**4. NEURAL NETWORKS FOR SPEECH CLASSIFICATION**

**4.1. Overview**

Nowadays, the use of neural networks has increased in scientific research field, where it is involved in multiple areas, and the reasons that led to use it extensively are:

* It implements the complex mathematical equations as well as simple equations.
* The speed in the implementation of the most complex calculations compared to their conventional counterparts.
* It gives the best results according to the results obtained in comparisons of scientific researches.

There are a large number of different types of networks, but they all are characterized by the following components: a set of nodes, and connections between nodes. The nodes can be seen as computational units; they receive inputs and process them to obtain an output. These units can do simple computations such as forward the input to the output or in some times they work as summation units or can do complex computations like derivation, integration and so on or they might contain another network. The connections determine the information flow between nodes. They can be unidirectional, when the information flows only in one way, or bidirectional, when the information flows in either way [33]. This is a general review of basic components of neural networks and more details about fundamentals of neural networks are illustrated in the following section.

**4.2. Fundamentals of neural networks**

As mentioned above all the neural networks participate several characteristics, these properties are the basis to configure any neural network, and they are [34]:

* Computational unites (nodes).
* Connections between nodes.
* Training the networks.

Figure 4.1 shows a simple structure of neural network and all basic components of neural network.



Figure 4.1: General structure of neural network.

Each neural network contains a potentially huge number of Computational units or nodes, they are responsible of conducting processing on the data depending on the function assigned to them, where there are multiple functions will be mentioned in details later in this chapter, and then broadcast the result value to the units in the next layer to make a processing on them again and so on until it is to get the final result in the output. All these units operate simultaneously, supporting massive parallelism. The units in a network are typically divided into input units, which receive data from the environment (such as raw sensory information); and/or hidden units, which may internally transform the data representation; and output units, which represent decisions or control signals.

The connections between nodes define the topology of the network. There are many topologies and some of them are illustrated in figure 4.2.The connection has a value called weight, typically ranging from -∞ to +∞, the values of all the weights predetermine the network’s computational reaction to any arbitrary input pattern; thus the weights encode the long-term memory of the network. Weights can change as a result of training, but they tend to change slowly, because accumulated knowledge changes slowly [17].



Figure 4.2: Neural network topologies: (a) unstructured, (b) layered, (c) recurrent, (d) modular [17].

Training of neural network can be defined as follows:

It is a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of training is determined by the manner in which the parameter changes take place (Mendel &Mc Claren, 1970). Training the network means resetting the values of the weights of the nodes so the network reeves the required computational behaviour for all patterns that will be classified. In some times changing or modifying the network topology is adopted to train the network, but modifying the weights of the nodes is more general than modifying the topology. in general, reset the value of any weight to zero mean deleting the corresponding node. When the network topology is changed this helps in improving the speed of learning by constructing the class of functions that the network can be capable to learn [17].

In some cases, it is not easy to set the weights that will enable the network to compute complex functions, but in other cases when the network is applied to do simple computations like in linear networks, an analytical solution is applied [34]. In this case the weights are given by equation (4.1) [17]:

$$w\_{ji}=\sum\_{p}^{}\frac{y\_{i}^{p}t\_{j}^{p}}{\left‖y^{p}\right‖^{2}} (4.1)$$

Where *y* is the input vector, *t* is the target vector, and *p* is the pattern index.

In general, networks are nonlinear and multi-layered, and their weights can be trained only by an iterative procedure, such as gradient descent on a global performance measure (Hinton, 1989). Multiple passes (called epochs or iterations) of training are applied on the entire training set to complete the learning of the network. The modification of the weights must be done gently because of the accumulated knowledge is distributed over all the weights, in this case the learning is continued without destroying the previous learning. A learning rate (Ɛ) is a small constant used to control the magnitude of weight modifications. It is important to find a suitable value for the learning rate because the speed of the learning is depend on this value, and also if this value is small, the learning will go forever; but if this value is large, all the previous knowledge will be disrupted. Here, it must be mentioned that there is no analytical method to find the optimal value for the learning rate, just trying different values and then choosing the best one. Most training procedures, including equation (4.1), are essentially variations of the Hebb Rule (Hebb, 1949), which reinforces the connection between two units if their output activations are correlated as in equation (4.2) [17]:

$$ Δw\_{ji}=Ɛy\_{i}y\_{j} (4.2)$$

The network is prepared to activate the second unit if only the first one is known during testing, by reinforcing the correlation between active pairs of units during training. The Delta Rule (also called the Widrow-Hoff Rule) is applied when there is a target value for one of the two units; this is the important variation of the above rule. If there is a correlation between the first unit’s activation yi and the second unit’s error (or potential for error reduction) relative to its target $t\_{j}$, the above rule reinforces the connection between these two units:

$$ Δw\_{ji}=Ɛy\_{i}(t\_{j}-y\_{j}) (4.3)$$

This rule decreases the relative error if $y\_{i}$ contributed to it, so that the network is prepared to compute an output $y\_{j}$ closer to $t\_{j}$ if only the first unit’s activation $y\_{i}$ is known during testing.

In the context of binary threshold units with a single layer of weights, the Delta Rule is known as the Perceptron Learning Rule, and it is guaranteed to find a set of weights representing a perfect solution, if such a solution exists (Rosenblatt, 1962). In the context of multi-layered networks, the Delta Rule is the basis for the backpropagation training procedure [17].

**4.3. Models of neuron**

A neuron is an information-processing unit that is fundamental of the operation of a neural network [35]. In accordance with the biological model, different mathematical models were suggested. The mathematical model of the neuron, which is usually utilized under the simulation of neural networks, is represented in figure 4.3. The neuron receives a set of input signals X1, X2, …,Xn (vector X) which usually are output signals of other neurons. Each input signal is multiplied by a corresponding connection weight (w) analogue of the synapse`s efficiency.

Weighted input signals come to the summation module corresponding to cell body, where their algebraic summation is executed and the excitement level of neuron is determined:

 I = $\sum\_{i=1}^{n}X\_{i}w\_{i}$ (4.4)



Figure 4.3: Mathematical neuron [36].

The output signal of neuron is determined by conducting the excitement level through the function f, called the activation function

 y = f (I - Ɵ) (4.5)

Where Ɵ, is the threshold of the neuron that has the effect of lowering the net input of the activation function. Usually the following activation functions are utilized as function f [36]:

* **Liner function (see figure 4.4):**

This type of functions is not used very often because it’s not very powerful, so nonlinear functions are suggested. Equation (4.6) represents threshold activation function.

 y = I (4.6)



Figure 4.4: Linear function.

* **Binary (threshold) function (see figure 4.5):**

It is a nonlinear function. In this type of functions, the output of the neuron takes on the value of 1 if the total internal activity level of that neuron is nonnegative and 0 otherwise. Equation (4.7) represents threshold activation function.

$ y=\left\{\begin{array}{c}1, if I\geq Ɵ\\0, if I<Ɵ\end{array}\right.$ (4.7)



Figure 4.5: Binary function.

* **Sigmoid function (see figure 4.6):**

The most common used function. In this thesis sigmoidal function is used as an activation function for neurons in artificial neural network. It may be tan or logistic as in equation (4.8).

$ y = \frac{1}{1+exp-(I)}$ (4.8)



Figure 4.6: Sigmoid function.

**4.4. Feed-Forward neural networks**

There are two types of architectures of neural networks they are Fully-connected, and Hierarchical. In fully-connected architecture all its elements are connected with each other. The output of every neuron is connected with inputs of all others and its own inputs. The number of connections in a fully-connected neural network is equal to c\*c, because of c links for each neuron. In the hierarchical architecture, the neural network may be differentiated by the neurons grouped into particular layers or levels. Each neuron in any hidden layer is connected with every neuron in the previous and the next layers.

In terms of the signal transference direction in networks, they are differentiated as networks with feedback loops (Feed-Back or Recurrent) and without feedback loops (Feed-Forward). In this thesis Feed-Forward networks were explained in brief details because they are the most used type.

The Feed-Forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. Feed-Forward neural networks can be either Single-layer or Multi-layer.

Neural networks that consist of just one layer that represent the output layer and the inputs are fed directly to the outputs via a series of weights are called single-layer neural network. Single-layer Feed-Forward neural network cannot solve complicated problems. A linear associative memory is an example of a single-layer neural network. Neural networks that have one or more hidden layers allow an increase in the computational power of the network, so multi-layer neural network are suggested to solve complicated problems [36]. Figure 4.7 shows an example of single-layer and multi-layer Feed-Forward neural networks. The function of the hidden neurons is to intervene between the external input and the network output. As the number of hidden layers is increased the network will be able to extract higher-order statistics, and an increase in reliability and computational power, and a decreased processing time are provided.



Figure 4.7: Fully connected Feed-Forward neural network (a) Single-layer (b) Multi-layer.

The nodes in the input layer receive the data from the surrounding environment and flow the output signals to the first hidden layer of the neural network. Then, the first hidden layer will flow the output signals which will be used as input signals to the second hidden layer and so on until the signals reach the output layer of the network. Typically, the neurons in each layer of the network receive signals from previous layer only. The signals of the nodes in the output layer of the network represent the overall response of the network to the signals that received by the nodes in the input layer [37]. For brevity, the network shown in figure 4.7 (b) is referred to as a 2-3-2 network in that it has 2 source nodes, 3 hidden neurons, and 2 output neurons. As another example, a Feed-Forward network with *p* source nodes, *h1* neurons in the first hidden layer, *h2* neurons in the second layer, and *q* neurons in the output layer is referred to as a *p-hl-h2-q* network.

The fully connected neural network is the network that each neuron in any layer of the network is connected to the all neurons in the next layer. So, if any communication link (synaptic connection) is missed from the neural network, in this case the neural network can be named as partially connected.

**4.5. Backpropagation algorithm**

The most used algorithm is backpropagation algorithm. It is multilayers and its architecture is fully connected feed forward neural network. Supervised learning is used in this algorithm. Supervised learning means that the input patterns and the target of the neural network are given, and then the network calculates the error (the difference between the network target and the obtained output).

The idea from developing backpropagation algorithm is to reduce the error between the target and the output of the neural network so the network can learn the training data.

The weights of the neural network are adjusted with random values in the beginning of the learning, and then they adjusted to the final values that make the error as minimal as. A weighted sum activation function is implemented in the neural networks that use backpropagation algorithm [38]. Equation (4.9) is used as an activation function.

 $A\_{j}\left(\overbar{x},\overbar{w}\right)=\sum\_{i=0}^{n}x\_{i}w\_{ji} (4.9)$

Linear neurons are the neurons that the activation function in them is equal to the output function. In this case the input of the neuron will be flowed to the output directly. Sigmoidal function is the most output function used because it can be applied to solve complex functions. Equation (4.10) shows the sigmoidal function.

 $O\_{j}\left(\overbar{x},\overbar{w}\right)=\frac{1}{1+e^{A\_{j}(\overbar{x},\overbar{w})}} (4.10)$

The sigmoidal function is represented as one for large positive numbers, close to 0.5 at zero and is represented as zero for large negative numbers as seen in figure 4.6.

Now, in this section the derivative of the backpropagation algorithm was done [38].The error function for the output of each neuron can be defined as in equation (4.11).

 $E\_{j}\left(\overbar{x},\overbar{w},d\right)=(O\_{j}\left(\overbar{x},\overbar{w}\right)-d\_{j})^{2} (4.11)$

After that, the square of the difference between the target and the output of the network is taken as in equation (4.12).

 $E\left(\overbar{x},\overbar{w},\overbar{d}\right)=\sum\_{j}^{}(O\_{j}\left(\overbar{x},\overbar{w}\right)-d\_{j})^{2} (4.12)$

The weights of the neural network can be adjusted by using the method of gradient descendent as shown in equation (4.13).

$$ Δw\_{ji}=-η\frac{ƏE}{Əw\_{ji}} (4.13)$$

This formula can be interpreted in the following way: the adjustment of each weight ($Δw\_{ji}$) will be the negative of a constant eta (η) multiplied by the dependence of the previous weight on the error of the network, which is the derivative of *E* in respect to $w\_{i}$. The size of the adjustment will depend on η, and on the contribution of the weight to the error of the function. If the weight contributes a lot to the error, the adjustment will be greater than if it contributes in a smaller amount. Equation (4.13) is used until the appropriate weights (the error is minimal) are founded [38].

Then from equation (4.11), the derivative of E in respect to $O\_{j}$is illustrated in equation (4.14).

$$ \frac{ƏE}{ƏO\_{j}}=2\left(O\_{j}-d\_{j}\right) (4.14)$$

Now, from equations (4.9 and 4.10), equation (4.15) is gotten.

$$ \frac{ƏO\_{j}}{Əw\_{ji}}=\frac{ƏO\_{j}}{ƏA\_{j}}\frac{ƏA\_{j}}{Əw\_{ji}}=O\_{j}\left(1-O\_{j}\right)x\_{i} (4.15)$$

And from equations (4.14 and 4.15), equation (4.16) is gotten.

$$ \frac{ƏE}{Əw\_{ji}}=\frac{ƏE}{ƏO\_{j}}\frac{ƏO\_{j}}{Əw\_{ji}}=2\left(O\_{j}-d\_{j}\right)O\_{j}\left(1-O\_{j}\right)x\_{j} (4.16)$$

And so, the adjustment to each weight will be in equation (4.17), from equations (4.13 and 4.16).

$$ Δw\_{ji}=-2η\left(O\_{j}-d\_{j}\right)O\_{j}\left(1-O\_{j}\right)x\_{j} (4.17)$$