Global Journal of Computer Science and Technology: E
Network, Web & Security
Volume 16 Issue 5 Version 1.0 Year 2016
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Comparative Analysis: Heart Diagnosis Classification using Bp-LVQ Neural Network Models for Analog and Digital Data

By D. Rajeswara Rao & Dr.JVR Murthy
KL University

Abstract- Decades onwards companies are creating massive data warehouses to store the collected resources. Even though the stored resources are available, only few companies have been able to know that the actual value stored in the database. Procedure used to extract those values is known as data mining. We use so-many technologies to apply this data-mining technique, artificial neural network(ANN) also includes in this data-mining techniques ,ANN is the information processing units which are similar to biological nervous systems. Backpropagation is one of the techniques that used for classification and LVQ (learning Vector Quantization) can be plotted under the competitive learning scheme which is also used for classification. This paper elaborates artificial neural networks, its characteristics and working of backpropagation and LVQ algorithms. In this paper we show the intriguing comparisons between backpropagation and LVQ (Learning Vector Quantization) for both analog and digital data. It also attempts to explain the results between back-propagation and LVQ.

Keywords: artificial neural networks (ANN),activation function,multi-layer-feedforward-network,sigmoid, least mean squared error, backpropagation, training, codebook, competitive networks, learning vector quantization.

GJCST-E Classification : F.1.1 C.2.1

COMPARATIVEANALYSISHEARTDIAGNOSISCLASSIFICATIONUSINGBPLVQNEURALNETWORKMODELSFORANALOGANDDIGITALDATA

Comparative Analysis: Heart Diagnosis Classification using Bp-LVQ Neural Network Models for Analog and Digital Data

D. Rajeswara Rao [α] & Dr.JVR Murthy [σ]

Abstract - Decades onwards companies are creating massive data warehouses to store the collected resources. Even though the stored resources are available, only few companies have been able to know that the actual value stored in the database. Procedure used to extract those values is known as data mining. We use so-many technologies to apply this data-mining technique, artificial neural network(ANN) also includes in this data-mining techniques ,ANN is the information processing units which are similar to biological nervous systems. Backpropagation is one of the techniques that used for classification and LVQ (learning Vector Quantization) can be plotted under the competitive learning scheme which is also used for classification. This paper elaborates artificial neural networks, its characteristics and working of backpropagation and LVQ algorithms. In this paper we show the intriguing comparisons between backpropagation and LVQ (Learning Vector Quantization) for both analog and digital data. It also attempts to explain the results between back-propagation and LVQ.

Keywords: artificial neural networks (ANN),activation function,multi-layer-feedforward-network,sigmoid, least mean squared error, backpropagation, training, codebook, competitive networks, learning vector quantization.

I. Introduction

Artificial neural networks (ANN), is often called as "neural networks", is a data processing model based on the biological neural network modeling[5]. Neural networks are widely pre-owned to understand the patterns and the connections in the data. The data may be the outcome of a market research effort, etc. Artificial neural networks have been successfully solved many complex practical issues. The Small processing units present in the network are called as "Artificial Neuron", which operates the information using a connectionist approach to perform complex computations[1][5]. Basically, neural network have layered architecture with interconnected neurons as from fig-1.1. The neural networks (ANN) can be generally be a either a multiple-layer or a single-layer networks. The multilayer structure of neural networks is shown in fig-1.1.

Author α: Department of Computer Science and Engineering, KL University, Andhra Pradesh. e-mail: rajeshduvvada@kluniversity.in
Author σ: Professor, Dept of computer science & engineering, JNTU-Kakinada, Andhra Pradesh.

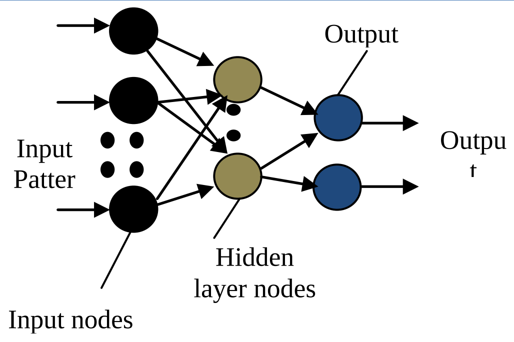


Fig. 1.1 : Architecture of Neural networks

Artificial neural networks had been developed based on the following hypothesis:

- The information is processed among many simple processing units, well known as "neurons".
- The signals are processed among these processing units which are known as neurons over the connection links among them.
- Each and every connection link among these neurons contains an weight, multiples with the transmitted signal.
- Each and every neuron or processing unit applies activation function to its net-input(weight multiplied with its signal input) comes from its previous unit.

Let consider a neuron h1 from fig-1.2, which receives inputs from input neurons y1,y2,y3. The weights on the connection from y1,y2,y3 are w1, w2, w3. The net-input N_y from the input nodes with the activations Y1,Y2,Y3 to the neuron h1 is defined as follows:

$$N\_y = w1Y1 + w2Y2 + w3Y3.$$

As from the final assumption pass this net input to the activation function given as h1= f(n_y).

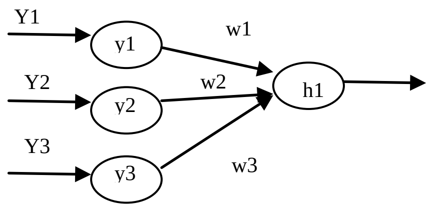


Fig-1.2 : Neuron output generation in ANN

Some simplifications are necessary to understand the intended properties and to attempt requires mathematical analysis. To implement the above assumptions the whole process of the neural networks are divided in to building blocks. The main building blocks of the neural networks are as follows:

- The Architecture of network.
- Initializing the weights to the nodes.
- Activation Functions.

a) Architecture of Neural Networks

The settlement of the neurons into several layers and the arrangement of the connection within and in-between the layers are known as the network architecture. The basic architecture of the simplest possible neural networks that performs classification subsists of a input layer units and a single output layer unit. Number of layers in the neural network can be outlined as the number of layers, which has weighted interconnected links among the neurons. Advanced neural network architecture consists of hidden layer along with the input layers and output layers. If the two layers of interconnected weights are present, then it is found to have hidden layer. The network architecture is divided into different types like Feed-Forward, Feed-back, Competitive. For back-propagation algorithm we are using Feed-Forward algorithm, where to LVQ (learning Vector Quantization) uses competitive network.

- Feed-Forward networks: These feed-forward networks have either a single layer of weights, where the neurons in the input layer are directly having connection links to the neurons in the output layer, or multiple layers with an interceding set of hidden neurons. Feed-back networks are also associated in two different ways i) Singlelayer ii) Multilayer. As in the single-layer feed-forward networks the weights from input layer does not influence the output layer. Whereas in multilayer feed forward networks one or many layer of nodes (units) between the input layer and the output layer units, so this network is used to solve the complex problems.

b) Setting the weights to the nodes:

The process of setting the weights enables the learning rules or training process. A neural network focusses on the way in which the weights can be changed. The method of tuning the weights on the connections among the network layers to attain the expected output is known as the network training. The internal process in the network training is called as learning. Basically, the training process is divided into three types i) supervised ii) Unsupervised and iii) Reinforcement training. For both Back-propagation and for LVQ we are using supervised learning to train the data.

Supervised Learning Rule: It is a procedure of contributing the networks with a sample of inputs and collating the output with a target output. Training process continues until we get the target output. The weights must adjust according to the algorithm. The various learning rules that follow the supervised learning are Delta rule, generalized delta rule, Competitive learning rule. Generalized delta rule is used to train the given data set in the back propagation algorithm, where as competitive learning is the process used to train dataset used for LVQ.

- Delta-Rule: This rule is purely based on the least mean squared error (LMS). The Mean squared error is nothing but the average of all the errors calculated between the target and actual values. This rule is used to minimize the error. Let discuss in detail, for a taken input data the output data is equated with a target output. If the difference between target and actual data is zero, no learning process is considered, otherwise the values of weights are adjusted to lessen the error obtained. The difference between the target output to the actual output value is defined as Δ (wij) = n* k_i *er_j, where n is the learning rate (α), k_i is the activation of unit and er_j is the difference between the target value and actual output value. This learning rule not only progress the weight vector nearer to the target weight vector, it does so in the most efficient way.

Generalized delta rule: Actually the delta rule uses the local information about the error, where the generalized delta rule deals with error information that is not local. The rule is stated in simple sense as follows for weights updating in a cycle after all the training patterns are presented as $W^{new}=w^{old}-n*(E(k))$ where n is learning rate and E(k) is the error difference between the target and actual output.

Competitive Learning Rule: In this competitive learning rule, the neurons present in the output-layer of the neural network compete among themselves to be in an active-state. The major idea behind this rule is that to allow the processing units (neurons) to challenge for the authority to answer a taken sets of inputs, such that only a output neuron (processing unit) challenge for the right to respond for a given subset of inputs. So that only a neuron in the output-layer is in an active-state at a time. The neuron which wins in the competition is known as winner-takes-all neuron. Let W_{kj} denotes the weight of input-layer node (unit) j to neuron. The neuron learns by altering the values of weights from inactive input mode to active input mode. If a neuron (processing unit) does not give acknowledgement to a particular input layout, then the learning does not happens in that particular neuron. If any of the neuron wins in the competition, then its weights are adjusted as follows.

ΔW_{kj}= n (X_j-W_{kj}), when neuron k wins the competition.

=0 ,when neuron k losses the competition.

As from above formulae “n” is well known as the learning- rate(α). The values of the weights are initially set to random values and those weights are being normalized during learning phase (either supervised or unsupervised). The winner-takes-all neuron is selected by using Euclidean distance.

c) Activation Function

The activation function is used to calculate the output comeback generated by neurons. Threshold function performs final mapping of activations of network neurons. The outcome of any neuron is a result of thresholding (internal activation). The aggregate of the weighted input signals is pertained with activation function to get the response. There may be linear and non-linear activation functions. Generally, the activation functions are classified into different types[2]:

i. Identity Function.
ii. Binary Step-Function.
iii. Bipolar step function.
iv. Sigmoidal function.
v. Ramp function.

We use sigmoidal function for the backpropagation algorithm, competitive activation function for the LVQ.

Sigmoid function: Generally these functions are represented by S-shaped curved. These functions are differentiated by its output ranges. Hyperbolic tangent activation function is the most important of all the sigmoid functions with the range of (-1,1). Logistic function has its range of values in between (0, 1). These functions are represented as follows:

a. *Hyperbolic tangent sigmoid activation function:*
$S(y)=tanh(y)= (e^{y}-e^{y})/(e^{y}+e^{y})$
b. *Logistic sigmoid function:*
$S(y)=1/(1+e^{yx}).$

The graphical representation of above sigmoid functions is shown in following FIG:1.3 (Logistic Function) , fig: 1.4 (Hyperbolic tangent function)

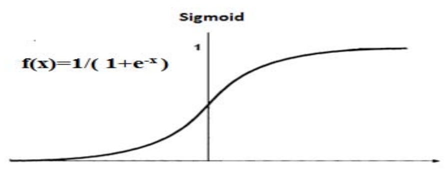


Fig:1.3 : Logistic sigmoid function

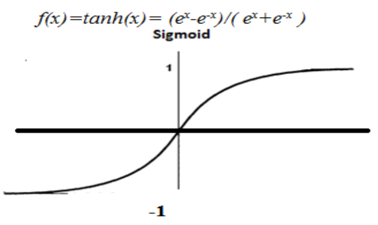


Fig:1.4 : Hyperbolic tangent sigmoid function

The representation of range of the each activation function are defined in table :1.1.

Table 1.1 : Description of activation functions

Function	Definition	Range
Identity	X	(-∞,∞)
Logistic	$S(x)=1/(1+e^{-x}).$	(0,1)
Hyperbolic tangent	$S(x)=(e^{x}-e^{-x})/(e^{x}+e^{-x})$	(-1,1)
Ramp	R(x)= x , x>=0 =0 , x<0 R(x)=max(x,0)	[-1,+1]
Step	O, if x<01, x>=1	[0,+1]

II. Overview of Backpropagation and LVQ

a) Backpropagation Algorithm

Backpropagation is one of the neural network learning algorithms, delineated to diminish the mean square error. Backpropagation is also well-known as the “error backpropagation”, because this algorithm is purely based on the error correction learning rule. This algorithm is used to train the multi-layer artificial neural network. Back propagation uses supervised learning rule, in which it generates error by comparing target output to actual output. The backpropagation algorithm could be broken down into four main steps[1][2]:

- Initialization of weights and bias.
- Implementation of feed forward technique to input training patterns.
- The method of calculating and backpropagating the associated errors.
- Weights Updation.

During the first stage, the weights are set-up to some random values (e.g., they ranges from [-1.0,1.0]or[-0.5,0.5])[2].Every processing unit in the network is associated with a bias (threshold), which is used to generate the net input. After the initialization of weights and bias, each training tuple is processed by remaining steps. First of all, the training tuple is pass to the networks input layer. During the process of feed forward of input training patterns, each input unit encounters an input signal and transfer these signals net-input to every hidden units in the network.Later each hidden unit in the network then figure-outs the activation function response. The activation function is known as the output response of the unit (neurons), where in backpropagation we use sigmoidal activation function. Fig: 2.1 show the neuron output generation in hidden and output layer diagrammatic representation. As from the fig: 2.1 the output of neuron is generated by using activation function i.e,

$$f(wp+b)= 1/(1+1+e^{-n}), \text{ where } n=wp+b.$$

Every hidden unit in the network then figure-outs the activation function as shown above and sends its signal to the output unit. The output unit performs the activation function and generates the outcome of the neural network for the given input pattern.

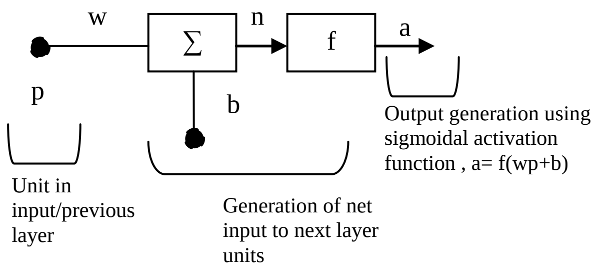

w
n
a
Σ
f
p
b
Unit in input/previous layer
Generation of net input to next layer units
Output generation using sigmoidal activation function , a= f(wp+b)

Fig. 2.1 : Activation function generation

During back-propagating the errors, each output units equates its calculated activation function value (a=f(wp+b)) with its target value to determine the error associated with the network. Based on the error, the factor ∂ is computed in backpropagation network for hidden and output layers. As in the final stage , the weights and the bias are updated based up on this factor ∂ and the activation. The backpropagation algorithm implementation is represented in flow chart from fig: 2.2

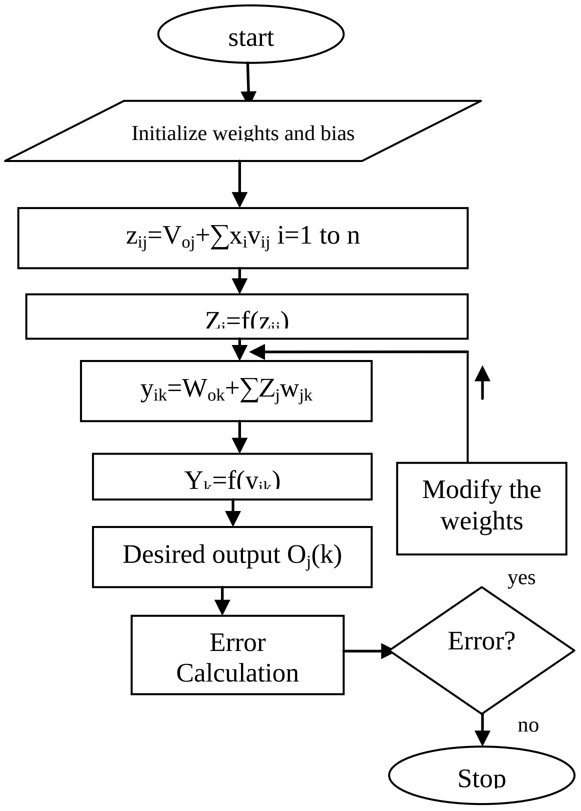


Fig: 2.2 : Flow chart of backpropagation algorithm

The algorithm used in the back-propagation network to train the network is implemented in four different stages is as follows:

Weights Initialization[2] :

Step-1: Initializing the weights and bias to random values (ranges from [-1.0,+1.0] or [-0.5,0.5]).

Step-2: Checking for the stopping condition, if it is false do the steps from 3 to 10.

Step-3: Foe each and every training set, perform the steps from 4 to 9 as mentioned below.

Feed-Forward of input training patterns[3]:

Step-4: Each and every input unit accepts the input x_i and transmits that input signal to hidden layer units.

Step 5: Each hidden unit in the network aggregates its weighted input signals. Activation function to z_{ij} is denoted by Z_j

$$z_{ij} = v_{oj}+\sum x_i V_{ij} \quad .i=1 \text{ to } n$$

$$Z_j= f(z_{ij})$$

The result obtained from this activation function is the input to next layer in the network.

Step 6: Each output unit in the network, aggregates its weighted input signals . Activation function applied to y_{ik} is denoted byY_k

$$y_{ik}= w_{ok}+\sum Z_j W_{jk}$$

$$Y_k=f\ (y_{ik})$$

Backpropagation of the errors:

Step 7: Error is calculated as

$$E(k)= \sum[O_j(k)-T_j(k)]2 \quad j=1 \text{ to } m$$

$$E=E(k)\ f(y_{ik})$$

Step 8: Find the mean squared error

$$E_t=1/2 \sum E \ k=1 \text{ to } N$$

Updating of weights and bias

Step 9: For the Output layer the weights and the bias are updated as follows

$\Delta W_{jk}=\alpha E_t z_j$. Updated weight is as follows W_{jk} (new) = W_{ik}(old) + ΔW_{ik}

$\Delta wok=\alpha E$. To update bias is w_{ok}(new) =w_{ok}(old) +Δw_{ok}

Similarly the values of weights and the bias are updated in the networks hidden layer is as follows:

$\Delta V_{ij}=\alpha E_t x_i$. The new weight is calculated as V_{ij}(new) =V_{ij}(old)+ΔV_{ij}

$\Delta v_{oj}=\alpha E$. Updated bias is v_{oj}(NEW)=v_{oj}(OLD)+Δv_{oj}

Step 10: Check the stopping condition.

Based upon the algorithm stated above the terms are defined as

x_i– Inputs that given to the input units.
v_{oj} – Bias used in the hidden layer units.
V_{ij} – Weights used in hidden layer units.
w_{ok} – Bias used for the outputunits.
W_{jk}– Weights that initialized in output layer.
α– Learning rate.

a) Learning Vector Quantization (LVQ)

Learning Vector Quantization (LVQ) algorithm is the prototype based supervised classification algorithm. It is a particular case of artificial neural network, which implements "winner-take-all" principle[2]. Winner-take-all is the computational principle applied by which neurons in layer compete with each other for activation. The neuron with highest activation stays active while other neurons shut down. LVQ is trained to classify the inputs according to the given targets. Training in LVQ occurs by performing the competition between the neurons. LVQ uses Euclidean distance to perform the competition between neurons. LVQ performs the classification for every target output unit by considering its input pattern i.e, it uses supervised learning technique.

LVQ defines the class boundaries based upon its prototypes. The prototypes are determined during the training procedure using a labeled dataset (the dataset that we take for training).LVQ system is represented by protocols which are defined in future of observed data. The class boundaries are not depends not only on prototypes but also on nearest neighbor rule and winner-takes-it-all. Weight vector for an output unit in a network is known as the "codebook vectors (CV)" or "reference". The architecture of the LVQ algorithm is as shown fig:2.3, fig:2.4 :

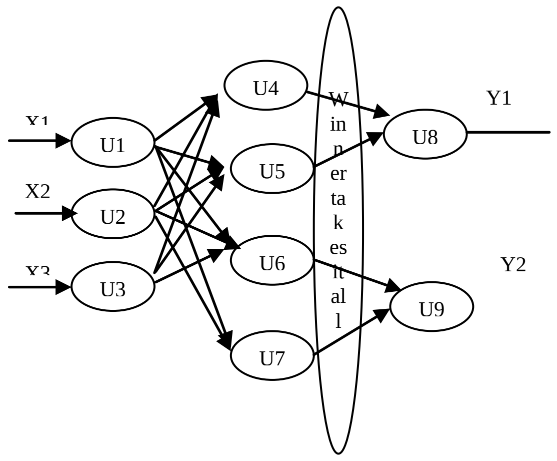


Fig. 2.3 : LVQ architecture

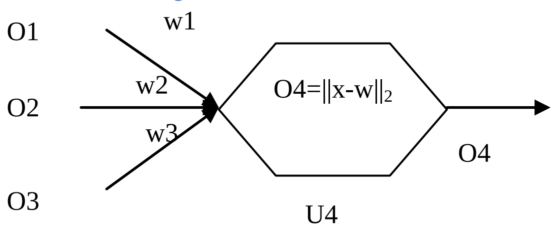
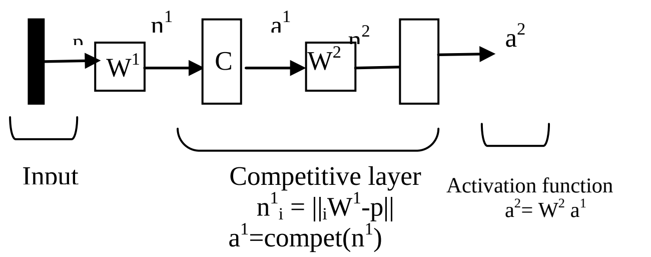


Fig. 2.4 : Inner working of neurons

To express in terms of neural networks, LVQ is a feed-forward neural network. Codebook vector is describe as weight vector(values of weights) of the interconnected weights between all the input layer neurons and hidden neurons. Learning method used in this LVQ algorithm is modifying the weights according to the rules specified and changing the position of code vector (CV) in the input space. Changing the position of CV is nothing but implementing the winner-takes-it-all principle by moving the winner closely if it correctly analyzes the data point or by moving the winner away if it analyzes the data point incorrectly. The working of LVQ is stated diagrammatically in the Fig:2.4



As from the above diagram the net input to the hidden layer is :

$n^1_i = ||_iW^1\text{-}p||$ where ${}_iW^1$ represents training vector i.e,, inputs given to the input layer p represents Weight vector for the units in next layer it is also called as the codebook vector.

Finally the net output of this input layer is passed to the activation function, where we use the competitive activation function for this LVQ algorithm. Competitive Activation Function which represents the input/output relation that purely derives by using the Euclidian rule in which

a^1= compet(n^1)

a^1= 1 neuron which wins the competition

=0 for all neurons.

Therefore the neuron whose weight vector is nearest to the input vector will gives output as 1, and the remaining neurons will gives the output as 0 as shown above. This states that the LVQ network purely competitve network . As initially stated that the neurons in input layer are considered as the same class, after this net output generation to the hidden layer the winning neuron represents a subclass. There may be different neurons that may win the competition, they all belongs to the same sub class.

The hidden layer of the LVQ (learning vector quantization) network combines all subclasses into a single class. As shown in the above figure W^2 done the whole process of combining all the sub classes. W^2 is represented in matrix, in which columns represent the subclasses and the rows represents the classes.

Note: W^2 matrix has a value of 1 in each column, eith the other values set to zero (0).The subclass of a particular class is denoted by the value of 1 in the row. Ex: W^2_{ij}=1 means j sub class is a part of ith class.

The input vector **X** is selected at random from the inputs given. If the class labels of the input vector **x** and a codebook vector (weight vector) **W** agree, the codebook vector **W** is moved in the direction of the input vector **X**. If the class labels of the input vector **X** and the ~~vector~~ codebook vector **w** is disagreed, the codebook vector **W** is moved away from the input vector **X.**

I. Ex: Let $\{W_j\}^1_{i=1}$ stand for the set of weighted vectors (codebook vectors), and the $\{X_i\}^N_{i=1}$ stand for the set of input vectors. Suppose, that the codebook vector W_c is the nearest to the input vector X_i . Let K_{wc} denote the class associated with the codebook vector W_c and K_{xi} denote the class label of the input vector X_i. The values of K_{wc} and K_{xi} are obtained from the W^2 . The codebook vector W_c is regulated as follows:

   If $K_{wc} = K_{xi}$,then W_c(New) = W_c(Old) + $\alpha_n[X_i - W_c(\text{Old})]$ where $0< \alpha_n<1$.

   If $K_{wc} \neq K_{xi}$,then W_c(New) = W_c(Old) - $\alpha_n[X_i - W_c(n)]$,where $0< \alpha_n<1$.

II. Remaining codebook Vectors are not modified.

The learning rate (α) is decreased. This whole LVQ process continues until the stopping condition fails.

Learning Vector Quantization Algorithm[2]*:*

Step-1: Initialize weights vectors (codebook vectors) and learning rate.

Step-2: Check for the stopping condition. If the condition is false, then perform the steps from 3 to 7.

Step-3: For every training input vector p, do the steps from 4-5

Step 4: Figure out J using Squared Euclidean distance

$E(j) = \sum (_jW^1\text{-}X_i)$ where X_i is input present in the input vector.

Find j when E(j)is minimum

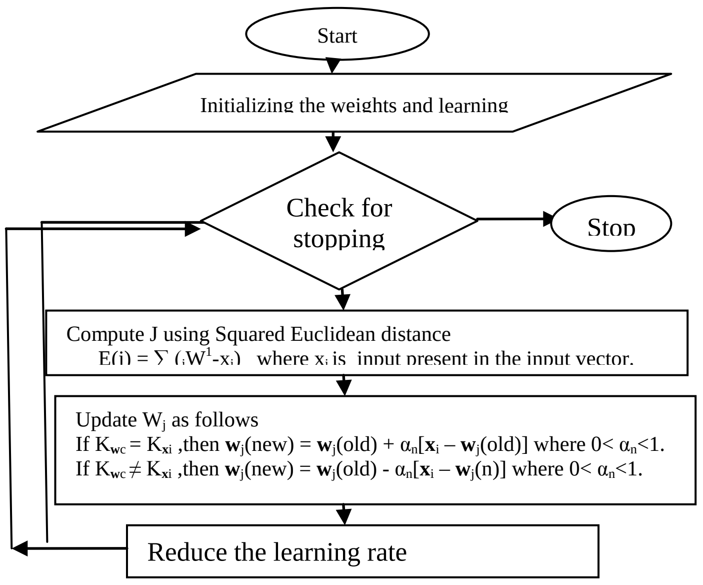
Step 5: The value of W_j is updated as follows

If $K_{wc} = K_{xi}$,then W_j(New) = W_j(Old) + $\alpha_n[X_i - W_j(\text{Old})]$ where $0< \alpha_n<1$.

If $K_{wc} \neq K_{xi}$,then W_j(New) = W_j(Old) - $\alpha_n[X_i - W_j(n)]$ where $0< \alpha_n<1$.

Step 6: Reduce the learning rate.

Step 7: Test for the stopping condition.



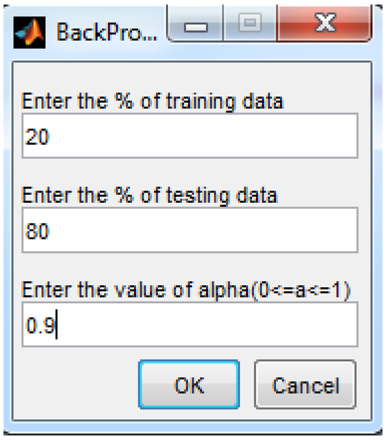
Start
Initializing the weights and learning
Check for stopping
Stop
Compute J using Squared Euclidean distance
E(j) = ∑ (jW¹-xi) where xi is input present in the input vector.
Update Wj as follows
If Kwc = Kxi ,then wj(new) = wj(old) + αn[xi – wj(old)] where 0< αn<1.
If Kwc ≠ Kxi ,then wj(new) = wj(old) - αn[xi – wj(n)] where 0< αn<1.
Reduce the learning rate

III. Comparision Between Backpropagation and LVQ

The practical implementation of back-propagation involves factors like choice of network architecture, momentum factor. While implementing these factors backpropagation algorithm associated with few problems like local minima. A local minimum is the problem that occurs frequently, used to change the weights frequently to minimize the error. As in this local minima, in some cases the error might have to rise part of more general fall. If this is the situation the algorithm will struck and the error will not be decreased further. So, for this drawback LVQ gives best results. In this paper we are comparing the efficiencies obtained for testing the heart disease dataset with both backpropagation and LVQ for the two different ranges (-1,1) and (0,1). The following are the results obtained while comparing the both algorithms. The programming is written for 100 instances of a heart diseases dataset from Cleveland with 14 attributes (13 +class attribute).

a) *BackPropagation*

In our paper we practice backpropagation algorithm with different learning rates and finally conclude, how the efficiency changed based upon the value of alpha (learning rate) . To allow fair comparison between backpropagation and LVQ a wide variety of parameter values are tested for each algorithm.

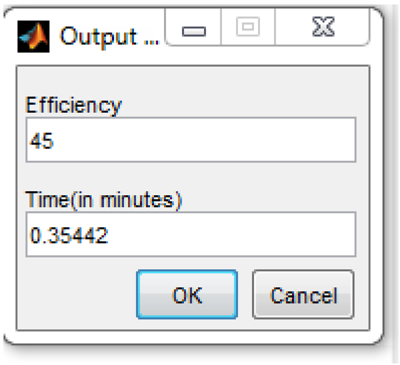


Enter the % of training data: 20

Enter the % of testing data: 80

Enter the value of alpha(0<=a<=1): 0.9

Fig: 3.1 : Input to backpropagation algorithm



Efficiency: 45

Time(in minutes): 0.35442

Fig. 3.2 : Output generated for fig:3.1

Global Journal of Computer Science and Technology (E) Volume XVI Issue V Version I Year 2016

The backpropagation network is trained on our dataset for different alpha values for different ranges and the observed results are mentioned in the below tables as follows:

When i)α=0.9 (learning rate)

Table. 3.1 : Efficiency obtained for backpropagation (digital) α=0.9

Sl.No	Training(%)	Testing(%)	Time(in minutes)	Efficiency(in%)
1	20	80	2.2	45
2	40	60	0.35	55
3	60	40	0.007	77.5
4	80	20	0.009	75

ii) α=0.8 (learning rate)

Table. 3.2 : Efficiency obtained for backpropagation (digital) α=0.8

Sl.No	Training(%)	Testing(%)	Time(in minutes)	Efficiency (%)
1	20	80	0.003	28.75
2	40	60	0.005	23.333
3	60	40	0.005259	50
4	80	20	0.008362	50

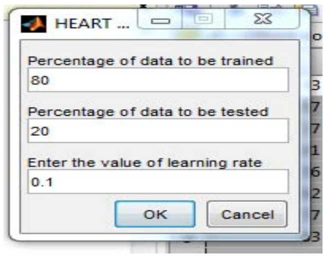
Table. 3.3 : Efficiency obtained for backpropagation (analog) α=0.1

Sl.No	Training(%)	Testing(%)	Time(min)	Efficiency(%)
1	20	80	0.0032099	38.75
2	40	60	0.006441	43.333
3	60	40	0.075057	40
4	80	20	0.010575	60

Table. 3.3 : Efficiency obtained for backpropagation (digital) α=0.1

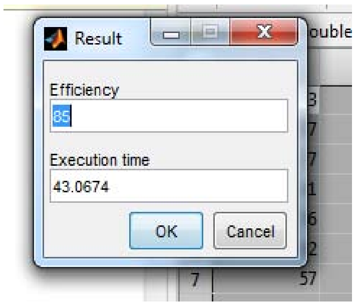
Sl.No	Training(%)	Testing(%)	Time(min)	Efficiency(%)
1	20	80	0.0032	62.5
2	40	60	0.00503	63.333
3	60	40	0.0066	60
4	80	20	0.00888	79

b) Learning Vector Quantization



Percentage of data to be trained
80
Percentage of data to be tested
20
Enter the value of learning rate
0.1

Fig.3.3 : Input to LVQ algorithm



Efficiency
85
Execution time
43.0674

Fig.3.4 : Output generated for fig:3.3

Varying the learning rate alpha from 0.1 to 0.9, it was found that the maximum efficiency is obtained at alpha α=0.1. The results that obtained for various alpha values are shown in the following tables.

Table. 3.4 : Efficiency variations in LVQ analog for α=0.9

Sl.No	Training(%)	Testing(%)	Time(min)	Efficiency (%)
1	20	80	23.7953	54
2	40	60	25.186	57
3	60	40	10.7664	60
4	80	20	10.164	70

Table. 3.5 : Efficiency variations in LVQ analog for α=0.1

Sl.No	Training(%)	Testing(%)	Time(min)	Efficiency
1	20	80	6.5829	64
2	40	60	6.2778	70
3	60	40	8.4187	70
4	80	20	7.175	85

Our paper also attempts to check the efficiency for different ranges i,e for analog (0,1) and bipolar (-1,1). Table:3.3 and Table:3.4 are the results obtained for analog, where the bipolar results are shown in Table:3.5.

Table. 3.6 : Efficiency variation in LVQ bipolar α=0.1

Sl.No	Training(%)	Testing (%)	Time(inmin)	Efficiency(%)
1	20	80	8.7658	70
2	40	60	9.0779	62
3	60	40	12.1897	80
4	80	20	97.8381	70

The better classification efficiency can be achieved by varying the learning rate. As from the above results , we found that the digital gave better efficiency than analog in vector quantization method. It is also found that maximum efficiency was obtained for alpha value 0.1.

IV. Conclusion

In this paper we present a supervised learning based approach to data-mining classification rules for a dataset. The classification is carried out using backpropagation and LVQ. We conclude that LVQ algorithm is one of the best in classification when compared to backpropagation. As from the results obtained for classifying our dataset, we can obtain better classification efficiency by varying the learning rate and it was found that maximum efficiency was obtained for alpha value 0.1 in both algorithms. Comparing the digital results (-1,1) with the analog results, it is found that the digital data gave better efficiency than analog in both back-propagation and LVQ algorithms. Overall comparison between the two algorithms states that the maximum efficiency is obtained in LVQ with high processing time.

References Références Referencias

1. Dr.YashPaul Singh. ,Alok Singh Chauhan (2009)' NEURAL NETWORKS IN DATA MINING',*Journal of Theortical and Applied Information Technology.*
2. S.N.Sivanandam.,S.N.Deepa.,S.Sumathi(2006)*Introduction to Neural Networks Using Matlab 6.0,*Noida: Tata MCGraw-Hill.
3. Athira Mayadevi Somanathan., V.Kalaichelvi(2014) 'An Intelligent Technique for Image Compression', *International Journal for Recent Development in engineering and Technology,*Volume 2(ISSN 2347-6435(online)).
4. A.K.Jain.,J.Mao., and K.M.Mohiuddin,Artificial Neural Networks: A tutorial, IEEE computer VOL.29,no.3,1996,pp.31-44.
5. Priyanka Gaur(n.d)'Neural Networks in Data Mining', *International Journal of Electronic and computer Science Engineering,*IJECSE,Volume 1, Number (ISSN 2277-1956/VIN3-1449-1453),pp. [Online].
6. Agrawal, R., Imielinski, T., Swami, A., "Database Mining : A Performance Persepective", *IEEE Transactions on Knowledge and Data Engineering,* PP.914-925, December 1993.
7. Zurada J.M., "An introduction to artificial neural networks systems", st.paul: West Publishing (1992).
8. Y.Bengio, J.M.Buhmann, M.embrechts, and J.M.Zurada, Introduction to the Special issue on neural networks for data mining and knowledge discovery. *IEEE Trans. Neural Networks.*
9. Haykin, S., *Neural Networks,* Prentice Hall International Inc., 1999.
10. Bradely, I., Introduction to Neural Networks, Multinet Systems Pty Ltd 1997.