# BiDirectional MLP Neural Networks

A.F. Nejad and T.D. Gedeon

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052 AUSTRALIA

## ABSTRACT

A new method of training neural networks is suggested that enables them to remember input patterns as well as output vectors, given either of them. This work could be a step ahead toward simulating the behaviour of networks closer to that of human associative memory. We shall indicate some applications of this approach in extracting meaning from neural networks and finding the centres of clusters.

In this paper we have applied a Bidirectional Neural Network(BDNN) to predict and analyse students final marks by using partial grades of students in a large subject (these partial marks give 40% of the overall grade). At the conclusion of this paper we have also included a discussion on the advantages and disadvantages of BDNNs and some of the important roles they may play in future work.

## I. INTRODUCTION

Since Ramón y Cajal described the contact between neurons there was a debate over the mechanisms of synaptic function until the 1950s, when physiologists found that synaptic transmission can be either electrical or chemical and where it is electrical the transmission is usually bidirectional (Kandel, Siegelbaum and Schwartz, 1991). We shall demonstrate the simulation of this kind of bidirectional transmission using neural networks and address some applications in real world problems.

Neural networks can be applied to many real world systems to perform classification, pattern recognition or prediction on the basis of input data. However, given the output data, standard neural network models are not able to produce any plausible input data, which is done easily by humans, unless another network is trained. For example, people can retrieve an image of an elephant from the word “elephant”, and when we see an elephant our mind will find the corresponding word. Networks which can produce plausible input values for a given output value could be used in many applications.

The extraction of rules from trained neural networks is seen as a way to improve the accessibility of neural networks (Yoda, Baba and Enbutsu, 1991, Hora, Enbutsu, and Baba, 1991, Towell and Shavlik, 1991, Bochereau and Bourgine, 1990, Gallant, 1988). There is an analogy with the traditional bottleneck of knowledge acquisition in expert systems.

Most methods of rule extraction use causal connections between inputs and outputs. There is some statistical indication that outputs have causal effects on inputs. Thus, if we were to explicitly allow connections to be made between the output and input values, more accurate rules could be extracted.

In the following sections we shall describe the training of such neural networks as a content addressable memory.

## II. Training BDNNs as content addressable memory

To use BDNNs as content addressable memories, it has been assumed that a complete set of training data with a one-to-one relation between input and output vectors is available, otherwise we should use some preparation techniques as described in following sections to create a one-to-one relationship between them, unless the problem is inherently not appropriate for such transformations of the vector spaces. We have used MLP networks with either one or two hidden layers for this task.

### II.i. Method of Training

We have applied the error back-propagation technique (Rumelhart, Hinton and Williams, 1986) in both reverse and forward directions to adjust the weight matrix of the network. In our experiments we did not need to use more hidden units or more hidden layer weights in training a bidirectional network in comparison to the case of training a network in the traditional way.

The network will be trained bidirectionally with the same weights in each direction. For each input and output node a threshold is assigned, but assigning extra threshold units to hidden layer units is not necessary. That is, the same threshold weights are used in both directions. Due to a flatter search space, usually a higher number of epochs will be necessary for the network to converge in comparison to those of traditional single direction back-propagation networks.

When quantities are related to each other in a specific numerical way, we use the concept of a function to unify them. By the definition of a function, we mean that each element in domain $A$ is mapped into precisely one element in range $B$. Therefore we can define $f'$ (reverse of function $f$) if and only if $f$ is a one-to-one function.

The remaining challenges to be solved are the cases where the function relating inputs to outputs is not reversible, or there is a relation which is many-to-one between inputs and outputs.

It should be noted that it is not suggested to use BDNNs for classes of problems which are inherently irreversible. For example, all the people who are authorised by a loan authorisation network are in one class. Since the result is a ‘yes’ or a ‘no’, we can not expect to map ‘yes’ to only a single input pattern.

In many cases, neural networks <u>are</u> used to map many input patterns to one output pattern (eg. XOR, Loan). We have used two techniques to avoid this problem. The first method is to use some statistical techniques to create a one-to-one relationship between input and output vectors if it is meaningful in the domain, allowing our two-directional training to proceed. Using statistical techniques in data preparation to increase the reliability of neural networks has been suggested by many researchers, thus our approach is appropriate. The second technique adds an extra output node for cases where the decision made by the neural network is essentially symbolic. There is some analogy with the use of mnemonics to add context to remember phone numbers, for example.

## II.ii. Experimental results

We implemented BDNNs for a few cases including XOR, character (digit) recognition, and student final mark prediction.

In the case of character recognition, our training patterns had a one to one mapping between input and output patterns, thus no modifications were required.

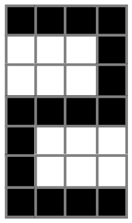
Figure 1.

Figure 1 shows an example of a training pattern. The network was a 28 input, 7 hidden units, and 10 output units. The network was trained as a BDNN, and performed at 100% accuracy in both directions on the training set. Note that for a content addressable memory, the use of a test set is neither possible, nor desirable, as we do not want generalisation, but retrieval.

In the case of the XOR and student final mark prediction, the function is not reversible.

For the XOR problem where a zero output is related to four input patterns we used one extra node in the output in order to make the function reversible.

| Inputs | | | | Output | Extra Node |
|---|---|---|---|---|---|
| 0 | 0 | 0 | ==> | 0 | 0.1 |
| 0 | 1 | 1 | ==> | 0 | 0.3 |
| 1 | 0 | 1 | ==> | 0 | 0.5 |
| 1 | 1 | 0 | ==> | 0 | 0.7 |
| 0 | 0 | 1 | ==> | 1 | 0.2 |
| 0 | 1 | 0 | ==> | 1 | 0.4 |
| 1 | 0 | 0 | ==> | 1 | 0.6 |
| 1 | 1 | 1 | ==> | 1 | 0.8 |

Table 1. XOR problem and using Output Extra Node

In this way each output pattern will be related to only one input pattern. This is analogous to the assignment of some otherwise meaningless mnemonics to a telephone number to improve our ability to recall it. The trained network classified correctly all the training patterns in the XOR problem.

For student final mark prediction case, we did not use an extra node, but we did some statistical manipulations on the input patterns (see figure 2) as data preparation.

| regno | crs | S | ES | Tut | L2 | TS | L4 | H1 | H2 | L7 | P1 | F1 | Mid | L10 | Fin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0275105 | 3400 | 1 | F | T9-ko | 3 | 4 | 2.5 | 17 | 17 | 3 | 5 | 14 | 10 | 2.5 | 56 |
| 0273169 | 3972 | 1 | FS | T1-yh | 2 | 3 | 3 | 20 | 19 | 3 | 16 | 14 | 33 | 2.5 | 80 |

| Student | Inputs | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| p01275312 | 1 | 0.5 | 1 | 0.8 | 0.7 | 0.7 | 0.7 | 1 | 0.8 | 0.4 | 1 | 0.3 | 0.49 | 0.5 |

Figure 2. Two example of students records and an example of a scaled record

For each class of final marks (Fail, Pass, Credit, Distinction) we found the mean of input values. We also calculated for each class of Pass and Credit six sub-classes and for each class of Fail and Distinction three sub-classes and their related input patterns. Assuming the monotonic behaviour of input nodes, we used some simple equations like ($80\% * X_i + 20\% * X_j$) to find the related value of inputs for each sub-class. For example, the six sub-classes of Credit are:

| | | |
|---|---|---|
| “90%CR-10%PS”, | | “90%CR-10%DN”, |
| “80%CR-20%DN”, | | “80%CR-20%PS”, |
| “70%CR-30%PS”, | and | “70%CR-30%DN”. |

This was possible because of our pre-assumption of the monotonic behaviour of inputs. We could also use some more sophisticated methods like Z scores to identify the expected value of inputs, or sensitivity analysis to exploit the behaviour of inputs. Alternatively we could use other clustering methods to identify the expected value of inputs, such as Kohonen networks, the best way is using BDNN itself to pre-classify the input patterns. This work is discussed elsewhere, and consists of training BDNNs to find cluster centres. The related input patterns for each of the sub-classes are those that have outputs in a sub-class.

Done properly this kind of splitting of input patterns into sub-classes will not degrade the learning of the network, but will increase the reliability of the resulting network as well as allow faster convergence. In many cases, such as the character recognition problem, there is no need to do this kind of statistical manipulations to prepare input patterns. As an example, suppose the case that we had the exact final mark of students which allowed us to define some intervals on the range of final marks, then we could easily associate the appropriate expected value to each of the input patterns of a class.

In training the student mark prediction BDNN, we used the same static momentum and learning rate in both directions. We intend to investigate whether the use of dynamic coefficients will result in faster convergence of the network. In BDNNs we have used the same thresholds in the hidden layer units in both directions. It is probable that using different thresholds could improve learning time, but would be a cost of partially decoupling the forward and reverse directions. We have yet to investigate whether this partial decoupling reduces BDNNs usefulness.

For the first fifty epochs the BDNN is trained normally in the forward direction. Then, the direction of training of the network is reversed. This direction of training is maintained until the overall normalised error of the output nodes (in the current direction) is less than or equal to the normalised error of the previous direction of training, or some maximum number of epochs have been spent in the current direction, or the overall error in the current direction is less than the error tolerance we have predefined for that direction. This sequence of reversals of direction of training continues until the error is below the error tolerance set for both directions.

The trained network could correctly classify 74% of the training patterns in the case of the student mark prediction problem. A number improvements are possible, such as removing the outliers, softening the sharp edges between the subclasses, and doing some preparation techniques on the training data such as using binary nodes (dummy variables) instead of continuous numbers for coding some fields like tutor-group and session number fields, or using the expected value of the lost fields instead of initialising them simply to zero. Finally, the assumption of monotonic increasing of values of some fields does not hold. Thus, the 74% performance is quite good. Note that we have retained the current encodings to maintain comparability with previous work using the same data set (Gedeon and Bowden, 1992, Slade and Gedeon, 1993). A future phase of this work will be to compare the extraction of rules from a BDNN with our previous rule extraction using this data set (Gedeon and Turner, 1993, Turner and Gedeon, 1993).

## III. DISCUSSION

We believe that the trained bidirectional neural networks will help us in solving the traditional bottle-neck of the acceptance and use of neural networks. This is the need to understand the contents of the black box, hence extracting rules from the trained network

Our method can also be used to enhance other techniques. For example using sensitivity analysis with a BDNN it will now be possible to measure the sensitivity of input values to output values as well as that of output values to input values.

Our method of bidirectional training of neural networks by learning both the forward and reverse tasks at one time will yield more powerful learning. The bidirectional learning aggregate gradient tends to be flatter, however we believe this may result in networks less susceptible to noise and providing better generalisation.

It is important to note that applications of bidirectional neural networks in control systems where the effect of a change in outputs should result in an appropriate change in input values should be considered as another important advantages of BDNNs.

## REFERENCES

Kandel, ER, Siegelbaum, SA and Schwartz, JH "Synaptic Transmission", *Principles of Neural Sciences*, 1991.

Edelman, GM, Gall, WE and Cowan, WM (eds.), *Synaptic Function*, New York: Wiley, 1987.

Yoda, M, Baba, K and Enbutsu, I "Explicit representation of knowledge aquired from plant historical data using neural networks," *International Joint Conference on Neural Networks*, San Diego, vol. 3, pp. 155-160, 1991.

Hora, N, Enbutsu, I and Baba, K "Fuzzy rule extraction from a multilayer neural net," *Proc. IEEE*, vol. 2, pp. 461-465, 1991.

Towell, GG and Shavlik, JW "The extraction of refined rules from knowledge-based neural networks," *Machine Learning*, August 1991.

Bochereau, L and Bourgine, P "Expert systems made with neural networks," *International Joint Conference on Neural Networks*, vol. 2, pp. 579-582, January 1990.

Gallant, SI "Connectionist expert systems," *Communications of the ACM*, vol. 31, no. 2, pp. 152-169, February 1988.

Gedeon, TD and Bowden, TG "Heuristic pattern reduction," *International Joint Conference on Neural Networks*, Beijing, pp. 449-453, November 1992.

Gedeon, TD and Turner, H "Explaining student grades predicted by a neural network," *Proceedings International Joint Conference on Neural Networks*, pp. 609-612, Nagoya, 1993.

Rumelhart, DE, Hinton, GE, Williams, RJ, "Learning internal representations by error propagation," in Rumelhart, DE, McClelland, *Parallel Distributed Processing*, Vol. 1, MIT Press, 1986.

Ruiz, R, Hernández, C and Mira, J "Method for mapping cardiac arrythmia in real time using micro-processor-based systems," *Medical & Biological Engineering & Computing*, vil. 22, pp. 160-167, 1984.

Slade, P and Gedeon, TD "Bimodal Distribution Removal," *Proceedings IWANN International Conference on Neural Networks*, Barcelona, 1993. also in Mira, J, Cabestany, J and Prieto, A, *New Trends in Neural Computation*, pp. 249-254, Springer Verlag, Lecture Notes in Computer Science, vol. 686, 1993.

Turner, H and Gedeon, TD "Extracting Meaning from Neural Networks," *Proceedings 13th International Conference on AI*, vol. 1, pp. 243-252, Avignon, 1993.