Chapter 2
Theory and Background

This chapter overviews the background and main definitions and basic concepts, useful to the development of this investigation work.

2.1 Neural Networks

Neural networks are composed of many elements (Artificial Neurons), grouped into layers and are highly interconnected (with the synapses), this structure has several inputs and outputs, which are trained to react (or give values) in a way you want to input stimuli (R values). These systems emulate in some way, the human brain. Neural networks are required to learn to behave (Learning) and someone should be responsible for the teaching or training (Training), based on prior knowledge of the problem environment problem [1].

Artificial neural networks are inspired by the architecture of the biological nervous system, which consists of a large number of relatively simple neurons that work in parallel to facilitate rapid decision-making [2].

An artificial neural network (ANN) is a distributed computing scheme based on the structure of the nervous system of humans. The architecture of a neural network is formed by connecting multiple elementary processors, this being an adaptive system that has an algorithm to adjust their weights (free parameters) to achieve the performance requirements of the problem based on representative samples [3, 4].

The most important property of artificial neural networks is their ability to learn from a training set of patterns, i.e. is able to find a model that fit the data [5, 6].

The artificial neuron consists of several parts (see Fig. 2.1). On one side are the inputs, weights, the summation, and finally the adapter function. The input values are multiplied by weights and added: $\sum x_i w_{ij}$. This function is completed with the addition of a threshold amount i. This threshold has the same effect as an entry with value -1. It serves so that the sum can be shifted left or right of the origin. After

F. Gaxiola et al., *New Backpropagation Algorithm with Type-2 Fuzzy Weights for Neural Networks*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-34087-6_2

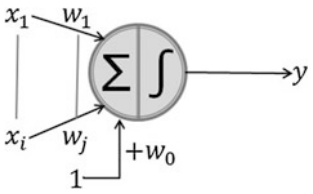

Fig. 2.1 Scheme of an artificial neuron

addition, we have the function f applied to the sum, resulting in the final value of the output, also called y_i [7], obtaining following the equation:

$$y_i = \left(\sum_{i=1}^{n} x_i w_{ij} \right) \tag{2.1}$$

where f may be a nonlinear function with binary output ± 1, a linear function f (z) = z, or as sigmoid logistic function: $f(z) = \frac{1}{1+e^{-z}}$.

A neural network is a system of parallel processors connected as a directed graph. Schematically each processing element (neuron) of the network is represented as a node. These connections establish a hierarchical structure that is trying to emulate the physiology of the brain as it looks for new ways of processing to solve real world problems. What is important in developing the techniques of Neural Networks (NNs) is if its useful to learn behavior, recognize and apply relationships between objects and plots of real-world objects themselves. In this sense, artificial neural networks have been applied to many problems of considerable complexity. Its most important advantage is in solving problems that are too complex for conventional technologies, problems that have no solution and/or that the algorithm of the solution is very difficult to find [8].

2.2 Ensemble Neural Networks

An Ensemble Neural Network is a learning paradigm where many neural networks are jointly used to solve a problem [9]. A Neural network ensemble is a learning paradigm where a collection of a finite number of neural networks is trained for the same task [10]. It originates from Hansen and Salamon's work [11], which shows that the generalization ability of a neural network system can be significantly improved through a number of neural networks in ensemble, i.e. training many neural networks and then combining their predictions. Since this technology behaves remarkably well, recently it has become a very hot topic in both neural networks and machine learning communities [12], and has already been successfully applied to diverse areas such as face recognition [13, 14], optical character recognition [15], scientific image analysis [16], medical diagnosis [17, 18], seismic signals classification [19], etc.

In general, a neural network ensemble is constructed in two steps, i.e. training a number of component neural networks and then combining the component predictions.

There are also many other approaches for training the component neural networks. Examples are as follows. Hampshire and Waibel [20] and Wei and Cheng [21] utilize different objective functions to train distinct component neural networks. Cherkauer [16] trains component networks with different number of hidden layers. Maclin and Shavlik [22] initialize component networks at different points in the weight space. Liu et al. [23] and Soltani [24] employ cross-validation to create component networks. Krogh and Vedelsby [25] and Opitz and Shavlik [26] exploit genetic algorithm to train diverse knowledge based component networks. Yao and Liu [27] regard all the individuals in an evolved population of neural networks as component networks [28].

2.3 Type-2 Fuzzy Systems

The basics of fuzzy logic do not change from type-1 to type-2 fuzzy sets, and in general, will not change for any type-n [29–33]. A higher-type number just indicates a higher "degree of fuzziness". Since a higher type changes the nature of the membership functions, the operations that depend on the membership functions change; however, the basic principles of fuzzy logic are independent of the nature of membership functions and hence, do not change. Rules of inference like Generalized Modus Ponens or Generalized Modus Tollens continue to apply.

The structure of the type-2 fuzzy rules is the same as for the type-1 case because the distinction between type-2 and type-1 is associated with the nature of the membership functions. Hence, the only difference is that now some or all the sets involved in the rules are of type-2 [34, 35].

As an example: Consider the case of a fuzzy set characterized by a Gaussian membership function with mean m and a standard deviation that can take values in $[\sigma_1, \sigma_2]$ i.e.,

$$\mu(x) = \exp\left\{-1/2\left[\frac{x-m}{\sigma}\right]^2\right\};\ \sigma\,\epsilon[\sigma_1, \sigma_2] \tag{2.2}$$

Corresponding to each value of σ we will get a different membership curve (as shown in Fig. 2.1). So, the membership grade of any particular x (except x = m) can take any of a number of possible values depending upon the value of σ i.e., the membership grade is not a crisp number, it is a fuzzy set. Figure 2.2 shows the domain of the fuzzy set associated with x = 0.7; however, the membership function associated with this fuzzy set is not shown in Fig. 2.2.

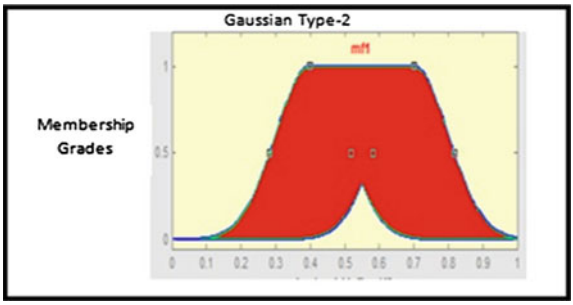

Fig. 2.2 A type-2 fuzzy set representing a type-1 fuzzy set with uncertain standard deviation

2.3.1 Gaussian Type-2 Fuzzy Set

A Gaussian type-2 fuzzy set is one in which the membership grade of every domain point is a Gaussian type-1 set contained in [0, 1].

2.3.2 Interval Type-2 Fuzzy Set

An interval type-2 fuzzy set is one in which the membership grade of every domain point is a crisp set whose domain is some interval contained in [0, 1].

2.3.3 Footprint of Uncertainty

Uncertainty in the primary memberships of a type-2 fuzzy set, $\tilde{A}$, consists of a bounded region that we call the "footprint of uncertainty" (FOU). Mathematically, it is the union of all primary membership functions [36, 37].

2.3.4 Upper and Lower Membership Functions

An "upper membership function" and a "lower membership functions" are two type-1 membership functions that are bounds for the FOU of a type-2 fuzzy set $\tilde{A}$. The upper membership function is associated with the upper bound of the FOU($\tilde{A}$). The lower membership function is associated with the lower bound of the FOU($\tilde{A}$).

2.3.5 Operations of Type-2 Fuzzy Sets

2.3.5.1 Union of Type-2 Fuzzy Sets

The union of $\widetilde{A}_1$ and $\widetilde{A}_2$ is another type-2 fuzzy set, just as the union of type-1 fuzzy sets A_1 and A_2 is another type-1 fuzzy set. More formally, we have the following expression:

$$\widetilde{A}_1 \cup \widetilde{A}_2 = \int_{x \in X} \mu_{\widetilde{A}_1 \cup \widetilde{A}_2}(x)/x \tag{2.3}$$

2.3.5.2 Intersection of Type-2 Fuzzy Sets

The intersection of $\widetilde{A}_1$ and $\widetilde{A}_2$ is another type-2 fuzzy set, just as the intersection of type-1 fuzzy sets A_1 and A_2 is another type-1 fuzzy set. More formally, we have the following expression:

$$\widetilde{A}_1 \cap \widetilde{A}_2 = \int_{x \in X} \mu_{\widetilde{A}_1 \cap \widetilde{A}_2}(x)/x \tag{2.4}$$

2.3.5.3 Complement of a Type-2 Fuzzy Set

The complement of a set is another type-2 fuzzy set, just as the complement of type-1 fuzzy set A is another type-1 fuzzy set. More formally we have:

$$\widetilde{\mathrm{A}'} = \int_{\mathrm{x}} \mu_{\widetilde{\mathrm{A}'1}}(\mathrm{x})/\mathrm{x} \tag{2.5}$$

where the prime denotes complement in the above equation. In this equation $\mu_{\widetilde{\mathrm{A}'1}}$ is the secondary membership function.

2.3.6 Type-2 Fuzzy Rules

Consider a type-2 Fuzzy System having r inputs $x_1 \in X_1, \ldots, x_{\mathrm{r}} \in X_{\mathrm{r}}$ and one output $\mathrm{y} \in \mathrm{Y}$. As in the type-1 case, we can assume that there are M rules; but, in the type-2 case the lth rule has the form:

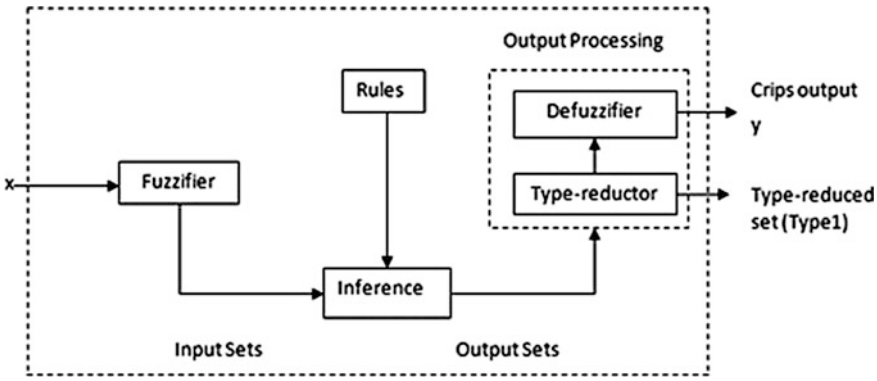

Fig. 2.3 Structure of a type-2 fuzzy logic system

$$R^1 : IF\ x_1\ is\ \widetilde{A}_1^1\ and \ldots x_p\ is\ \widetilde{A}_p^1,\ THEN\ y\ is\ \widehat{Y}^1 \quad \mathrm{I} = \mathrm{I}, \ldots M \tag{2.6}$$

The rules represent a type-2 relation between the input space $X_1 \times \cdots \times X_r$, and space output set space Y, of the type-2 fuzzy system. In the type-2 fuzzy system (Fig. 2.3), as in the type-1 fuzzy system crisp inputs are first fuzzified into fuzzy input sets than then activate the inference block, which in the present case is associated with type-2 fuzzy sets [38].

2.4 Optimization

Regarding optimization, we have the following situation in mind: there exists a search space V, and a function:

$$g : V \to \mathbb{R} \tag{2.7}$$

And the problem is to find: ***arg ming(v)***.

$$v \in V \tag{2.8}$$

Here, V is vector of decision variables, and g is the objective function. In this case we have assumed that the problem is one of minimization, but everything we say can of course be applied mutatis mutandis to a maximization problem. Although specified here in an abstract way, this is nonetheless a problem with a huge number of real-world applications.

In many cases the search space is discrete, so that we have the class of combinatorial optimization problems (COPs). When the domain of the g function is continuous, a different approach may well be required, although even here we note that in practice, optimization problems are usually solved using a computer, so that

in the final analysis the solutions are represented by strings of binary digits (bits) [39, 40].

There are several optimization techniques for neural networks, some of these are: evolutionary algorithms [41], ant colony optimization [42] and Particle swarm [43].

2.5 Genetic Algorithms

Genetic algorithms were introduced for the first time by a Professor of the University of Michigan, John Holland [44]. A genetic algorithm is a mathematical highly parallel algorithm that transforms a set of mathematical individual objects with regard to the time using operations based on evolution. The Darwinian laws of reproduction and survival of the fittest can be used, and after having appeared of natural form a series of genetic operations that can be used [45, 46]. Each of these mathematical objects is usually a chain of characters (letters or numbers) of fixed length that adjusts to the model of the chains of chromosomes, and one associates to them a certain mathematical function that reflects the fitness.

A Genetic Algorithm (GA) [47] assumes that a possible solution to a problem can be seen as an individual and is represented by a set of parameters. These parameters are the genes of a chromosome, representing the structure of the individual. This chromosome is evaluated by a fitness function, providing a fitness value to each individual, as to the suitability it has to solve the given problem. Through genetic evolution using crossover operations (matching of individuals to produce children) (example: one point crossover, see Fig. 2.4) and mutation that simulates the biological behavior, combined with a selection process (example: single mutation, see Fig. 2.4), the population of individuals is eliminating those with less ability, and tends to improve overall fitness of the population, to produce a solution close to optimal performance for the given problem. John Holland was aware of the importance of natural selection [48], and in the late 60s developed a technique that allowed incorporating it into a computer program. His goal was to get computers to learn for themselves. The technique proposed by Holland was originally called "reproductive plans" [49].

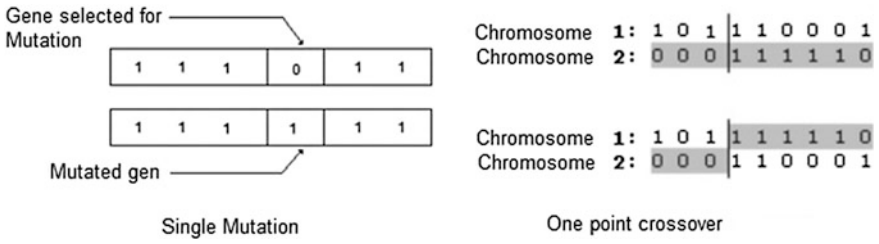

Fig. 2.4 Example of genetic operators

The fitness function determines the best individual where each individual must represent a complete solution to the problem that is being optimized. Therefore the three most important aspects of using genetic algorithms are:

(a) Definition of the fitness function.
(b) Definition and implementation of the genetic representation.
(c) Definition and implementation of the genetic operators.

Therefore, to solve a problem using a GA should take the following steps:

1. Define the chromosome or individual's structure and its representation.
2. Determine the evaluation criteria or adaptation of individuals.
3. Determine the genetic operators to use.
4. Define the stopping criterion of the algorithm.
5. Define a criterion for replacement between consecutive generations.

The basic genetic algorithm is as follows:

Step 1 Represent the problem variable domain as a chromosome of a fixed length; choose the size of a chromosome population N, the crossover probability pc and the mutation probability pm.

Step 2 Define a fitness function to measure the performance, or fitness, of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.

Step 3 Randomly generate an initial population of chromosomes of size $N = x_1, x_2, \ldots, x_n$.

Step 4 Calculate the fitness of each individual chromosome: $f(x_1), f(x_2), \ldots, f(x_n)$

Step 5 Select a pair of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness.

Step 6 Create a pair of offspring chromosomes by applying the genetic operators—crossover and mutation.

Step 7 Place the created offspring chromosomes in the new population.

Step 8 Repeat Step 5 until the size of the new chromosome population becomes equal to the size of the initial population, N.

Step 9 Replace the initial (parent) chromosome population with the new (offspring) population.

Step 10 Go to Step 4, and repeat the process until the termination criterion is satisfied.

2.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a bio-inspired optimization method proposed by Eberhart and Kennedy [50] in 1995. PSO is a search algorithm based on the behavior of biological communities that exhibits individual and social behavior [38, 51], and examples of these communities are groups of birds, schools of fish and swarms of bees [52–54].

A PSO algorithm maintains a swarm of particles, where each particle represents a potential solution. In analogy with the paradigms of evolutionary computation, a swarm is similar to a population, while a particle is similar to an individual. In simple terms, the particles are "flown" through a multidimensional search space, where the position of each particle is adjusted according to its own experience and that of its neighbors. Let x_i denote the position i in the search space at time step t; unless otherwise stated, t denotes discrete time steps. The position of the particle is changed by adding a velocity, $v_i(t)$, to the current position, i.e.

$$\mathrm{x_i(t+1)} = \mathrm{x_i(t)} + \mathrm{v_i(t+1)} \tag{2.9}$$

With $\mathrm{x_i}(0) \sim U(X_{min}, X_{max})$.

It is the velocity vector the one that drives the optimization process, and reflects both the experimental knowledge of the particles and the information exchanged in the vicinity of particles.

For PSO, the particle velocity is calculated as:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)\left[\hat{y}_{ij}(t) - x_{ij}(t)\right] \tag{2.10}$$

where $v_{ij}(t)$ is the velocity of the particle i in dimension j at time step t, c_1 and c_2 are the positive acceleration constants used to scale the contribution of cognitive and social skills, $\hat{y}_{ij}(t)$ is the best position, respectively, and $r_{1j}(t)$, and $r_{2j}(t) \sim U(0,1)$ are random values in the range [0, 1].

The best personal position in the next time step $t + 1$ is calculated as:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(x_i(t+1)) \geq f\, y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(x_i(t+1)) > f\, y_i(t)) \end{cases} \tag{2.11}$$

where $f : \mathbb{R}^{nx} \rightarrow \mathbb{R}$ is the fitness function, as with EAs, the goal is measuring the fitness function closely corresponding to the optimal solution, for example the objective function quantifies the performance, or the quality of a particle (or solution).

The overall best position $\hat{y}(t)$, at time step t, is defined as:

$$\hat{y}(t) \epsilon \{y_o(t), \ldots, y_s(t)\} f(y(t)) = \min\{f(y_o(t)), \ldots f(y_s(t)),\} \tag{2.12}$$

where s is the total number of particles in the swarm, more importantly, the above equation defining and establishing $\hat{y}(t)$ the best position is uncovered by either of

the particles so far as this is usually calculated as the best personal best position. The global best position can also be selected from the particles of the current swarm, in which case is expressed as:

$$\hat{y}(t) = \min\{f(x_o(t)), \ldots f(x_{ns}(t)),\} \tag{2.13}$$

A few important and interesting modifications in the basic structure of PSO are as follows:

Shi and Eberhart [55] introduced a new parameter called inertia weight (ω) into the velocity vector, Eq. (2.10), after which the equation becomes:

$$v_{ij}(t+1) = \omega * v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_1 r_{1j}(t)\left[\hat{y}_{ij}(t) - x_{ij}(t)\right] \tag{2.14}$$

The inertia weight ω is employed to control the impact of the previous history of velocities on the current velocity, thereby influencing the trade-off between global and local exploration abilities of the particles. It can be a positive constant or even a positive linear or nonlinear function of time. A larger inertia weight encourages global exploration while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Suitable selection of the inertia weight provides a balance between global and local exploration abilities and thus requires less iteration on an average to find the optimum [46]. Initially the inertia weight was kept static during the entire search process for every particle and dimension. However, with the due course of time dynamic inertia weights were introduced.

Clerc [56, 57] introduced a new parameter called constriction factor 'K' that improves PSO's ability to constrain and control velocities. Eberhart and Shi [45] later verified that K, combined with constraints on V_{max}, significantly improved the PSO performance. The value of the constriction factor is calculated as follows:

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = c_1 + c_2, \varphi > 4 \tag{2.15}$$

With the constriction factor K, the PSO equation for computing the velocity is:

$$v_{ij}(t+1) = K\left\{v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_1 r_{1j}(t)\left[\hat{y}_{ij}(t) - x_{ij}(t)\right]\right\} \tag{2.16}$$

Usually, when the constriction factor is used, φ is set to 4.1 ($c_1 = c_2 = 2.05$), which gives the value of constriction factor K as 0.729. Carlisle and Dozier [58] showed that cognitive and social values of $c_1 = 2.8$ and $c_2 = 1.3$ also yield good results for their chosen set of problems.

The constriction approach is effectively equivalent to the inertia weight approach. Both approaches have the objective of balancing exploration and exploitation, and thereby improving convergence time and the quality of solution

found. It should be noted that low values of ω and K result in exploitation with little exploration, while large values result in exploration with difficulties in refining solutions [51].

2.7 Time Series

A time series is a set of observations for the sequence of data x_t each one being recorded at a specific time. A discrete time series is one in which the set T_O of times at which observations are made is a discrete set, as is the case, for example, when observations are made at fixed time intervals. A continuous time series are obtained when observations are recorded continuously over some time interval, e.g., when $T_0 \in [0, 1]$

Loosely speaking, a time series $\{X_t, t = 0, \pm 1, \ldots\}$ is said to be stationary if it has statistical properties similar to those of the "time-shifted" series $\{X_t, t = 0, \pm 1, \ldots\}$, for each Restricting attention to those properties that depend only on the first-and second-order moments of $\{X_t\}$, we can make this idea precise with the following definitions. Let $\{X_t\}$ be a time series with $\left[EX_t^2\right] < \infty$. Then mean function of $\{X_t\}$ is $\mu_x(t) = E(X_t)$.

The covariance function of $\{X_t\}$ is:

$$Y_x(r, s) = Cov(X_r, X_s) = E[(X_r - \mu_x(r))(X_s - \mu_x(s))] \tag{2.17}$$

for all integer r and s.

$\{X_t\}$ is (weakly) stationary if:

(i) $\mu_x(t)$ is independent of t.
(ii) $Y_x(t + h(t))$ is independent of t for each t.

Remark 1 Strict stationary time series $\{X_1, t = 0, \pm 1, \ldots\}$ is defined by the condition that $(X_1 \ldots X_n)$ and $(X_{(1+h)} \ldots X_{(n+h)})$ have some joint distributions for all integers h and $n > 0$. It is easy to check that if $\{X_t\}$ is also weakly stationary. Whenever we the use term stationary well shall mean weakly stationary.

Remark 2 In view of condition (ii), whenever we use term covariance function with reference to a stationary time series $\{X_t\}$ we shall mean the function Y_x of one variable, defined [40, 59–64] as:

$$Y_x(h) := Y_x(h, 0) = Y_x(t + h, t) \tag{2.18}$$

The function $Y_x(.)$ will be referred to as the auto covariance function and $Y_x(h)$ as its value at log h [60].

Let $\{X_t\}$ be a stationary time series. The autocovariance function (ACVF) of $\{X_t\}$ at log h is:

$$\mathrm{Y_x(h)} = \mathrm{Cov}(\mathrm{X_{(1+h)}}, X_t) \tag{2.19}$$

The autocorrelation function (ACF) of $\{X_t\}$ at log h is

$$p_x \equiv \frac{Y_x(h)}{Y_x(0)} = Cor(\mathrm{X_{(1+h)}}, X_t) \tag{2.20}$$

2.8 Human Iris Biometrics

The biometrics systems can operate like a identification system or like a verification system. In the first case the objective of the system is identify the individual taking the biometric characteristic and comparing with the biometric characteristics existents in the database. In the first case, the objective of the system is identify the individual, this action is achieved with the obtaining the biometric characteristic of the individual and compares with the biometric characteristics in existence in the database.

In the second case the individual provide at system their identity, the system compare their biometric characteristic that is associated with the database to confirm that the individual really is he.

The first use of the iris was presented in Paris, where criminals were classified according to the color of their eyes following a proposal by the French ophthalmologist Bertillon [65]. Research in visual identification technology began in 1935. During that year an article appeared in the 'New York State Journal of Medicine', which suggested that "the pattern of arteries and veins of the retina could be used for unique identification of an individual" [66].

After researching and documenting the potential use of the iris as a tool to identify people, ophthalmologists Flom and Safir patented their idea in 1987 [67]; and later, in 1989, they patented algorithms developed with the mathematician Daugman. Thereafter, other authors developed similar approaches [66].

In 2001, Daugman also presented a new algorithm for the recognition of people using the biometric measurement of Iris [68].

In 2005, Roy proposes an iris recognition system for the identification of persons using support vector machine [69].

In 2006, Cho and Kim presented a new method to determine the winner in LVQ neural network [70].

In 2009, Sarhan used the discrete cosine transform for feature extraction and artificial neural network for recognition [71]; Abiyev and Altunkaya presented the iris recognition system using neural network [72].

The literature has well documented the uniqueness of visual identification. The iris is so unique that there are no two irises alike, even twins, in all humanity. The probability of two irises producing the same code is 1 in 1078, becoming known that the earth's population is estimated at approximately 1010 million [73], it is almost impossible to occur.

2.9 Historical Development

The backpropagation algorithm and its variations are the most useful basic training methods in the area of research of neural networks. However, these algorithms are too slow for practical applications.

When applying the basic backpropagation algorithm to practical problems, the training time can be very high. In the literature we can find that several methods have seen proposed to accelerate the convergence of the algorithm [3–5].

There exists many works about adjustment or managing of weights but only the most important and relevant for this research will be considered here.

Momentum method—Rumelhart, Hinton y Williams suggested adding in the increased weights expression a momentum term β, to filter the oscillations that can be formed a higher learning rate that lead to great change in the weights [7, 74].

Adaptive learning rate—focuses on improving the performance of the algorithm by allowing the learning rate changes during the training process (increase or decrease) [74].

Conjugate Gradient algorithm—A search of weight adjustment along conjugate directions. Versions of conjugate gradient algorithm differ in the way in which a constant βk is calculated.

- Fletcher-Reeves Update [75].
- Polak-Ribiere Update [75].
- Powell-Beale Restart [76, 77].
- Scaled Conjugate Gradient [78].

Kamarthi and Pittner [79], focused in obtaining a weight prediction of the network at a future epoch using extrapolation.

Ishibuchi et al. [80], proposed a fuzzy network where the weights are given as trapezoidal fuzzy numbers, denoted as four trapezoidal fuzzy numbers for the four parameters of trapezoidal membership functions.

Ishibuchi et al. [81], proposed a fuzzy neural network architecture with symmetrical fuzzy triangular numbers for the fuzzy weights and biases, denoted by the lower, middle and upper limit of the fuzzy triangular numbers.

Feuring [82], based on the work by Ishibuchi where triangular fuzzy weights are used, developed a learning algorithm in which the backpropagation algorithm is used to compute the new lower and upper limits media weights. The modal value of the new fuzzy weight is calculated as the average of the new computed limits.

Castro et al. [83], uses interval type 2 fuzzy neurons for the antecedents and interval of type 1 fuzzy neurons for the consequences of the rules. This approach handles the weights as numerical values to determine the input of the fuzzy neurons, as the scalar product of the weights for the input vector.

Gedeon [84], using a discrete selection (following the Hebbian paradigm) performs the weight adjustment. Monirul and Murase [85], as a strategy used the same weights in epochs where the output does not change. Meltser et al. [86], performed a weight adjustment of the network through the BFGS Quasi-Newton method

(Broyden-Fletcher-Goldfarn-Shanno). Barbouinis and Theocharis [87], performed the weights updating using the identification of recursive error prediction (RPE). Yeung et al. [88], used a new training objective function to adjust the weights for a network with radial basis functions.

Neville et al. [89], worked with sigma-pi networks, which are transformed for performing a second task of assignation for which they were initially trained. Casasent and Natarajan [90] used weights with complex values and the non linear square law. Yam and Chow [91] developed an algorithm to find the initial optimal weights of feed forward neural networks based on the Cauchy inequality and a linear algebraic method. Draghici [92], calculates a range of weights for a category of given problems and ensures that the network has the capacity to solve the given problems with integer weights in that range.

There are also recent works on type-2 fuzzy logic that have been developed in time series prediction, like that of Castro et al. [93], and other researchers [29, 94, 95].

In literature, the bio-inspired algorithms have proved that using them for optimization in different areas of research is possible find an optimal results. There are many works of different bio-inspired algorithms, but only works of the genetic algorithm and particle swarm optimization and the most important and relevant for this paper will be considered here: Valdez et al. [96], performed a hybridization of GA and PSO algorithm and integral the concept of fuzzy logic in it, and other researchers [97, 98].

References

1. Ni, H., Yiu, H.: Exchange rate prediction using hybrid neural networks and trading indicators. Neurocomputing **72**(13–15), 2815–2823 (2009)
2. Dow, J.: Indexes. http://www.djindexes.com. Accessed 5 Sept 2010
3. Cazorla, M., Escolano, F.: Two Bayesian methods for junction detection. IEEE Trans. Image Process. **12**(3), 317–327 (2003)
4. Martinez, G., Melin, P., Bravo, D., Gonzalez, F., Gonzalez, M.: Modular neural networks and fuzzy Sugeno integral for face and fingerprint recognition. Adv. Soft Comput. **34**, 603–618 (2006)
5. De Wilde, P.: The magnitude of the diagonal elements in neural networks. Neural Netw. **10**(3), 499–504 (1997)
6. Salazar, P.A., Melin, P., Castillo, O.: A new biometric recognition technique based on hand geometry and voice using neural networks and fuzzy logic. Soft Comput. Hybrid Intell. Syst. 171–186 (2008)
7. Phansalkar, V.V., Sastrq, P.S.: Analysis of the back-propagation algorithm with momentum. IEEE Trans. Neural Netw. **5**(3), 505–506 (1994)
8. Park, Y.N., Murray, T.J., Chen, C.: Predicting sun spots using a layered perceptron neural network. IEEE Trans. Neural Netw. **1**(2), 501–505 (1996)
9. Sharkey, A.: Combining Artificial Neural Nets: Ensemble and MODUlar Multi-Net Systems. Springer, London (1999)
10. Sharkey, A.: One combining Artificial of Neural Nets. Department of Computer Science University of Sheffield, U.K. (1996). Shimshoni, Y.: Intrator classification of seismic signal by

integrating ensemble of neural networks. IEEE Trans. Signal Process. **461**(5), 1194–1201 (1998)
11. Hansen, L.K., Salomon, P.: Ensemble methods for handwritten digit recognition. In: Proceedings of IEEE Workshop on Neural Networks for Signal Processing, Helsingoer, Denmark, pp. 333–342. IEEE Press, Piscataway, NJ (1992)
12. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley (1989)
13. Gutta, S., Wechsler, H.: Face recognition using hybrid classifier systems. In: Proceedings of ICNN-96, Washington, DC, pp. 1017–1022. IEEE Computer Society Press, Los Alamitos, CA (1996)
14. Hansen, L.K., Salomon, P.: "Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. **12**(10), 993–1001 (1990)
15. Drucker, H., Schapire, R., Simard P.: Improving performance in neural networks using a boosting algorithm. In: Hanson, S.J., Cowan, J.D., Giles, C.L. (eds) Advances in Neural Information Processing Systems 5, Denver, CO, Morgan Kaufmann, San Mateo, CA, pp. 42–49 (1993)
16. Cherkauer, K.J.: Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks. In: Chan, P., Stolfo, S., Wolpert, D. (eds) Proceedings of AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, pp. 15–21. AAAI Press, Menlo Park, CA (1996)
17. Cunningham, P., Carney, J., Jacob, S.: Stability problems with artificial neural networks and the ensemble solution. Artif. Intell. Med. **20**(3), 217–225 (2000)
18. Mencattini, A., Salmeri, M., Mertazzoni, B., Lojacono, R., Pasero, E., Moniaci, W.: Local meteorological forecasting by type-2 fuzzy systems time series prediction. In: CIMSA 2005—IEEE International Conference on Computational Intelligence for Measurement Systems and Applications Giardini Naxos, Italy, pp. 20–22 (2005)
19. Singh, S.R.: A simple method of forecasting based on fuzzy time series. Appl. Math. Comput. **186**(1), 330–339 (2007)
20. Hampshire, J., Waibel, A.: A novel objective function for improved phoneme recognition using time- delay neural networks. IEEE Trans. Neural Netw. **1**(2), 216–228 (1990)
21. Wei, L.-Y., Cheng C.-H.: A hybrid recurrent neural networks model based on synthesis features to forecast the Taiwan stock market. Int. J. Innov. Comput. Inf. Control **8**(8), 5559–5571
22. Maclin, R., Shavlik, J.W.: Combining the predictions of multiple classifiers: using competitive learning to initialize neural networks. In: Proceedings of IJCAI-95, Montreal, Canada, Morgan Kaufmann, San Mateo, CA, pp. 524–530 (1995)
23. Liu, F., Quek, C., See, G.. Ng: Neural network model for time series prediction by reinforcement learning. In: Proceedings of the International Joint Conference on the Neural Networks, Montreal, Canada (2005)
24. Soltani, S.: On the use of the wavelet decomposition for time series prediction. Neurocomputing **48**(1–4), 267–277 (2002)
25. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Tesauro, G., Touretzky, D., Leen, T. (eds.) Advances in Neural Information Processing Systems 7, Denver, CO, pp. 231–238. MIT Press, Cambridge, MA (1995)
26. Opitz, D.W., Shavlik, J.W.: Generating accurate and diverse members of a neural network ensemble. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems 8, Denver, CO, pp. 535–541. MIT Press, Cambridge, MA (1996)
27. Yao, X., Liu, F.: Evolving neural network ensembles by minimization of mutual information. Int. J. Hybrid Intell. Syst. **1**, 12–21 (2004)
28. Xue, J., Xu, Z., Watada, J.: Building an integrated hybrid model for short-term and mid-term load forecasting with genetic optimization. Int. J. Innov. Comput. Inf. Control **8**(10), 7381–7391 (2012)

29. Karnik, N., Mendel, M.: Applications of type-2 fuzzy logic systems to forecasting of time-series. Inf. Sci. **120**(1–4), 89–111 (1999)
30. Karnik, N., Mendel, M.: "Introduction to type-2 fuzzy logic systems. IEEE Trans. Signal Process. **2**, 915–920 (1998)
31. Karnik, N., Mendel, J.M.: Operations on type-2 set. Fuzzy Set Syst. **122**, 327–348 (2001)
32. Tong, R.M., Nguyen, H.T.: Fuzzy sets and applications: selected papers by L.A. Zadeh. In: Yager, R.R., Ovchinnikov, S. (eds) Wiley, New York (1987)
33. Zadeh, L.A.: Fuzzy sets, information and control. **8**, 338–353 (1965)
34. Fazel Zarandi, M.H., Rezae, B., Turksen, I.B., Neshat, E.: A type-2 fuzzy rule-based expert system model for stock price analysis. Expert Syst. Appl. **36**, 139–154 (2009)
35. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, p. 614. Prentice Hall, (1997)
36. Jilani, T.A., Burney, S.: A refined fuzzy time series model for stock market forecasting. Phys. A Stat. Mech. Appl. **387**, 2857–2862 (2008)
37. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems. Introduction an New Directions, pp. 213–231. Prentice-Hall Inc. (2001)
38. Eberhart, R., Shi, Y., Kennedy, J.: "Swam Intelligence", San Mateo. Morgan Kaufmann, California (2001)
39. Andreas, A., Sheng, W.: Introduction Optimization, Practical Optimization Algorithms and Engineering Applications, pp. 1–4. Springer (2007)
40. Antoniou, A., Sheng, W.: Practical optimization algorithms and engineering applications "introduction optimization". In: Antoniou, A., Sheng, W. (eds), pp. 1–4. Springer (2007)
41. Yen, J., Langari, R.: Fuzzy Logic: Intelligence, Control and Information. Prentice Hall (1999)
42. Dorigo, M., Stutzle, T.: Ant colony optimization "ant colony optimization theory". A Bradford Book The Milt Press London England, Massachusetts Institute of Technology, pp. 121–151 (2004)
43. Valle, Y., Venayagamoorthy, G.K., Mohagheghi, S., Hernandez, J.-C., Harley, R.G.: Particle swarm optimization: basic concepts, variants and applications in power systems. In: IEEE Transactions on Evolutionary Computation, pp. 171–195 (2008)
44. Holland, J.: Adaptation in natural and artificial systems. University of Michigan Press (1975)
45. Eberhart, R, Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 84–88 (2000)
46. Eberhart, R., Shi, Y.: Comparison between genetic algorithms and particle swarm optimization. In: Proceedings of the Seventh Annual Conference on Evolutionary Programming, pp. 611–616 (1998)
47. Mao, J.: A case study on bagging, boosting and basic ensembles of neural networks for OCR. In: Proceedings of IJCNN-98, Anchorage, AK, vol. 3, pp. 1828–1833. IEEE Computer Society Press, Los Alamitos, CA (1998)
48. Reeves, C., Row, J.: Genetic algorithms: principles and perspectives, "optimization", pp. 4–8. Kluwer Academic Publishers, New York (2002)
49. Whitley, L.D.: Foundations of Genetic Algorithms, vol. 2, p. 332. Morgan Kaufman Publishers (1993)
50. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of 6th International Symposium on Micro Machine and Human Science (MHS), pp. 39–43 (1995)
51. Engelbrech. P.: Fundamentals of Computational of Swarm Intelligence: Basic Particle Swarm Optimization, pp. 93–129. Wiley (2005)
52. Escalante, H.J., Montes, M., Sucar, L.E.: Particle swarm model selection. J. Mach. Learn. Res. **10**, 405–440 (2009)
53. Kennedy, J., Eberhart, R.: Particle swam optimization. In: Proceedings of IEEE International Conference on Neural Network (ICNN), vol. 4, pp. 1942–1948 (1995)
54. Kim, D., Kim, C.: Forecasting time series with genetic fuzzy predictor ensemble. IEEE Trans. Fuzzy Syst. **5**(4) (1997)

55. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proceedings of the IEEE Congress of Evolutionary Computation, pp. 69–73 (1998)
56. Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1951–1957 (1999)
57. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multimodal complex space. IEEE Trans. Evol. Comput. **6**, 58–73 (2002)
58. Carlisle, Dozier, G.: An off-the-shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, USA, pp. 1–6 (2001)
59. Brockwell, P.D., Davis, R.A.: Introduction to Time Series and Forecasting, pp. 1–219. Springer, New York (2002)
60. Cowpertwait, P., Metcalfe, A.: Time Series. Introductory Time Series with R, pp. 2–5. Springer, Dordrecht (2009)
61. Davey, N., Hunt, S., Frank, R.: Time Series Prediction and Neural Networks. University of Hertfordshire, Hatfield, UK (1999)
62. Plummer, E.A.: Time series forecasting with feed-forward neural networks: guidelines and limitations. University of Wyoming (2000)
63. Wei, W.W.S.: Time Series Analysis: Univariate and Multivariate Methods, vol. 1, pp 40–100. Addison-Wesley (1994)
64. Zhang, J., Man, K.F.: Time series prediction using recurrent neural network in multi-dimension embedding phase space. IEEE Int. Conf. Syst. Man Cybernet. **2**, 11–14 (1998)
65. Tisse, C., Torres, L.M., Robert M.: Person identification technique using human iris recognition. Universidad de Montepellier (2000)
66. López, J.,: Estado del Arte: Reconocimiento Automático del Iris Humano", Politécnico Colombiano, y Javier González Patiño, Universidad Nacional de Colombia, Scientia et Technica Año XI, No 29, pp. 77–81 (2005)
67. Khaw, P.: Iris recognition technology for improved authentication, pp. 1–17. Sala de Lectura de Seguridad de la Información, SANS Institute (2002)
68. Daugman, J.: Statistical richness of visual phase information: update on recognizing persons by iris patterns. Int. J. Comput. Vis. **45**(1), 25–38 (2001)
69. Roy, K., Bhattacharya, P.: Iris recognition with support vector machines. Advances in biometrics. Lect. Notes Comput. Sci. **3822**, 486–492 (2005)
70. Cho, S., Kim, J.: Iris recognition using LVQ neural network. Adv. Neural Netw. Lect. Notes Comput. Sci. **3972**, 26–33 (2006)
71. Sarhan, A.: Iris recognition using discrete cosine transform and artificial neural networks. J. Comput. Sci. **5**, 369–373 (2009)
72. Abiyev, R., Altunkaya, K.: Neural network based biometric personal identification with fast iris segmentation. Int. J. Control Autom. Syst. **7**(1), 17–23 (2009)
73. Sánchez, O., y González, J.: Control de Acceso basado en Reconocimiento de Iris. Corporación Universitaria Tecnológica de Bolívar, Facultad de Ingeniería Eléctrica, Electrónica y Mecatrónica, Cartagena de Indias, Monografía, Noviembre 2003, pp. 1–137
74. Hagan, M.T., Demuth, H.B., Beale M.H.: Neural Network Design, pages 736. PWS Publishing (1996)
75. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**, 149–154 (1964)
76. Powell, M.J.D.: Restart procedures for the conjugate gradient method. Math. Program. **12**, 241–254 (1977)
77. Beale, E.M.L.: "A Derivation of Conjugate Gradients. In: Lootsma, F.A. (ed.) Numerical methods for nonlinear optimization, pp. 39–43. Academic Press, London (1972)
78. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. **6**, 525–533 (1993)
79. Kamarthi, S., Pittner, S.: Accelerating neural network training using weight extrapolations. Neural Netw. **12**(9), 1285–1299 (1999)

80. Ishibuchi, H., Morioka, K., Tanaka, H.: A fuzzy neural network with trapezoid fuzzy weights. In: Fuzzy Systems, vol. 1, IEEE World Congress on Computational Intelligence, pp. 228–233 (1994)
81. Ishibuchi, H., Tanaka, H., Okada, H.: Fuzzy neural networks with fuzzy weights and fuzzy biases. In: IEEE International Conference on Neural Networks, vol. 3, pp. 160–165 (1993)
82. Feuring, T.: Learning in fuzzy neural networks, neural networks. In: IEEE International Conference on, vol. 2, pp. 1061–1066 (1996)
83. Castro, J., Castillo, O., Melin, P., Rodríguez-Díaz, A.: A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks. Inf. Sci. **179**(13), 2175–2193 (2009)
84. Gedeon, T.: Additive neural networks and periodic patterns. Neural Netw. **12**(4–5), 617–626 (1999)
85. Monirul Islam, Md., Murase, K.: A new algorithm to design compact two-hidden-layer artificial neural networks. Neural Netw. **14**(9), 1265–1278 (2001)
86. Meltser, M., Shoham, M., Manevitz, L.: Approximating functions by neural networks: a constructive solution in the uniform norm. Neural Netw. **9**(6), 965–978 (1996)
87. Barbounis, T.G., Theocharis, J.B.: Locally recurrent neural networks for wind speed prediction using spatial correlation. Inf. Sci. **177**(24), 5775–5797 (2007)
88. Yeung, D., Chan, P., Ng, W.: Radial basis function network learning using localized generalization error bound. Inf. Sci. **179**(19), 3199–3217 (2009)
89. Neville, R.S., Eldridge, S.: Transformations of Sigma–Pi nets: obtaining reflected functions by reflecting weight matrices. Neural Netw. **15**(3), 375–393 (2002)
90. Casasent, D., Natarajan, S.: A classifier neural net with complex-valued weights and square-law nonlinearities. Neural Netw. **8**(6), 989–998 (1995)
91. Yam, J., Chow, T.: A weight initialization method for improving training speed in feedforward neural network. Neurocomputing **30**(1–4), 219–232 (2000)
92. Draghici, S.: On the capabilities of neural networks using limited precision weights. Neural Netw. **15**(3), 395–414 (2002)
93. Castro, J., Castillo, O., Melin, P., Mendoza, O., Rodríguez-Díaz, A.: An interval type-2 fuzzy neural network for chaotic time series prediction with cross-validation and Akaike test. Soft Comput. Intell. Control Mobile Robot. 269–285 (2011)
94. Abiyev, R.: A type-2 fuzzy wavelet neural network for time series prediction. Lect. Notes Comput. Sci. **6098**, 518–527 (2010)
95. Pulido, M., Melin, P., Castillo, O.: Genetic optimization of ensemble neural networks for complex time series prediction. IJCNN 202–206 (2011)
96. Valdez, F., Melin, P., Castillo, O.: Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pp. 2114–2119 (2009)
97. Sanchez, D., Melin, P.: Modular neural network with fuzzy integration and its optimization using genetic algorithms for human recognition based on iris, ear and voice biometrics. Soft Comput. Recogn. Based Biometrics 85–102 (2010)
98. Valdez, F., Melin, P., Parra, H.: Parallel genetic algorithms for optimization of modular neural networks in pattern recognition. IJCNN 314–319 (2011)

http://www.springer.com/978-3-319-34086-9

New Backpropagation Algorithm with Type-2 Fuzzy Weights for Neural Networks

Gaxiola, F.; Melin, P.; Valdez, F.

2016, IX, 102 p. 94 illus., Softcover
ISBN: 978-3-319-34086-9