A COMPARISON OF EXTREME LEARNING MACHINES AND BACK-PROPAGATION TRAINED FEED-FORWARD NETWORKS PROCESSING THE MNIST DATABASE

Philip de Chazal[1,2], Jonathan Tapson[2], and André van Schaik[2]

[1]University of Sydney, School of Electrical and Information Engineering, Australia
[2]University of Western Sydney, MARCS Institute, Australia

ABSTRACT

This paper compares the classification performance and training times of feed-forward neural networks with one hidden layer trained with the two network weight optimisation methods. The first weight optimisation method used the extreme learning machine (ELM) algorithm. The second weight optimisation method used the back-propagation (BP) algorithm. Using identical network topologies the two weight optimization methods were directly compared using the MNIST handwritten digit recognition database. Our results show that, while the ELM weight optimization method was much faster to train for a given network topology, a much larger number of hidden units were required to provide a comparable performance level to the BP algorithm. When the extra computation due to larger number of hidden units was taken in to account for the ELM network, the computation times of the two methods to achieve a similar performance level was not so different.

Index Terms— Extreme Learning Machine, Back-propagation, Multilayer feed-forward network, MNIST database.

1. INTRODUCTION

Feed-forward neural networks using random weights were first suggested by Schmidt et al. in 1992 [1] but were not a widely used method until Huang et al. popularised them as Extreme Learning Machines (ELM) [2,3] in 2005. The ELM is a multi-layer feed-forward neural network topology and algorithm that offers fast training and flexible non-linearity for function regression and classification tasks. Its principal benefit is that the network parameters are calculated in a single pass during the training process, which offers a significant improvement in implementation time over conventional back-propagation-trained feed-forward networks. In its standard form it has an input layer that is fully connected to a hidden layer with conventional non-linear activation functions. The hidden layer is fully connected to an output layer with linear activation functions. The number of hidden units is often much greater than the input layer with a fan-out of 5 to 20 hidden units per input element frequently used. A key feature of ELMs is that the weights connecting the input layer to the hidden layer are set to random values, usually uniformly distributed in some predefined range. This simplifies the requirements for training to one of determining the hidden to output unit weights, which can be achieved in a single pass. By randomly projecting the inputs to a much higher dimensionality, it is possible for the algorithm to find a hyperplane which approximates a desired regression function, or represents a linear separable classification problem [4].

A common way of calculating the hidden to output weights is to use the Moore-Penrose pseudo-inverse [5] applied to the hidden layer outputs using labelled training data, or some regularized version of the pseudo-inverse [6].

In this paper we compare the classification performance and training times of ELM classifiers and back-propagation trained networks with identical network topology processing the MNIST database.

2. MNIST DATABASE

The MNIST handwritten digit recognition database was used throughout this study [7][8]. The database has 60,000 training and 10,000 testing examples. Each example is a 28*28 pixel 256 level grayscale image of a handwritten digit between 0 and 9. The 10 classes are approximately equally distributed in the training and testing sets.

We performed no preprocessing of the data and use 784 pixels values for each image as our network inputs.

3. DATA PROCESSING

If we consider a particular sample of input data $\mathbf{x}_k \in \mathbb{R}^{L\times 1}$ where k is a series index and K is the length of the series, then the forward propagation of the local signals through a fully connected network with one hidden layer can be described by:

$$y_{n,k} = \sum_{m=1}^{M} w_{nm}^{(2)} g\left(\sum_{l=1}^{L} w_{ml}^{(1)} x_{l,k} \right) \quad (1)$$

Where $\mathbf{y}_k \in \mathbb{R}^{N\times 1}$ is the output vector corresponding to the input vector $\mathbf{x}_k$, l and L are the input layer index and number of input features respectively, m and M are the hidden layer index and number of hidden units respectively, and n and N are the output layer index and number of output units respectively. $w^{(1)}$ and $w^{(2)}$ are the weights associated with the input to hidden layer and the hidden to output layer linear sums respectively. $g(\ )$ is the hidden layer non-linear activation function. The output layer activation is a linear function.

3.1 Extreme Learning Machines

With ELM, $w^{(1)}$ are assigned randomly which simplifies the training requirements to task of optimisation of the $w^{(2)}$ only. The choice of linear output neurons further simplifies the optimisation problem of $w^{(2)}$ to a single pass algorithm. The weight optimisation problem for $w^{(2)}$ can be stated as

$$y_{n,k} = \sum_{m=1}^{M} w_{nm}^{(2)} a_{m,k} \text{ where } a_{m,k} = g\left(\sum_{l=1}^{L} w_{ml}^{(1)} x_{l,k} \right) \quad (2)$$

We can restate this as matrix equation by using $\mathbf{W}^{(2)} \in \mathbb{R}^{N\times M}$ with elements $w_{nm}^{(2)}$, and $\mathbf{A} \in \mathbb{R}^{M\times K}$ in which each column contains outputs of the hidden unit at one instant in the series $\mathbf{a}_k \in \mathbb{R}^{M\times 1}$, and the output $\mathbf{Y} \in \mathbb{R}^{N\times K}$ where each column contains output of the network at one instance in the series as follows:

$$\mathbf{Y} = \mathbf{W}^{(2)}\mathbf{A}. \quad (3)$$

The optimisation problem involves determining the matrix $\mathbf{W}^{(2)}$ given a series of desired outputs for $\mathbf{Y}$ and a series of hidden layer outputs $\mathbf{A}$.

We represent the desired outputs for $\mathbf{y}_k$ using the target vectors $\mathbf{t}_k \in \mathbb{R}^{N\times 1}$ where $t_{n,k}$ has value 1 in the row corresponding to the desired class and 0 for the other N-1 elements. For example $\mathbf{t}_k = [0,1,0,0]^T$ indicates the desired target is class 2 (of four classes). As above we can restate the desired targets using a matrix $\mathbf{T} \in \mathbb{R}^{N\times K}$ where each column contains the desired targets of the network at one instance in the series. Substituting $\mathbf{T}$ in for the desired outputs for $\mathbf{Y}$, the optimisation problem involves solving the following linear equation for $\mathbf{W}$:

$$\mathbf{T} = \mathbf{W}^{(2)}\mathbf{A}. \quad (4)$$

In ELM literature $\mathbf{W}$ is often determined by the taking the Moore-Penrose pseudo-inverse $\mathbf{A}^+ \in \mathbb{R}^{K\times M}$ of $\mathbf{A}$ [3]. If the rows of are $\mathbf{A}$ are linearly independent (which normally true if K>M) then $\mathbf{W}^{(2)}$ maybe calculated using

$$\mathbf{W}^{(2)} = \mathbf{T}\mathbf{A}^+ \text{ where } \mathbf{A}^+ = \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}$$

$$\text{i.e. } \mathbf{W}^{(2)} = \mathbf{T}\mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1} \quad (5)$$

This minimises the sum of square error between networks outputs $\mathbf{Y}$ and the desired outputs $\mathbf{T}$, i.e. $\mathbf{A}^+$ minimizes

$$\left\|\mathbf{Y}-\mathbf{T}\right\|_2 = \sum_{k=1}^{K}\sum_{n=1}^{N}\left(y_{n,k}-t_{n,k}\right)^2 \quad (6)$$

We refer to networks trained with above algorithm as ELM networks.

3.2 Back-propagation trained Feed-forward Network

The second method to determine the network parameters was to minimise the network error by iteratively back-propagating the gradient of the error with respect the network weights and using a numerical optimisation algorithm to reduce the network error. As the output stage of our network was linear we only required to iteratively calculate the input to hidden layer weights. The hidden to output weights could be calculated at each iteration using equation 5 of the ELM classifier.

For a fully connected network, a sum of square error function and *tanh* activation functions for the hidden units, the gradient of the error E with respect the input to hidden network weights $\frac{\partial E}{\partial w_{ml}^{(1)}}$ is given by [7]:

$$\frac{\partial E}{\partial w_{ml}^{(1)}} = \frac{1}{K}\sum_{k=1}^{K}\left(1-a_{\mathrm{m},k}^2\right)\left(\sum_{n=1}^{N}\left(y_{n,k}-t_{n,k}\right)w_{nm}^{(2)}\right)x_{l,k} \quad (7)$$

Using matrices as defined in section 3.1 and the matrix $\mathbf{X} \in \mathbb{R}^{L\times K}$ in which each column contains inputs $\mathbf{x}_k \in \mathbb{R}^{L\times 1}$ at one instant in the series, equation 7 can be restated in matrix form as

$$\frac{\partial E}{\partial \mathbf{W}^{(1)}} = \frac{1}{K}\left(\left(\mathbf{1}-\mathbf{A}\times\mathbf{A}\right)\times\left(\mathbf{W}^{(2)T}\left(\mathbf{Y}-\mathbf{T}\right)\right)\right)\mathbf{X}^T \quad (8)$$

where $\times$ is an element-by-element multiplication operator.

Having calculated $\frac{\partial E}{\partial \mathbf{W}^{(1)}}$, a gradient descent algorithm is used to update the input to hidden layer weights using

$$\mathbf{W}^{(1)} = \mathbf{W}^{(1)} - \lambda\frac{\partial E}{\partial \mathbf{W}^{(1)}} \quad (9)$$

where λ is the learning rate parameter. We refer to networks trained with above algorithm as BP networks

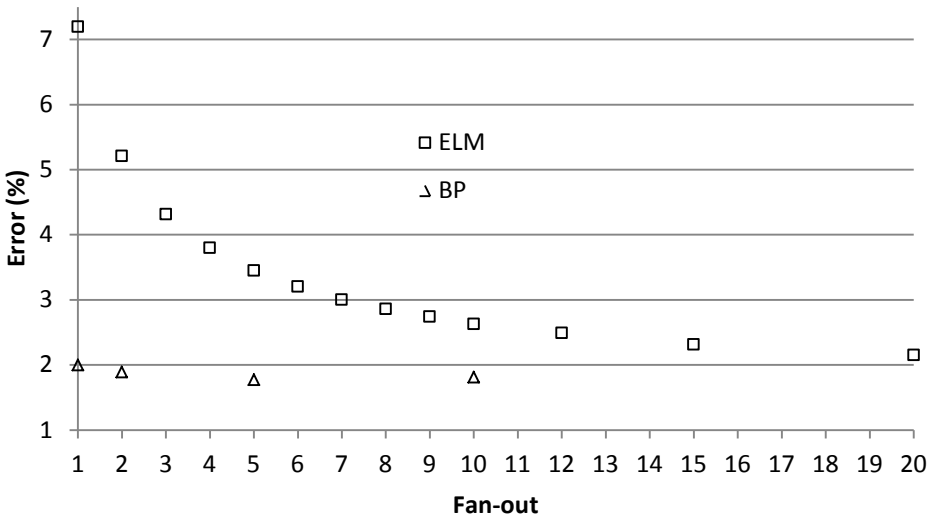

Fig. 1. The error rate of the ELM and the BP on the MNIST database for fan-out varying between 1 and 20. All results at each fan-out are averaged from 10 repeats of the experiment.

Table 1. The error rate (%) of the ELM and and the BP networks on the MNIST database. Results averaged from 10 repeats.

	Fan-out												
	1	2	3	4	5	6	7	8	9	10	12	15	20
ELM	7.20	5.21	4.32	3.80	3.45	3.20	3.00	2.86	2.74	2.63	2.49	2.31	2.15
BP	2.00	1.89			1.77					1.81			

Table 2. Computation times (in seconds). The elapsed time is shown for training the ELM and BP networks on the 60,000 images from MNIST database using MATLAB R2012a code running on 2012 Sony Vaio Z series laptop with an Intel i7-2640M 2.8GHz processor and 8GB RAM.

	Fan-out													
	1	2	3	4	5	6	7	8	9	10	12	20	15	20
ELM	6.2	13.9	24.9	37.8	53.3	68.5	88.2	111	136	162	228	630	339	630
BP	900	2040			7200					72000				

4. EXPERIMENTS

We applied the ELM and BP weight calculation methods to the MNIST data. Both the ELM and BP algorithms were applied directly to the unprocessed images and we trained the networks by providing all data in batch mode. The random values for the input layer weights were uniformly distributed between -0.5 and 0.5 after scaling the pixel data so that the minimum value was 0 and maximum value was 1. The *tanh* function was used as the hidden layer activation function. In order to perform a direct comparison of the two methods we used the following protocol:

For fan-out of 1 to 20 hidden units per input, repeat 10 times

(i) Assign random values to the input layer weights.

(ii) Determine ELM network weights using data from (i).

(iii) Train the BP network weights using data from (i).

(iv) Evaluate both networks on the 10,000 test data examples and store results.

For the BP network we used a learning rate of $\lambda = 0.01$ and stopped training after 100 iterations. Due to the long training times of the BP networks we evaluated networks for a fan-out of 1, 2, 5 and 10 only. For the ELM networks we evaluated at fan-outs of 1 to 10, 12, 15 and 20.

We averaged the results for the 10 repeats of the experiment for each fan-out and compared the misclassification rates. Both training algorithms were implemented in Matlab using matrix format.

5. RESULTS AND DISCUSSION

The classification performance results for experiments are shown in Fig. 1 and Table 1. The computation time is shown in Table 2. The error rate of the ELM network decreased as the fan-out value was increased and the trend of the curve in Fig. 1 suggest that further reduction in the error rate may be achieved with a fan-out number greater than 20. The highest error rate was 7.2% at a fan-out of 1 and the minimum error rate was 2.15% at a fan-out of 20. The error rate of BP networks did not vary much with fan-out value. It ranged between 2.0% at a fan-out of 1 to a minimum of 1.77%.at a fan-out of 5 with a value.

The error rate of the BP networks outperformed the ELM networks at every fan-out and, notably, the error rate of the smallest BP network (fan-out=1) outperformed the best of the ELM networks (fan-out=20). One of the claimed benefits of the ELM networks is the fast training time. Table 2 shows this to be the case with the training times of the ELM networks approximately 0.65% of the BP networks for a given fan-out number. This result was unsurprising given that we were training the network for 100 iterations. The training time for a fan-out of 1 BP network was 900 seconds which was 50% greater than the training time of the fan-out of 20 ELM network (600 seconds). Given that the BP network at fan-out 1 (error 2.0%) outperformed the ELM network at fan-out 20 (error 2.15%) our results suggest that a much larger number of hidden units are required by the ELM network to achieve a similar performance to the BP network on this database. Given a desired performance level, the training times of the two networks may not be so different.

Our results are comparable with the best of the published results for the MNIST database processing raw pixel values [4] [11]. We have used a simple gradient descent algorithm with a fixed learning rate. More sophisticated optimization algorithms such as adaptive learning rates and the inclusion of a momentum term would likely improve network training speeds [9]. These methods would decrease the difference the computational penalty of the BP algorithm relative to the ELM algorithm.

6. CONCLUSION

This study has shown that the feed-forward neural networks networks with one hidden layer trained with the extreme learning machine and the back-propagation algorithms can successfully classify the MNIST database. For a desired performance level the back-propagation network resulted in a smaller number of hidden units and the computational training requirements of the two optimisation methods was not so different.

7. RELATION TO PRIOR WORK

The work presented here has focused on side-by-side comparison of two training methods for multilayer feed-forward networks. The results for the ELM classifiers processing the MNIST dataset was presented in [8]. All other work is this publication is original.

8. REFERENCES

1. W.F. Schmidt, M.A. Kraaijveld, and R.P. Duin, "Feed forward neural networks with random weights", *Proceedings of 11th IAPR international conference on pattern recognition methodology and systems*, pp 1–4, 1992.
2. G.B. Huang, Q.Y. Zhu and C.K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.
3. G. Huang, G.B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review", *Neural Networks*, vol. 61, pp. 32-48, Jan. 2015.
4. J. Tapson and A. van Schaik, "Learning the pseudoinverse solution to network weights," *Neural Networks*, vol. 45, pp. 94–100, Sep. 2013.
5. R. Penrose and J. A. Todd, "On best approximate solutions of linear matrix equations," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 52, pp 17-19, 1956.
6. B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge Univ. Press, 1996.
7. Y. LeCun and C. Cortes, "The MNIST database of handwritten digits", Available at: http://yann.lecun.com/exdb/mnist.
8. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86(11), pp. 2278-2324, Nov. 1998.
9. S. Haykin, "*Neural Networks. A comprehensive foundation*." 2nd Edition, Prentice Hall, 1998.
10. P. de Chazal, J. Tapson and A. van Schaik, "Learning ELM network weights using linear discriminant analysis", *Proceedings of ELM-2014 Volume 1,* pp 183-191, 2015.
11. http://yann.lecun.com/exdb/mnist/