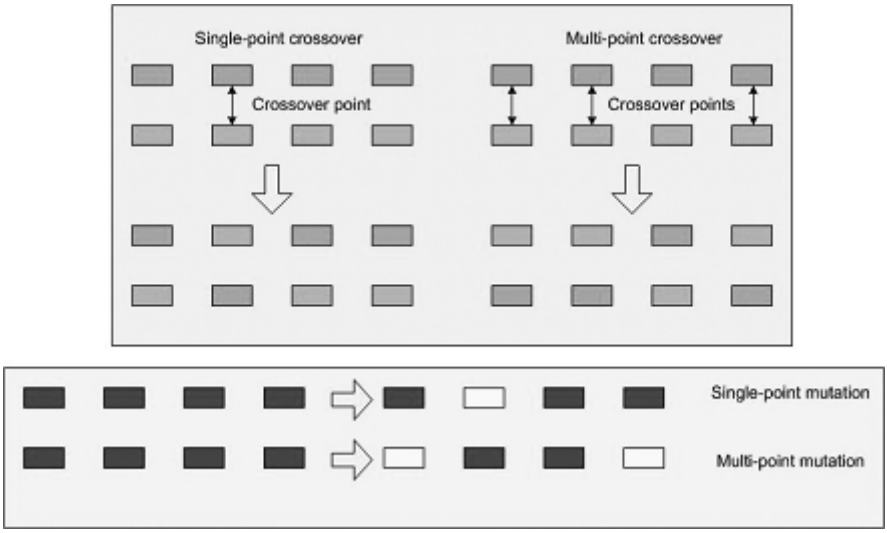

Fig. 5.9. The illustration showing how crossover and mutation operators work

The selection operator selects the as many survival chromosomes as possible from a given population based on their fitness values. The aim of

the selection is to increase the occurrence of fitter chromosomes in the population over subsequent generations. There exist a wide number of selection techniques (Forrest, 1985; Goldberg and Deb, 1991; and Grefenstette and Baker, 1989); however, a detailed discussion on the selection techniques is beyond the scope of this book. Further reading can be found in any genetic algorithm book.

The crossover operator breaks and then swaps some parts of two parental chromosomes. The mutation operator represents a mechanism through which a randomly chosen gene (or several genes) is (are) changed to some other gene. The mutation introduces diversity to the population and guarantees that the population is not trapped at a local maximum. Figure 5.9 shows a typical case of crossover and mutation operators.

As mentioned before, the genetic algorithm simulates the process of natural evolution. Analogies between the two can be found in Table 5.3.

Table 5.3. A comparison of the genetic algorithm and the natural evolutionary mechanism

Natural evolution	Genetic algorithm
Environment	Given problem
Chromosome	Binary string
Fitness of phenotype (probability of survival)	Fitness function
Locus	A position on the string
Selection, recombination, crossover, and mutation	Genetic operators
A population of chromosomes that suits to the environment	The optimal solutions to a given problem

The genetic algorithm is a simple computational model compared to the natural mechanism; however, complex and interesting structures have been developed using genetic algorithms. Most genetic algorithms consist of the following steps (Coley, 1999; Ghanea-Hercock, 2003; and Goldberg, 1989):

Step 1 (a) Encode the problem variables as a chromosome, representing a fixed-length binary string.
(b) Choose a population size, N.

(c) Define a fitness function to measure the probability that a chromosome will be selected as a parent chromosome to further generate new chromosomes.

Step 2 Randomly generate a population of chromosomes of size, N.

Step 3 Test each chromosome in the population with the fitness function.

Step 4 Perform the following sub-steps until termination condition such as specified best fitness values, is satisfied.

(a) Select a pair of chromosomes from the population with the higher fitness value as parent chromosomes for reproduction.
(b) Apply the genetic operators to selected parent chromosomes to create a pair of offspring chromosomes.
(c) Allow the offspring chromosomes and their parents to form the new population.
(d) Replace the current chromosome population with the new population.
(e) Calculate the fitness value of each chromosome of the new population.

Step 5 Output the optimal solutions to a given problem as the fittest chromosomes.

Genetic algorithms have a number of advantages.

- A genetic algorithm is a parallel search, that is, in each generation several solutions are checked at once. It generates optimized and robust solutions via powerful operators; for example, bad solutions are filtered out by selection, and local optimal solutions can be avoided by mutation.
- A genetic algorithm can provide good solutions even if very little information about the problem is provided. As a result, genetic algorithms are widely used in classification and optimization.

However there are limitations with the genetic algorithm.

- Encoding a given problem in a suitable representation (for example, bit string) is difficult and often changes the nature of the problem being investigated. Natural evolution does not always produce a good solution. Nor does a genetic algorithm. It frequently converges to a local optimum.
- A genetic algorithm involves several parameters, such as representation, population size, and fitness function. In practice, it is difficult to define

or create these parameters due to the lack of guidelines for choosing them.

It is expected that new developments in genetic algorithms may overcome the limitations.

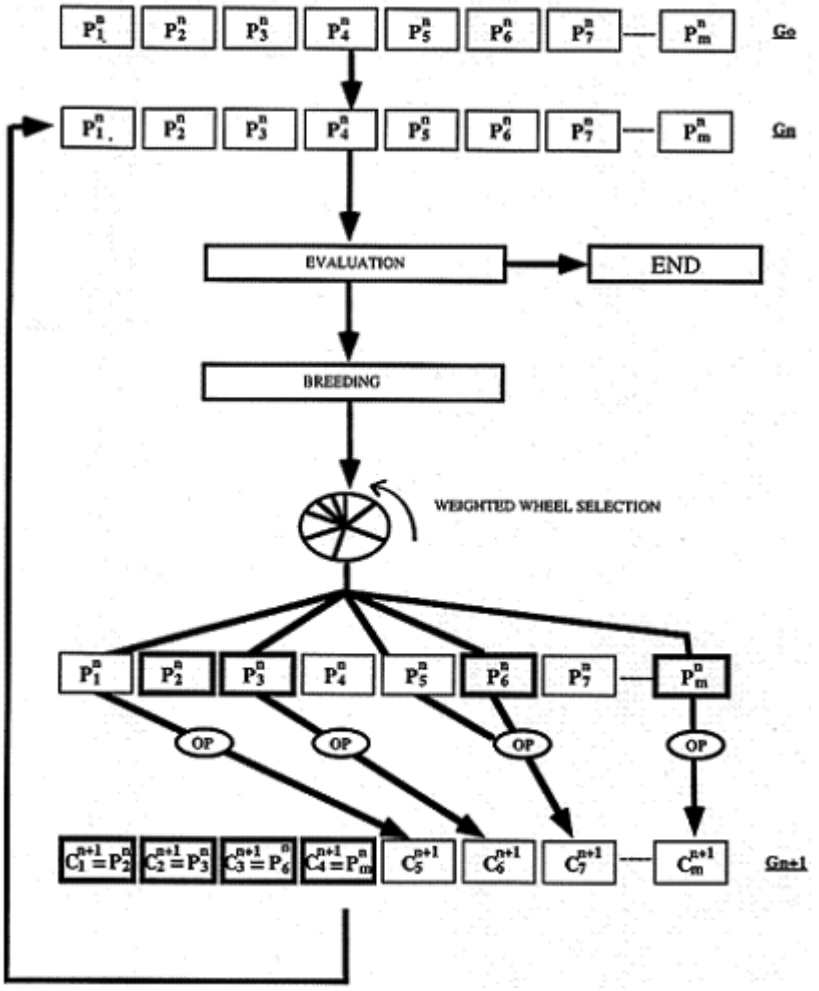

Fig. 5.10. The layout of the SAGA algorithm (Notredame and Higgins, 1996)

The genetic algorithm has been successfully applied for solving many practical problems in many disciplines, in particular, in bioinformatics. Genetic algorithms have been used to solve multiple sequence alignment

problems. One well known approach is SAGA (Ohno-Machado et al., 2002). SAGA randomly creates an initial population of alignments and evolves them in a quasi-evolutionary manner. Through each generation, the fitness of the population is gradually improved. The authors show that SAGA outperforms the most common solution of the multiple alignment problem that uses progressive approach (Barton and Sternberg, 1987; Feng and Doolitle, 1987; and Thompson et al., 1994). The layout of the SAGA algorithm is shown in Fig. 5.10. The first generation initially creates a random population (G_0) consisting of a set of alignments. The subsequent generations are derived from better parents, as measured by multiple alignment quality. When creating children, genetic operators are involved in selecting the better parents, in mixing the contents, and in modifying a single parent. These steps are repeated iteratively to increase the fitness of the population until no more improvement can be made.

In addition to SAGA, there are a few approaches (Chellapilla and Fogel, 1999; Isokawa et al., 1996; Nguyen et al., 2002; Wayama et al., 1995; and Zhang and Wong, 1997) that have applied genetic algorithms to multiple sequence alignment.

Genetic algorithms have been commonly applied to a set of RNA sequences to find common RNA secondary structures (Benedetti and Morosetti, 1995; Chen et al., 2000; Gultyaev et al., 1995; Shapiro and Navetta, 1994; Shapiro et al., 2001; and Wu and Shapiro, 1999). The early proposed methods can deal only with a single RNA sequence, while the latest improved methods can be used to determine RNA structures in RNA sequences.

The trend to use pure genetic algorithms to analyze gene expression data has diminished. The new techniques tend to combine genetic algorithm with other computational methods, such as the K-nearest Neighbor Method (Li et al., 2001) and the neural network (Keedwell and Narayanan, 2003), to solve gene expression problems. They are called neural-genetic hybrid methods. Keedwell and Narayanan use a genetic algorithm to select a set of genes for classification and use a neural network to determine the fitness of the genes.

The steps that are to be followed in neural-genetic hybrid methods can be seen in Fig. 5.11. Preprocessing, to convert each attribute in the dataset into binary field, is the first step. Then, the genetic algorithm initializes a random population of chromosomes. The population becomes the input to the neural network. The network is trained till the desired output (minimum error) is produced. The error from each chromosome acts as a fitness function to determine mutation, crossover, and selection for the next generation of chromosomes. The generation creation process is iterated until

the maximum number of generations is satisfied, that is, until the correct classification of genes is finally discovered.

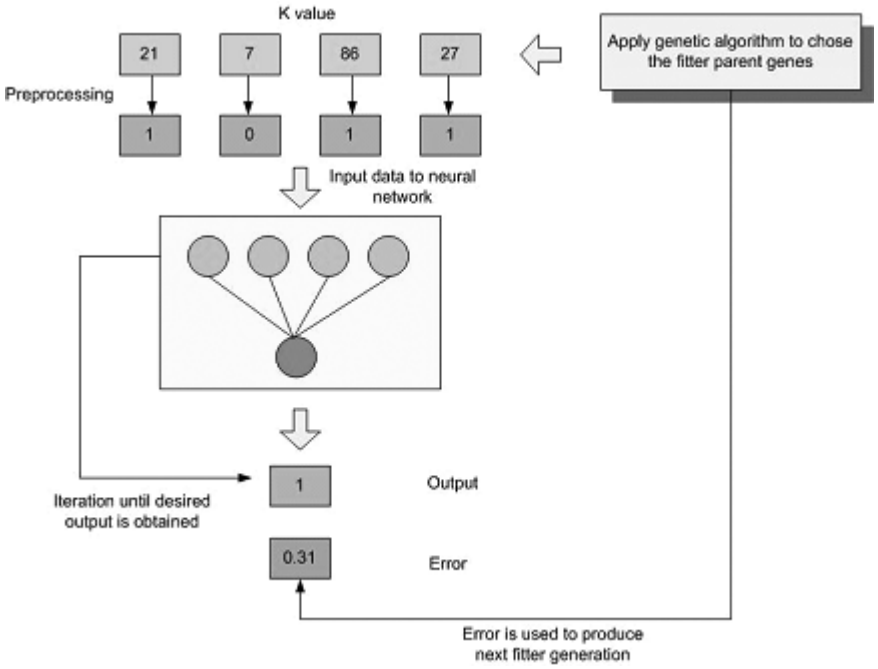

Fig. 5.11. The visual layout of neural-genetic algorithm (adapted from Keedwell and Narayanan, 2003)

5.5 Fuzzy System

A fuzzy system is an expert system that uses a collection of fuzzy membership functions and rules, instead of Boolean logic, to reason about data. It provides a rich meaningful addition to classical logic. The basic concepts of a fuzzy system include fuzzy logic and fuzzy set theory. In order to understand a fuzzy expert system, related terminologies and theories are first explained.

A characteristic of human mind is its ability to reason about vague and ambiguous terms. For example, today may feel hot because the temperature is more than 33°C. If the temperature tomorrow is 31°C, human senses can immediately interpret it as moderately hot. However, a computer with conventional logic cannot replicate that statement. The reason is

that in conventional logic a statement is either true or false, and not multi-valued or partially true or false.

There have been attempts to emulate the way human senses work with computers so that they are able to respond like human. This is the starting point of fuzzy logic. Fuzzy logic is a superset of conventional logic. It can describe partial truth or uncertainty. In fuzzy logic, a true statement can range from completely true through half truth to completely false. In other words, it is possible that a statement is 0.75 true, or not completely true. The multivalued logic has been widely studied since the last century, but the most significant breakthrough was the theory of fuzzy sets proposed by Lotfi Zadeh. Based on fuzzy sets, fuzzy logic can be defined as a set of mathematical principles for knowledge representation based on degree of membership rather than on crisp membership.

A crisp set is a set in classical logic where elements either belong or do not belong to the set, whereas a fuzzy set is a set in fuzzy logic where members have a degree of membership or degree of truth that ranges from 0 to 1. Let us consider Table 5.4 to make things clear.

Table 5.4. Crisp and fuzzy membership

		Degree of membership of "Hot day"	
Day	Temperature, ° C	Crisp	Fuzzy
1	5	0.0	0.0
2	10	0.0	0.0
3	25	0.0	0.1
4	27	0.0	0.3
5	29	0.0	0.5
6	30	0.0	0.8
7	33	1.0	1.0
8	35	1.0	1.0
9	40	1.0	1.0

In crisp set theory, days 1 through 9 fall into only two groups (hot and not hot), depending on their temperatures (°C). If the temperature is less than 33°C, the day is considered not hot. Unlike crisp set theory, fuzzy set theory classifies its members (days) by regarding the degree of truth (hotness). Therefore, there are more than two groups of days. It is a cold day if the temperature is less than or equal 10°C and it is hot day if the temperature is equal to or more than 33°C. For example, days 1 and 2 are hot with

0 degree of truth; days 3 through 6 are hot with 0.10, 0.30, 0.50, and 0.80 degrees of through respectively; and days 7 through 9 are considered hot day with 1 degree of truth. The question is how to find the degree of membership.

In classical set theory, the degree of membership can be calculated by a characteristic function. For example, the crisp set "Hot day" can be defined as

$$f_{Hot\ day}(temperature) = \begin{cases} 1 & if\ temperature \geq 33 \\ 0 & otherwise \end{cases} \tag{5.4}$$

The function $f_{Hot\ day}(temperature)$ maps each temperature value onto 0 or 1. In a fuzzy set, the mapping function, called membership function, maps each temperature value onto the real interval [0, 1]. In the "Hot day" case, the membership function can be defined as

$$\mu_{Hotday}(temperature) = \begin{cases} 1 & if\ temperature \geq 33 \\ 0 & if\ temperature \leq 10 \\ between\ 0\ and\ 1 & if\ 10 < temperature < 33 \end{cases} \tag{5.5}$$

In a manner similar to that of producing crisp and fuzzy sets of "Hot day", crisp and fuzzy sets of "Fine day" and "Cold day", and their degrees of membership, can be obtained as shown in Table 5.5.

Table 5.5. Crisp and fuzzy set temperatures, with members Cold, Fine, and Hot

		Degree of membership					
		Crisp			Fuzzy		
Day	Temperature °C	Cold	Fine	Hot	Cold	Fine	Hot
1	5	1.0	0.0	0.0	1.0	0.0	0.0
2	10	1.0	0.0	0.0	1.0	0.0	0.0
3	25	0.0	1.0	0.0	0.8	0.8	0.1
4	27	0.0	1.0	0.0	0.5	1.0	0.3
5	29	0.0	1.0	0.0	0.3	1.0	0.5
6	30	0.0	1.0	0.0	0.1	0.8	0.8
7	33	0.0	0.0	1.0	0.0	0.0	1.0
8	35	0.0	0.0	1.0	0.0	0.0	1.0
9	40	0.0	0.0	1.0	0.0	0.0	1.0

The characteristic and membership functions that describe how to translate from temperature in value to temperature in words, and vice versa, are shown in Fig. 5.12.

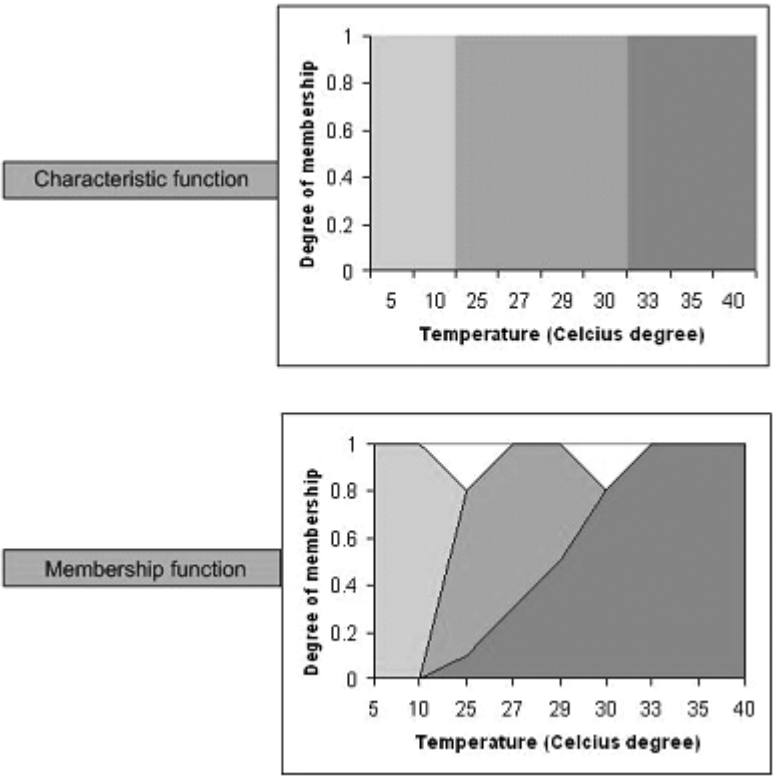

Fig. 5.12. Characteristic and membership functions of Cold, Fine, and Hot day

Another important concept of fuzzy set theory is the linguistic variable. It is used to construct the fuzzy rules. For example, "Temperature" is called a linguistic variable, and "hot", "fine", and "cold" are called linguistic values. A simple fuzzy rule can be represented in the following way:

$$\text{IF} \quad x \text{ is A} \quad \text{THEN} \quad y \text{ is B} \tag{5.6}$$

where x and y represent linguistic variables and A and B are linguistic values. One can use operations, including EQUAL, COMPLEMENT (NOT), CONTAINMENT, UNION (OR), and INTERSECTION (AND), to construct a more complex fuzzy rule such as:

$$\begin{gathered}\text{IF (x is A) AND (y is B) AND (…) OR NOT (…) THEN } z \text{ is } Z \\ \text{IF } x \text{ is A THEN (y is B) AND (…) AND } z \text{ is } Z\end{gathered} \tag{5.7}$$

where *x, y*, and *z* are variables and A, B, and Z are values. The IF part of the rule is called the rule antecedent and the THEN part of the rule is called the rule consequent. If the antecedent part is true with a degree of membership, then consequent part is also true with the same degree. The outputs of fuzzy sets are aggregated into a single. Then the single output is transformed to a single output number. Many researchers have proposed techniques (Mamdani and Assilian, 1975) that facilitate the whole process from the beginning to the end. The most significant technique is fuzzy inference. Fuzzy inference is a tool used to evaluate a knowledge base. It takes a given input and fires an output by using the theory of fuzzy sets. Generally, it consists of four steps: fuzzification of the input variables, rule evaluation, aggregation of the rule outputs, and defuzzification, as shown in Fig. 5.13.

The development of the fuzzy expert system (FES) is an iterative process. A typical process involves the following four steps:

- determine problem input and output variables and their ranges,
- define fuzzy sets and construct fuzzy rules,
- perform fuzzy inference process, and
- evaluate and tune the system.

The basic structure of a fuzzy system can be seen in Fig. 5.14. It consists of four basic components: a fuzzifier, an inference engine, a defuzzifier, and a knowledge base. More details of these components are beyond the scope of this chapter; however, they can be found in general artificial intelligence or soft computing books (Kruse et al., 1994).

Fuzzy systems have been successfully applied to several areas in practice. In bioinformatics, fuzzy systems play an important role for building knowledge-based systems. Most systems involve fuzzy logic-based and fuzzy rule-based models. They can control and analyze processes and diagnose and make decisions in biomedical sciences (Adriaenssens et al., 2004; Boegl et al., 2004; Saritas et al., 2003; Sarkar and Leong, 2003; Schneider et al., 2003; Seker et al., 2003; and Virant-Klun and Virant, 1999).

A fuzzy expert system for the diagnosis of prostate cancer (Saritas et al., 2003), has been explained in detail. The authors use prostate-specific antigen (PSA), age, and prostate volume (PV) as input parameters, and prostate cancer risk as output. The input values are converted to the linguistic variables, with degree of truth. This conversion is done by the membership function. Then, 80 rules are formed. The output of each rule has a degree of truth, obtained from fuzzy operation (MIN and MAX). Finally, the

fuzzy outputs are converted into real output values. Figure 5.15 shows the overview of the FES system and its components.

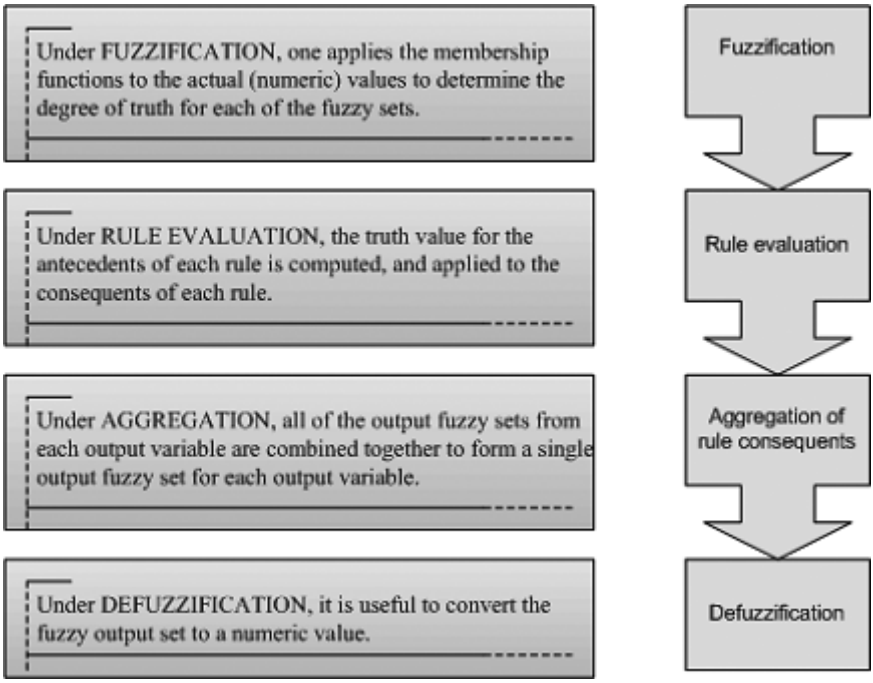

Fig. 5.13. The general steps of fuzzy inference

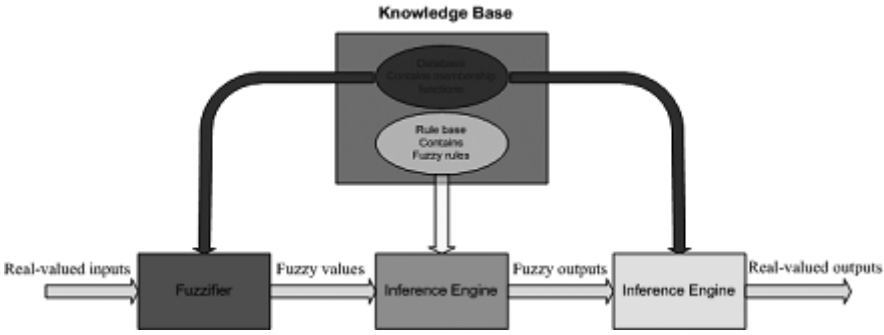

Fig. 5.14. The basic components of fuzzy expert system

In addition, fuzzy logic has been recently applied to analyze (Woolf and Wang, 2000) and to classify (Ohno-Machado et al., 2002) gene expression data. The fuzzy logic-based classifier gives results similar to those of other

classifiers, but are much simpler and easier to interpret (Ohno-Machado et al., 2002). Fuzzy logic accounts for noisy data from a large number of biological patterns.

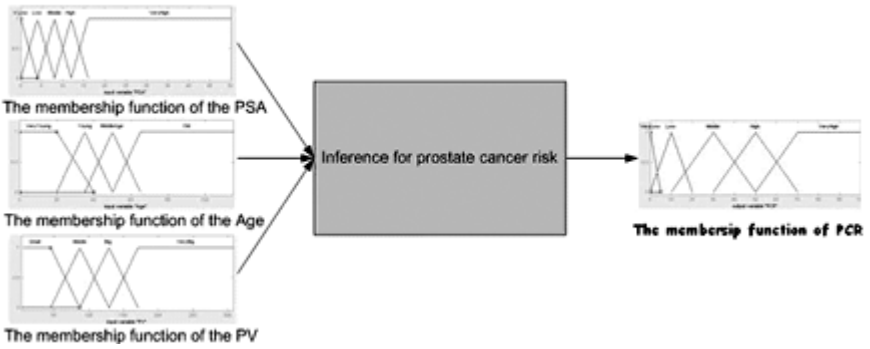

Fig. 5.15. The structure of the fuzzy expert system (FES) for diagnosis of prostate cancer (adapted from Saritas et al., 2003)

References

Adeli, H. (1995) Machine learning : neural networks, genetic algorithms, and fuzzy systems. New York: Wiley.

Adriaenssens, V., Baetsb, B.D., Goethalsa, P.L.M. and Pauwa, N.D. (2004) Fuzzy rule-based models for decision support in ecosystem management. Science of The Total Environment, vol. 319, pp 1-12.

Agatonovic-Kustrin, S., Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. Journal of Pharmaceutical and Biomedical Analysis 22(5): 717-727.

Azuaje, F. (2003) A computational evolutionary approach to evolving game strategy and cooperation. IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 33, pp 498-503.

Baldi, P. and Brunak, S. (2001) "Bioinformatics The Machine Learning Approach", The MIT Press.

Baldi, P., Brunak, S., Frasconi, P., Pollastri, G. and Soda, G. (2000) Bidirectional IOHMMs and Recurrent Neural Networks for Protein Secondary Structure Prediction. In: Protein Sequence Analysis in the Genomic Era, R. Casadio and L. Masotti, Eds. CLUEB, Bologna, Italy.

Baldi, P. and Brunak, S. (1998) Bioinformatics: the Machine Learning Approach. MIT Press.

Barton, G.J. and Sternberg, M.J.E. (1987) A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons. Journal of Molecular Biology, vol. 198, pp 327-337.

http://www.springer.com/978-3-540-20873-0

Bioinformatics Technologies
Chen, Y.-P.P. (Ed.)
2005, XV, 396 p., Hardcover
ISBN: 978-3-540-20873-0