Simplifying Hand Written Digit Recognition Using A Genetic Algorithm

A. D. Parkins & A. K. Nandi
Signal Processing and Communications Group,
Dept. Electrical Engineering & Electronics., University of Liverpool,
Liverpool, L69 3GJ, UK
Tel: +44 151 7944525; Fax: +44 151 7944540
e-mail: a.d.parkins@liv.ac.uk, A.Nandi@liv.ac.uk

ABSTRACT

For recognition in image data, the large number of features can cause an unnecessary increase in the complexity of the chosen classifier. It is important to select from the available features only those that contribute new information. In many cases, an excess of features not only does not aid classification but in fact actively reduces performance. In this paper we use reduced complexity of both classifier and feature set to improve accuracy and speed of computation for the identification of handwritten digits. We show that performances comparable with existing classifiers can be achieved with a 200 fold smaller network

1 INTRODUCTION

In many classification problems, the size of the feature set can be an important consideration – larger numbers of features increase the classifier complexity and non-contributory "noise" features can degrade performance. The problem is one of separating the features which aid classification from those that hinder. One technique for dealing with this situation is the use of a genetic algorithm as a feature selector. This paper is concerned with the use of a genetic algorithm to find an optimum feature set for the classification of hand written digits.

2 FEATURE SELECTION

The term feature selection is applied to the task of selecting those features that are most useful to a particular classification problem from all those available. Consider a feature set, $\mathbf{F} = \{f_0, f_1, \ldots, f_N\}$. If f_0 and f_1 are dependent, that is they always move together, then one of these could be discarded and the classifier has no less information to work with. This has the benefit that computational complexity is reduced as there is a smaller number of inputs. Often, a secondary benefit found is that the accuracy of the classifier increases. This implies that the removed features were not adding any useful information but they were also actively hindering the recognition process.

A. D. Parkins is supported by the EPSRC

3 NEURAL NETWORKS

Artificial neural networks (ANNs) have been shown ([1],[2]) to be good classifiers in hand written character recognition problems.

The ANN used for this work was a feed-forward network using a hyperbolic activation function with momentum and per-neuron adaptive learning enabled (see section 3.1). Simulations performed with no GA component showed that, using all 256 features, classification performance was not increased significantly when more than fifteen hidden layer neurons were used. Therefore, when the genetic algorithm component was added to keep network complexity at a minimum and provide a constant state for comparison, the number of hidden neurons was fixed at fifteen.

3.1 Adaptive Learning

Adaptive learning is used to adjust the learning rate of the network, η, during the training phase. The method used for the network uses a combination of techniques. Adjustments are made to learning rate based on the size of the change in the MSSE of the output. That is, if the MSSE of the neuron output decreases by 50%, then the learning rate for that neuron is increased by 50%. There is also a limiter applied, so that if the MSSE increases by more than 100%, the learning rate does not jump by such a big amount to cause the next backpropagation phase to move a huge distance on the error surface; this tends only to apply in early epochs when the improvements are huge. This has the added benefit that there are no parameters to be set. Much of this combines heuristics proposed by others – adjustment of η based on gradient from Silva and Almeida's method; more cautious increases of η than decreases from Jacobs' delta-bar-delta; and a limiting factor from Riedmiller and Braun's RPROP. See [3] for descriptions and comparisons of these learning heuristics.

4 GENETIC ALGORITHMS

Genetic algorithms (GAs) have been successfully used to select feature subsets in classification problems ([4], [5]). Being a directed search rather than an exhaustive search, population members cluster near good solutions; however, the GA's stochastic component does not rule out wildly different solutions, which may turn out to be better. This has the benefit that, given enough time and a well bounded problem, the algorithm can find a global optimum. This makes them well suited to feature selection problems – they can find near optimum solutions using little or no *a priori* knowledge. An introduction to genetic algorithms can be found in [6].

The genetic algorithm used in this case is from the GAlib[7] C++ library. A steady state GA is used, this is similar to the algorithms described in [8]. In each generation a new temporary population is created and evaluated. The worst performers from the combined population are removed to reduce the population to its original size and the process is repeated.

4.1 Genome Encoding

A genetic algorithm evaluates a population of test genomes and then selects a number of them to continue to the next generation. As such, before using a genetic algorithm a coding scheme must be devised, where all problem solutions are mapped on to a genome. In this case a fixed subset of features are to be selected from a global pool. Each available feature is assigned a unique number and the genome is simply a list of the feature numbers contained in the subset being evaluated, with no feature allowed to appear more than once.

4.2 Crossover and Mutation

Mutation is the random corruption of a new genome. In this case a feature mutator is used – a new random feature from the total available is selected then a random feature in the genome is selected for mutation and the old feature is replaced with the new.

Crossover is the combining of two parent genomes to make two new child genomes. These child genomes are related but different from the parent genomes. A union crossover is performed in this case. The two sets of parent features are added together and any duplicates are removed, the first child genome is created by taking the first N features from the union, the second child is created by taking the last N features from the union (N is the size of a genome, which is fixed for each simulation).

5 DATA SET

The data set used is a subset of the United States Postal Services (USPS) digit set [9], a representative sample of which is shown in figure 1. Each entry is a 16×16 image with each pixel represented by a gray level in the range $[-1, 1]$. This set is divided at source into a training and test set. In our case we have reduced the training set such that all digits appear with equal frequency so that there is no bias toward any one particular digit. This training set is then further divided into a training and validation set (see table 1).

Set	digits × classes = total
Training	$360 \times 10 = 3600$
Validation	$180 \times 10 = 1800$
Testing	2700 assorted

Table 1: Data set divisions and breakdown

The validation set is normally used to stop training a classifier before over-fitting occurs by measuring divergence in performance on the training set and performance on the validation set. In the work presented here the validation set is not used, as early stopping does not apply (section 6).

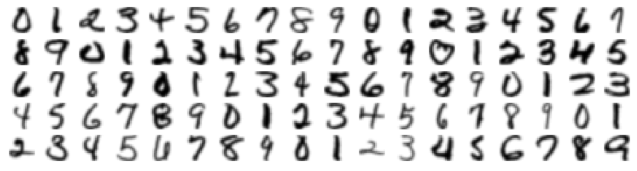

Figure 1: Sample of images from the USPS data set

6 SIMULATION

The simulation environment was set to produce fixed length genomes. For each simulation the first population of twenty genomes was generated by selecting random subsets of a fixed length from the total features available and evaluating them. This evaluation consisted of creating the appropriate feature subset for each example as specified by the genome under evaluation then training an MLP network for 50 epochs using the training section of the set. The final performance measure for that genome is then given by classification percentage on an unseen test set. In each generation a temporary population, equal in size to the real population is created (see section 4.2) by combining members from the real population using roulette wheel selection. Each member of this temporary set is then evaluated. This gives a "score" for each genome in the current generation and for each member of the temporary set. The worst ninety percent of the real set is then replaced with the best members from the temporary set. This process was repeated for 40 generations and for different genome sizes (or numbers of features).

In other simulations with this data set it has been found that performance improves as the number of training epochs is increased. However, in separate simulations using different test criteria where one criteria is better than another early on, it remains better no matter how many training epochs are performed. Therefore, a low number of epochs was used for the feature selection phase to lower computation time.

7 RESULTS

7.1 Performance of ANN alone

As an example of the performance of the neural network component alone, a simulation was performed over 4,400 epochs using all 256 available features; the first five hundred epochs of which are shown in figure 2. The plots after this point continue in the same way, performance has saturated at approximately 88%. Surprisingly, the validation and training sets do not diverge; this may be because the training set is very representative and as such does not permit over-fitting. Alternatively, the low number of hidden neurons may have reduced the number of free parameters enough to keep the network general.

Table 2, and figure 3 show final performance (after 4,400 epochs) for a number of different hidden layer sizes. These results highlight the difficulty of classifying this data set. The test set scores are consistently lower than the validation set – this is surprising because the network is never trained on either. If both sets were of equal difficulty then we would expect similar performance from both. The results imply that the test set contains more difficult (or significantly different) examples than either the training or validation set.

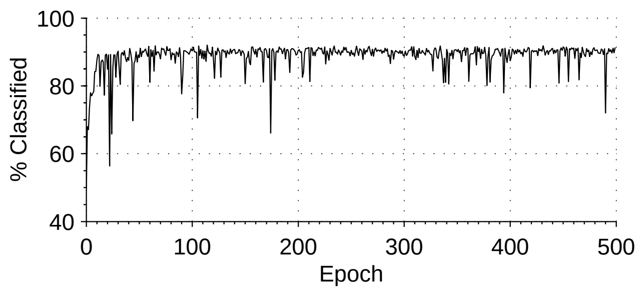

Figure 2: Performance vs. epoch with all features (256) used

No. of Hidden Neurons	Training	Validation	Test
5	48.47	47.50	44.87
10	89.61	90.17	85.67
15	94.19	91.06	86.40
20	97.08	93.33	87.47
25	95.17	93.00	88.00
30	97.44	92.88	87.93
35	97.97	92.39	88.60
40	97.92	92.78	88.47

Table 2: Performance with number of hidden layer neurons for a train-until-zero gradient network

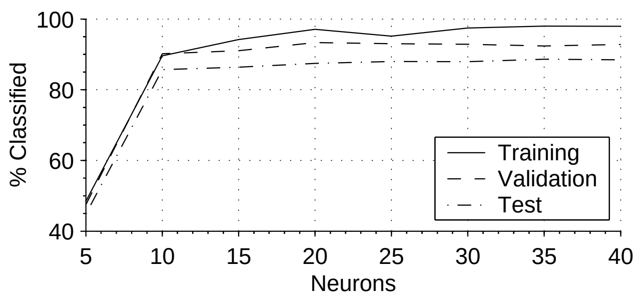

Figure 3: Performance with number of hidden layer neurons

7.2 Performance of feature selected set

Table 3 and figure 4 show performance with different numbers of features. For every number of features selected the GA has selected a "good" selection for the subset. As the number of features increase, the performance quickly saturates at approximately 90%. This occurs near the forty feature point; after which, a doubling in number of features gives less than 1% increase in performance. Table 2 shows that for 15 neurons with all 256 features, after 4,400 epochs the performance is 86.4%; performance has decreased with a large number of features.

Features	% Classified	Features	% Classified
1	25.1	55	89.7
5	69.6	60	89.4
10	79.5	65	89.7
15	83.9	70	89.2
20	85.3	75	89.4
25	85.9	80	89.1
30	87.9	85	89.9
35	88.0	90	90.0
40	88.8	95	90.2
45	88.3	100	89.3
50	88.5		

Table 3: Performance, after 50 training epochs, of a 15 hidden neuron neural network using best N features selected using 40 generations of GA

8 CONCLUSION

Table 4 shows a summary of performances that others have achieved using the same data set as has been used in this work. Although at this stage the work in this paper does not achieve performance of these levels, it has the benefit of using a much reduced complexity classifier. For example [10] uses the same network architecture as was introduced in [11]; this uses approximately 100,000 connections operations to evaluate an image. The network in this work uses 3,990 connections when

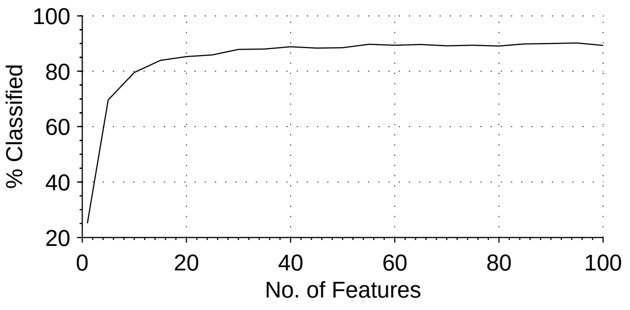

Figure 4: Performance vs. no.of features selected

used with all 256 features and 450 when used with 20 features; the difference is due to the radically reduced hidden layer size and the pre-selection of subset of significant features. In both of these cases performance is approximately 90%, this is a 6% performance degradation, relative to [10]'s performance, with network that is 220 fold smaller in number of connections.

Method	% Classified
Human [12]	97.5
Convolutive Neural Network [10]	95.8
Kernel density based [13] Bayesian classifier	97.8
Support Vector Machine [14]	97.0

Table 4: Performances on the USPS data set

Direct comparisons with the alternative classifier architectures used in [13] and [14] are more difficult. For the support vector case, $\mathbf{x}_i \cdot \mathbf{x}_j$ is calculated for each support vector, where $\mathbf{x}_i$ is an input vector and $\mathbf{x}_j$ is a support vector. Each of these dot products represents a series of multiply and add operations similar to those that take place in a hidden layer neuron. To achieve the performance shown in table 4, multiple binary classifiers were used each with hundreds of support vectors – making the support vector classifier many times more complex than the ANN classifier.

9 References

[1] Y. Le Cun, "Comparison of learning algorithms for handwritten digit recognition," in *International Conference on Artificial Neural Networks, Paris* (F. Fogelman and P. Gallinari, eds.), pp. 53–60, 1995.

[2] P. Suganthan, "Structure adaptive multilayer overlapped soms with supervision for handprinted digit classification," 1998.

[3] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms," *Int. Journal of Computer Standards and Interfaces*, vol. 16, pp. 265–278, 1994.

[4] L. B. Jack and A. K. Nandi, "Genetic algorithms for input selection in condition monitoring," in *Proceedings of COMADEM 99*, (Oxford, UK), pp. 381–388, Coxmoor Publishing, 1999.

[5] L. B. Jack and A. K. Nandi, "Genetic algorithms for feature selection in machine condition monitoring with vibration signals," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 147, no. 3, pp. 205–212, 2000.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison Wesley, 1989.

[7] M. Wall, "Galib: A C++ genetic algorithms library." available from http://lancet.mit.edu/ga .

[8] K. DeJong, *An Analysis of the Behaviour of of a Class of Genetic Adaptive Systems.* PhD thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

[9] "Cedar cdrom 1: Usps office of advanced technology database of handwritten cities," 1992.

[10] Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 2, (Denver 1989), pp. 396–404, Morgan Kaufmann, San Mateo, 1990.

[11] Y. Le Cun, B. Boser, J. Denker, D. Hendersen, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, p. 541, 1989.

[12] P. Simard, Y. Le Cun, and J. Denker, "Efficient pattern recognition using a new transformation distance," in *Advances in Neural Information Processing Systems* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), vol. 5, pp. 50–58, Morgan Kaufmann, San Mateo, CA, 1993.

[13] D. Keysers, J. Dahmen, T. Theiner, and H. Ney, "Experiments with an extended tangent distance," in *Proceedings 15th International Conference on Pattern Recognition*, (Barcelona, Spain), 2000.

[14] B. Schölkopf, P. Simard, A. Smola, and V. Vapnik, "Prior knowledge in support vector kernels," in *Advances in Neural Inf. Proc. Systems* (M. I. Jordan, M. J. Kearns, and S. A. Solla, eds.), vol. 10, pp. 640–646, MIT Press, 1998.