Classification of Multilayer Neural Networks Using Cross Entropy and Mean Square Errors

Hussein Rady

El-Shorouk Academy, Higher Institute for Computer & Information Technology,
Tel.:0106093311, E-mail: dr_hussein_rady@yahoo.com

Abstract: *The last years have witnessed an increasing attention to entropy based criteria in adaptive systems. Several principles were proposed based on the maximization or minimization of entropic cost functions. One way of entropy criteria in learning systems is to minimize the entropy of the error between two variables: typically one is the output of the learning system and the other is the target. In this paper, a classification of multilayer Back Propagation (BP) Neural Networks was proposed. The usual mean square error(MSE) minimization principle is substituted by the minimization of cross-entropy (CE) of the differences between the multilayer perceptions output and the desired target. These two cost functions are studied, analysied and tested with three different activation functions namely, the trigonometric (Sin) function, the hyperbolic tangent function, and the sigmoid activation function. The analytical approach indicates that the results are encourage and promising and that the cross entropy cost function is a more appropriate error function than the usual mean square error.*

Keywords ─ Cross-entropy, Mean Square error, Activation Function, Learning Rate and Neural Network.

1. INTRODUCTION

Neural Networks have emerged as an important tool for classification. The recent vast research activities in neural classification have established that neural networks are a promising alternative to various conventional classification methods. The advantage of neural networks lies in the following theoretical aspects. First, neural networks are data driven self-adaptive methods in that they can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model. Second, they are universal functional approximators in that neural networks can approximate any function with arbitrary accuracy. Third, neural networks are nonlinear models, which makes them flexible in modeling real world complex relationships. Finally, neural networks are able to estimate the posterior probabilities, which provides the basis for establishing classification rule and performing statistical analysis[24].

For classification problems[20], supervised learning methods train a classifier on a set of labeled training examples which fall into several classes. The classifier can then be used to predict the class of a new instance. Each instance is represented by a set of features, which have to be carefully chosen[28, 29].

The BP method is a technique used in training multilayer neural networks in a supervised manner. It also known as the error BP algorithm, that is based on the error-correction learning rule. It consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern is applied to the input nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as an actual response of the network. During the forward pass the synaptic weights of the networks are all fixed. During the backward pass, the synaptic weights are all adjusted in accordance with an error-correction rule. The actual response of the network is subtracted from a desired response to produce an error signals, this error signal is then propagated backward through the network. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The weight adjustment is made according to the generalized delta rule to minimize the error[28].

Gradient based methods are one of the most widely used error minimization methods used to train back propagation networks. The BP training algorithm is a supervised learning method for multilayered feed-forward neural networks [21]. It is essentially a gradient descent[22] local optimization technique which involves backward error correction of network weights. Despite the general success of back propagation method in the learning process, several major deficiencies are still needed to be solved. The convergence rate of back propagation is very low and hence it becomes unsuitable for large problems. Furthermore, the convergence behavior of the back propagation algorithm depends on the choice of initial values of connection weights and other parameters used in the algorithm such as the learning rate and the momentum term [21]. Improving the training efficiency of neural network based algorithms is an active area of research and numerous papers have been proposed in the literature. Early days of BP algorithms saw improvements on (i) selection of better activation function; (ii) selection of dynamic learning rate and momentum [23, 28]. Later, various optimization techniques were suggested. For improving the efficiency of error minimization process or in other words the training efficiency [21].

The behavior of an artificial neural network depends on both the weights and the activation function (energy function) that is specified for the units. Because the standard BP algorithm descends along the gradient of the error surface, the use of any activation function that has a large gradient than that of the sum of the squared error or any other cost function at higher energy values would make for faster training [19].

The idea behind Error Entropy Minimization (EEM) [9, 15, 17] is to replace the MSE, as the cost function of a learning system, with the entropy of error. The

minimization of the error entropy results in the minimization of the divergence between the joint probability density functions (pdf_s) of input-target and input-output signals. This suggests that the distribution of the output of the system is converging to the distribution of the targets. Let the error e(j) = T(j) – Y(j) represent the difference between the target T of the j output neuron and its output Y, at a given time t. The MSE of the variable e (j) can be replaced by its EEM counterpart. For EEM, we can use (1) cross entropy[1,4,13], (2) Shannon entropy[3, 10, 14], (3) relative entropy[5, 25, 26, 27], (4) Renyi's entropy [6,8,11,12], (5) Differential entropy[7] or any other suitable entropy technique.

The organization of the paper presented in the following sections. In the next section, the idea of entropy is outlined. In section three, Least Mean Square Error Function is presented. In section four, Cross entropy Error Function is studied. Activation functions are presented in section five. Simulated results are discussed in section six. And finally in section seven, the conclusions are outlined.

2. ENTROPY

As defined in information theory, entropy is a measure of the uncertainty of a particular outcome in a random process [2, 16]. The entropy of a random variable is a measure of the uncertainty of the random variable; it is a measure of the amount of information required on the average to describe the random variable. Entropy is a nonlinear function to represent information we can learn from unknown data. In the learning process, we learn some constraints on the probability distribution of the training data from their entropy [25].

The entropy $\boldsymbol{H(x)}$ of a discrete random variable $\boldsymbol{X}$ is defined by

$$\boldsymbol{H(x) = -\sum_x p(x) \log p(x)} \quad \rightarrow \quad (1)$$

where $\boldsymbol{p(x)}$ is the probability mass function. The log is to the base 2.

$\boldsymbol{\log \frac{1}{p(x)}}$ The entropy of $\boldsymbol{x}$ can also be interpreted as the expected value of , where $\boldsymbol{x}$ is drawn according to probability mass function $\boldsymbol{p(x)}$. thus

$$\boldsymbol{H(x) = E_p \log \frac{1}{p(x)}} \quad \rightarrow \quad (2)$$

$$\boldsymbol{= -E_p \log p(x)} \quad \rightarrow \quad (3)$$

The joint entropy $\boldsymbol{H(x,y)}$ of a pair of discrete random variables $\boldsymbol{(X,Y)}$ with a joint distribution $\boldsymbol{p(x,y)}$ is defined as

$$\boldsymbol{H(x,y) = -\sum\sum p(x,y) \log p(x,y)} \quad \rightarrow \quad (4)$$

which can also be expressed as

$$H(x,y) = -E\log p(x,y) \quad \rightarrow \quad (5)$$

For a continuous random variable X, the entropy H(x) [2] is defined by:

$$H(x) = -\int f(x)\log[f(x)]dx \quad \rightarrow \quad (6)$$

where $f(x)$ represents a probability density function of the variable X.
Similar to Eq.(6), uncertainty of two variables, X and Y, can be described by joint entropy:

$$H(x,y) = -\iint f(x,y)\log[f(x,y)]\,dxdy \quad \rightarrow \quad (7)$$

Here, $f(x,y)$ represents the joint probability density function of variables X and Y. The joint and marginal entropies are related

$$H(x,y) = H(x) + H(y) - T(x,y) \quad \rightarrow \quad (8)$$

If X and Y are independent, $T(x,y) = 0$, the uncertainty of Y after obtaining X is the same as the original (marginal) uncertainty of Y. If the variables are perfectly correlated, then the knowledge of one variable gives complete information about the other variable, in that case

$$T(x,y) = H(y).$$

where $T(x,y)$ represents the information transferred from x to y.
The uncertainty of Y, given X, denoted as $H(y/x)$, is equal to

$$H(y/x) = H(y) - T(x,y) \quad \rightarrow \quad (9)$$

3. LEAST MEAN SQUARE ERROR FUNCTION

In statistics, the Mean Squared Error (MSE) of an estimator is one of many ways to quantify the amount by which an estimator differs from the true value of the quantity being estimated. MSE measures the average of the square of the "error." The error is the amount by which the estimator differs from the quantity to be estimated. The difference occurs because of randomness or because the estimator doesn't account for information that could produce a more accurate estimate.

The Least Mean Square (LMS) cost function has been used more frequently than any alternative cost function in Neural Networks. It yields good performance with large data bases on real world. LMS error cost function is the most often used error function despite being criticized for its lack of convergence speed and a higher possibility of being trapped in a local minima in the network training process [14].

For the general multi-class in Multilayer Neural Networks, let us consider the problem of assigning an input vector $\mathbf{x} = \{x_i : i = 1, \ldots, D\}$ to one of M classes $\{c_i : i = 1, 2, \ldots, M\}$. Let $\mathbf{c_i}$ denote the corresponding class of $\mathbf{x}$, $\{y_i(x) : i = 1, \ldots, M\}$ the outputs of the network, and $\{d_i : i = 1, 2, \ldots, M\}$ the target outputs for all output nodes. With the least mean sequence cost function, the network parameters are chosen to minimize the following:

$$\Delta = E\{\sum_{i=1}^{M}[y_i(x) - d_i]^2\} \quad \rightarrow \quad (10)$$

where $\mathbf{E\{.\}}$ is the expectation operator. Denoting the joint probability of the input and the i[th] class by $p(\mathbf{x}, \mathbf{c_i})$, we obtain

$$\Delta = \int \sum_{j=1}^{M}\{\sum_{i=1}^{M}[y_i(x) - d_i]^2\} p(x, c_i)dx \quad \rightarrow \quad (11)$$

Substituting $p(x, c_i) = p(c_i|x)p(x)$ in eq (11) gives:

$$\Delta = \int\{\sum_{j=1}^{M}\sum_{i=1}^{M}[y_i(x) - d_i]^2 p(c_i|x)\}p(x)dx \quad \rightarrow \quad (12)$$

$$= E\{\sum_{j=1}^{M}\sum_{i=1}^{M}[y_i(x) - d_i]^2 p(c_j|x)\}$$

$$= E\{\sum_{j=1}^{M}\sum_{i=1}^{M}[y_i^2(x)p(c_j|x) - 2y_i(x)d_i p(c_j|x) + d_i^2 p(c_j|x)]\} \quad \rightarrow \quad (13)$$

But since, $y_i^2(x)$ is a function only of x and $\sum_{j=1}^{M} p(c_j|x) = 1$, we obtain

$$\Delta = E\{\sum_{i=1}^{M}[y_i^2(x) - 2y_i(x)\sum_{j=1}^{M} d_i p(c_j|x) + \sum_{j=1}^{M} d_i^2 p(c_j|x)]\}$$

$$= E\{\sum_{i=1}^{M}[y_i^2(x) - 2y_i(x)E\{d_i|x\} + E\{d_i^2|x\}]\} \quad \rightarrow \quad (14)$$

$$= E\{\sum_{i=1}^{M}[y_i(x) - E\{d_i|x\}]^2\} \quad \rightarrow \quad (15)$$

$$= E\{\sum_{i=1}^{M}[y_i(x) - E\{d_i|x\}]^2\} + E\{\sum_{i=1}^{M} var\{d_i|x\}\} \quad \rightarrow \quad (16)$$

Where $var\{d_i|x\} = E\{d_i^2|x\} - E^2\{d_i|x\}$ is a conditional variance of Δ is achieved by choosing network parameters to minimize the first term of eq.(16) which is simply the mean-squared error between the network output $y_i(x)$ and the conditional expectation of the target outputs. Thus, when network parameters are chosen to minimize a LMS cost function, outputs estimate the conditional expectation of the target outputs so as to minimize the mean squared error [14].

4. CROSS-ENTROPY ERROR FUNCTION

In general, the cross entropy[1,13] is an iterative method, which involves the following two phases: First, generation of a sample of random data according to a specified random mechanism. Second, updating the parameters of the random

mechanism, on the basis of the data, in order to produce a "better" sample in the next iteration.

The significance of the cross-entropy concept is that it defines a precise mathematical framework for deriving fast, and in some sense "optimal" updating rules.

To investigate the cross entropy error function, we will discuss Network training using the maximum likelihood principle, and then derive Network training using the cross entropy cost function in the following two subsections:

4.1 NETWORK TRAINING USING THE MAXIMUM LIKELIHOOD PRINCIPLE

Let $\boldsymbol{x}$ denotes the input vector $(x_1, x_2, \dots, x_m)$ and $\boldsymbol{t}$ denotes the corresponding target vector $(t_1, t_2, \dots, t_c)$. The network has to estimate the joint probability $\boldsymbol{p(x,t)}$, where

$$p(x,t) = p(t|x)p(x) \quad \rightarrow \quad (17)$$

If we assume that the patterns are drawn independently from the true distribution, we can define the likelihood function $\boldsymbol{L}$ with respect to a training data set $\boldsymbol{T}$

$$L(T) = \prod_{n=1}^{N} p(x_n, t_n) = \prod_{n=1}^{N} p(t_n|x_n)p(x_n) \quad \rightarrow \quad (18)$$

In order to estimate the distribution $p_w(x_n, t_n)$, we use the maximum likelihood principle with respect to the weight vector $\boldsymbol{w}$.

$$L_w(T) = \prod_{n=1}^{N} p_w(t_n|x_n) \quad \rightarrow \quad (19)$$

$$= \prod_{n=1}^{N} p_w(t_n|x_n) \quad \rightarrow \quad (20)$$

Since $\prod_{n=1}^{N} p(x_n) = 1$ and it doesn't depend on the weight vector $\boldsymbol{w}$.

4.2 NETWORK TRAINING USING THE CROSS ENTROPY COST FUNCTION

Equivalently to the maximum likelihood principle, we can define an error function by taking the negative logarithm of the likelihood function L_w

$$E = -\ln L_w(T) = -\sum_{n=1}^{N} \ln p_w(t_n|x_n) \quad \rightarrow \quad (21)$$

For a two class problem that is represented by a single output unit. A target value of 1 corresponding to class c_1 and a value of o to class c_2, respectively. After training the network, the class-conditional probabilities are given by

$$p(t|x) = o^t(1-o)^{1-t} \quad \rightarrow \quad (22)$$

From equation (20) and (22) we get the model specific likelihood function

$$L_w(T) = \prod_{n=1}^{N} O_n^{t_n}(w)\left(1 - O_n(w)\right)^{1-t_n} \quad \rightarrow \quad (23)$$

Taking the negative logarithm, we obtain:

$$E(w) = -\sum_{n=1}^{N}\left[t_n \ln O_n(w) + (1 - t_n)\ln\left(1 - O_n(w)\right)\right] \rightarrow \quad (24)$$

The last equation corresponding to a Cross Entropy (CE) function between the two distributions $p(t|x)$ and $p(o|x)$. The CE is minimized if the distribution of the model output $p(o_n|x)$ and the distribution of the target values $p(t_n|x)$ are equal. Because the outputs o_n depend on the vector $w$its distribution can be adapted to the distribution $p(t_n|x)$ of the target values [4].

For a multiclass case, the CE is usually defined on the base of a 1-of-c coding scheme. Since the output values represent probabilities they must lie in the range between 0 and 1 their sum must be 1. Instead of using a 1-of-c coding scheme, a c-1-of-2 scheme was proposed [4] and the mentioned CE function for the multiclass case as the sum over all two class problems is as follows:

$$E_{CE}(w) = -\sum_{z=1}^{c}\sum_{n=1}^{N_c}\left[t_{zn} \ln O_{zn}(w) + (1 - t_{zn})\ln\left(1 - O_{zn}(w)\right)\right] \rightarrow \quad (25)$$

where t_{zn} and O_{zn} denote the n-th target and output values for the z-th pattern class.

5. ACTIVATION FUNCTIONS

The activation function, defines the output of a neuron in terms of thee induced local field. It is applied on the input summation for limiting the amplitude of the neuron's output. The computation of the local gradient for each neuron of the multilayer perceptron requires knowledge of the derivative of the activation function associated with that neuron. For this derivative to exist, we require the activation function to be continuous. In basic terms, differentiability is the only requirement that an activation function has to satisfy.

Why use activation functions

Activation functions for the hidden units are needed to introduce nonlinearity into the network. Without nonlinearity, hidden units would not make nets more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). The reason is that a linear function of linear functions is again a linear function. However, it is the nonlinearity (i.e., the capability to

represent nonlinear functions) that makes multilayer networks so powerful. Almost any nonlinear function does the job, except for polynomials. For BP learning, the activation function must be differentiable, and it helps if the function is bounded; the sigmoidal functions such as logistic and tanh and the Gaussian function are the most common choices. Functions such as tanh or arctan that produce both positive and negative values tend to yield faster training than functions that produce only positive values such as logistic[18], because of better numerical conditioning[30].

For hidden units, sigmoid activation functions are usually preferable to threshold activation functions. Networks with threshold units are difficult to train because the error function is stepwise constant, hence the gradient either does not exist or is zero, making it impossible to use BP or more efficient gradient-based training methods. Even for training methods that do not use gradients - such as simulated annealing and genetic algorithms – sigmoid units are easier to train than threshold units. With sigmoid units, a small change in the weights will usually produce a change in the outputs, which makes it possible to tell whether that change in the weights is good or bad. With threshold units, a small change in the weights will often produce no change in the outputs [30].

6. SIMULATED RESULTS

Usually error back propagation for neural network learning is made using MSE as the cost function. In this article, we proposed the use of the minimization of the error entropy besides the MSE as a cost function for classification purposes. In terms of the entropy measure, the cross entropy approach have been tested with good results when compared to MSE.

In this section, and on a simulated data sets, we discuss some the figures and tables showing the behavior resulting from the practical implementations as a comparison between the Mean Square error and their counterpart using the cross entropy errors.

6.1 USING RANDOM INITIAL WEIGHTS:

Cross Entropy and the trigonometric activation function (Sin). Figure1-a, represents the actual output when the target output = 1. The weights taken random between zero and one. In this case, the number of iterations = 33, and the value of the minimum error = 0.00653027040600151. we take the learning rate = 0.6 as a constant learning rate. The figure shows the behavior of the actual output for the last fourteen iterations that converges gradually towards the target output. While Figure 1-b shows the cross entropy error that is minimized gradually step by step after each iteration and finally tends to a value which is less than the stopping criterion.

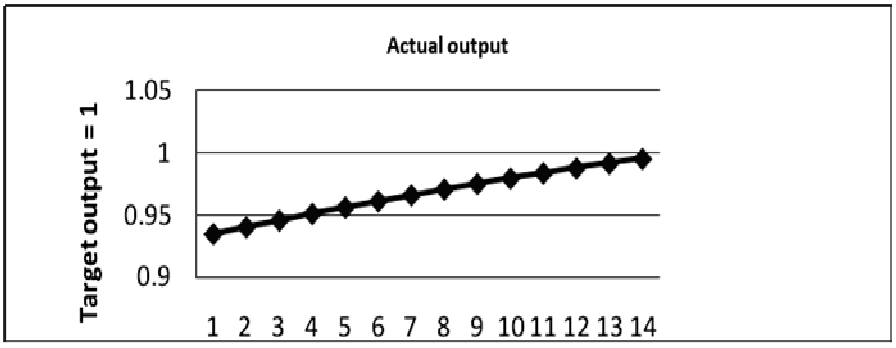

Figure 1-a: Actual Output For the Final 14 Iterations Using Trig Activation Function and Cross Entropy.

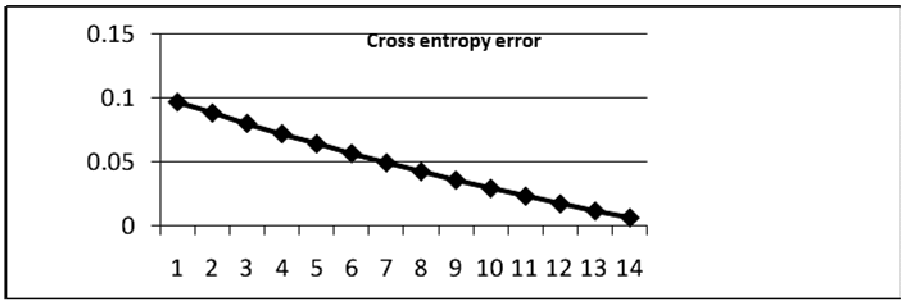

Figure 1-b: Cross Entropy Error For The Final Fourteen Iterations Using Trig Activation Function.

MSE with the trigonometric activation function. Figure 2-a represents the actual output when the target output = 1. The weights taken random between zero and one. In this case the number of iterations = 74, and the value of the minimum error = 0.00999108352645411. We take the learning rate = 0.6 as a constant learning rate. For a certain input pattern, and according to the target output, we find that for the last fourteen iterations that the actual output grows gradually and finally tends towards the value of the target output. In Figure 2-b, the mean square error decreases gradually verifying the stopping criterion at the end of the last iteration.

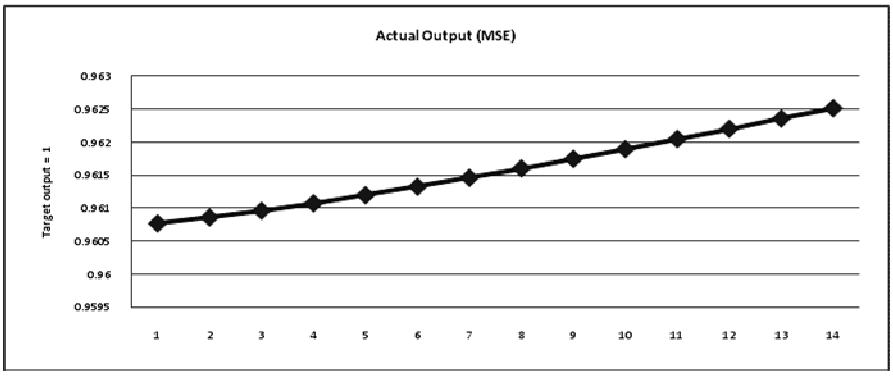

Figure 2-a: Actual Output For The Final Fourteen Iterations Using MSE And The Trig Activation Function.

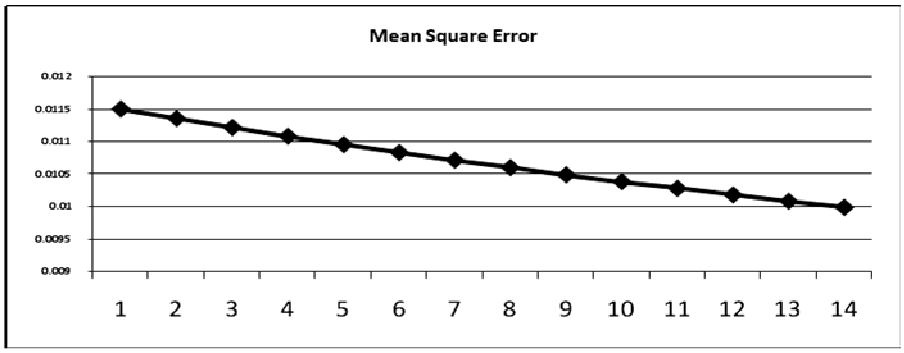

Figure2-b : MSE For The Final Fourteen Iterations Using Trig Activation Function.

Figure 3 shows that the Cross Entropy error is less than the MSE in the last fourteen iterations. As a comparison between every iteration using the cross entropy error and its counterpart using the MSE, we find that the cross entropy error is more converged to the stopping criterion early than the MSE. On the other hand, the rate of convergence (which pertains to the "speed" at which a convergent sequence approaches its limit) in cross entropy is more than the rate of convergence using MSE.

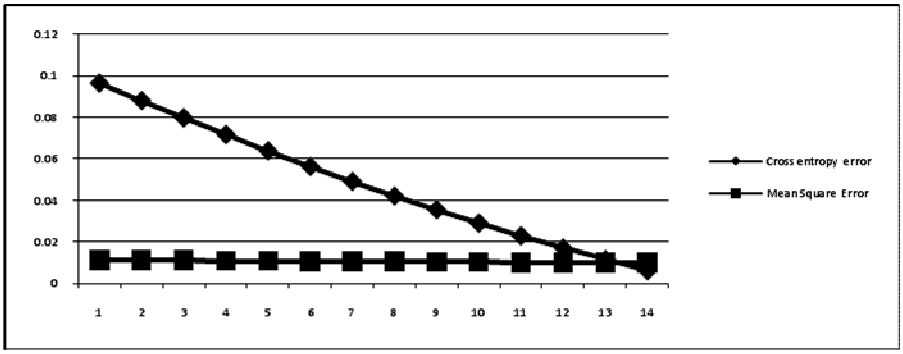

Figure 3: A Comparison Between CE Error And MSE For The Last 14 Iterations Using Trig Function.

CE Error & MSE using the sigmoid function. In this case, the initial weights and bias also taken random between zero and one. The number of iterations using cross entropy = 7398, but the number of iterations using Mean Square Error = 279. We take the learning rate a constant value and is equal to 0.6. in this case the number of iterations required to satisfy the stopping criterion condition using cross entropy is greater than the number of iterations using Mean Square Error. But in each iterations of the last fourteen, we find that the cross entropy error is less than that of the MSE using the sigmoid activation function as shown in Figure 4.

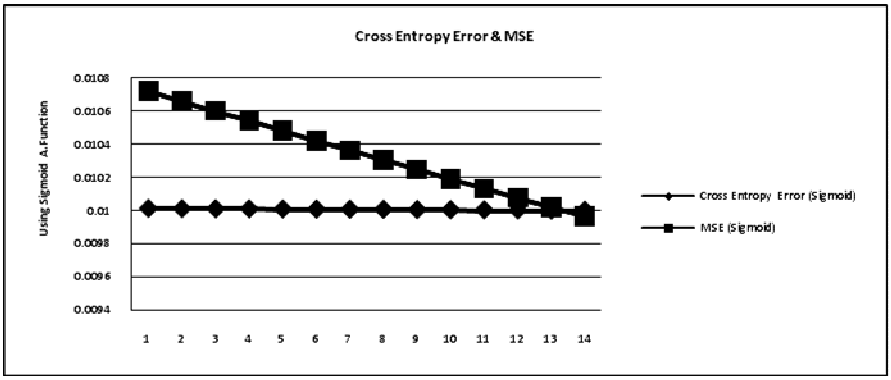

Figure 4 CE Error and MSE For The Last 14 Iterations Using The Sigmoid Function.

CE error & MSE using the (tanh –trig- sigmoid)activation functions. Fig 5-a represents the CE errors using the three functions. The weights taken random between zero and one. Using tanh function , the number of iterations = 677 , and the value of the minimum error = 0.00997437697541428. We take the learning

rate = 0.6 as a constant learning rate. Comparing the cross entropy error using the three activation functions, we find that:

1) The number of iterations using the sigmoid function is greater than the number of iteration using the tanh activation function.
2) The number of iterations using the tanh activation function is greater than the number of iterations using the trig function.

Accordingly, for the last fourteen iterations, we find that the trig function speeds the convergence than the tanh function which also speeds the convergence than the sigmoid function.

Figure 5-b illustrates the convergence using MSE. The convergence is speed using the tanh function rather than using the trig function. Also the trig function is better in convergence than the sigmoid function.

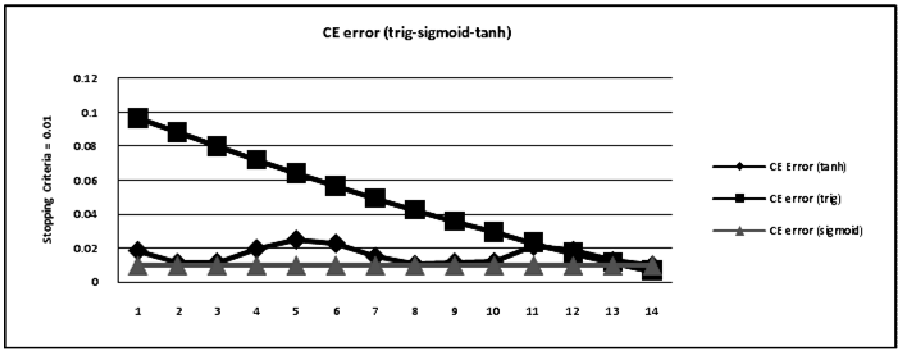

Figure 5-a : The CE Error Using The Activation Functions (Tanh-Trig – Sigmoid) Function.

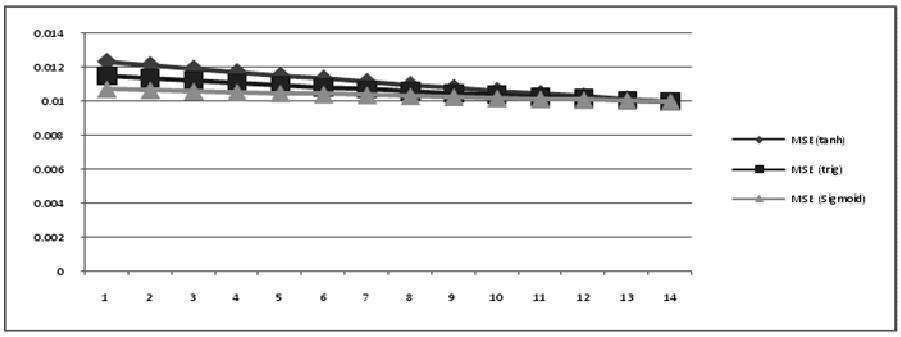

Figure 5-b The MSE Using The Activation Functions (Tanh – Trig – Sigmoid).

In Table1: a comparison between the Cross Entropy and Mean Square outputs using three different activation functions for the last iteration. These functions are:

the trig function, the sigmoid function and the tanh function. These outputs resulting using random weights between zero and one. The learning rate applied here is a constant learning rate = 0.6. the stopping criteria = 0.01 in all cases. As a comparison between the CE error and MSE, we find that:

1. In case of using cross entropy, the rate of convergence using trig function is greater than the rate of convergence using tanh function which is greater than the rate of convergence using the sigmoid function.
2. In case of using mean square, the rate of convergence using trig function is greater than the rate of convergence using tanh function which is greater than the rate of convergence using sigmoid function.

Table1: Comparing the Last Iteration For (Trig-Sigmooid-Tanh) Functions Using Random Initial Weights.

Activation Function	Actual Output (CE)	Degree of Conv. (CE)	Error (CE)	No. of Iteration (CE)	Actual Output (MS)	Degree of Conv. (MS)	Error (MS)	No. of Iteration (MS)
Trig	0.995	99.5%	0.0065	33	0.962	96.2%	0.0099	74
Sigmoid	0.993	99.3%	0.0099	7398	0.874	87.4%	0.00996	279
Tanh	0.9931	99.31%	0.00997	677	0.889	88.9%	0.00996	69

Finally, although the weights and the bias are taken random, which affects the accuracy of the results to some extent, we find that the cross entropy error is more appropriate than using the mean square error.

6.2 USING A FIXED INITIAL WEIGHTS

Figure 6-a represents the CE error using the three (trig-tanh-sigmoid) activation functions. The figure indicates that the rate of convergence using the trig function is more than the rate of convergence using the tanh function. On the other hand, the rate of convergence using the tanh function is also greater than its counterpart in case of using the sigmoid function. We also notice that the number of iterations using the trig function is less than the number of iterations using the tanh function, which is less than the number of iterations using the sigmoid function.

In case of using the MSE as in Figure 6-b, we find that by using the trig activation function, the training process is more converged (fastest) than using the tanh function (fast) which is more converged than the sigmoid function (slow).

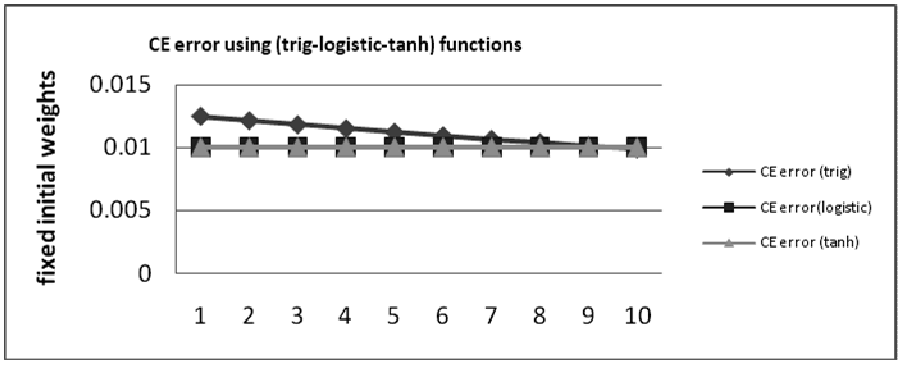

Figure 6-a: Comparing the CE Errors For The Last Ten Its Using (Trig – Logistic – Tanh) Activation Function.

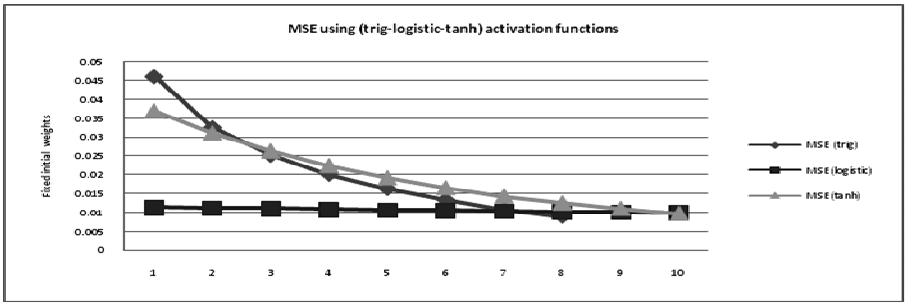

Figure 6-b: Comparing the MSE for the Last Ten Its Using (Trig – Logistic – Tanh) Activation Function.

Figure 7-a,b,c indicates that the actual output degree of convergence towards the target output using the cross entropy is greater than the degree of convergence using the MSE through the last ten iterations in case of using any of the three mentioned activation functions. This indicates that the cross entropy error is more accurate than the MSE.

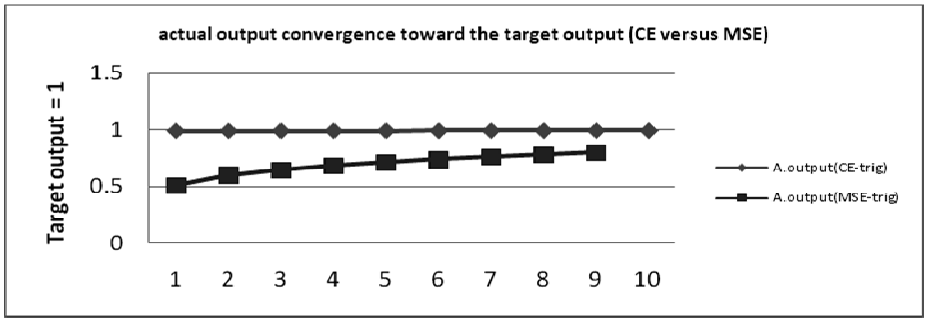

Figure 7-a: Actual Output Convergence Towards The Target Output = 1 Using Trig Function.

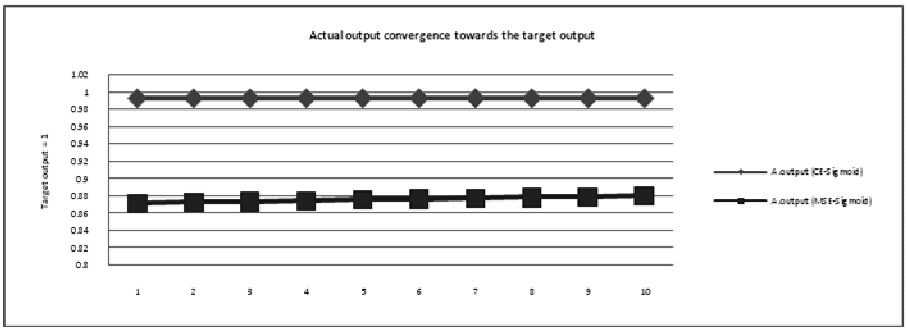

Figure 7-b: Conv. Behavior For The Actual Output Towards The Target Output = 1 using CE error & MSE (sigmoid fn).

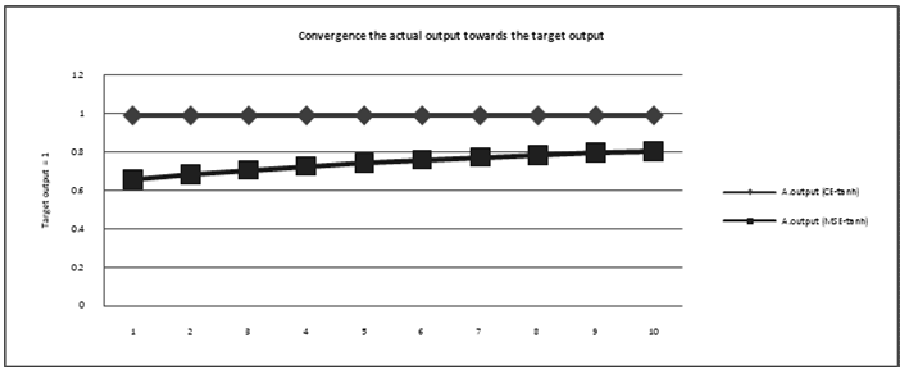

Figure 7-c: Actual Output Conv. Behavior Towards The Target Output = 1 (CE Versus MSE) Using Tanh Function.

In Table 2 a comparison between the Cross Entropy and Mean Square outputs using three different activation functions. These functions are: the trig function, the sigmoid function and the tanh function. These outputs resulting using constant

initial weights between zero and one. The learning rate applied here is a constant learning rate which is equal to 0.6. the stopping criterion = 0.01 in all cases.

Table 2: Comparing the Last Iteration For (Trig-Sigmooid-Tanh) Functions Using Constant Initial Weights.

Activation Function	Actual Output (CE)	Degree of Conv. (CE)	Error (CE)	No. of Iteration (CE)	Actual Output (MS)	Degree of Conv. (MS)	Error (MS)	No. of Iteration (MS)
Trig	0.993	99.3%	0.0098	67	0.798	79.8%	0.0089	9
Sigmoid	0.993	99.3%	0.0099	7865	0.88	88%	0.0099	251
Tanh	0.993	99.3%	0.00999	7331	0.806	80.6%	0.0097	16

Table 2 also shows the following observations:

- The number of iterations using the MSE is less than their CE counterpart.
- The actual output degree of convergence to the target output using the CE is more than their MSE counterpart. This means that the stopping criterion using MSE must be smaller than stopping criterion using the CE to reach to a satisfactory results.

In table 3 a comparison between the Cross Entropy using the logarithm to the base two and the Cross Entropy using the logarithm to the base e by applying three different activation functions. These functions are: the trig function, the sigmoid function and the tanh function. These outputs resulting from using constant initial weights between zero and one. The learning rate applied here is a constant earning rate = 0.6. The stopping criteria = 0.01 in all cases. Comparing the cross entropy to the base 2 and the cross entropy to the base e, we find that for the three activation functions, the number of iterations using the base e is less than the number of iterations using the base 2.

Table 3: Comparing the Last Iteration For (Trig-Sigmooid-Tanh) Function Using Cross Entropy (Base 2 – Base E)

Activation Function	Actual Output (base 2)	Degree of Conv. (base2)	Error (base2)	No. of Iterations (base2)	Actual Output (base e)	Degree Of Conv. (base e)	Error (base e)	No. of Iterations (base e)
Trig	0.993	99.3%	0.0098	67	0.99	99%	0.0097	54
Sigmoid	0.993	99.3%	0.0099	7865	0.99	99%	0.0099	4314
Tanh	0.993	99.3%	0.00999	7331	0.999	99.9%	0.0099	3500

Table 4 illustrates the mean and standard deviation for the last ten iterations for the actual output using a fixed initial weight. Comparing the three activation functions, we find that the cross entropy less diverged (more converged) than the mean square error because Mean-Cross > Mean-MSE and STD-cross < STD-MSE for all the three activation functions (Trig - Sigmoid – Tanh).

Table 4 : The Mean And Standard Deviation For The Last Ten Iterations For The Actual Output

Actual Output	Mean	Standard Deviation
CE-Trig	0.9923	0.0006
MSE-Trig	0.6909	0.09457
CE-Sigmoid	0.993090437	0.00000158
MSE-Sigmoid	0.876215	0.00279
CE-Tanch	0.99309	0.000001407
MSE-Tanh	0.744	0.05000664

Table 5 : shows that the mean and standard deviation for the error to the last ten iterations using fixed initial weights. Investigating this table, we find that:

1. Using trig fn, mean of the error (CE) < mean of the error (MSE) .
   STD of the error(CE) < STD of the error (MSE).
   This means that cross entropy is more converged to the target output than MSE.
2. Using Sigmoid fn, mean of the error (CE) < mean of the error (MSE) .
   STD of the error(CE) < STD of the error (MSE).
   This means that cross entropy is more converged to the target output than MSE.
3. Using tanh fn, mean of the error (CE) < mean of the error (MSE).
   STD of the error(CE) < STD of the error (MSE).

This means that cross entropy is more converged to the target output than the MSE.

Table 5 : The Mean & STD For The Last 10 Iterations For The Error Using Fixed Initial Weights (trig-tanh-sigmoid).

Error	Mean	Standard Deviation
CE-Trig fn	0.011120777	0.000880312
MSE-Trig fn	0.021712104	0.012577033
CE-Sigmoid fn	0.010003371	0.0000023046
MSE-Sigmoid fn	0.01068208	0.000476141
CE-Tanch fn	0.010002612	0.00000204458
MSE-Tanh fn	0.0.020016973	0.009078251

7- DISCUSSION AND CONCLUSIONS

We have presented, in this paper a new way of performing classification by using the cross entropy of the error between the actual outputs and the desired targets, as well as the most often used mean square error cost function. A comparative approach were investigated between these two errors using three different activation functions, the trigonometric, the hyperbolic tangent and the sigmoid activation functions. In fact, besides, the cross-entropy cost function has much less local minima compared to the mean squared error function, the approach employed here shows that cross entropy has significant, practical advantages over Mean Squared Error in the following points:

1- The actual outputs using cross entropy error is more converged to the target output rather than using MSE.
2- The number of iterations resulted in training the neural network using the trigonometric (Sin) function is less than the number of iterations using the hyperbolic tangent function.
3- The number of iterations using the hyperbolic tangent function is less than the number of iterations using the sigmoid function.
4- From (2) and (3), we conclude that the trigonometric function is the fastest activation function since it speeds the convergence better than the other two functions. While the hyperbolic tangent function is fast, the sigmoid activation function less fast (slow).
5- Using the trigonometric function, the mean of the cross entropy error is less than the mean square error. And the standard deviation of the cross entropy error is less than the standard deviation of the mean square error.

6- Using the hyperbolic tangent function, the mean of the cross entropy error is less than the mean square error. And the standard deviation of the cross entropy error is less than the standard deviation of the mean square error.
7- Using the sigmoid activation function, the mean of the cross entropy error is less than the mean square error. And the standard deviation of the cross entropy error is less than the standard deviation of the mean square error.
8- With the three functions, we find that using the mean square error, a poor convergence speed was obtained. While using the Cross entropy error we obtain a more convergence speed.

As a future work some other entropy techniques can be applied such as shannon's entropy, relative entropy, reinyi's entropy for classification purposes.

References:

[1] Yamada T., S-aito K. and Ueda N. (2003). "Gross-Entropy Directed Embedding of Network Data".
[2] Markus M., Christina W. and Demessie M. (2003). "Uncertainty of Weekly Nitrate-Nitrogen Forecast Using Artificial Neural Networks." Journal of Environmental Engineering.
[3] Silva L.M. and Marques S. (2005). "Neural Network Classification using Shannon's Entropy." European Symposium on Artificial Neural Networks.
[4] Joost M. and S-chiffmann W. "Speeding up Backpropagation Algorithms by using cross-Entropy combined with pattern normalization." International Journal of Uncertainty, Fuzziness and knowledge-Based Systems. World scientific publishing company.
[5] Poland W. B. and Shachter R. D. (1993). "Mixtures of Gaussians and Minimum Relative Entropy Techniques for Modeling Continues Uncertainties". In Uncertainty in Artificial Intelligence.
[6] S-antos J. M., Alexander L. A., and Marques J. (2004). "The Error Entropy Minimizing Algorithm for Neural Network Classification".
[7] Shwarz S., Zibalevoky M., and Schechner Y.Y. (2005). "Fast kernel entropy estimation and optimization".
[8] Erdogmus D., and Principe J. "An on-line adaptation algorithm for adaptive system training with minimum error entropy: stochastic information gradient".
[9] Entropy D., and Principe J. "Companies of entropy and mean square error criteria in adaptive system training using higher order statistics.
[10] Erdogmus D., Rao Y., Principe J., Romero O., and Bentanzos A. (2003). "Recursive Least Squares for an Entropy Regularized MSE Cost Function".
[11] SANTOS J., Alexander L., De s-a M. J, and sereno F. (2004). "optimization of the error entropy minimization algorithm for Neural Network Classification".

[12] Santos M. J., De sa M. J., and Alexander A. L."Neural Networks Trained with the EEM Algorithm: Tuning the Smoothing Parameter".
[13] Teahan J. W. "Text classification and segmentation using minimum cross-entropy".
[14] See G. N., Erdogan S. S., and Hiang K. C. "An adaptive scheme for switching Neural Network Cost Function".
[15] Alexander A. L. and de sa M. J. (2006). "Error Entropy Minimization for LSTM Training".
[16] WuXu J., Erdogmus D., and Principe C. J (2005). "Minimum Error Entropy Luenberger Observer".
[17] Brand M. (1999). "Pattern discovery via entropy minimization".
[18] Ng S. G, chan H. K., Erdogan S. S., and Singh H."Neural Network Learning Using Entropt Cycle".
[19] Rady H. (2007). "Different aspects for enhance the Back probagation Neural Networks".
[20] Krissilov V. A., Krissilov A. D., and Oleshko D. N. "Application of the sufficiency principle in Acceleration of Neural Networks Training".
[21] Nawi. M. N, Ransing S. R. and Ransing R. M (2007). "Am Improved Conjugate Gradient Based Learning Algorithm for Back Propagation Neural Networks".
[22] Arono, Alpaydin, E. "An Incremental Neural Network Construction Algorithm for Training Multilayer Perceptrous".
[23] Ahlawat A. and Pandey S. (2007)."A Variant of Back Propagation Algorithm for Multilayer Feed-Forward Network".
[24] Zhang P. G. (2000). "Neural Networks for Classification: A Survey".
[25] Starzyk J. and Guo Y. (2001). "An Entropy-based Learning Hardware Organization Using FPGA".
[26] Adah T."A General Probabilistic For mulation ofr Neural Classifications".
[27] Adah T., Liux. and Sonmez M.(1997). "Conditional Distribution Learning with Neural Networks and Its Application to Channel Equation".
[28] Dao P., Vemuri R. “A Performance Comparison of Different Back Propagation Neural Networks Methods in Computer Network Intrusion Detection”.
[29] Koehn P. “Combining Multiclass Maximum Entropy Text Classifiers with Neural Network Voting. http://www.flipdog.com/ .
[30] “Why activation functions”.http://www.fags.org/faqs/ai-faq/neural-nets/.