A Modular Neural Network Architecture with Additional Generalization Abilities for Large Input Vectors

Albrecht Schmidt*
The Intelligent Systems Group
Department of Computing
The Manchester Metropolitan University

Zuhair Bandar†
The Intelligent Systems Group
Department of Computing
The Manchester Metropolitan University

Abstract

This paper proposes a two layer modular neural system. The basic building blocks of the architecture are multilayer Perceptrons trained with the Backpropagation algorithm.

Due to the proposed modular architecture the number of weight connections is less than in a fully connected multilayer Perceptron.

The modular network is designed to combine two different approaches of generalization known from connectionist and logical neural networks; this enhances the generalization abilities of the network.

The architecture introduced here is especially useful in solving problems with a large number of input attributes.

1 Introduction

The multilayer Perceptron (MLP) trained by the Backpropagation (BP) algorithm has been used to solve real-world problems in prediction, recognition, and optimization.

If the input dimension is small the network can be trained very quickly. However for large input spaces the performance of the BP algorithm decreases [3]. In many cases it becomes difficult to find a parameter set which leads to convergence towards an acceptable minimum. This makes it often very difficult to find a useful solution, especially in recognition where large input spaces are common.

A lot of research is being done to overcome these problems; many of the ideas include modularity as a basic concept.

In [4] a locally connected adaptive modular neural network is described. This model employs a combination of BP training and a Winner-Take-All layer.

A modular neural system using a self organizing map and a multilayer Perceptron is presented in [2]. It is applied in a cosmic ray space experiment.

In this paper a modular neural network is proposed to enhance the generalization ability of neural networks for high dimensional inputs. The network consists of several MLPs. Each of the modules is trained by the BP algorithm. The number of weight-connections in the proposed architecture is significantly smaller than in a comparable monolithic network.

The modular architecture is introduced, a training algorithm is given, the operation of the network is described, and experiments are presented.

2 The Network Architecture

The proposed network system consists of a layer of input modules and an additional decision module. All sub-networks are MLPs. Each input variable is connected to only one of the input modules. These connections are chosen at random. The outputs of all input modules are connected to the decision network. The structure is depicted in Figure 1.

The following parameters are assumed: the dimension of the input vector is l and the number of classes is k.

One of the design issues is to select the number of inputs per module in the first layer (n); this decision determines the number of input modules $m = \lceil \frac{l}{n} \rceil$. (It is assumed that $l = m*n$; if this is not the case the spare inputs may be connected to constant inputs or the size of one of the networks may be altered.) Each network in the first layer has $\lceil \log_2 k \rceil$ outputs. This is the required number to represent all the classes in a binary code.

The decision network has $m * \lceil \log_2 k \rceil$ inputs. The number of outputs is k, one neuron for each class.

The number of weights is much less than in a fully connected monolithic MLP with the same number of hidden neurons.

3 Training the System

The training occurs in two stages. All the modules are trained using the Backpropagation algorithm [6].

In the first phase all sub-networks in the input layer are trained. The training set for each sub-network is

* Email: aschmidt@hydra.informatik.uni-ulm.de
† Email: Z.Bandar@doc.mmu.ac.uk

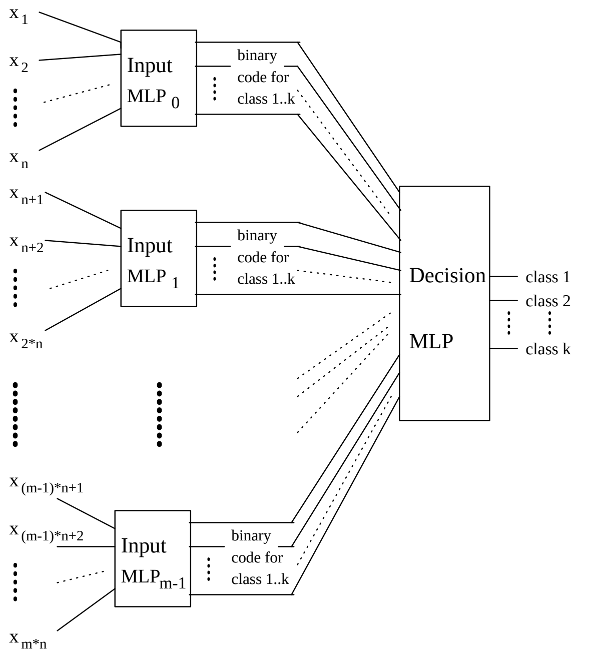

Fig. 1: The Multiple Neural Network Architecture.

selected from the original training set. The training pair for a single module consists of the components of the original vector which are connected to this particular network (as input vector) together with the desired output class represented in binary coding.

All input modules can be trained in parallel very easily because they are all mutually independent.

In the second stage the decision network is trained. The training set for the decision module is built from the output of the input layer together with the original class number. To calculate the set each original input pattern is applied to the input layer; the resulting vector together with the desired output class (represented in a 1-out-of-k coding) form the training pair for the decision module.

The original training set is: $(x_1^j, x_2^j, \ldots, x_l^j; d^j)$ for all $j = 1, \ldots, t$. Where $x_i^j \in R$ is the ith component of the jth input vector, d^j is the class number, and t is the number of training instances.

The module MLP_i is connected to::
$x_{i \cdot n+1}, x_{i \cdot n+2}, \ldots, x_{(i+1) \cdot n}$

The training set for the network MLP_i:
$(x_{i \cdot n+1}^j, x_{i \cdot n+2}^j, \ldots, x_{(i+1) \cdot n}^j; d_{BIN}^j)$
for all $j = 1, \ldots, t$

The mapping performed by the input layer:
$\Phi : R^{n*m} \mapsto R^{m*\lceil \log_2 k \rceil}$.

The training set for the decision network:
$(\Phi(x_1^j, x_2^j, \ldots, x_l^j); d_{BIT}^j)$ and $j = 1, \ldots, t$.

The mapping of the decision network:
$\Psi : R^{m*\lceil \log_2 k \rceil} \mapsto R^k$.

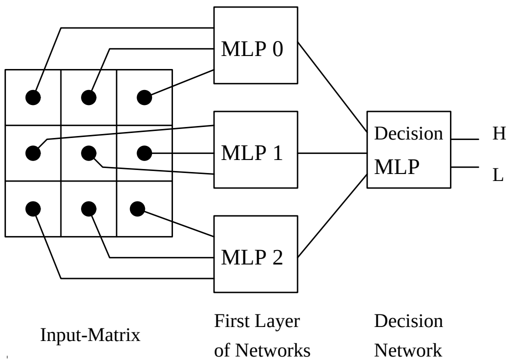

Fig. 2: The Example Architecture.

4 Calculation of the Output

The mapping of the whole network is:
$\Phi \circ \Psi : R^l \mapsto R^k$.

The response r for a given test input $(a_1, a_2, \ldots, a_l)$ is determined by the following function:
$r = \Psi(\Phi(a_1, a_2, \ldots, a_l))$.

The k-dimensional output of the decision module is used to determine the class number for the given input. In the experiments the output neuron with the highest response was chosen as the calculated class. The differences between the winning neuron and the runner-up may be taken as a measure of accuracy.

5 On Generalization

The ability to generalize is the main property of neural networks. This is how neural networks can handle inputs which have not been learned but which are *similar* to inputs seen during the training phase. Generalization can be seen as a way of reasoning from a number of examples to the general case. This kind of reasoning is not valid in a logical context but can be observed in human behaviour.

The proposed architecture combines two methods of generalization.

One way of generalizing is built-in to the MLP. Each of the networks has the ability to generalize on its input space. This type of generalization is common to connectionist systems.

The other method of generalization is due to the architecture of the proposed network. It is a way of generalizing according to the similarity of input patterns. This method of generalization is found in logical neural networks [1, p172ff].

To explain the behaviour more concretely the following simplified example of a recognition system is given.

A 3x3 input retina with the architecture shown in Figure 2 is assumed. Each of the nine inputs reads a continuous value between zero and one, according to the recorded gray level (black=1; white=0).

Fig. 3: The Training Set.

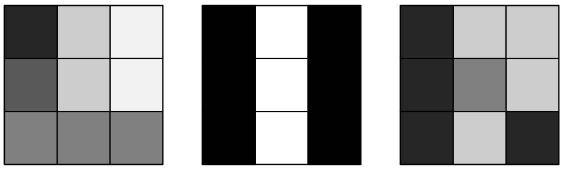

Fig. 4: The Test Set.

The network should be trained to recognize the simplified letters 'H' and 'L'. The training set is shown in Figure 3. The desired output of the input networks is '0' for the letter 'H' and '1' for the letter 'L'.

The training subsets for the networks MLP_0, MLP_1, and MLP_2 are:

MLP_0	MLP_1	MLP_2
(1,0,1;0)	(1,1,1;0)	(1,0,1;0)
(1,0,0;1)	(1,0,0;1)	(1,1,1;1)

After completing the training of the first layer of networks it is assumed that the calculated output is equivalent to the desired output.The resulting training set for the decision network is:

$$(\Phi(1,0,1,1,1,1,1,0,1);1,0) = (0,0,0;1,0)$$
$$(\Phi(1,0,0,1,0,0,1,1,1);0,1) = (1,1,1;0,1)$$

After the training of the decision network the assumed response of the system to the training set is:

$$r_H = \Psi(\Phi(1,0,1,1,1,1,1,0,1)) = \Psi(0,0,0) = (1,0)$$
$$r_L = \Psi(\Phi(1,0,0,1,0,0,1,1,1)) = \Psi(1,1,1) = (0,1)$$

To show the different effects of generalization three distorted characters, shown in Figure 4 are used as the test set:

The first character tests generalization within the input modules, the second shows the generalization on the number of correct sub-patterns, and the third character is an example of a combination of both. (The figures in the input vectors are according to the gray-level in the pattern; the outputs are taken from a typical neural network).

$$r_1 = \Psi(\Phi(0.9,0.2,0.1,0.7,0.2,0.1,0.5,0.5,0.5)) = \Psi(0.95,0.86,0.70) = (0.04,0.96) \;\Rightarrow\; 'L'$$

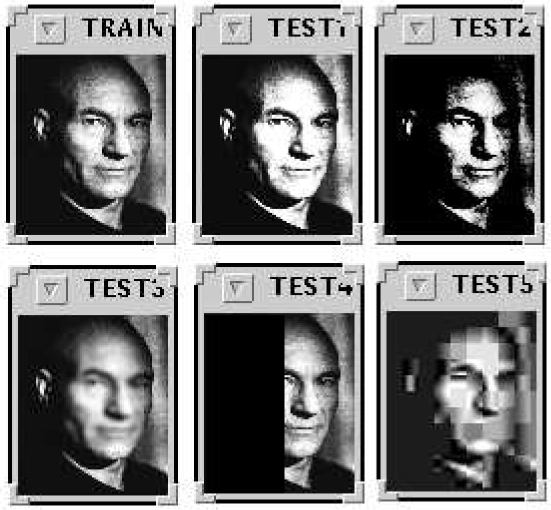

Fig. 5: Original and Distorted Pictures.

$$r_2 = \Psi(\Phi(1.0,0.0,1.0,1.0,0.0,1.0,1.0,0.0,1.0)) = \Psi(0,0.49,0) = (0.91,0.09) \;\Rightarrow\; 'H'$$

$$r_3 = \Psi(\Phi(0.9,0.2,0.2,0.9,0.5,0.2,0.9,0.2,0.9)) = \Psi(0.92,0.65,0.09) = (0.15,0.89) \;\Rightarrow\; 'L'$$

6 Experiments

The proposed architecture was tested with different real-world data sets. The number of input attributes was between eight and 12000.

Throughout the experiment it appeared that the modular network converged for a large range of network parameters. Particularly for huge input spaces it was often very difficult to find an appropriate learning coefficient for a monolithic network, whereas convergence was no problem for the modular structure.

The time needed to train the modular network was much shorter than that for a monolithic network. In most cases it took less than half the time to train the network to a similar performance. For larger input spaces the training was up to ten times quicker (without parallel training).

For small input spaces (up to 60 attributes) the memorization and generalization performance of the modular network and a monolithic MLP were very similar on the real-world data sets.

One task was to memorize five pictures of different faces. Each gray-level pictures had a size of 75 by 90 pixels (6750 continuous input variables). The original pictures are from [5]. After training the generalization performance was tested with distorted pictures. In Figure 5 one training picture (upper left) and some degenerations of this picture are shown.

The modular network had a much higher recognition rate on the manually distorted pictures.

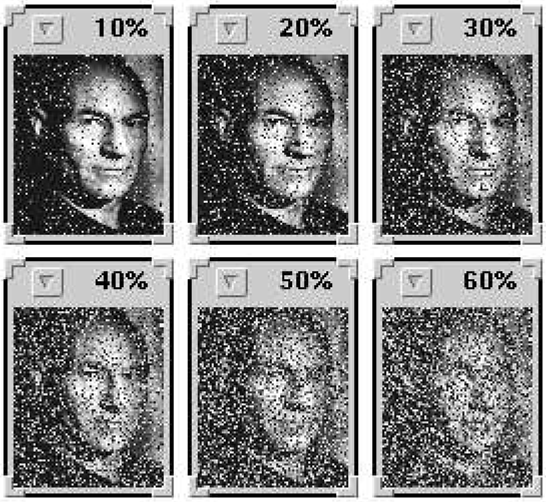

Fig. 6: Examples of Noisy Test Pictures.

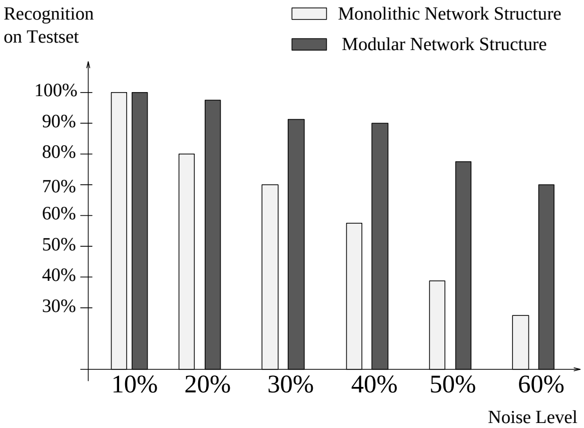

Fig. 7: The Performance on Noisy Inputs.

Another comparison was made on the ability to recognize noisy inputs. The noise on the pictures was generated randomly. In Figure 6 pictures with different noise-levels are shown. The modular network could recognize pictures with a significant higher noise-level than the single MLP; the results are shown in Figure 7.

From the above experiments it can be seen that the modular network has superior generalization abilities on high dimensional input vectors.

7 Limitations

The network is less useful for problems with very small input dimensions. The network has the ability to solve problems which are not linear separable. The proposed architecture has certain theoretical limitations; statistically neutral problems (like the XOR-problem) can not be learned. Monolithic MLPs are able to learn such problems but the generalization performance is very poor [7].

8 Conclusion

The usage of a modular architecture consisting of small MLPs to solve real world problems is demonstrated in this paper.

It is shown that for different real world data sets the training is much easier and faster with a modular architecture.

Two different approaches of generalization are combined in this model. It is demonstrated that this results in a generalization advantage on high dimensional input vectors.

Due to the independence of the modules in the input layer parallel training is readily feasible.

References

[1] Igor Aleksander and Hellen Morton. An Introduction to Neural Computing. Second Edition. Chapman & Hall 1995.

[2] R. Bellotti, M. Castellano, C. De Marzo, G. Satalino. Signal/Background classification in a cosmic ray space experiment by a modular neural system. In: Proc. of the SPIE - The International Society for Optical Engineering. Vol: 2492, Iss: pt.2, Page 1153-61. 1995.

[3] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: benchmarking studies. In: Proc. IEEE International Conference on Neural Networks. Page 61-67. San Diego. 1988.

[4] L. Mui, A. Agarwal, A. Gupta, P. Shen-Pei Wang. An Adaptive Modular Neural Network with Application to Unconstrained Character Recognition. In: International Journal of Pattern Recognition and Artificial Intelligence. Vol: 8, Iss: 5, Page 1189-204. October 1994.

[5] Picture-Directory. University Stuttgart. FTP From: `ftp.uni-stuttgart.de /pub/graphics/pictures`

[6] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In: D.E. Rumelhart, and J.L. McClelland (Eds.), Parallel distributed processing. Volume I: Foundations. Cambridge, MA: MIT Press 1986.

[7] J.V. Stone and C.J. Thorton. Can Artificial Neural Networks Discover Useful Regularities?. In: Artificial Neural Networks. Page 201-205. Conference Publication No. 409 IEE. 26-28 June 1995.