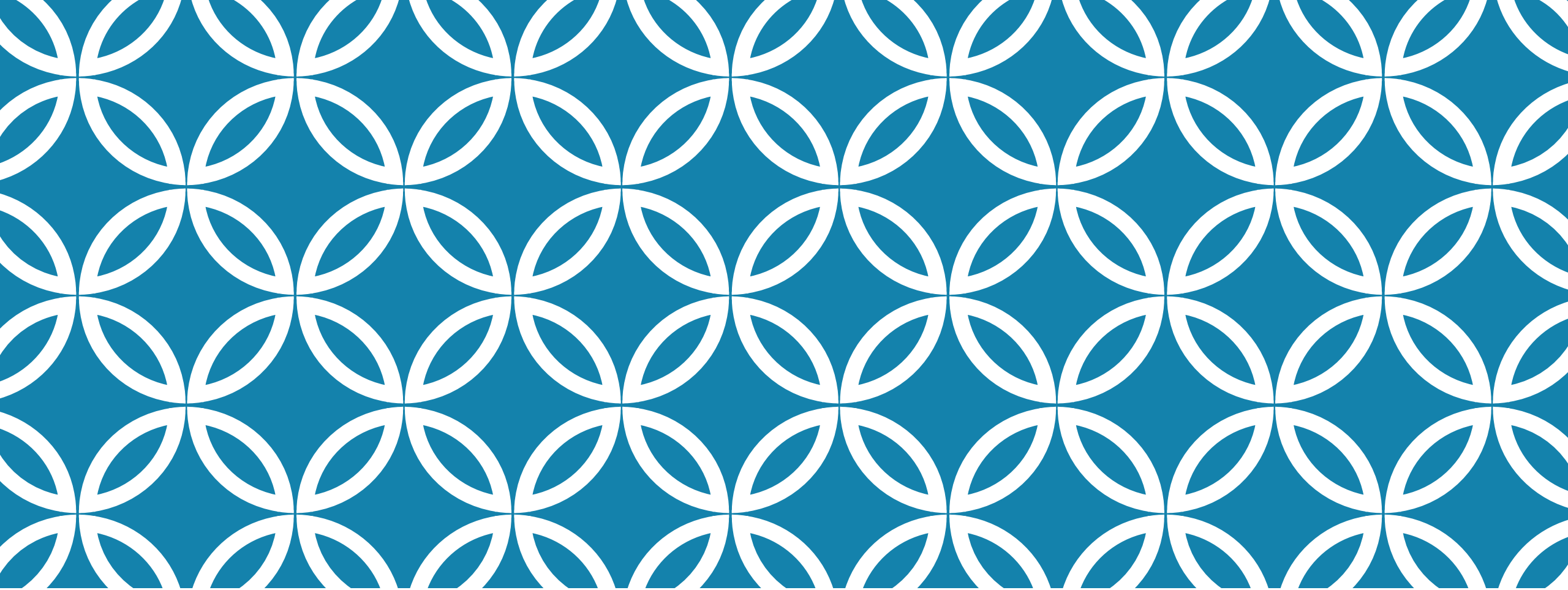


INTRODUCTION TO SOFT COMPUTING

WHAT IS SOFT COMPUTING?

Definition 1: According to Professor Lofti Zadeh, soft computing is “an emerging approach to computing, which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision”

Definition 2: Another commonly used definition of soft computing given by Professor Zadeh states, “The guiding principle of soft computing is to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost”

Definition 3: “Optimization is about finding optimal values for parameters of an object or a system which minimizes an objective (cost) function”, Kasabov.

SOFT COMPUTING VERSUS HARD COMPUTING

S.NO	Soft Computing	Hard Computing
1.	Soft Computing is liberal of inexactness, uncertainty, partial truth and approximation.	Hard computing needs a exactly state analytic model.
2.	Soft Computing relies on formal logic and probabilistic reasoning.	Hard computing relies on binary logic and crisp system.
3.	Soft computing has the features of approximation and dispositionality.	Hard computing has the features of exactitude(precision) and categoricity.
4.	Soft computing is stochastic in nature.	Hard computing is deterministic in nature.
5.	Soft computing works on ambiguous and noisy data.	Hard computing works on exact data.
6.	Soft computing can perform parallel computations.	Hard computing performs sequential computations.
7.	Soft computing produces approximate results.	Hard computing produces precise results.
8.	Soft computing will emerge its own programs.	Hard computing requires programs to be written.
9.	Soft computing incorporates randomness .	Hard computing is settled.
10.	Soft computing will use multivalued logic.	Hard computing uses two-valued logic.

SOME COMPUTATIONAL LIMITATIONS OF HARD COMPUTING

Model complexity and assumptions

Computationally expensive problems

Space complexity

SC APPLICATIONS:

Biometrics

Bioinformatics

Biomedical systems

Robotics

Vulnerability analysis

Character recognition

Data mining

Music

Natural language processing

SC APPLICATIONS:

Multiobjective optimizations

Wireless networks

Financial and time series prediction

Image processing

Toxicology

Machine control

Software engineering

Information management

Picture compression

Noise removal

Social network analysis

ARTIFICIAL INTELLIGENCE

“Artificial intelligence is the simulation of human intelligence on a machine, so as to make the machine efficient to identify and use the right piece of “knowledge” at a given step of solving a problem”, Konar.

“Artificial intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.”, McCarthy.

PROBLEM SOLVING IN AI

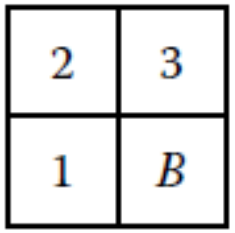
The concepts of AI are not restricted to a single set of algorithms. Rather, they extend to the intelligent use of a large number of tools and techniques and to many more algorithms that are designed per the problem requirements.

The most commonly used and most basic set of algorithms are the graph search algorithms.

A graph is defined as a collection of vertices and edges and may be denoted by G(V, E).

Graph algorithms have many applications in scheduling, planning, game solving, and other such problems.

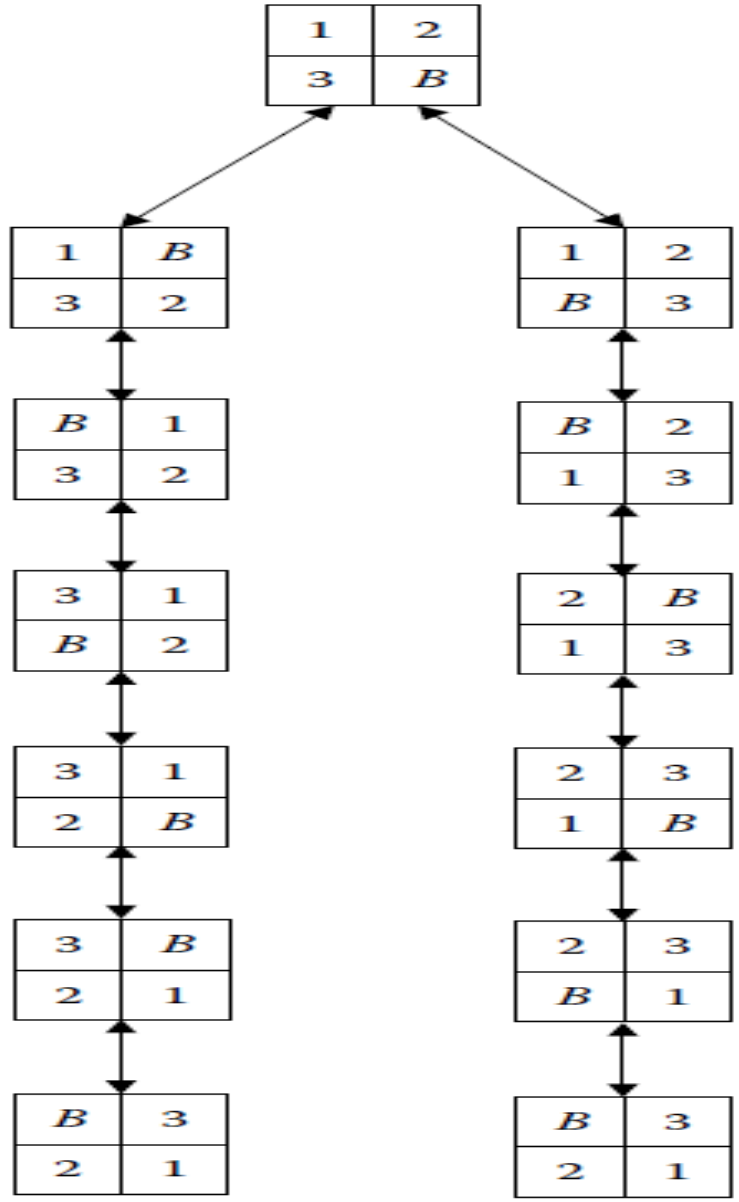
To better understand the concepts of graph and search, let us take an example of the famous four-puzzle problem. In this problem, there are four cells.



The board is shuffled, and our task is to arrange the numbers in a particular order.

Let us say that we denote a board by <P, Q, R, S>, where P is the top left cell, Q is the top right cell, R is the bottom left cell, and S is the bottom right cell.

Let B denote the blank. Possible states are <1, 2, 3, B>, <1, 2, B, 3>, <1, 3, B, 2>, and so on.


1 2
3 B
1 B
3 2
1 2
B 3
B 1
3 2
B 2
1 3
3 1
B 2
2 B
1 3
3 1
2 B
2 3
1 B
3 B
2 1
2 3
B 1
B 3
2 1
B 3
2 1

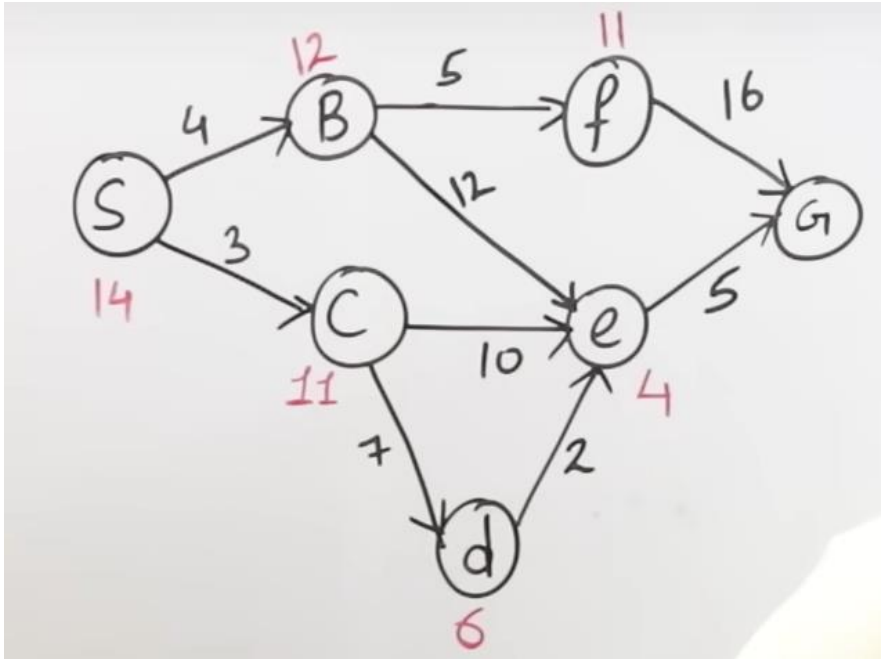
One of the most commonly used algorithms for real life problems is the A* algorithm

The main idea of A* is to evaluate each node based on two parameters:

1. g(n): the actual cost to get from the initial node to node n. It represents the sum of the costs of node n outgoing edges.

2. h(n): Heuristic cost (also known as "estimation cost") from node n to destination node n. This problem-specific heuristic function must be acceptable, meaning it never overestimates the actual cost of achieving the goal.

The evaluation function of node n is defined as f(n) = g(n) + h(n).



Algorithm 1.1

A*(graph, source, goal)
Step 1: closed, open ← empty list
Step 2: Calculate cost of open
Step 3: Add source to open
Step 4: While open is not empty
Do
Step 5: Extract the node *n* from open with the least total cost
Step 6: If *n* = goal position, then break
Step 7: For each vertex *v* in expansion of *n*
Do
Step 8: if *v* is in neither open nor closed list, then compute costs of *v* and add to open list with parent *n*
Step 9: if *v* is already in open list or closed list, then select the better of the current solution or the solution present already and update the cost and parent of the vertex
Step 10: if *v* is already in closed list, then select the better of the current solution or the solution present already and update the cost and parent of the vertex and all children from the vertex
Step 11: Add *n* to closed

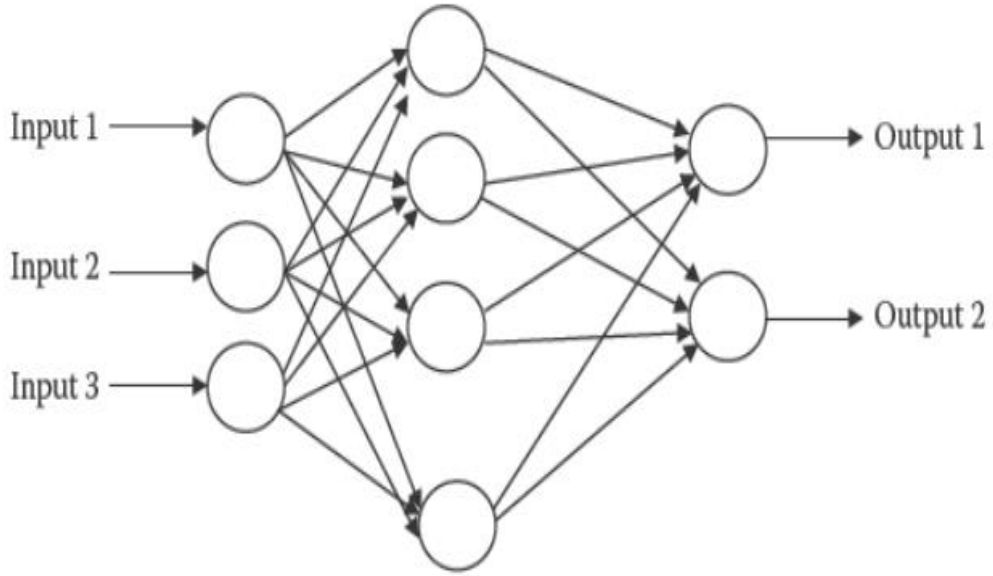
SOFT-COMPUTING TECHNIQUES

1. Artificial Neural Networks:

*The artificial neural network (ANN) is an attempt to imitate the functionality of the human brain. The human brain is believed to be composed of millions of small processing units, called neurons, that work in parallel. Neurons are connected to each other via neuron connections. Each individual neuron takes its inputs from a set of neurons. It then processes those inputs and passes the outputs to another set of neurons. The outputs are collected by other neurons for further processing.

*ANNs are able to learn past data, just like the human brain. The network learns the past data by repeating the learning process for a number of iterations. ANNs learn the data and then reproduce the correct output whenever the same input is applied. The precision and correctness of the answer depends on the learning and the nature of the data given. Sometimes an ANN may refuse to learn certain data items, while other times it may be able to learn complex data very easily. The precision depends on how well the neural network was able to learn the presented data.

*ANNs have another extraordinary capability that allows them to be used for all sorts of applications: Once they have been well taught the historical data, they are able to predict the outputs of unknown inputs with quite high precision. This capability, known as generalization, results from the fact that ANNs can imitate any sort of function, be it simple or complex. This gives ANNs the power to model almost any problem that we see in real life applications.



Furthermore ANNs are highly resistant to noise. Thus, if there is any noise in the input data, ANNs are still able to perform appreciably well. This ability makes it possible to make robust applications for ANNs.

2. FUZZY SYSTEMS

Impreciseness is always prevalent in systems. Observations might reveal that you are never able to make discrete rules over a set that always holds precisely true. Rather the rules are always imprecise in nature. We can never work over definite systems that have definite rules. It is from the same observations that fuzzy systems evolved. The impreciseness that is introduced in the system makes them work in a better way. Fuzzy systems are hence being used to model many real life applications.

Fuzzy systems are implemented by a set of rules called *fuzzy rules*, which may be defined on sets. A nonfuzzy system has very discrete ways of dealing with rules—either a rule fires fully or it does not fire at all, depending on the truth of the expression in the condition specified. On the other hand, in fuzzy systems, the conditions are true to a certain degree and false to a certain degree. The degree of truthfulness and falseness decides the degree to which the rule fires. Hence, every rule is fired to some degree. The outputs of all the rules are aggregated to get the system's final output.

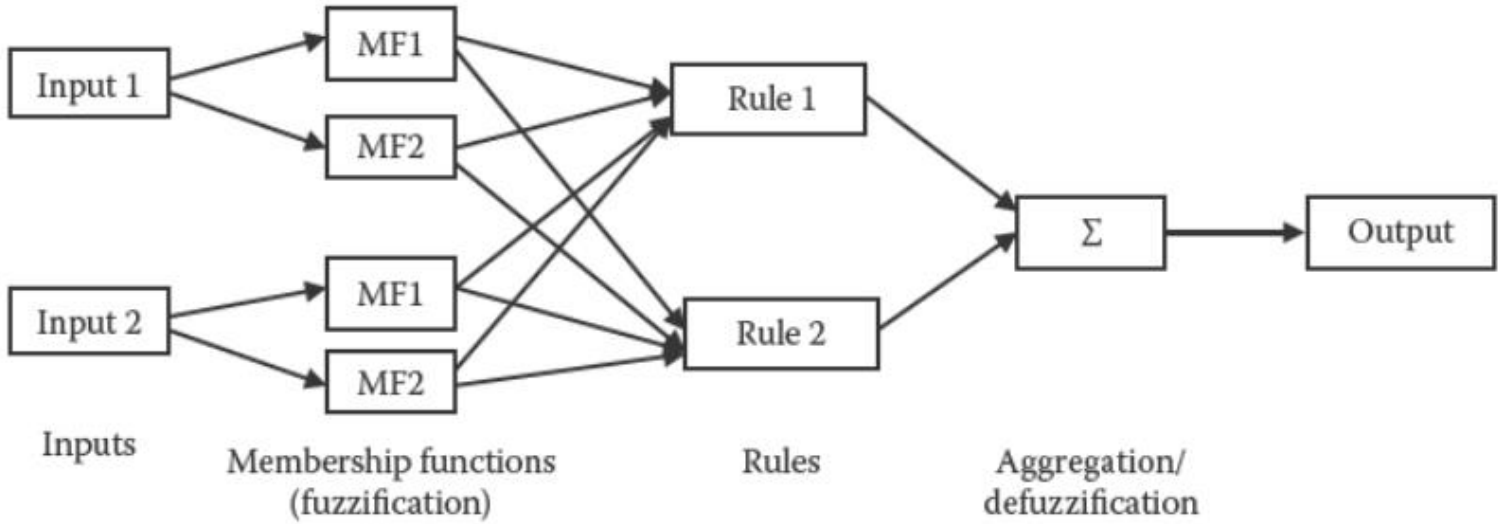


FIGURE 1.4 The architecture of the fuzzy inference system.

The general procedure of working with fuzzy systems consists of input modeling. Then the membership functions are applied over the crisp inputs to get the fuzzified inputs. Then the rules are applied over these inputs to generate the fuzzy outputs. The various outputs are then aggregated and defuzzified to get the final crisp output. This procedure is given in Figure 1.4.

Fuzzy systems are also known as fuzzy inference systems (FIS). These intelligent systems generate outputs from given inputs using fuzzy logic. These systems are defined as "systems formulating the mapping from a given input to an output using fuzzy logic" (MATLAB Guide).

3. EVOLUTIONARY ALGORITHMS

* Natural evolution results from the fusion of male and female chromosomes to generate one or more offspring. The offspring contain mixed characters of both the male and female counterparts.

* The offspring may be fitter or weaker than the participating parents. This process of evolution leads to the development of one generation from the previous generation.

*Over time, the newer generations evolve, and the older ones die out. This survival process goes on and on, following Charles Darwin's theory of survival of the fittest.

*According to this theory, only the characteristics of the fittest individuals in a population go to the next generation. Others die out.

* Evolutionary algorithms work in a similar way.

*The first step is to generate a random set of solutions to the given problem comprising a ***population***.

*These random individuals are then made to participate in the ***evolution*** process.

*From these solutions, a few individuals with high fitness are chosen through a method called ***selection***. These individuals are then made to generate offspring.

* This process of generating offspring by two parents is known as ***crossover***.

* The system then randomly selects some of the newly generated solutions and tries to add new characteristics to them through a process known as ***mutation***.

* Once the new generation is ready, the same process is applied to that **new generation**.

* The fitness of any one solution may be measured by a function known as the **fitness function**.

* As we move through newer and newer generations, we find that the quality of solution continues to improve.

* This improvement is very rapid over the first few generations, but with later generations, it slows.

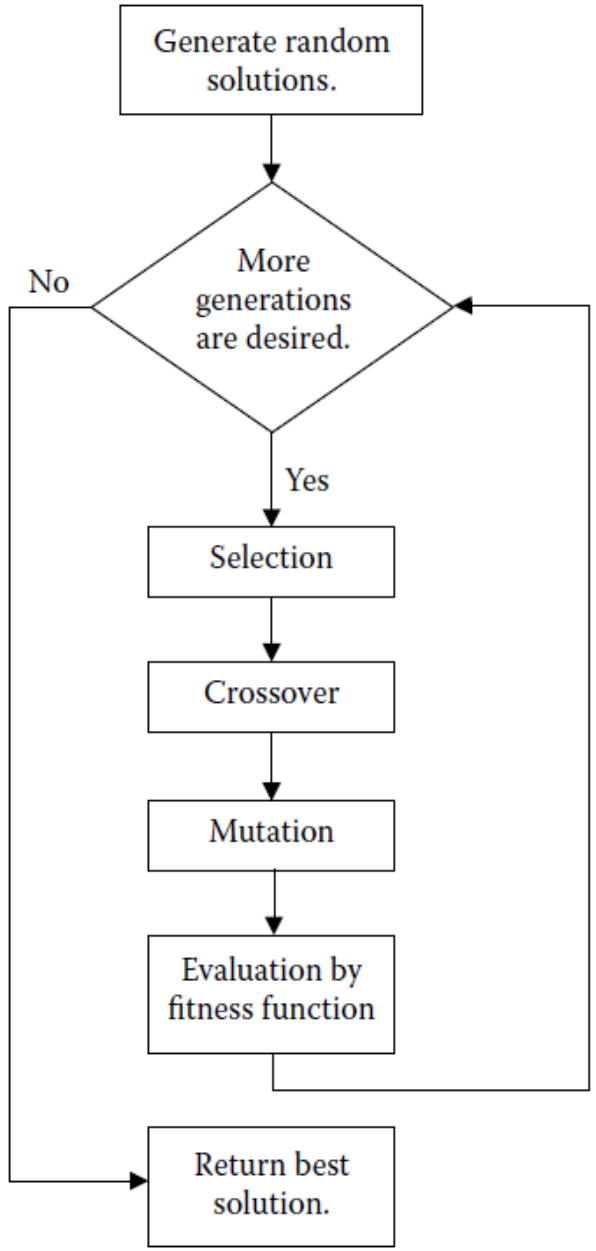


FIGURE 1.5 Simple genetic algoirthm.

4. HYBRID SYSTEMS

* Each system discussed above has some advantages and some disadvantages. In hybrid systems, we try to mix the existing systems, with the aim of combining the advantages of various systems into a single, more complex system that gives better performances.

* Many hybrid systems have been proposed, and many models have been made to better cater to the data. One of the best examples is the fusion of **ANNs** and **FIS** to make **fuzzy-neuro systems**.

* In this hybrid model, a fuzzy system is built over ANN architecture and is then trained using the back propagation algorithm (BPA). These hybrid systems are highly beneficial in real life applications, such as in controllers.

* These controllers primarily use the fuzzy logic approach, but optimization of the various parameters is done using ANN training.

* This combination makes the systems perform with a very high degree of precision. An evolution from this hybrid model is the **adaptive neuro-fuzzy inference system (ANFIS)**, which is similar to the neuro-fuzzy system but is adaptive in nature.

* Another system that is being used in hybrid models are the evolutionary algorithms.

* These systems are being added to ANNs to give better performance and obtain the global minima.

* Eas give the hybrid systems a good performance optimization. They are also being used to make an evolutionary model of ANNs.

* This system starts from a simple ANN; as time continues, the system evolves the ANN, based on performance, by constantly adding neurons.

* These systems are also found to give very good performance.

* In a similar system, EAs evolve the fuzzy system to ensure that the generated system gives a high performance.

EXPERT SYSTEMS

* Expert system (ES) design is motivated by this concept of experts. In this type of design, we try to make systems that can act autonomously.

* Once these systems are presented with the conditions as inputs, they work over the solutions. The solution given by these systems is then implemented.

* These systems are a complete design of whatever it might take to draw conclusion once the problem is given. To reach the conclusions may require preprocessing, processing algorithms, or any other procedures.

* These systems may require a historical database or a knowledge base to learn from.

* Expert systems are being built as an end tool to meet the customer's various needs. These tools behave just like an expert who takes care of all the subject's needs. These tools have made it possible for people to enjoy the fruits of soft computing without knowing anything much about the underlying system.

* An expert system may be defined as “knowledge-based systems that provide expertise, similar to that of experts in a restricted application area” (Kasabov, 1998).

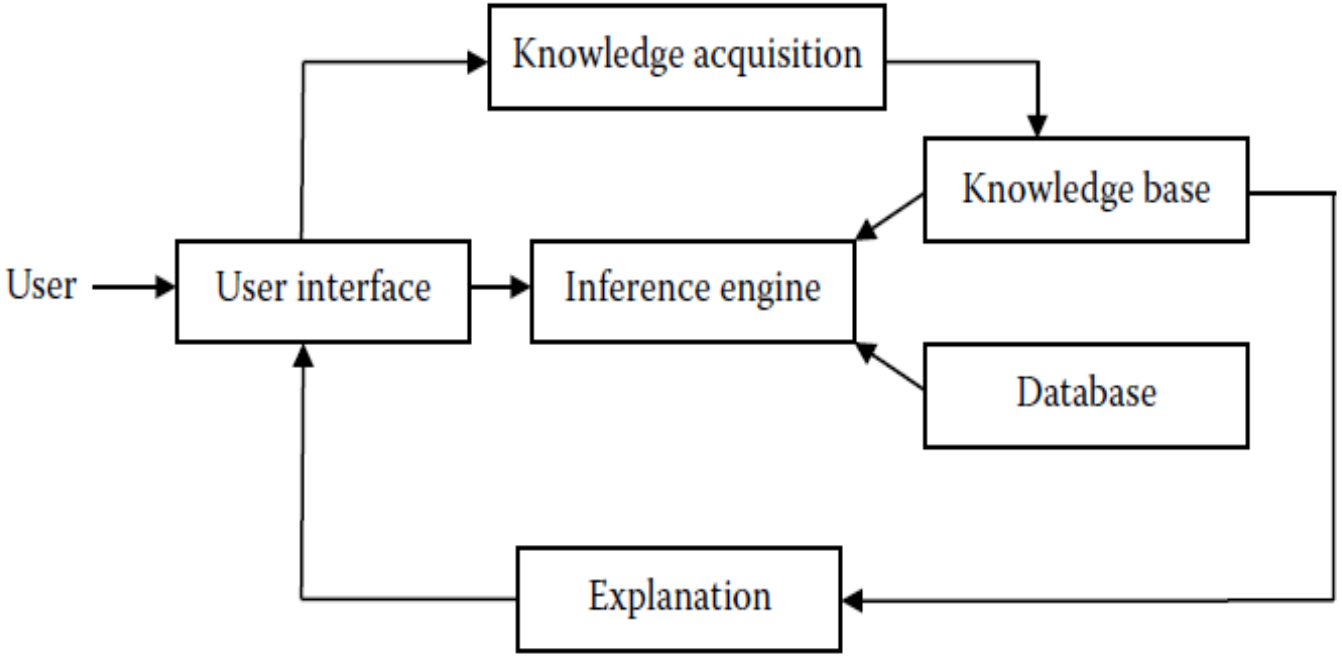


FIGURE 1.6 The architecture of an expert system.

* **User interface:** This is where the user interacts with the system. The user can select options, give inputs, and receive outputs. The user interface is designed to keep all the functionality available in the system in some or the other way.

* **Knowledge base:** This is the pool of knowledge available in the system. Knowledge may be represented in many ways. Standard AI and soft-computing techniques are used for this purpose.

* **Database:** This module is a collection of data. All past or historical data are stored here. The database itself has the potential to generate a huge amount of knowledge once any learning technique, or any other means, is applied over it.

* **Inference engine:** This module is responsible for interpreting the knowledge or its inference. It assembles and presents the output to the user in the form desired. This engine controls the system's overall working.

* **Knowledge acquisition:** This module traps data or inputs and adds them to the pool so the system can learn new things from the data that have been presented. Knowledge acquisition is responsible for making the system robust against the newer inputs.

* **Explanation:** This module is responsible for tracing the execution of the entire process. It records how the various inferences were made and why the specific rules were fired. In expert systems, it is responsible for the reasoning behind the generation of the output from the inputs.

TYPES OF PROBLEMS

A. Classification:

* In classification problems, we are supposed to map the inputs to a class of outputs. The output set consists of discrete values or classes. The system is supposed to see the inputs and select one of the classes to which the input belongs.

* Every input can map only to a single class. Hence, once an input is presented, if the system maps it to the correct class, the answer is regarded as correct.

* If the system maps it to the incorrect class, the answer is incorrect. In the case of classification problems, the system tries to learn the differences between the various classes based on the inputs presented.

* The differentiation among the classes is very simple if there are many differences between the two classes. However, if the two classes are very similar, it becomes very difficult to differentiate.

B. Functional Approximation:

* In functional approximation problems, the output is continuous and is an unknown function of the inputs. For every input, there is a specific output in the entire output space.

* The system is supposed to predict the unknown function that maps the inputs to the outputs. The system tries to predict the same unknown function in the learning phase.

* Whenever the input is provided, the system calculates the outputs. In these systems, it is very difficult for any input to get the precisely correct output. But the output given by the system is usually very close to the actual output.

* The closeness depends on the input given and the kind of learning the system was able to perform. The system's performance may be measured by the deviation of the actual output to the desired output.

* The purpose of such a system's design and learning is to make it possible to imitate the function that maps the inputs to the outputs. Here performance may be measured by the mean percent deviation or root mean square error.

C. Optimizations:

* In optimization, we are given a function, known as the objective function.

* The aim is to minimize or maximize the value of the objective function by adjusting its various parameters.

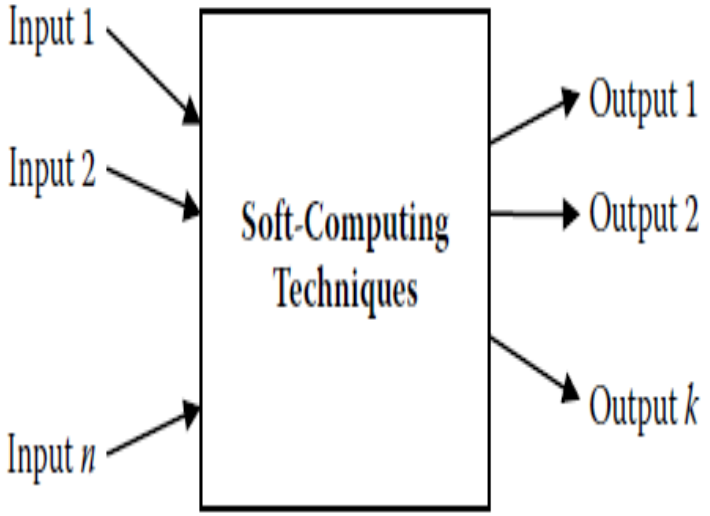
* Each combination of parameters marks a solution that may be good or bad, depending on the value of the objective function.

* Soft-computing techniques generate the best possible parameter set that gives the best values of the objective function considering the time constraints.

MODELING THE PROBLEM

* Soft-computing problems usually involve some situation, and we need to understand that situation to accordingly generate an output. The foremost job is to think about how to represent the entire situation.

* We try to model the problem in a way that can be given as input to the machine in order to generate output. It is important for the machine to be presented with knowledge, data, and inputs or outputs in a form with which it can work.



Input:

* For any soft-computing technique, it is important to identify the inputs well. Too many inputs may make the system very slow. Hence, we only select those inputs that can appreciably affect the output.

* If our system has some feature that is not making a fine contribution to the output, that contribution may be neglected from the list of inputs. The input set contains the candidates that affect the output by a reasonable amount.

* Often while making the system, we may be required to guess the system inputs, because the manner in which different aspects of the problem or different inputs affect the output may not be known at all.

* The decision of selecting inputs is made using common sense. Let us say, for example, that we are making a system that recognizes some circular object in an image. The color may not be an important input to us, because all objects would be circular whatever color they may be.

* Many times we may be required to adopt feature extraction techniques that extract relevant features from the situation that forms the system's input.

* In feature extraction, "features are essential attributes of an object that distinguishes it from the rest of the objects. Extracting features of an object being a determining factor in object recognition is of prime consideration" (Konar, 2000). The uses of this technique can easily be imagined in the context of practical applications of soft-computing techniques.

* Another term of importance is dimensionality reduction or feature selection means to select only the best features for the application.

Outputs:

* Output modeling is as important as input modeling. Outputs must be well known in advance. Because the system can only give valid outputs, not abstract ones, we need to decide the number of outputs and the type of every output according to the problem.

* As was the case with inputs, the number of outputs must be limited to keep the complexity of the system low. A very complex system requires a huge amount of time for processing.

* The amount of data required for training might also be very high. Thus, it is better to limit the number of outputs.

* As was the case for inputs, the outputs must lie between finite ranges of values. It is also desirable for the outputs in the training data to cover the entire output space with equal distribution.

* This may help the system to perform better.

System:

* We now know the inputs and the outputs. We next need to build a system that maps the inputs to the outputs. The system may be built using any of the AI or soft-computing approaches that are covered in this text.

* A combination of two or more of these techniques may also be used to better cater to the needs of the problem. Once the inputs and outputs are known, designing the system using the tools and techniques presented in this text is not a very difficult task.

* To get the best efficiency, however, we need to thoroughly understand the technique we are using. We need to know how it works, as well as its merits and demerits.

* We must use the technique that we are convinced is the best means for solving the problem. In addition, if the existing methods do not perform well, we may need to use a combination of them, along with some intelligent techniques.

* All of this requires a good understanding of the problem and the systems.

MACHINE LEARNING

* Machine learning is the ability of a machine to learn from the given data and to improve its performance by learning. The motivation is for the machine to find the hidden facts in the data.

* By doing so, the machine will better be able to get the correct results if the same data are again given as input. The machine summarizes the entire input data into a smaller set that can be consulted to find outputs to inputs in a manageable amount of time.

* Thus, machine learning may be formally defined as “computer methods for accumulating, changing, and updating knowledge in an AI computer system” (Kasabov, 1998).

A. HISTORICAL DATABASE

* The main focus in machine learning is to provide a good database from which the machine can learn. The underlying assumption is that the outputs can always be reproduced by some function of the inputs.

* Hence it is important for the system to be provided with enough data so the machine can try to predict the correct function. To a large extent, the system's performance depends on the volume of data.

* The data need to be sufficiently large; otherwise, the machine will make false predictions that might look correct to the user. In addition, the system must be well validated against all sorts of inputs to ensure that it can cater to the needs of any future input.

* The diversity of the inputs is also important, and data from all sections of the input space need to be presented.

B. DATA ACQUISITION

* The standard practice for good machine learning is to make a database for all types of inputs. We use means specific to the problem to collect data for the purpose of machine learning.

* For example, a speech recognition system requires numerous people to speak the words to be identified multiple times. All the audio clips may then be recorded and stored in the system, and the inputs and outputs may be calculated and stored in the database.

* In this case, we would need to conduct a survey to collect live data from people. Many standard problems have standard databases that are available on the Internet.

* A few examples of these database repositories are the University of California—Irvine (UCI) Machine Learning Repository, the Time Series Data Library (TSDL), Delve data sets, and the UCI Knowledge Discovery in Databases (KDD) Archive.

* These databases have been built by researchers around the world and donated to be shared under public domain. These databases act as benchmark databases to analyze the performance of any soft-computing technique.

B. PATTERN MATCHING

* Pattern matching is a collection of methods that can tell us whether and to what degree two given patterns or inputs match. Pattern-matching techniques can hence be used for any recognition system in which we are trying to match any given input to the known patterns available in a database.

* The pattern that corresponds to the highest match is regarded as the final pattern. These techniques may also be used for verification systems in which the degree of matching decides the authenticity of the person.

* For example, in a character recognition system, we have a number of ways in which all the letters can be written. Whenever any input is given, we need to match the input with the available database.

* This would give us the correct letter that the input represents. Pattern recognition is a concept similar to pattern matching, which is mostly used for recognition systems.

* Pattern recognition may be defined as “the scientific discipline whose goal is the classification of objects (or patterns) into a number of categories or classes. Depending upon the problem, these objects may be images or signal waveforms or any type of measurements that need to be classified” (Theodoridis and Koutroumbas, 2006).

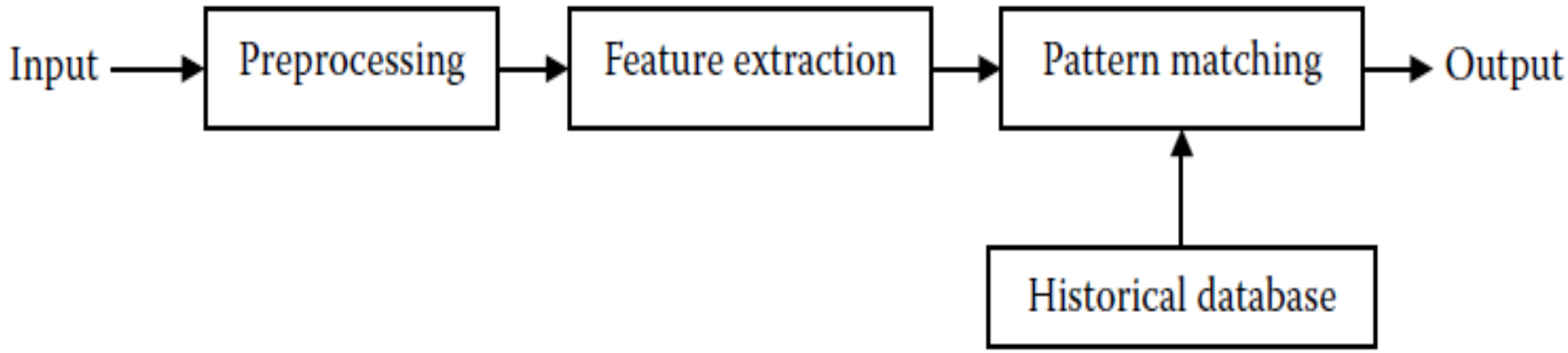


FIGURE 1.10 The various steps involved in a recognition system.

HANDLING IMPRECISENESS

* So far, we have talked about the amount of data and about the ways and means of collecting this data. Any sort of experimentation has limitations, and no matter how hard you try, data will always have some amount of impreciseness.

* This impreciseness can pose a serious limitation to the system. Suppose that your system was supposed to learn data. If there is too much noise, the system may learn the wrong rule, which in turn can make the system behave abnormally.

* Any system will fail to perform unless the input is perfect. However, the effect of impreciseness in soft-computing techniques is actually rather low. Soft-computing techniques are appreciably resistant to noise and uncertainties prevalent in the data.

* They are able to perform well even when placed in highly noisy conditions. The level to which the system can handle noise and uncertainty decides the robustness of the system.

A. UNCERTAINTIES IN DATA

* Uncertainties occur in data when we are not sure that the data presented are correct. It is possible that the data might be wrong.

* Because of this uncertainty, we cannot decide whether the data should be used for machine learning. If the data are wrong, the training and learning will suffer.

* One example of uncertainty is when two sensors get the same data but the data do not match each other. Ideally when two sensors are used for the same reading, the data should match.

* If the data do not match, then there is uncertainty as to whether we should accept the data. If the data do need to be accepted, then we must determine which is correct.

B. NOISE

* Noise in the data means that the data present in the database are not the same data present in the system. Rather the data in the database have been deformed or changed to a different value.

* The amount of deformation depends on the amount of noise. Noise may be present in data for various reasons. It may be due to poor instruments used to measure data or due to mixing of unwanted signals.

* Consider our example of speech recognition. If the person speaking is standing where a train can be heard in the background, the voice our system receives would be that of the speaker as well as of the train.

* Hence the system will not get the correct data and may behave abnormally.

* Two very common forms of noise are prevalent in soft-computing systems. The first is **impulse noise**, in which an attribute, or a set of attributes, is highly noisy for some very small amount of data.

* The rest of the data has **permissible noise**. In our speech recognition example, the data being recorded to make the database is from people speaking. If the spoken words of one speaker got mixed with a sudden noise of dropping glass, then this recording would reveal an impulse noise.

* All other recordings would not have much noise. The other form of noise occurs when some general noise is randomly distributed through the entire data set.

CLUSTERING

* Clustering is an effective way to limit these data. Clustering groups data into various pools. Each pool can then be represented by a representative element of that group; this element has average values of the various parameters.

* The entire pool may be replaced by this representative, which conveys the same information as the entire pool. In this way, we are able to limit the data without much loss of performance.

* Clustering may be formally defined as “the process to partition a data set into clusters or classes, where similar data are assigned to the same cluster whereas dissimilar data should belong to different clusters” (Melin and Castillo, 2005).

* The distance measure, or the difference in parameters between the two clusters, measures the closeness of the clusters to each other.

A. K -MEANS CLUSTERING

Algorithm 1.2

k-Means Clustering (k)

```
C ← random k centers
While change in cluster centers is not negligible
Do
    For all elements e in data set
    Do
        Find the center c_i, which is closest to e
        Add e to cluster i
    For all centers c_i in C
    Do
        C_i ← arithmetic mean of all elements in cluster i
```

* To begin, k random points are chosen. These points are regarded as the centers of the k clusters. The elements are then distributed to these clusters. Any element belongs to the cluster that is closest to it.

* After all the elements have been distributed in this manner, the cluster centers are modified. Each cluster has a set of elements given to it during distribution. The new cluster center lies at the mean of all the cluster members. All the k cluster centers are modified in this manner.

* We again start with the distribution of elements to the clusters. We repeatedly do this until the changes in the cluster centers become very small or the centers stop changing their positions.

B. FUZZY C-MEANS CLUSTERING

* Another commonly used clustering algorithm is the fuzzy C-means clustering. This algorithm is also iterative, which means that the steps are repeated until the stopping criterion is met.

* The clustering improves with every step. In this algorithm, rather than associating every element with a cluster, we take a fuzzy approach in which every element is associated with every cluster by a certain degree of membership.

* This is why the algorithm is prefixed fuzzy. The degree of membership is large if the element is found near the cluster center.

* However, if the element is far from the cluster center, then the degree of membership is low.

For any element, the sum of the degree of belongingness to all clusters is always 1, as is represented in Equation 1.1:

$$\sum_{i=1}^{k} u(e,i) = 1 \tag{1.1}$$

where $u(e, i)$ denotes the membership of the element e to the cluster i.

The center of any of the clusters is the mean of the points weighted by the degree of membership to this center. The center c_i for any cluster i can be given by Equation 1.2:

$$c_i = \frac{\sum_e e * u(e,i)^m}{\sum_e u(e,i)^m} \tag{1.2}$$

The degree of belongingness of any element e to the cluster i is the inverse of the distance between the element and the cluster center c_i. This is given by Equation 1.3:

$$u(e,i) = \frac{1}{\text{distance}(e,c_i)} \tag{1.3}$$

The coefficients are normalized and fuzzified with a real parameter $m > 1$ in order to make their sum 1. This degree of membership of any element e to cluster i can hence be given by Equation 1.4:

$$u(e,i) = \frac{1}{\sum_j \left(\frac{\text{distance}(c_k,e)}{\text{distance}(c_j,e)}\right) 2/(m-1)} \tag{1.4}$$

C. SUBTRACTIVE CLUSTERING

* In subtractive clustering, however, the number of clusters does not need to be specified in advance. It thus may be used to find clusters when we do not wish to specify the number of clusters; instead, we expect the algorithm to inspect the data and decide the number of clusters on its own.

* This is a fast algorithm that is used to estimate the number of clusters and the cluster centers in a data set. Due to lack of space in this text and the complexity of the algorithm, we do not study this algorithm in detail.

* As an example, we will use the IRIS classification data from the UCI Machine Learning Repository to carry out clustering by these three algorithms. The IRIS data have four features that were measured: sepal length in centimeters, sepal width in centimeters, petal length in centimeters, and petal width in centimeters.

HAZARDS OF SOFT COMPUTING

* Every system has a liability associated with it. This is most obvious when the system fails. Every system has certain assumptions and constraints that change with time.

* It cannot be guaranteed that any system will continue performing without failures, as errors and bugs are prone to every system. Thus, it is possible for a system to generate wrong answers or even crash.

* The case of soft computing is even more special. Even after long research, we have not been able to achieve 100 percent accuracy.

* This means that soft-computing systems are not to be relied upon blindly. Every soft-computing system has some small amount of error that must be considered.

* Normally such an error is acceptable to the user or application, and hence it does not make much of a difference. But in certain conditions, such an error can prove bad.

ROAD MAP FOR THE FUTURE

* Soft-computing techniques have seen research in every domain, ranging from practical applications to theoretical foundations. As a result, these techniques give much better performance and are able to solve problems very efficiently.

* There is, however, still a great deal of scope for things to be done in the future to enable people to better enjoy the fruits of soft computing and to tap the full potential of soft computing.

* Soft computing is still far behind in providing the functionality of the human brain. The brain

is still the biggest motivator and challenge for engineers working on soft computing.

* Both the computational power and performance of the human brain is beyond the reach of our present systems.

* The existing systems also have far to go in terms of robustness. Performance is still a major issue for various systems. In addition, so many domains still remain untouched by soft computing, and soft-computing techniques are still in the growing stage in many other fields.

* Hence, we find a very bright future and much potential for further research into the fields of soft computing. We can surely expect many wonders in the near future as a result of this field.

A. ACCURACY

* The best accuracies of well-established systems may revolve around 99 percent to 99.9 percent. These accuracies are achieved when many assumptions have been imposed on a system.

* The accuracies of some of the more complex problems, however, are much lower. Attempts are being made to build better architectures and to extract better features in order to improve this accuracy.

* Simultaneous work in all the fields, be it biometrics, bioinformatics, or some other field, are being carried out in the hopes that these systems will one day be able to cross all barriers to achieve 100 percent accuracy.

* Systems are also prone to noise. Many different preprocessing techniques have been devised for specific applications to compress a good amount of noise.

* Along with better preprocessing, better recognition systems might result in good performances in real life applications.

B. INPUT LIMITATIONS

* It is well known that the performance of soft-computing techniques largely depends on the size of the input data set, with higher data sets being a big boon to the system.

* Presently all systems have a limited data set through which they are trained and through which performance is tested. The limited data-set size gives good results in experimentation but has huge limitations in real life problems.

* The types of data are also limited in experimentation. In real life problems, the data size needed for a system to perform might be too high.

* A speaker recognition system might work well for ten speakers. But for the practical real-life problem, if we increase the number of speakers to 10,000, the system might refuse to work at all.

* This imposes a serious issue on deployment of these systems into the real world.

C. COMPUTATIONAL CONSTRAINTS

* Every system needs to perform under a given time. Even though computation power has increased rapidly over the past few decades, an increase in expectations and performance of soft-computing techniques means we need greater computation power.

* An exponential rise in the size of data sets and in the number of inputs has also had a deep effect on the computational requirements. If computation power continues to grow, we may soon have systems that can perform much better in a shorter time.

* The increase in computation would mean a greater autonomy in the selection of features, data sets, or complexity.

D. ANALOGY WITH THE HUMAN BRAIN

* The human brain is one of the most magnificent developments and is responsible for the intelligence of human beings. The history of humankind narrates the wonderful creations that the human brain is capable of producing.

* We now give a brief glimpse into the kind of system the brain is. It is estimated that the human brain contains 50 to 100 billion neurons.

* Of these, about 10 billion are cortical pyramidal cells, which pass signals to each other via approximately 100 trillion synaptic connections. Even the most complex machines that we make, including our super computers, are too small in computation when compared with the brain's massive structure.

* The large computation in parallel by the biological neurons is what makes it possible for the brain to do everything it does. It will take a very long time for engineers to build any machine that is comparable to the human brain.

* But who knows, one day artificial life and artificial humans might turn out to be a reality.

E. WHAT TO EXPECT IN THE FUTURE

* We have already discussed how performance, robustness, and computation play a major role in changing the face of soft computing in the future. We have even discussed that in the future, soft computing will move from the laboratories and into the practical areas of industry and consumer life.

* But there is much more that is possible in the very near future. For years, one of the major problems with soft computing has been its multidisciplinary nature.

* To make effective systems, either the soft-computing experts will have to master other domains or people from other domains will have to master soft computing.

* This problem has prevented soft computing from entering many domains. As a result, there is a considerable amount of work that is yet to be done by soft computing in many domains.

* Making newer and newer models by mixing existing models is another very active field of research. We may soon find many innovative ways to handle the problems that soft-computing techniques are currently facing.