# Soft Computing

## 3.1. Introduction:

A simple example that might draw the line between our way of thinking and doing the calculations in our minds in a *soft way* and how computers and machines think and do the computations in a *hard way* is to demonstrate the different algorithms used in both ways. Let us say, you are now reading this work in electronic format and you want to find the beginning of "chapter 5", for example, and you expect that chapter to start with the word "chapter 5". Searching for the word "chapter 5" using the *search algorithm* embedded in the software you are using for reading this document, and using your *human intelligent algorithm* in finding the required word demonstrates the difference between the *hard computing* and the *soft computing*, respectively. If you observe the search algorithm embedded in the software you use, you will find it comparing each word in the document with the required words and it usually starts from where you are and compares each word in the following text to the required words. If you start, for instance, from a page that is just next to the page that includes the required word, the search algorithm will do the *hard computing* and compares all the words until the end of the file then starts again from the beginning until reaching the required page. Human minds, on the other hand, think in a different way and perform *intelligent search* where they can change the starting point at any time and skip pages with thousands of words if they have the doubt that this is not the probable place for the required word. To find the beginning of "chapter 5" in a book of ten chapters, for example, human will decide to start at the middle of the book or to start at a *random* page then go backward or forward from that point according to how the words in that page *match* or *fit* the *search criterion*. It is the *soft computing* embedded in our intelligent brains that saves us the work of reading and comparing each word (the *hard computing* that followed by the traditional computing algorithms) and going directly to our required target based on a *fitness* criterion. We should notice that, computers can find the required word faster than the human being assigned the same task and this has nothing to do with advances in the *computation algorithms* itself but might be the advances in the *speed* of doing the tedious calculations. In fact, computers reach results faster, not because they are smarter or more intelligent but because they are doing the calculations *faster*, with the speed of the used electronics and hardware, and in a *systematic* way that does not affect with the *fatigue* that might affect humans in repeating the same task many times.

In our contemporary times, scientists are not content with the available resources and usually think of improving and advancing techniques to serve the human needs. In the middle of the 1950s, scientists decided to combine the advantages of computers (e.g. artificial, fast, systematic…, etc) and the advantages of human minds (e.g. intelligent, rational, interacting with uncertainty…, etc) and initiated a new science of that time which is called "***Artificial Intelligent***", usually abbreviated as **AI**. The name itself implies the two main advantages of the required systems of being *intelligent* as human and *artificial* as machines and computers and AI

can be defined as "the branch of computer science that is concerned with the automation of intelligent behavior" [1]. Therefore, the field of artificial intelligent attempts to understand intelligent entities (human) and implements this understanding in building intelligent entities. AI addresses one of the ultimate puzzles, "how is it possible for a slow, tiny brain, whether biological or electronic, to perceive, understand, predict, and manipulate a world far larger and more complicated than itself?", "how do we go about *creating* something with those properties?". Therefore, AI deals with the problem of creating an "*intelligent agent*" [2].

An agent is an object that is able to *perceive* its environment through *sensors* and acting upon that environment through *effectors*. Human "agents" use their eyes, ears, skins…,etc as sensors and their hands, legs…,etc as effectors. On the other hands, robotic agents use cameras, infra-red detectors…, etc as sensors and motors, mechanical arms…,etc as effectors. While the brain is used in human agents to respond to the sensor's signal with an appropriate effectors signal, the software (*program*) embedded in the robotic processor (architecture) performs a similar task in connecting the robot's sensors with its effectors, and the main goal of AI is to elevate the interaction of robots to their environment to reach the level of the *rational* human interaction. It is also a part of the AI to make the agent more *autonomous*; i.e. behaving according to its *own experience* rather than on knowledge of the environment that has been built-in by the designer. Achieving these goals of AI requires the *design* of the "agent program" that runs on the computing device, that is called "architecture" and *learning* becomes an important design stage, and the process equation turns to be: *agent = architecture + program*.

The following sections will focus on two "*soft-computing program*" designs that are built in our agents to imitate the human agent's *mapping*. This mapping refers to the processing of the received *percepts* from the sensorial inputs (in our case, images represent the inputs that are captured using the computer vision systems described in the last chapter) and *generating actions* through the effectors (in our case, evaluation decisions on input materials represented in the form of an alarm signal or a signal that is directed to a mechanical arm that remove defective parts…, etc). The task of the soft-computing program is to replace the procedure of "looking-up table of actions" corresponding to a set of percepts; because the *table-driven-agents* are not intelligent enough to deal with different situations that pop-up in daily life's judgments.

### 3.2. Artificial Neural Networks (ANN):

It is believed for long time that the brain is the center of intelligence and thinking in human beings. Therefore a mathematical model for the operation of the brain is very important and will be introduced here after understanding the biological system and its simplest computing elements. The fundamental functional unit of the nervous system (including the brain as the center of thinking) is the neuron (nerve cell). Each neuron consists of a cell body that contains the nucleus and there is a number of fibers branching out from the cell body and called *dendrites* and a single long fiber called the *axon*. Dendrites branch around the cell in a network form and the axon stretches out for a long distance compared to the cell body. Eventually, the axon also

branches into strands and substrands that connect to the dendrites and cell bodies of other neurons. The connecting junction is called a *synapse* and each neuron forms synapses with anywhere from a dozen to a hundred thousand other neurons, as demonstrated in Figure 1.

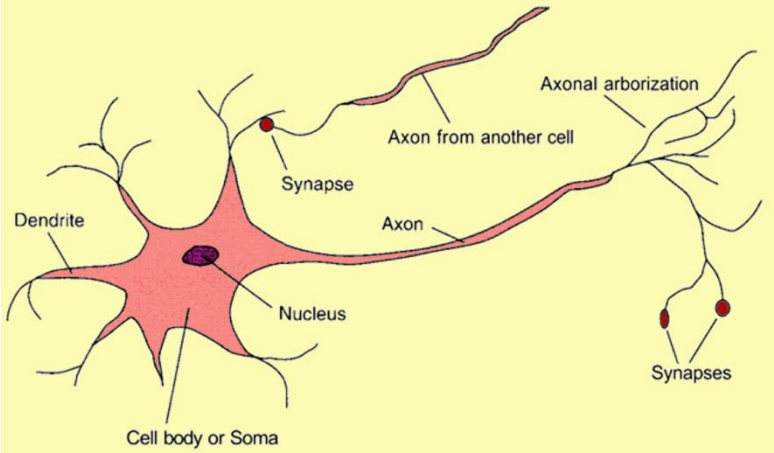


Figure 1. The parts of a nerve cell (or neuron). The length of the axon is usually about 100 times the diameter of the cell body (image reproduced from ref. [2])

Signals propagate from neuron to anther through electrochemical reactions where chemical transmitter substances are released from the synapses and enter the dendrite, raising or lowering the electrical potential of the cell body. When the potential reaches a *threshold*, an electrical pulse or *action potential* is sent down the axon. The pulse spreads out along the branches of the axon, eventually reaching synapses and releasing transmitters into the bodies of other cells. Synapses that increase the potential are called *excitatory*, and those that decrease it are called *inhibitory*. Neurons also form new connections with other neurons, and sometimes entire collections of neurons can migrate from one place to another and these mechanisms are thought to form the basis for learning in the brain and a collection of those neurons can lead to thought, action, and consciousness.

### *3.2.1. Neuron model*

In biomimicry of the nervous system, the artificial neural network (ANN) appears as a mathematical representation for the neuron model. ANN is composed of a number of *nodes*, or *units*, connected by *links* and each link has a numeric *weight* associated with it. Weights are the primary means of long-term storage in neural networks, and *learning stage* usually takes place by updating these weights. Some of the units are connected to the external environment, and can be designated as *input* or *output* units. As demonstrated in Figure 2, each unit has: a set of *input links* from other units, a set of *output links* to other units, a current *activation level*, and a *means of computing* the activation level at the next step in time given its inputs and weights. Therefore, each unit does a local computation based on inputs from its neighbors, but without the need for any global control over the other units in the neural network. The design of a neural network for a specific task requires decisions on: how many units to be used, what kind of units are appropriate, and how the units are to be connected to form a network. After designing the

*network architecture* based on these questions, the weights of the network are *initialized* and a *learning algorithm* is used to *train* the *weights* based on a set of data for the task. It is also important to decide how to encode the data in terms of numeric inputs and outputs of the network.

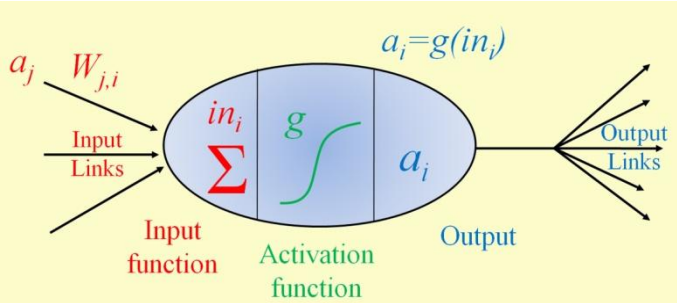


Figure 2. Simple computing unit (Node)

In artificial neurons, each unit performs a simple computation by receiving signals from its input links and computing a new activation level that is sent along each of the output links. The computation of the activation level is based on the values of each input signal received from the neighboring node and the weights on each input link. The computation in the neuron has a *linear component*, called the *input function* ($in_i$) that computes the *weighted sum* of the unit's input values. The neuron also has a *nonlinear component* called the *activation function* (g) that transforms the weighted sum into the final value that serves as the unit's *activation value* ($a_i$) and it is common to use the same activation function for all units in the network.

The total weighted input ($in_i$) is the sum of the input activations times their respective weights:

$$in_i = \sum_j W_{j,i} a_j = \boldsymbol{W_i} \cdot \boldsymbol{a_i} \qquad (1)$$

Where $\boldsymbol{W_i}$ denotes the weights on links into node *i*, and $\boldsymbol{a_i}$ refers to the set of input values for the $i^{th}$ node, and the dot product denotes the sum of the pairwise products. The elementary computation step in each unit computes the new activation value for the unit by applying the activation function ($g$) to the result of the input function:

$$a_i = g(in_i) = g(\sum_j W_{j,i}\, a_j) \qquad (2)$$

There are different mathematical forms for the activation function ($g$), also called the *transfer function*, such as the step, sign, and sigmoid functions, as demonstrated Figure 3. The step function has a threshold *t* such that it outputs a 1 when the input is greater than its threshold, and outputs a 0 otherwise. The biological motivation is that a 1 represents the firing of a pulse down the axon, and a 0 represents no firing where the threshold represents the minimum total weighted input necessary to cause the neuron to fire.

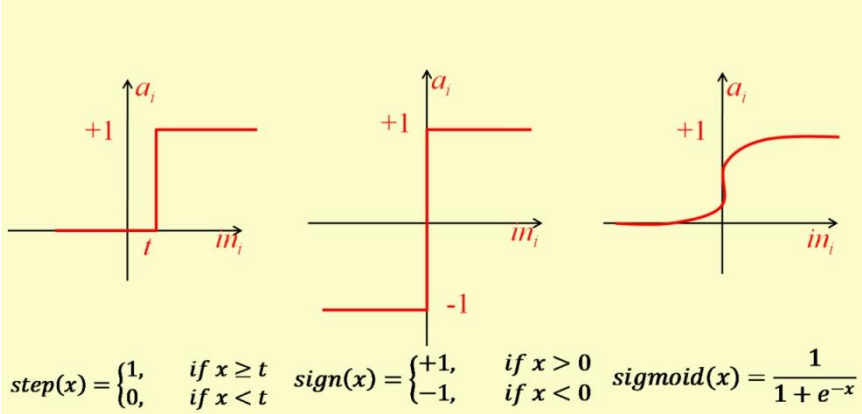


Figure 3. Three different activation (transfer) functions for neuron computational units

It is mathematically more convenient to replace the threshold with an extra input weight which allows for a simpler learning element because it needs only to worry about adjusting weights, rather than adjusting both weights and thresholds. Thus, instead of having a threshold $t$ for each unit, we add an extra input whose activation $a_0$ is fixed at -1. The extra weight $W_{0,i}$ associated with $a_0$ serves the function of a threshold at $t$, provided that $W_{0,i}\ a_0 = -t$. Then all units can have a fixed threshold at 0. These two representations for thresholds are mathematically equivalent:

$$a_i = step_t\left(\sum_{j=1}^{n} W_{j,i}\ a_j\right) = step_0\left(\sum_{j=0}^{n} W_{j,i}\ a_j\right) \qquad (3)$$

Where $W_{0,i}=t$ and $a_0=-1$.

### *3.2.2. Network structures*

There are different kinds of network structures that result in very different computational properties, and the most important structures that need to be distinguished are the *feed-forward* and the *recurrent* networks. In a feed-forward network, links are unidirectional, and there are no cycles, while in a recurrent network, the links can form any arbitrary topology. We usually deal with networks that are arranged in *layers*, and in a *layered feed-forward network*, each unit is linked only to units in the next layer; there are no links between units in the same layer, no links backward to a previous layer, and no links that skip a layer. The significance of the lack of cycles is that computation can proceed uniformly from input units to output units. The activation from the previous time step plays no part in the computation, because it is not fed back to an earlier unit. Hence, a feed-forward network simply computes a function of the input values that depends on the weight settings and it has no internal state other than the weights themselves. Figure 4 shows a simple example of a layered feed-forward network with two layers where the input units (square nodes) serve to pass activation to the next layer and may not be considered as an individual layer because no computations take place at these nodes.

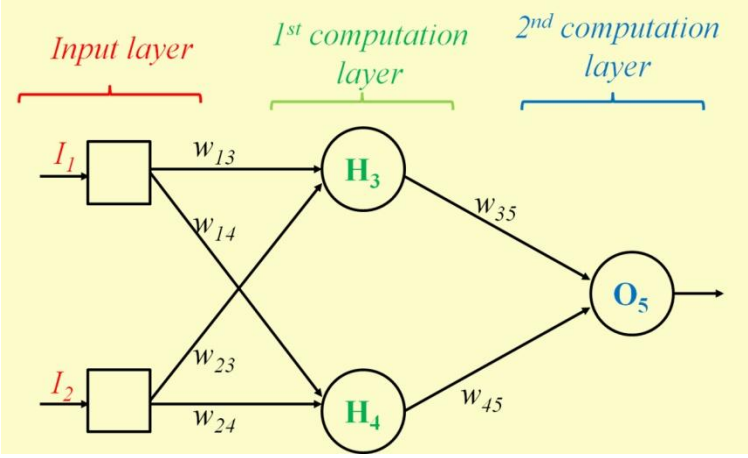


Figure 4. Layered feed-forward network

The activation value of each of the units in the input layer in Figure 4 is determined by the environment, and at the right-hand end of the network is the output unit that represents the effectors from the network model. In between, the nodes labelled $H_3$ and $H_4$, have no direct connection to the outside world and are called *hidden units*, because they cannot be directly observed by noting the input/output behavior of the network. Some networks, called *perceptrons*, have no hidden units and this simplifies their learning process, but it also means that perceptrons are very limited in what they can represent. Networks with one or more layers of hidden units are called *multilayer networks* and with one layer of sufficient number of hidden units it is possible to represent any continuous function of the inputs; while with two hidden layers it is also possible to represent discontinuous functions. The numbers of hidden layers and the units per layer are of great importance in the design of the ANN where too small network might be incapable of representing the desired function. On the other hand, too big networks tempt to *overfit* the data because there are too many parameters (i.e., weights) in the model and the ANN will memorize all the examples by forming a large lookup table and will not be able to generalize well to inputs that have not been seen before.

With a fixed structure and fixed activation functions $g$, the models represented by a feed-forward network are restricted to have a specific parameterized structure. The weights chosen for the network determine which of these models is actually represented. For example, the network in Figure 19.7 calculates the following function:

$$\begin{aligned} a_5 &= g\left(W_{3,5}\, a_3 + W_{4,5}\, a_4\right) \\ &= g\left(W_{3,5} g\left(W_{1,3} a_1 + W_{2,3} a_2\right) + W_{4,5} g\left(W_{1,4} a_1 + W_{2,4} a_2\right)\right) \end{aligned} \quad (4)$$

Where $g$ is the activation function and $a_i$ is the output of the $i^{th}$ node.

It should be noticed that; because the activation functions *g* are nonlinear, the whole network represents a model of complex *nonlinear function* and if you think of the weights as *parameters* or *coefficients* of this nonlinear function, then the learning stage becomes a process of tuning these parameters to fit the data in the training set, analogically to the *nonlinear regression* modeling commonly used in statistics, and that is what ANN perform.

Although the biological neurons in our brains are not feed-forward networks, some regions of the brain are largely feed-forward and somewhat layered, but there are widespread back-connections and, therefore, the brain forms recurrent networks. Because activation is fed back to the units that caused it, recurrent networks have internal state stored in the activation levels of the units which means that computation can be less orderly than in feed-forward networks. Recurrent networks can become unstable, or oscillate, or exhibit chaotic behaviors and with a given set of inputs, it can take a long time to compute a stable output, and learning is made more difficult. On the other hand, recurrent networks can implement and model more complex designs and the *Hopfield networks* as well as the *Boltzmann machines* are two common examples of the recurrent networks that are used in such modeling.

### *3.2.3. Back-propagation learning*

Learning in a feed-forward network with one computation layer (e.g. perceptrons) is a straight-forward process where the error (the difference between the network output and required target) is fed back to the network and the weights are adjusted to reduce this error. The problem, however, is to assess the blame for an error and divide it among the contributing weights and this is an easy task in perceptrons because there is only one weight between each input and the output. But in multilayer networks, there are many weights connecting each input to an output, and each of these weights contributes to more than one output. The back-propagation algorithm is a sensible approach to dividing the contribution of each weight where it provides a way of dividing the calculation of the gradient among the units, so the change in each weight can be calculated by the unit to which the weight is attached using only local information. Similar to the learning algorithm in the perceptron, back-propagation tries to minimize the error (*E*) between each target output (*T*) and the output actually computed by the network (*O*), that is:

$$E = \frac{1}{2}\sum_i (T_i - O_i)^2 \qquad (5)$$

This error function represents the *sum square error* which is selected because it allows easier differentiation in the next calculation steps. This error function can be expressed as a function of the weights by substituting the output value (*O*) in the previous equation, which can be expressed for a two-layer network as:

$$E = \frac{1}{2}\sum_i (T_i - g(\sum_j W_{j,i}\, a_j))^2$$
$$= \frac{1}{2}\sum_i (T_i - g(\sum_j W_{j,i}\, g(\sum_k W_{k,j}\, I_k)))^2 \quad (6)$$

Where $I_k$ represent the $k^{th}$ input to the network.

Notice that although the $a_j$ term in the first line represents a complex expression, it does not depend on $W_{j,i}$. Also, only one of the terms in the summation over $i$ and $j$ depends on a particular $W_{j,i}$, so all the other terms are treated as constants with respect to $W_{j,i}$ and will disappear when differentiated. Hence, when we differentiate the first line with respect to $W_{j,i}$ we obtain:

$$\frac{\partial E}{\partial W_{j,i}} = -a_j (T_i - O_i) g'(\sum_j W_{j,i}\, a_j)$$
$$= -a_j (T_i - O_i) g'(in_i) = -a_j \Delta_i \quad (7)$$

With:

$$\Delta_i = (T_i - O_i) g'(in_i) = Err_i\, g'(in_i) \quad (8)$$

Therefore, the update rule for the weights in the output layer becomes:

$$W_{j,i}^{new} = W_{j,i}^{old} + \eta\, a_j\, \Delta_i \quad (9)$$

Where $\eta$ is an arbitrary value that is called *the learning rate* and controls the speed of the network learning. For updating the connections between the input units and the hidden units, we need to define a quantity analogous to the error term for output nodes. Here is where we need the error back-propagation where the idea is that; hidden node $j$ is "responsible" for some fraction of the error $\Delta_i$ in each of the output nodes to which it connects. Thus, the $\Delta_i$ values are divided according to the strength of the connection between the hidden node and the output node, and propagated back to provide the $\Delta_i$ values for the hidden layer. The propagation rule for the $\Delta_i$ values is the following:

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i \quad (10)$$

And the update rule for the weights leading into the layer becomes:

$$W_{k,j}^{new} = W_{k,j}^{old} + \eta\, I_k\, \Delta_j \quad (11)$$

Similar to the calculation of $W_{j,i}$, the derivation of the gradient error with respect to $W_{k,j}$ is slightly more mathematically involved, but has a similar result where:

$$\frac{\partial E}{\partial W_{k,j}} = -I_k \Delta_j \quad (12)$$

To obtain the update rules for the weights, we have to remember that the object is to *minimize* the error, so we need to take a small step in the direction opposite to the gradient.

As can be observed from this derivation, the derivative of the activation function *g* is required during the computation, so the *sign* and the *step* functions are not used in back-propagation networks. Back-propagation networks usually use the sigmoid function or some other variants that are easily differentiable. The *sigmoid* also has the convenient property that the derivative $g' = g(1 - g)$, so that little extra calculation is needed to find $g'(in_j)$.

### *3.2.4. Closing remarks on ANN*

Although their popularity, artificial neural networks (ANN) still questionable as an optimum soft-computing method for machine learning and producing the required artificial intelligence. ANN can be evaluated from different points of view such as their:

- **Expressiveness**: Neural networks are attribute-based representations (i.e. most suitable for mapping continuous inputs and outputs), and do not have the expressive power of other logical representations (e.g. decision tree systems).
- **Computational efficiency**: Computational efficiency depends on the amount of computation time required to train the network to fit a given set of examples. If there are *m* examples, and $|W|$ weights, each training epoch takes $O(m|W|)$ time. However, work in computational learning theory has shown that the worst case number of epochs can be exponential in *n*, the number of inputs. In practice, time to convergence is highly variable, and vast arrays of techniques have been developed to try to speed up the process using tunable parameters. Local minima in the error surface are also a problem and at the cost of some additional computation, other techniques (e.g. simulated annealing method) can be used to assure convergence to a global optimum.
- **Generalization**: Neural networks can do a good job of generalization especially on functions for which they are well-suited. These seem to be functions in which the interactions between inputs are not too complicated, and for which the output varies smoothly with the input. There is no theorem to be proved here, but it does seem that neural networks have had reasonable success in a number of real-world problems.
- **Sensitivity to noise**: Because neural networks are essentially doing nonlinear regression, they are tolerant of noise in the input data. They simply find the best fit given the constraints of the network topology. On the other hand, it is often useful to have some idea of the degree of certainty of the output values and ANN do not provide probability distributions on the output values.
- **Transparency**: Neural networks are essentially black boxes, even if the network does a good job of predicting new cases, many users will still be dissatisfied because they will have no idea why a given output value is reasonable. If the output value represents, for example, a decision to perform open heart surgery, then an explanation is clearly in order. With decision trees and other logical representations, the output can be explained as a logical derivation and by appeal to a specific set of cases that supports the decision. This is not currently possible with neural networks.

- **Prior knowledge**: Learning systems can often benefit from prior knowledge that is available to the user or expert and prior knowledge can mean the difference between learning from a few well-chosen examples and failing to learn anything at all. Unfortunately, because of the lack of transparency, it is quite hard to use one's knowledge to "lead" a network to learn better.

All these considerations suggest that simple feed-forward networks, although very promising as construction tools for learning complex input/output mappings, do not fulfill our needs for a comprehensive theory of learning in their present form. Researchers in AI, psychology, theoretical computer science, statistics, physics, and biology are working hard to overcome these difficulties.

### 3.3. Modeling human judgment (fuzzy logic):

In situations where a great deal of human judgment must be made about a product or a system, most traditional models fail to fully characterize the nature of human judgment. In this case, a method of rule-based decision making should be used in conjunction with artificial intelligence systems and process control. The objective of this would be to emulate the thought process used by human beings. Such a method is now commonly known as 'fuzzy logic', a revolutionary approach to system identification and control pioneered by Lotfi Zadeh in 1965. Fuzzy Logic is a multivalued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/white…, etc. Notions like "rather warm" or "pretty cold" can be formulated mathematically and processed by computers. In this way an attempt is made to apply a more human-like way of thinking in the programming of computers. To realize the importance of the fuzzy logic and getting started with its terminology, it is important at the beginning to distinguish between the *crisp set* and the *fuzzy set* which will be defined in the following section.

#### *3.3.1. Crisp and fuzzy sets*

In classical mathematics we are familiar with what we call *crisp sets* which usually have "sharp" edges that identify the starting and ending points of a group (set) of points. The idea of crisp sets even consumes a lot of work with calculus beginners who start with learning the "limit" of a function and its continuity. A set of discrete points $X$ for the first five integers is defined in mathematics as $X=\{1, 2, 3, 4, 5\}$ and any point does not belong to this group is outside the set $X$. The continuous set of numbers $Y = [1, 5]$ represents the same period and includes the natural numbers. These two sets are *crisp* in a way that the number 1.3 is not part of the set $X$ (that is $1.3 \notin X$) while it belongs to $Y$ ($1.3 \in Y$). Since $Y$ is a "crisp" set, any value with any slightly lower value than 1 or slightly higher value than 5 will not belong to the set Y (therefore, $0.999 \notin Y$ and $5.001 \notin Y$). The two sets $X$ and $Y$ are graphically represented in Figure 5, where the vertical axes

in this figure represent the "membership ($\mu$)" of a point on the horizontal axis to the required set. Any point that belongs to ($\in$) a set has a "full membership" with $\mu$=1, and any point that does not belong to ($\notin$) a set has "no membership" with $\mu$=0. Therefore, a "crisp set" can be defined as a set that has a "*binary membership*"; either $\mu$=0 (no membership) or $\mu$=1 (full membership). The principle of membership helps to define the set *X* mathematically as:

$$X = [x \mid P(x)] \tag{13}$$

That statement read as, the set *X* consists of all points *x* as long as the property *P*(*x*) holds true, and in this case the property *P*(*x*) is true when the membership $\mu(x)$=1.

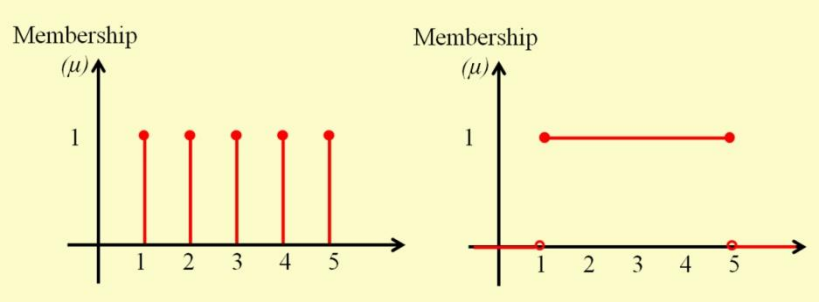


Figure 5. Membership representation to the set X={1, 2, 3, 4, 5} (left), and the set Y=[1, 5] (right)

The application of this crisp set in real life will be more complicated, because human have different sets that do not have such crisp edge; for example sets of: hot vs. cold, tall vs. short, young vs. old…, etc. Let us consider the age-set for "young" vs. "old" and consider the age of 20 years as a limit for a person to be consider as a "young". The representation of this set mathematically will be *Young* = [0, 20]; because the age starts at zero and we decided the limit for it to be 20 years old. The question for a person whose age is 15 years old will be very simple, and the answer definitely will be yes, this person belongs to the group of "young" people and his membership to this group will be $\mu$=1. Similarly, a person with 20 years old will have membership $\mu$=1 to the group of the "young" people. But in reality, it does not make sense for human to decide that a person on the day of being exactly 20 years old to be "young" and "not young" the next day. Human decisions about this problem tend to consider a person of even 21 years old to be "still young" or for someone of 25 years as "young enough". Therefore, the limit within the age-set between the "young" and the "old" is "*fuzzy*", not "crisp", and the membership to each set is "*continuous*", not "*binary*". The continuity of the membership suggests a "*membership function*" and any point *A* can take a value from zero to one; *i.e.* $\mu_A = [0, 1]$.

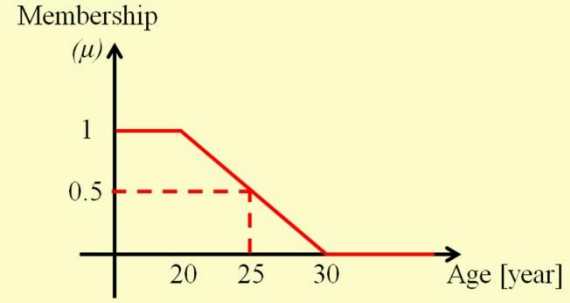


Figure 6. Membership function for the age-set of "young" people

A graphical representation for the principle of the "*fuzzy set*" of age-set and the membership function to the group of "young" people is demonstrated in Figure 6. Based on this figure, a person of age 25 years will be considered 50% belonging to the group of "young" people; and to describe this mathematically, we set the point $A$=25 and its membership $\mu(A) = \mu_A = 0.5$. Similarly, for a person of age 21 years: $A$=21 and $\mu(A) = \mu_A = 0.9$ that is he/she is 90% belonging to the group of "young" people. It is clear that, the membership function does not have to be linear as we considered in this example. In fact, membership function takes different shapes such as Gaussian, sigmoid, trapezoid, triangular…, etc. Setting the parameters of these membership functions allows the final decision for a point to resemble the human judgment about that point.

The concept of membership function is often confused with the probability concept. There is an obvious similarity between the two where both membership values and probability values exhibit a range from 0 to 1. However, the interpretation of this range of values is substantially different in the two cases. Fuzzy membership provides a measure of judgment, whereas the probability indicates the proportion of times the result is true or false in the long run. It is also crucial to understand that the fuzzy set theory is a means of specifying how well an object satisfies a *vague description*, not uncertainty about external world. For example, consider the proposition "Alex is tall", and the question: "Is this true, given that Alex is 175 cm?" Most people would hesitate to answer "true" or "false," preferring to say, "sort of." This is not a question of uncertainty about the external world, as we are sure of Alex's height, but it is a case of *vagueness* or *uncertainty* about the *meaning* of the *linguistic term* "tall".

### *3.3.2. Operations on Fuzzy Sets*

Human used to apply some logical rules during their decision making and daily life judgments. We face many situations where our decisions should be made based on a "*logical sense*"; consider the tipping at a restaurant, for example, as a problem where you decide the amount of a tip based on the quality of food you eat. The logical rule for such problem will be: "if the food is

*good*, then tip is *high*". Notice the two "vague" and "fuzzy" terms used in this statement "good" and "high", and the questions "how good is good?" and "how high is high?" reflects back to the fuzzy logic. The application of fuzzy sets for this problem will make it easier to deal with this "uncertainty" of the limits and the vagueness of these terms.

This simple problem becomes more complicated as we consider other criteria in our judgment. For the same tipping problem, the logical rule in the human mind will consider not only the quality of food, but also the quality of service at the restaurant to determine the total amount of tip. The logical rule for the problem in this case will be: "if the food is *good* AND the service is *good*, then the tip is *high*". Notice the AND operator that is used to connect the two statements in our judgment, which poses the question of how to perform the "*logical operations*" (AND, OR, NOT) on fuzzy sets? In fuzzy logic terminology, these logical operations are defined as the "*fuzzy intersection*" or "*conjunction*" (AND), the "*fuzzy union*" or "*disjunction*" (OR), and the "*fuzzy complement*" (NOT). To perform these operations on fuzzy sets, the membership functions are subjected to some functions that replace the logical operators where: AND = *min* function, OR = *max*, and NOT = *additive complement*. This principle can be represented graphically for a two fuzzy intervals *A* and *B* where *A*= [5, 8] with a trapezoid membership function shown in Figure 7, and *B* = 4 with a triangular membership function shown in the same figure.

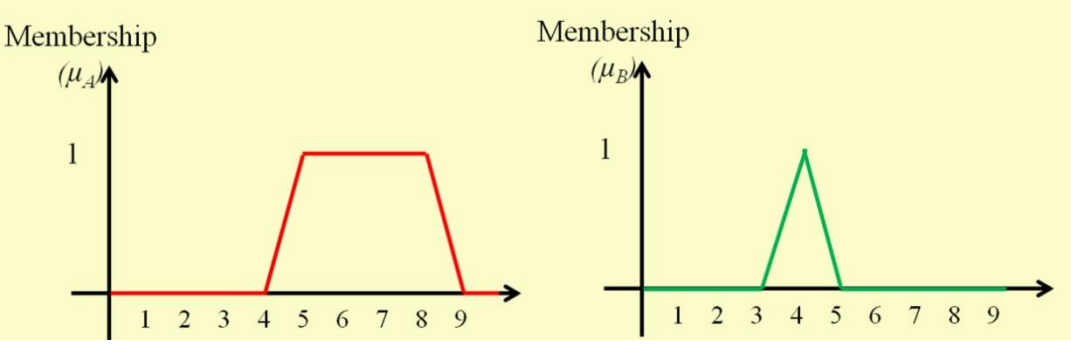


Figure 7. Membership functions for the fuzzy sets A=[5, 8] (left) and B=4 (right)

The application of the AND operator on both membership functions $\mu_A$ AND $\mu_B$ gives their intersection ($\mu_A \cap \mu_B$) and results in the can be obtained by applying the *minimum* function as shown in blue color at Figure 8.

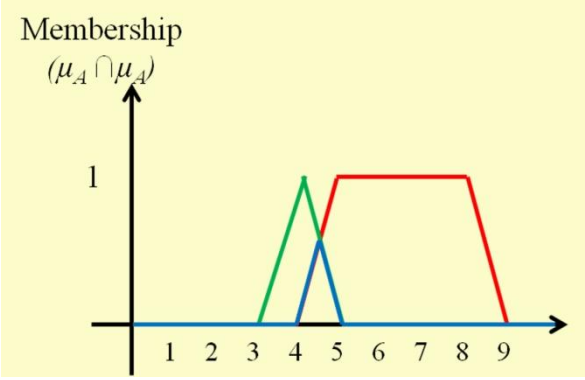


Figure 8. The AND operator applied on the two fuzzy sets A and B

The OR operator can be replaced with the *maximum* function and its application on the two fuzzy sets gives their union ($\mu_A \cup \mu_B$) as shown in blue color at Figure 9.

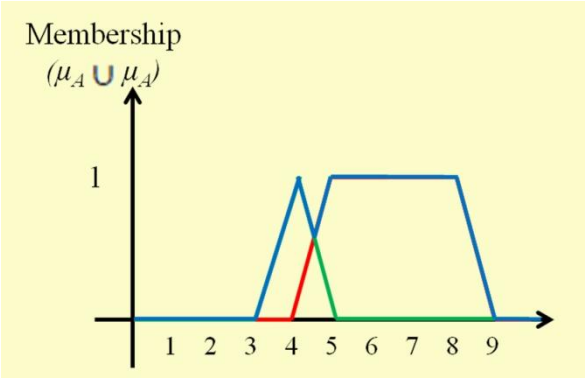


Figure 9. The OR operator applied on the two fuzzy sets A and B

Finally, the negation of the membership function $\mu_A$ can be seen as the blue curve in .

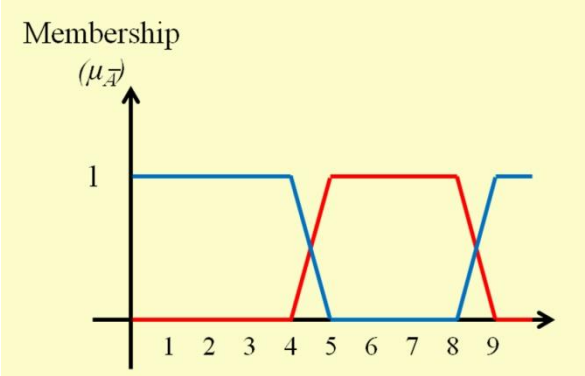


Figure 10. . The NOT operator applied on the fuzzy set A

The application of these principles and the fuzzy logic controller can be demonstrated with a practical textile example for a quantitative grading of cotton fibers using the fuzzy membership functions, based on the work presented by Elmoghazy [3] and explained in the following section.

### *3.3.3. Cotton grading*

Cotton fiber is traditionally graded by subjective classing; in which a classer observes the appearance and the color of fibers and touches the fibers to determine their smoothness or roughness to the hand. Quantitatively, the color appearance is measured using color reflectance, *Rd*, and fiber roughness can be evaluated through testing a combination of parameters including fiber friction, bulk resiliency under cyclic pressing and un-pressing of the fibers, and the energy consumed to open the fiber bulk. This simple case illustrates the superiority of expert human judgment of certain phenomena to laboratory testing, as it takes many quantitative parameters and a significant amount of time to make a judgment of cotton grade that a classer can make in a few seconds. When a cotton classer observes, touches and handles the cotton sample, all these quantitative parameters come into play in a very complex interactive fashion that only the human brain supported by experience can comprehend. In recent years, efforts were made to convert subjective grading of cotton into instrumental grading. These efforts involved comparative analysis between subjective measures and instrumental measures, with expert classer opinion being one of the main references for calibrating the objective measures. This is a case where fuzzy logic analysis can be very useful as it can result in characterizing human judgment on the basis of a combination of expert opinion supported by instrumental grading.

To demonstrate the above point, let us take one of the quantitative parameters, say color reflectance, *Rd*, and assume that it is the only quantitative parameter required to make a judgment on the cotton grade. The traditional approach to relating the quantitative value (*Rd* values) to human judgment has been based on the discrete classification of the parameter of interest. For example, experts in the field may come together and establish the following criteria:

• Poor grade < 75 *Rd*, • Middle grade 75–80 *Rd*, • Good grade > 80 *Rd*

These criteria may be represented by the crisp set shown in Figure 11. The number zero is given for both poor and good grades, and the number 1.0 is given for middle grade cotton. This approach is very common in many human judgments owing to its simplicity. Indeed, people often use terms such as ‘good or bad’, and ‘true or false’ to characterize their judgments. The problem with this approach, however, is that it often masks a great deal of information and judgment resolution. A cotton classer being a human may easily judge extreme conditions such as extremely dark or extremely light color. As colors deviates from these extremes, human judgment may become progressively fuzzy. In this case, the crisp set with binary scoring may not be accurate.

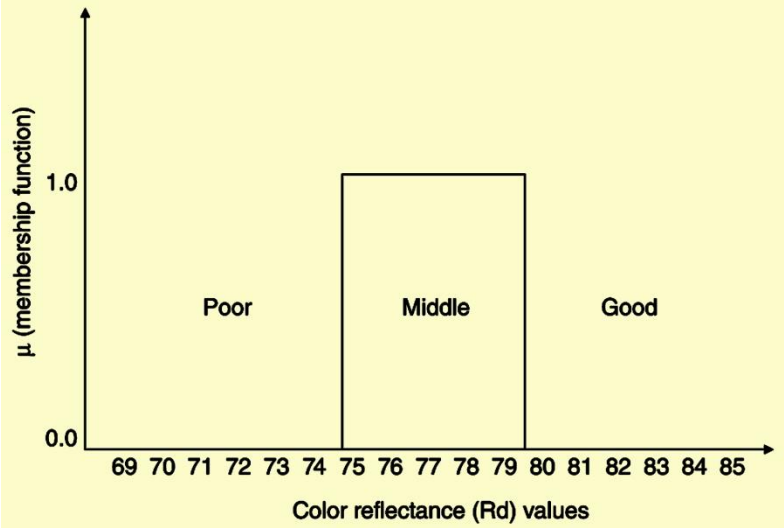


Figure 11. Crisp set–middle cotton grade (Reproduced from Ref. [3])

If the grading criteria of *Rd* listed above are accepted by experts, we will find that the classer's judgment of very poor and very good grades represent a clear cut. As we approach the middle grade from the low or the high value, different classers may have different opinions about whether the cotton is 'poor to middle', 'definitely middle' or 'middle to good'. This fuzziness can be resolved by the use of the fuzzy logic membership function. This is developed based on asking a panel of classing experts to vote on the matter. The fraction of the panel that regards cotton with a given *Rd* value as 'middle grade' will give a number from 0 and 1, which will indicate the strength of their judgment. An expert who examines a cotton sample of, say, 77 *Rd* value, may determine that it is too close to the poor category, but not close enough to be judged poor. In this case, he/she may give a number close to zero (say 0.3). This number implies the uncertainty of his/her judgment about how to consider this cotton sample, a 'middle' or 'poor grade'. Similarly, an expert who examines a cotton sample of, say, 79 *Rd* value, may determine that it is too close to the good category. As a result, he/she may again give a number close to zero; say 0.2 which implies a judgment uncertainty of middle grade. Finally, an expert who examines a cotton sample of, say, 78 *Rd* value, may determine that it is definitely middle-grade cotton and gives a number 1 or close to 1 (say, 0.8) to imply judgment certainty.

The above example illustrates one way in which fuzzy logic handles the extent of certainty of human judgment; it is through a continuous pattern, not a discrete pattern, of decision-making. The fuzzy set derived in this way is illustrated in Figure 12. It indicates a more resolute judgment, particularly in the representation of 'middle cotton grade'.

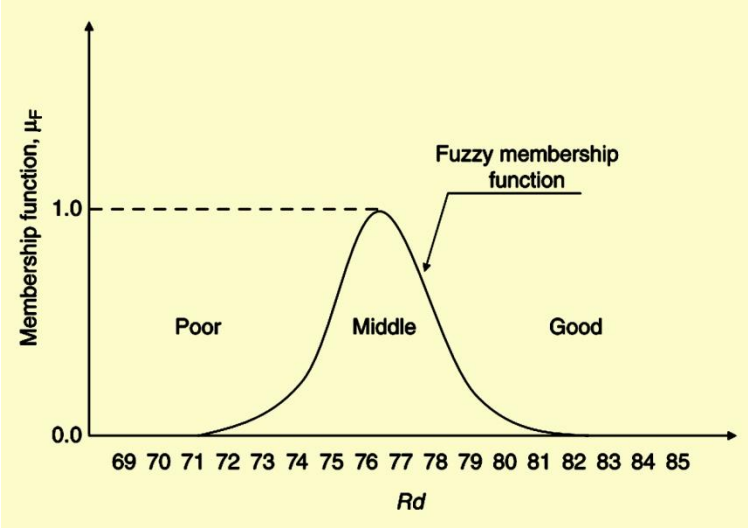


Figure 12. Fuzzy set–middle cotton grade (Reproduced from Ref. [3])

Fuzzy sets corresponding to poor or good grades might also be defined in the same way. This is illustrated in Figure 13 where the three sets developed overlap. This reflects the fact that a certain value of color reflectance may lead to a two-way judgment. The figure also indicates that there are no sharp changes in *Rd* groups. As *Rd* increases, membership of the 'good grade' gradually increases to 1.0; or as *Rd* decreases, membership of the 'poor grade' gradually declines to 0. Thus, each of the fuzzy sets described in Figure 13 can be regarded as the definition of a corresponding linguistic value, in this case 'poor grade', 'good grade' and 'middle grade'.

This point leads us to two different variables related to grade:

1. Color in *Rd*: a numerical variable with integer numerical values

2. Color group: a linguistic variable taking the linguistic values 'poor grade', 'good grade' and 'middle grade'.

In the context of determining cotton grade, color *Rd*, though numerical, represents the underlying measurement, which in this case drives everything else, but it is not easy to be interpreted. Color group, on the other hand, ranges over a more limited set of values. This makes it easier to comprehend and easier to use.

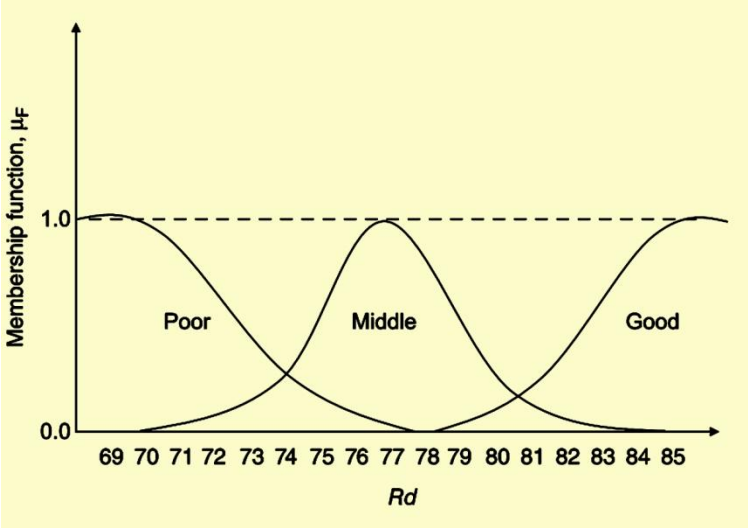


Figure 13. Fuzzy sets for cotton grade categories (Reproduced from Ref. [3])

This example illustrates the concept of membership function where, in practice, there are many parameters and many ways to characterize each parameter. In addition, parameters may indeed interact with one another leading to many decisions regarding the subjective measure. As we indicated earlier, cotton grade reflects many parameters that are interacting in a complex manner. Accordingly, the above approach should be expanded to accommodate all these parameters, individually and combined. This gives rise to a combination matrix of inputs, in response to which a fuzzy logic controller or a decision scheme can be developed.

It should be pointed out that the above example was only presented as a simple demonstration of some elements of fuzzy logic analysis. In recent years, fuzzy logic analysis has been used in numerous applications in which human judgment was accurately represented or simulated for control or modeling. In the area of developing fibrous products, many subjective phenomena such as appearance, hand and comfort can be characterized using fuzzy logic analysis.

**References:**

[1] G. F. Luger, *Artificial intelligence: structures and strategies for complex problem solving*. Pearson education, 2005.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, Third edition*. 2014.

[3] Y. Elmogahzy, *Engineering Textiles: Integrating the Design and Manufacture of Textile Products*. Elsevier Science, 2008.