Journal of Computer Science 4 (7): 578-584, 2008
ISSN 1549-3636

Effect of Hidden Layer Neurons on the Classification of Optical Character Recognition Typed Arabic Numerals

[1]Nidal F. Shilbayeh and [2]Mahmoud Z. Iskandarani
[1]Faculty of Information Technology for Graduate Studies,
Middle East University for Graduate Studies, P.O. Box 42, Post Code: 11610
[2]Faculty of Science and Information Technology, Al-Zaytoonah Private University of Jordan,
P.O. Box 911597, Post Code: 11191, Amman-Jordan

Abstract: Problem statement: The effect of varying the number of nodes in the hidden layer and number of iterations are important factors in the recognition rate. In this paper, a novel and effective criterion based on Cross Pruning (CP) algorithm is proposed to optimize number of hidden neurons and number of iterations in Multi Layer Perceptron (MLP) neural based recognition system. Our technique uses rule-based and neural network pattern recognition methods in an integrated system in order to perform learning and recognition of dynamically printed numerals **Approach:** The study investigates the effect of varying the size of the network hidden layers (pruning) and number of iterations (epochs) on the classification and performance of the used MLP. The optimum number of hidden neurons and epochs is experimentally established through the use of our novel Cross Pruning (CP) algorithm and via designing special neural based software. The designed software implements sigmoid as its shaping function. **Results:** Experimental results are presented on the classification accuracy and recognition. Significant recognition rate improvement is achieved using 1000 epochs and 25 hidden neurons in our MLP OCR numeral recognition system. **Conclusions/Recommendations:** Our approach has a significant improvement in learning and classification of any numeral, character MLP based recognition system.

Key words: Neural network, MLP, pruning, pattern recognition, classification, OCR

INTRODUCTION

A major problem in applying neural networks is specifying the size of the network. Even for moderately size networks the number of parameters may become large compared to the number of data[1-4].

Artificial neural networks have been successfully applied to problems in pattern classification, function approximation, optimization, pattern matching and associative memories[20]. Multilayer feed forward networks trained using the back propagation-learning algorithm is limited to searching for a suitable set of weights. This initiates the problem of selecting appropriate topology (number of hidden layers) and weights to solve the learning problem in hand. Two small networks are unable to adequately learn the problem while excessively large networks tend to over fit the training data and consequently result in poor generalization and weak performance.

Most practical learning problems are known to be computationally complex and hard to optimize. Network pruning offers an excellent approach for dynamically determining an appropriate network topology. Pruning techniques involves training different network sizes and compare the performance in terms of classification and error. The process comprises methodical and consistence elimination or addition of neurons which subsequently removes or adds weights to the hidden layers of the system.

Most of available systems capture inputs as a sequence of coordinate points, taking into account character blending and merging and the problem of characters that have close similarity to each other. This is solved via pre-processing of the characters prior to recognition, hence, performing a shape recognition process. The post-processing recognition process can be achieved using zones that define directions of travel, where characters are recognized as connected zones using a lookup table for matching and classification.

The primary task of OCR recognition is to take an input and correctly assign it as one of the possible output classes. This process can be divided into two

Corresponding Author: Nidal F. Shilbayeh, Faculty of Information Technology for Graduate Studies, Middle East University for Graduate Studies, P.O. Box 42, Post Code: 11610

general stages: feature selection and classification. Feature selection is critical to the whole process since the classifier will not be able to recognize from poorly selected features[21].

Some requirements for character recognition system design suggest themselves such as:

- To create a system which can recognize non-perfect numerals
- To create a system which can use both static and dynamic information components
- To perform recognition with image data presented at field rate or as fast as possible
- To recognize as quickly as possible, even before the full image sampling is completed
- To use a recognition method which requires a small amount of computational time and memory
- To create an expandable system which can recognize additional types of characters
- High recognition rate
- In the event of an error, non-recognition is implemented instead of miss-recognition, especially in critical conditions
- Recognizes inputs quickly
- Require minimal training

In this study network performance is examined as a function of both network size and number of iterations. This is carried out using a feed forward multi layer perceptron neural model. This system captures and intelligently recognizes numerals. There are many proposed methods for on-line recognition, which use a wide variety of pattern recognition techniques. Neural networks have been proposed as good candidates for character recognition. Studies have also been carried out for OCR recognition, comparing techniques such as dynamic programming, Hidden Markov Models and recurrent neural networks[13-16].

MATERIALS AND METHODS

System design: Figure 1 shows the used OCR MLP. In designing our OCR system the following factors are considered:

Initial weights: Since we used a gradient decent learning algorithm that is known to dynamically treat all weights in similar manner and to avoid a situation of no learning, random function is included in our algorithm. Such a function is specifically designed to guarantee that the network will not end in global minima which will adversely affect the system performance and will result in massive errors and miss classifications.

Learning rate: The learning rate is bounded due to the following:

- Two small a learning rate will cause the system to take a very long time to converge
- To large a learning rate will cause the system to oscillate indefinitely and diverge

Based on the above two mentioned factors our learning rate was chosen to be equal 0.25.

Transfer function: Even though some literature indicates that anti symmetric functions will cause the system to learn faster, this research used the standard sigmoid function which is known to be a stable function due to its range from 0-1.

All the above is considered in our research when hidden layer neurons and number of epochs are closely examined.

Mathematical modeling: Figure 2 shows our designed and implemented MLP neural engine used in our OCR system and characterized by the following three main equations:

$$out_i^{(0)} = in_i \rightarrow \text{The output of the input layer} \quad (1)$$

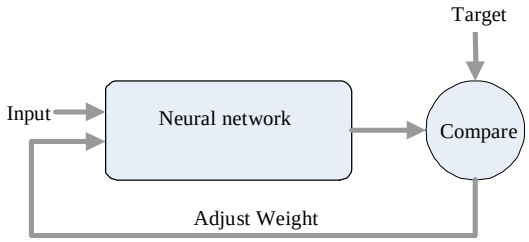

Fig. 1 Used OCR MLP system

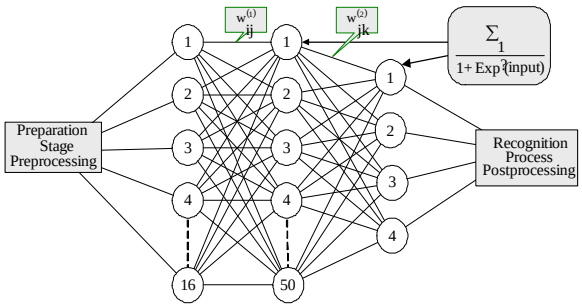

Fig. 2: OCR neural classification system

$$out_j^{(1)} = f\left(\sum_j out_i^{(0)} \cdot w_{ij}^{(1)}\right) \rightarrow \text{The output of the hidden layer} \quad (2)$$

$$out_k^{(2)} = f\left(\sum_j out_j^{(1)} \cdot w_{jk}^{(2)}\right) \rightarrow \text{The output of the output layer} \quad (3)$$

Using Eq. 1-3 we can rewrite Eq. 3 to become:

$$out_k^{(2)} = f\left(\sum_j out_j^{(1)} \cdot w_{jk}^{(2)}\right) = f\left(\sum_j f\left(\sum_i in_i w_{ij}^{(1)}\right).w_{jk}^{(2)}\right) \quad (4)$$

Learning in an MLP is known to occur via modification of weights as indicated in the following equations:

$$\Delta w_{jk}^{(2)} = \eta \sum_p \left(t\arg_k - out_k^{(2)}\right) \cdot out_k^{(2)} \cdot \left(1 - out_k^{(2)}\right) \cdot out_j^{(1)} \quad (5)$$

$$\Delta w_{ij}^{(1)} = \eta \sum_p \left(t\arg_k - out_k^{(2)}\right) \cdot out_j^{(1)} \cdot \left(1 - out_j^{(1)}\right) \cdot out_i^{(0)} \quad (6)$$

Where η the learning rate and p is is the pattern on which the equation is applied.

It is clear that for the output layer the hidden layer appears as an input layer and for the input layer the hidden layer appears as an output layer.

Equation 5 and 6 are obtained using the sigmoid function as the shaping function, which takes the following form:

$$f(input) = \frac{1}{1 + \exp(-input)} \quad (7)$$

Applying Eq. 7 to Eq. 1, 2 and 3 we obtain the shaped outputs for the three layers used in our MLP.

$$layer(0): out_i^{(0)} = in_i \quad (8)$$

$$layer(1): out_j^{(1)} = \frac{1}{1 + \exp\left(-\left(\sum_i out_i^{(0)} \cdot w_{ij}^{(1)}\right)\right)} \quad (9)$$

$$layer(2): out_k^{(2)} = \frac{1}{1 + \exp\left(-\left(\sum_i out_j^{(1)} \cdot w_{jk}^{(2)}\right)\right)} \quad (10)$$

Calculation of weight adjustments in our used MLP follows the following steps:

- Applying Eq. 9, 10 and 11 will produce all layers outputs
- Using Eq. 5 and 6, the necessary weight modification values are obtained
- Adding the results of 5 and 6 to the original weights according to the following equations will update the weight matrix

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \quad (11)$$

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk} \quad (12)$$

Each time Eq. 11 and 12 are applied, a single cycle is completed (epoch). However there is a need to compute the sum squared error for each two layers and for the overall network using the following equation

$$E(w_{ik}) = \frac{1}{2}\sum_p \sum_j \left(t\arg_k - out_k\right)^2 \quad (13)$$

RESULTS

Table 1-3 show the effect of varying both number of hidden neurons and number of epochs on the recognition of Arabic numerals, while Table 4 and 5 clearly shows classification accuracy and sum squared

Table 1 Recognition results using 100 epochs

Hidden	Reference									
Neurons	0	1	2	3	4	5	6	7	8	9
5	7	7	0	7	0	7	5	7	7	7
10	7	3	5	7	6	7	6	7	7	7
15	0	0	0	1	1	5	6	7	1	1
20	7	7	6	5	4	5	5	7	5	7
25	0	1	2	1	0	5	6	7	8	9
30	0	1	2	1	4	5	6	7	8	9
35	0	1	2	9	4	5	6	7	9	9
40	2	1	2	3	4	5	6	7	8	9
45	0	1	2	3	4	5	6	7	8	9
50	0	1	2	3	4	5	6	7	8	9

Table 2 Recognition results using 1000 epochs

Hidden	Reference									
Neurons	0	1	2	3	4	5	6	7	8	9
5	2	0	2	9	4	5	5	7	9	9
10	7	7	7	7	5	5	6	7	6	6
15	0	1	2	3	4	5	6	7	8	9
20	0	1	2	3	4	5	6	7	8	9
25	0	1	2	3	4	5	6	7	8	9
30	0	1	2	3	4	5	6	7	8	9
35	0	1	2	3	4	5	6	7	8	9
40	0	1	2	3	4	5	6	7	8	9
45	0	1	2	3	4	5	6	7	8	9
50	0	1	2	3	4	5	6	7	8	9

errors. Table 6 and 7 show the extracted weight matrices for both input-hiddenand hidden-output, which is made available via the specially Designed Brain Database (DBD) that automatically adds the new trained engine weights to its contents while keeping track of all previously used weights.

Table 3 Recognition results using 10000 epochs

Hidden	Reference									
Neurons	0	1	2	3	4	5	6	7	8	9
5	6	7	6	6	5	5	6	7	6	6
10	7	7	0	7	7	7	7	7	7	7
15	7	7	2	1	7	7	7	7	8	9
20	0	1	2	3	4	5	6	7	8	9
25	0	1	2	3	4	5	6	7	8	9
30	0	1	2	1	1	7	7	7	8	9
35	0	1	2	3	4	5	6	7	8	9
40	0	7	2	3	3	5	6	7	8	5
45	0	1	2	3	4	5	6	7	8	9
50	0	1	2	3	4	5	6	7	8	9

Table 4 Percentage classification

Hidden Neurons	Percentage Classification 100 Epochs	Percentage classification 1000 Epochs	Percentage classification 10000 Epochs
5	10	40	30
10	20	30	10
15	40	90	50
20	30	90	100
25	80	100	100
30	90	100	60
35	80	100	100
40	90	100	70
45	100	100	100
50	100	100	100

DISCUSSION

Traditionally, pruning is carried out on hidden layers and hidden layer neurons however we introduce in our work the principle of Cross Pruning (CP) or what we can call the pruning matrix whereby we carried out pruning on both hidden layer neurons and number of epochs which serves as a two dimensional controlling matrix function for the design and implementation of MLP based recognition systems[5-12,17-19].

Figure 3 shows the effect of pruning and learning process on the classification accuracy and recognition rate of our MLP based OCR system. It is clear from the graphs that best accuracy and classification is obtained when using 1000 epochs and 25 hidden neurons. It observed that using 100 epochs resulted in under learning and poor recognition of tested numerals with slow oscillations due to slow recognition time while at the other extreme of 10000 epochs' instability and fast oscillations which indicates shorter recall time and over learning with slightly better recognition than 100 epochs case.

Table 5 Classification and sum squared errors

Hidden Neurons	Sum squared error (100 Epochs)	Sum squared error (100 Epochs)	Sum squared error (100 Epochs)
5	0.5090	0.35100	0.50000
10	0.5060	0.50080	0.50000
15	0.1270	0.00390	0.13070
20	0.5033	0.00294	0.00048
25	0.1127	0.00293	0.00032
30	0.0538	0.00581	0.02530
35	0.0986	0.00330	0.00028
40	0.0855	0.05110	0.12500
45	0.0536	0.00106	0.00490
50	0.0150	0.00220	0.00014

Table 6: Hidden-output weight matrix

$W_{1:20}$	$W_{21:40}$	$W_{41:60}$	$W_{61:80}$	$W_{81:100}$	$W_{101:120}$	$W_{121:140}$	$W_{141:160}$	$W_{161:180}$	$W_{181:200}$
-0.7384	-0.716210	0.123308	0.887697	0.014582	0.302913	0.444217	-0.946150	-0.797750	-0.553820
0.187194	0.253935	0.438999	-0.455240	0.383340	0.682899	0.493849	0.095120	0.400409	-0.135570
-0.21763	-0.317750	-0.483780	0.502167	-0.171300	-0.063630	0.298933	-0.295410	-0.962740	0.557803
0.75599	-0.743930	-0.913310	-0.280830	0.712710	-0.129340	-0.103260	-0.448650	-0.692260	-0.491680
-0.06732	0.333320	-0.656110	-0.824540	-0.224590	0.358697	-0.560870	-0.639430	-0.671780	0.939861
-0.01499	-0.419550	-0.458600	-0.640310	-0.681850	-0.103110	-0.320470	-0.240550	0.646042	-0.329020
-0.54418	0.777422	0.416540	-0.477290	0.872413	0.650348	0.730660	-0.541800	0.929757	0.775350
-0.96502	-0.753570	0.263159	-0.238450	0.537143	-0.839140	-0.864750	0.550791	0.242063	0.444053
-0.30225	-0.965410	0.682166	0.236304	-0.064590	0.691711	0.622521	0.081475	-0.810210	-0.132490
-0.61636	-0.040450	-0.576440	-0.886810	0.438242	0.787011	-0.067430	0.939009	-0.432320	0.183444
-0.3216	0.691387	-0.420860	-0.045920	-0.788150	0.812514	-0.938130	0.392654	-0.854190	-0.148270
-0.38814	0.850727	0.848113	-0.108260	0.919719	0.676802	-0.629370	0.827983	-0.848960	-0.126400
0.733219	0.683133	-0.790960	0.072790	-0.914960	0.636549	0.729384	0.508201	0.291481	0.102784
0.478894	0.129642	-0.502290	-0.577420	0.597251	-0.025530	-0.326980	0.383929	-0.723240	0.405845
0.287646	-0.666060	0.892322	0.069366	0.411091	0.684220	0.157430	0.382600	-0.523940	0.899859
0.599062	-0.035420	0.801635	0.915521	0.793179	-0.740560	-0.366740	0.384021	0.287710	0.083222
-0.1732	0.717059	0.169277	-0.607440	-0.448910	0.795384	-0.206470	-0.237550	0.999136	-0.088080
-0.46738	0.401398	0.654608	0.789937	-0.560170	0.277750	-0.575580	-0.645910	-0.699230	0.164478
0.623036	-0.101940	-0.191160	0.487330	-0.338820	-0.064980	0.571663	-0.226630	-0.536870	0.565854
-0.39686	0.634015	-0.609550	0.601572	0.209401	-0.474910	-0.114810	-0.192100	0.649275	0.759553

Table 7: Input-hidden weight matrix

$W_{1:80}$	$W_{81:160}$	$W_{161:240}$	$W_{241:320}$	$W_{321:400}$	$W_{401:480}$	$W_{481:560}$	$W_{561:640}$	$W_{641:720}$	$W_{721:800}$
0.4027637	-0.5742482	0.4578005	0.3506368	0.4889897	-0.7937924	0.6900722	0.7386004	0.1621374	0.08943069
-0.9394081	0.5275164	0.3039289	0.3546679	-0.8948634	-0.4981773	0.5236177	0.1255918	0.02172875	-0.7196238
0.2715555	-0.9868838	0.566203	-0.02375829	-0.2717465	0.3931757	-0.3423754	-0.8909732	-0.3296283	0.2728473
-0.03968	0.05863595	0.252403	-0.9144807	-0.6807699	-0.740478	0.2266734	0.6004941	-0.08280802	0.4745111
0.3040608	-0.2098812	0.1497356	-0.7893106	-0.4596812	-0.5857326	0.9813191	-0.8242253	0.9734839	-0.2927119
0.9177358	0.8665442	0.4649401	0.3511703	-0.2300031	0.5636785	0.207006	-0.8241148	0.2893493	0.3706515
0.5170184	0.1996459	0.8727201	0.756754	0.7771298	0.6495029	-0.5639268	-0.6273028	-0.7374705	-0.3295215
0.2732477	0.3882072	-0.5358007	0.9339237	-0.4577699	0.6648569	0.9054878	0.06583047	0.4556599	0.7762465
0.7173284	-0.3660754	0.6203038	-0.404348	-0.737568	-0.8246866	-0.5976046	0.5107228	-0.08674395	-0.1818048
0.1217806	-0.6624317	-0.7646356	-0.09424472	-0.06968236	-0.1506116	0.07121849	0.4781699	0.3045056	-0.5104096
0.07334101	-0.7207505	0.5150603	0.1776365	0.816187	-0.4141697	0.9611114	0.04445636	0.5330824	0.4586812
0.5989089	0.1713688	-0.8236611	0.5338252	0.3649998	-0.02449656	-0.5226543	0.8147132	-0.4309499	0.4765797
-0.7435418	-0.2814783	0.06640446	0.4583012	0.4742988	-0.615513	0.7459036	0.471594	-0.9520611	-0.9776284
0.4560516	-0.7184691	-0.3504963	0.5807471	-0.2458103	-0.7014725	-0.6365523	0.4729977	-0.2741036	-0.5384662
-0.2844523	-0.6458999	-0.4846817	-0.09921277	-0.5338472	-0.4915675	-0.3838378	0.9276489	-0.276848	0.2894331
0.2372303	0.7266023	-0.05246234	0.7535331	0.5503564	-0.1023617	0.4169786	0.06210828	-0.4869988	0.2445779
-0.8533303	0.2134658	0.9368902	0.1316968	0.01246727	-0.9517418	-0.2873505	-0.4615091	-0.8263949	0.4376153
0.8698895	-0.185235	-0.01419973	-0.9369984	-0.6430304	0.5166867	-0.3937235	-0.6549392	-0.02366066	-0.4419119
0.720616	0.2001985	0.1631669	0.4466995	0.5059577	-0.2744631	-0.04152358	0.5931977	-0.4105011	-0.5270852
0.9822793	-0.6490953	-0.466223	-0.362231	0.386508	0.1315789	0.6776335	0.720247	0.5609334	0.7746992
0.130761	0.1854728	0.08881676	0.4130713	0.1373132	0.1906747	-0.68247	-0.9002808	0.5888907	-0.8944479
0.7599008	0.2378552	-0.7019129	-0.08336782	0.5825293	0.0567224	0.1932461	0.5544424	0.5532365	-0.7206247
-0.9762734	-0.8364061	0.4221071	-0.7074538	0.5289856	0.7035743	-0.7572664	-0.2141463	-0.1094543	-0.3662399
0.6535139	0.01272058	0.669554	-0.6199858	-0.2816968	0.9030333	-0.06063604	-0.8201234	0.06257796	0.8180423
-0.124642	-0.03868783	-0.5563072	0.8469802	0.4392909	-0.7403876	-0.3108054	0.1080376	-0.6303986	-0.1021138
-0.2487924	0.5026224	-0.5531621	0.1329718	-0.7154934	-0.7999933	0.7955377	0.867866	0.8541708	0.6352832
-0.1254867	0.09243691	-0.164282	0.5893282	0.09932339	0.7234284	0.7840034	-0.6695539	-0.4414564	0.8673059
-0.8729672	0.5567579	-0.6508687	0.1522658	0.4784813	0.99435	-0.228879	-0.4084904	-0.1538179	-0.2864037
-0.1228353	0.8375009	-0.8562697	-0.9575676	0.2451824	-0.9669744	-0.6727759	0.2369081	0.155996	-0.07735026
-0.02130246	0.8491538	-0.3695831	0.509994	0.1075547	0.2376773	0.9815466	0.889946	0.5833659	0.9267151
0.993313	0.6275498	-0.06609309	-0.9318753	0.7018617	-0.9633139	-0.5743164	0.5381321	0.6058215	-0.7234541
0.8668251	0.7272983	-0.3922126	0.3862865	-0.2009599	0.2757683	0.7469592	0.6466744	0.4997585	0.6765501
-0.1307865	-0.5017568	-0.7919058	-0.1708602	0.1489846	-0.9748036	0.1970729	-0.4632543	-0.6909636	-0.2882332
0.868274	-0.1501629	-0.5836711	0.1835632	-0.9502873	-0.7481124	-0.5200894	0.09443712	0.2829361	-0.2245717
0.2274157	-0.8232306	-0.8608409	0.4463428	-0.1291562	-0.4781126	-0.8931388	0.1126469	0.5310367	-0.5484906
-0.04397917	0.9258308	0.803066	-0.4451454	0.7125041	0.7366681	0.6746178	0.859165	0.1523354	0.6970699
0.7271146	-0.1359288	-0.3108472	0.9137675	0.6473619	0.3929852	0.000522971	0.5911685	0.5666376	0.5139772
0.4181058	0.3680756	0.2724695	-0.3413157	0.3382721	-0.6029437	-0.1897171	0.355485	0.3641362	-0.7225113
-0.2926854	-0.7251142	0.172788	0.333968	0.1540912	0.8407756	-0.6064237	-0.4434458	-0.7745754	-0.2807461
-0.3718596	0.1729856	-0.6039732	-0.5469859	-0.08401561	0.01681995	0.4809766	0.3462908	-0.2447932	0.07205892
-0.1246935	-0.001904607	0.05424154	-0.9638165	-0.6227468	-0.6579195	-0.9180652	-0.828838	-0.8000821	0.3543421
0.9798777	0.8395383	-0.7180603	0.3407526	-0.3723979	0.6811182	-0.4354174	-0.2172592	0.6386118	0.3603978
0.502956	-0.2601079	0.08016837	-0.3601862	0.3490895	-0.9054531	-0.3649281	-0.9769224	0.3790487	-0.8414851
0.2983451	-0.3602171	-0.8834872	-0.02067542	-0.3579752	-0.5623651	-0.2566938	0.9518476	0.8631823	0.1903112
-0.8066941	-0.4877459	-0.9106113	0.6257092	-0.2339734	-0.9667512	-0.1655031	-0.4088031	0.6694485	-0.8085202
-0.4599602	-0.9465902	-0.1999667	0.5668044	0.2605982	0.2467735	0.1882441	0.2711713	0.349649	0.7733011
-0.2200106	-0.6707653	0.1552821	0.4777716	-0.447848	0.9801008	-0.3799556	0.8121806	-0.6129495	-0.3531443
0.09468651	0.678884	0.5029879	-0.5828888	-0.7733152	0.5334148	0.7205176	0.3213549	-0.08416295	0.04413629
-0.5780426	-0.526436	0.5945143	0.957369	-0.06949651	-0.01141512	0.6064757	-0.2277743	-0.2184962	0.1734616
0.5010579	0.4336402	0.7910044	0.1367664	-0.4882035	-0.8472621	-0.7385323	-0.6405675	0.7034245	0.6969848
0.0240835	-0.103937	-0.8366741	0.5730613	-0.9353296	-0.09019721	0.7328931	-0.8466457	-0.2441002	-0.2330819
-0.3062971	-0.6441593	0.03598547	-0.7842853	-0.6092103	-0.8618925	0.8844128	0.9799628	-0.9812381	0.5106094
0.8810116	0.8103698	0.8241347	-0.1918596	-0.1667918	-0.7568897	0.1074661	-0.7499849	0.3443114	0.7940618
-0.3944154	0.8201954	0.1163032	-0.6552939	-0.8813763	0.433351	-0.7195308	-0.9347756	-0.09974432	-0.7248745
-0.0526315	0.04547942	-0.3550509	0.1684808	0.7993644	-0.2781183	-0.3900293	0.118498	0.009644389	-0.7873284
-0.7494919	-0.5196548	-0.2814426	-0.7282584	-0.9788196	-0.8464692	-0.04906607	0.3369732	-0.7751868	-0.8671787
0.7306157	-0.7814056	0.8940948	-0.3867959	0.9386183	-0.6581401	0.3146409	0.9216598	-0.6756982	0.549008
0.9967136	-0.2496932	-0.9466045	0.5100813	0.3126473	0.2020435	0.93873	-0.677737	0.6658506	-0.3544593
-0.5458428	-0.7092103	-0.1072458	-0.5693442	0.1865319	0.72098	0.2958328	0.03146422	-0.9038397	-0.3043357
0.1896145	0.1374807	0.2978959	-0.289686	0.4519632	-0.2592914	-0.5885859	-0.9541717	0.3153632	-0.5988715
0.03487837	0.5679888	-0.3121825	-0.3268677	0.3694144	-0.6730913	-0.120999	-0.04816735	0.4311661	-0.9077691
-0.7808204	0.2397215	-0.4229872	-0.7993419	-0.6491747	-0.4928489	0.9082472	-0.7384703	-0.7218807	0.4714384
0.8809975	0.7097527	0.1075469	-0.002158523	-0.4197012	-0.4751045	0.8994881	-0.3756913	0.7107702	0.351272
0.1263788	-0.2810292	0.7965841	0.3420091	-0.7542398	-0.3287528	0.563827	0.5353613	-0.6327147	0.5344031
0.07273018	0.8083819	-0.7374464	0.477353	-0.4372696	-0.9915627	0.03162563	-0.8209654	-0.09066308	0.8921143
0.06119061	-0.9360478	0.1404021	0.4427812	-0.4532437	0.5431318	0.1207836	0.9925215	0.6233919	-0.02733278
-0.0117346	0.1008302	-0.9454139	-0.1928834	0.1648692	0.6752542	0.7506486	0.6935402	-0.1644028	-0.3550512
0.4046729	0.4544296	0.5438457	0.80725	0.4749591	-0.05473113	-0.9559631	-0.9589543	-0.9891291	-0.2387173
0.937555	0.3759311	0.7556323	-0.09756911	-0.5735968	-0.4889206	0.9899219	-0.9781553	0.4243807	0.387745
0.8675952	-0.4604728	-0.04342389	0.2051451	-0.1980705	-0.4389157	0.5491173	-0.1154988	-0.7591417	0.4569664
-0.1335851	-0.07693541	-0.7801665	0.9072319	0.3607754	0.5437602	0.7155222	-0.7436293	-0.00459516	-0.2215229
0.3781345	0.5846503	-0.2932053	0.845283	-0.4970367	0.07266116	-0.19116	0.1362457	0.9948268	-0.8985794
0.6934396	0.4444977	0.7907108	0.2731951	-0.004777312	0.4301721	-0.03275907	-0.1148297	-0.4176236	-0.3548368

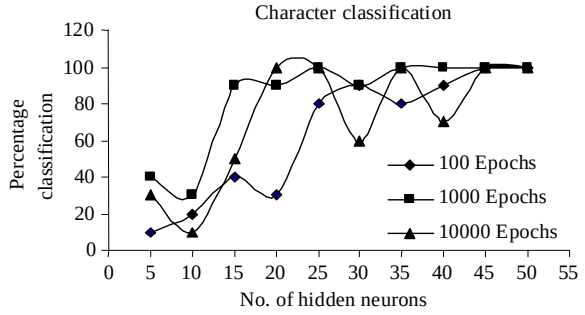

Fig.3: Effect of cross pruning on character classification

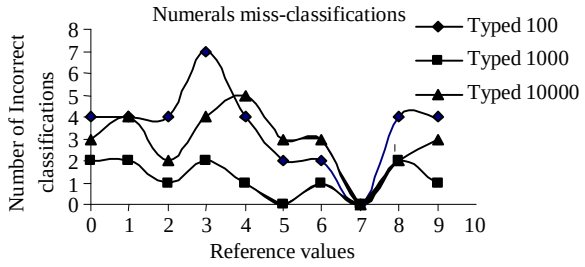

Fig. 4: Effect of cross pruning on classification errors

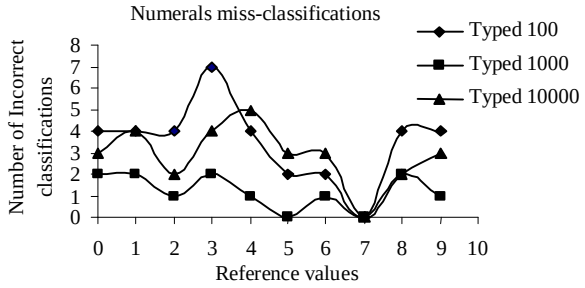

Fig. 5: Effect of cross pruning on miss-classification

Figure 4 demonstrates application of Eq. 13 in conjunction with our cross pruning approach to obtain a much more reliable and reflective representation for the errors that have occurred as a result of miss classification. It is clear from the diagram that the greatest percentage of miss classification occurs when using a 100 epochs with larger oscillation period compared to a 1000 epochs the number of tested hidden neurons with the difference of amplitude reduction in the sum squared errors at earlier stage due to the large number of cycles but still unstable due to over teaching and over fitting compared to the 100 epochs case which suffers instability due to under learning and under fitting. It is clear from the curve that the optimum number of epochs that produce a stable classification system with minimum number of errors as a function of hidden layer neurons is a 1000 epochs.

Figure 5 deals with an important aspect of designing our MLP OCR system and in general contributes to designing a much more accurate classification system through the plotting and analyzing of what we can call Statistical Classification Reach (SCR) whereby the figure indicates through the use of our Cross Pruning (CP) technique the effect of varying both number of epochs and number of hidden layer neurons on the absolute statistical difference of numeral miss classification which can be generalized to cover character and word miss classification. It is clear from Fig. 5 and by applying our SCR technique that our system will be most unstable when using 100 and 10000 epochs with low number of hidden layer neurons and it also shows that the numeral the system had most difficulty in learning and classifying is the numeral 3 as it shares common properties with numerals 2, 8. The figure also indicates that epoch numbers and hidden neuron numbers had no effect on the classification of the numeral 7 as its composition is simple and does not share many features with other numerals; hence it is difficult to be confused or misclassified.

CONCLUSION

It is known that the choice of hidden units depend on many factors such as the amount of noise in the training data, the type of the shaping or activation function, the training algorithm, the number of input and output units and number of training patterns.

Overall, deciding how many hidden layer neurons to use is a complex task which this research tried to clarify and quantify. In doing so and to have a meaningful characterization for the system performance in terms of classification and to include and analyze the dependency of all the mentioned factors the concept of cross pruning is introduced together with the Statistical Classification Reach (SCR) and weight matrix auto update. These two techniques helped in deciding the optimal number of hidden layer neurons and number of epochs and pinpointed to the numerals that the system found difficult to classify. Our approach is certainly novel in terms of reducing time and effort and simplifying approaches to system design and modification which will assist in eliminating any weak points found in numeral based system in terms of numeral, MLP based recognition system.

REFERENCES

1. Thivierge, J.P., F. Rivestand and T.R. Shultz, 2003. A dual-phase technique for pruning constructive networks. Proceedings of the IEEE International Joint Conference on Neural Networks, July 20-24, IEEE Xplore Press, USA., pp: 559-564. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1223407.
2. Parekh, R., J. Yang and V. Honavar, 2000. Constructive neural-network learning algorithms for pattern classification. IEEE Trans. Neural Networks, 11: 436-451. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=839013&isnumber=18100.
3. Engelbrecht, A., 2001. A new pruning heuristic based on variance analysis of sensitivity information. IEEE Trans. Neural Networks, 12: 1386-1399. DOI: 10.1109/72.963775.
4. Parekh, R., J. Yang and V. Honavar, 2000. Constructive neural network learning algorithms for multi-category pattern classification. IEEE Trans. Neural Networks, 11: 436-451. Digital Object Identifier 10.1109/72.839013.
5. Yang, J., R. Parekh and V. Honavar, 2000. Comparison of performance of variants of single-layer perceptron algorithms on non-separable data. Neural, Parallel Sci. Comput., 8: 415-438. http://citeseerx.ist.psu.edu/viewdoc/summary;jsessionid=4B392260D248E984926072CCCC1A647B?doi=10.1.1.70.9449.
6. Chen, C.H. and V. Honavar, 1999. A neural network architecture for syntax analysis. IEEE Trans. Neural Networks, 10: 94-114. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.5727
7. Yang, J., R. Parekh and V. Honavar, 1998. DistAl: An inter-pattern distance based constructive learning algorithm. Technical Report TR97-05. http://archives.cs.iastate.edu/documents/disk0/00/00/01/50/index.html.
8. Yao, X., 1993. A review of evolutionary artificial neural networks. Int. J. Intell. Syst., 4: 539-567. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.9598
9. Cantú-Paz, E., 2003. Pruning neural networks with distribution estimation algorithms. Lecture Notes Comput. Sci., 2723: 790-800. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.5167.
10. Amin, H., K.M. Curtis and B.R. Hayes Gill, 1997. Dynamically pruning output weights in an expanding multilayer perceptron neural network. Proceedings of the 13th International Conference on Digital Signal Processing, July 2-4, Santorini, Greece, pp: 991-994. DOI: 10.1109/ICDSP.1997.628530.
11. Chen, H.H., M.T. Manry and H. Chandrasekaran, 1999. A neural network training algorithm utilizing multiple sets of linear equations. Neurocomputing, 25: 55-72. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.4223.
12. Ponnapalli, 1999. A formal selection and pruning algorithm for feedforward artificial network optimization. IEEE Trans. Neural Networks, 10: 964-968. DOI: 10.1109/72.774273.
13. Polikar, R., L. Udpa, S. Udpa and V. Honavar, 2001. Learn++: An incremental learning algorithm for supervised neural networks. IEEE Trans. Syst., Manand Cybernetics-Part C: Appl. Rev., 31: 497-508. DOI: 10.1109/5326.983933.
14. Costa, M., A. Braga and B. Menezes, 2002. Constructive and pruning methods for neural network design. Proceeding of the 7th Brazilian Symposium on Neural Networks, Nov. 11-14, IEEE Computer Society, USA., pp: 49-54. http://doi.ieeecomputersociety.org/10.1109/SBRN.2002.1181434.
15. Grippo, L., 2000. Convergent on-line algorithms for supervised learning in neural networks. IEEE Trans. Neural Network, 11: 1284-1299. DOI: 10.1109/72.883426.
16. Burgess, N., 1994. A constructive algorithm that converges for realvalued input patterns. Int. J. Neural Syst., 5: 59-66. DOI: 10.1109/72.839013.
17. Reed, R., 1993. Pruning algorithms: A survey. IEEE Trans. Neural Network, 4: 740-747. DOI: 10.1109/72.248452.
18. Hancock, P.J.B., 1992. Pruning neural nets by genetic algorithm. Proceedings of the International Conference on Artificial Neural Networks, 1992, Elsevier Science, Amsterdam, Netherlands, pp: 991-994. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.3661.
19. Lim, T.J., W.Y. Loh and Y.S. Shih, 2000. A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. Mach. Learn., 40: 203-228. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.5864.
20. Setiono, R. and A. Gaweda, 2000. Neural network pruning for function approximation. Proceeding of the International Joint Conference on Neural Networks, July 24-27, IEEE Computer Society, Los Alamitos CA., pp: 443-448. http://cat.inist.fr/?aModele=afficheN&cpsidt=14172954.
21. de Kruif, B.J. and T.J.A. de Vries, 2003. Pruning error minimization in least squares support vector machines. IEEE Trans. Neural Networks, 14: 696-702. DOI: 10.1109/TNN.2003.810597.