BiDirectional Neural Networks and Class Prototypes

A.F. Nejad and T.D. Gedeon
Department of Artificial Intelligence
School of Computer Science and Engineering
The University of New South Wales
Sydney 2052 AUSTRALIA
Email: {akbar,tom}@cse.unsw.edu.au

ABSTRACT

Among the numerous existing neural network models, there are only a few bidirectional neural networks. None of these bidirectional models are based on multi layer perceptrons. These models also have limitations which have prevented their popularity. We have designed Bidirectional Neural Networks (BDNNs) based on multi layer perceptrons trained by a generalised form of the error back-propagation algorithm. They can be trained as either associative memories or cluster centroid finders. This gives the network new abilities and enables us to design powerful data representation techniques which are a key factor in reducing network generalisation error. We demonstrate applications of this approach to extracting meaning from neural networks, and finding the centers of clusters. This work could be a step towards simulating via the behaviour of artificial neural networks clustering methods closer to that of the biological brain. BDNNs may also be used as a simulation tool for evaluating some major cognitive psychology theories such as prototypes and dual-code theory. The successful and promising results of applying BDNNs on two real world problems (including missing and noisy attributes) as well as some artificial data sets is reported.

1 . Introduction

Since Ramon y Cajal described the contact between neurones there had been a debate over the mechanisms of synaptic function until the 1950s. Then, physiologists found that synaptic transmission can be either electrical or chemical, and where it is electrical the transmission is usually bidirectional [8,2]. We shall demonstrate the simulation of this kind of bidirectional transmission using artificial neural networks and examine some applications to real world problems.

Most neural networks can be applied to real world systems to perform classification, pattern recognition or prediction tasks on the basis of input data. Given the output data, however, these neural network models are not able to produce any plausible input data unless another network is trained specifically for that task. This is done easily by humans. For example, we can retrieve an image for an elephant from the word *elephant*, and when we see an elephant we will find the corresponding word. Networks which can produce plausible input values for a given output value have many applications. We are particularly interested in their use for constructing more accurate rules when extracting rules from trained neural networks.

A new model of training neural networks is suggested that enables them to remember input patterns as well as output vectors, given either of them. They may be trained as associative memories and cluster centroid finders and are capable of classification or prediction, in real world problems.

Bidirectional associative memories [9,10] and the bidirectional version of counterpropagation networks [6] have been developed to learn bidirectional mappings. The problem with these approaches is their low capacity, low efficiency and multiple learning. By multiple learning we mean the different learning methods which are used in the first and second layer of bidirectional version of counterpropagation networks. BDNNs avoid the problems of these approaches. Moreover, BDNNs have the power to determine expected cluster centroids, at a desired level of optimisation, for each class while minimising the effects of outliers.

Two real world data sets which include missing values and noisy attributes have been used in the experimental part of this study to evaluate BDNNs.

We have applied BDNNs to predict and analyse *Student Final Marks (SFM)* by having partial grades of students in a relatively large subject. This data set consists of 153 samples. Each pattern had fourteen input attributes, and four outputs have been used to classify the marks. Each record comprises student information, assessment and subsequent final mark for a sample of students from a first year computer science subject at The University of New South Wales.

The exam mark which contributes 60% of the weight to the final mark is omitted. This contributes a significant amount of noise to the data. The educational justification is the usefulness of a final mark predictor for students to use before the final examination is attempted. We have trained a BDNN

using 100 samples, while the other 53 unseen samples have been used to measure the generalisation ability of the BDNN.

We have also applied BDNNs to predict *Gross National Product (GNP)* for most of the developing countries in the world. The data set contains some socio-economic information about 143 developing countries from UNCTAD [18]. Each record has fourteen attributes such as *Total Population, Education Illiteracy Rate, and Health Population Per Doctor.* We have separated the data into a training set consisting of 100 records and a test set which contains the remaining 43 records which will not be seen during training by the neural network. The output contains four classes: *Very Poor Countries, Poor Countries, Middle Countries, and Rich Countries.*

The extraction of meaning from trained neural networks is seen as a way to improve the acceptability of neural networks [19,7,16,1,3]. Most methods of rule extraction use causal connections between inputs and outputs. There is some statistical indication that the outputs have causal effects on inputs. Thus, if we were to explicitly allow connections embodded in the network to be made between the output and input values as well as the usual input to output connections, it is possible that more accurate rules could be extracted.

In the following sections we shall describe the training of BDNNs for two different tasks:
a) as a context addressable memory; and
b) to find the center of a cluster.

Subsequently, we will demonstrate how an understanding of the behaviour of the network can be obtained by investigating the properties of such cluster centers. This will be done by examining the class prototypes, comparing their representative values and manually providing an interpretation of the network behaviour.

2. Training BDNNs as Content Addressable Memories

For the purposes of this investigation 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 would use some preparation techniques as described in the following sections to create a one-to-one relationship between them, with the premise that this is not done when the problem is inherently not appropriate for such transformations of the vector spaces. We have used MLP networks with either one or two layers of hidden layers for context addressable memories.

2.1. Method of Training

We have applied the error back-propagation technique [14] 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 in training a bidirectional network in comparison to the case of training a network in the traditional unidirectional way.

When trained from left to right in Figure 1, the weights on the connections between layers A and B, and the weights between layers B and C are used. The biases of layers B and C are also used. When training in the reverse direction, the same weights between B and C, and A and B, are used. The biases of neurons in layer C are not used, and those of B and A are used. Note that the same biases are used in layer B in both directions. This would be the case for multiple hidden layers also.

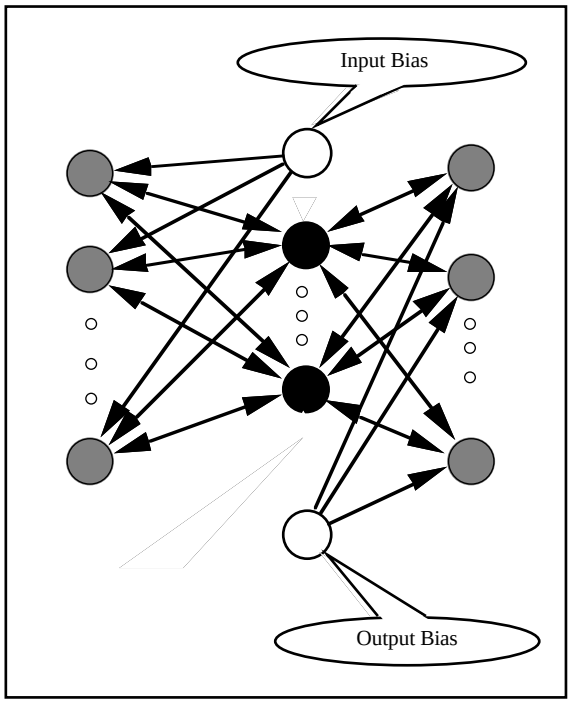

Figure 1. BDNN Topology (hidden biases not shown)

Due to a flatter search space, sometimes a higher number of epochs may be necessary for the network to converge in comparison to traditional unidirectional networks. The remaining challenge is the case where the function relating inputs to outputs is not invertible, or there is a relation which is many-to-one between inputs and outputs. In the next section we shall 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.

3. Implementation and experiments

To implement our technique, some issues should be addressed. The first of these is invertibility. When quantities are related to each other in a specific numerical way, we use the mathematical concept of a function to unify them. By the definition of a function, we mean that each element in *A* (domain) is

mapped into precisely one element in B (range). Therefore we can define f' (inverse of function f) if and only if f is a one-to-one (not many-to-one) function. In the implementation of BDNN as associative memories we must address how to solve this problem. In many cases neural networks are used to map many input patterns to one output pattern (eg. OR, XOR, Loan, and so on).

Two techniques have been used to solve this problem. In the first technique, we manipulated the training data to make the data suitable (one-to-one) for input to the BDNN. Using statistical techniques for data preparation to increase the generalisation and the reliability of neural networks have been suggested by many researchers.

It should also be noted that we do not suggest useing bidirectional neural networks for classes of problems which are inherently not invertible. For example, to put all the people who are authorised for by a loan authorisation network 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 the second technique, extra outputs have been used in the output layer.

4. Experimental results

We have implemented BDNNs for a few cases where the function is invertible (for example, our character recognition patterns), and where the function is not invertible (for example, XOR and Student Final Marks).

In the case of character recognition (Figure 2), where our training patterns had a one-to-one mapping between input and output patterns, we simply used the patterns unchanged.

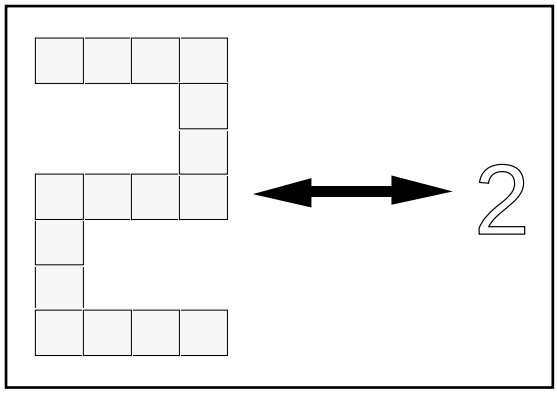

Figure 2. A character recognition pattern

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 invertible, as shown in Table 1.

In this way, each output pattern will be related to only one input pattern, forming a one-to-one mapping.

Table 1. XOR problem and using output extra node

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

This is analogous to the assignment of some otherwise meaningless mnemonics to a telephone number to improve our ability to recall it (the psychological term for this approach is called *narrative stories*). The trained network correctly classified all of the training patterns in the XOR problem.

Table 2. Three patterns of student marks.

Student No.	0275000	0275105	0275169
Course No.	3400	3400	3972
Stage	1	2	1
Enrol. Status	F	F	FS
Tutor Group	T10-yh	T9-ko	T1-yh
Lab. #2	2.5	2.5	2.5
Tut. Ass.	3.0	3.5	3.0
Lab. #4	3.0	2.5	3.0
Assign H1	18.0	17.0	19.5
Assign H2	4.5	17.5	19.0
Lab. #7	3.0	3.0	3.0
Assign. P1	14.0	5.0	16.0
Assign. F1	18.5	14.5	14.0
Mid-Exam	24.0	10.5	33.0
Lab. #10	2.5	2.5	2.5
Final Mark	Credit	Pass	Dist

For student final mark prediction we did not use an extra node, but did some statistical manipulations on the input patterns (Table 2) as data preparation. For each class of final marks (Fail, Pass, Credit, Distinction) we found the mean of the input values for each attribute. We also calculated for each of the Pass and Credit classes six sub-classes and for each of Fail

and Distinction three sub-classes, and their related input patterns.

Supposing monotonic behaviour of input values, we used some simple equations like *Z = 80%X + 20%Y* to find the related value of inputs for each sub-class. *X* is the mean vector or characteristic pattern of a class and *Y* is the mean vector of one of its neighbour classes. For example, from the mean vector of class credit *'C'*, the following vectors may be calculated as six new representative patterns.

These are:
Z1 = 90%C - 10%P
Z2 = 90%C - 10%D
Z3 = 80%C - 20%D
Z4 = 80%C - 20%P
Z5 = 70%C - 30%P
Z6 = 70%C - 30%D

D and *P* are the mean vectors for calsses distinction and pass respectively. This was possible because of our pre-assumption of the monotonic behaviour of inputs. We could also use some more sophisticated methods like the Z score to identify the expected value of inputs, or sensitivity analysis methods to exploit the behaviour of inputs. We have also found we can use BDNN to pre-classify the input patterns (see the section entitled "Training BDNNs as cluster center finders").

Done properly this kind of subclassification of input patterns will not cause a malfunction of the network, and may increase the generalisation and the reliability of the network leading to faster convergence.

In training a BDNN for predicting student final marks, 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. It is probable that using different biases could improve learning time, but would be at the cost of partially decoupling the forward and reverse directions. We have yet to investigate whether this partial decoupling reduces the overall usefulness of our BDNNs.

For the first 50 epochs, the BDNN is trained normally in the forward direction. Then, the direction of training of the network is reversed. A direction of training is maintained until the overall normalised error is less than the 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 this direction. This sequence of reversals of direction of training continues until the error is below the error tolerance set for either direction.

The trained BDNN correctly classified 74% of student mark training patterns. A number of 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 inputs instead of continuous numbers to code fields like session number, or by substitution of missing values with the most expected values.

Finally the assumption of monotonic increase of values of some fields does not hold. Thus, the 74% performance is quite good.

Note that we have used the current encoding to maintain comparability with previous work using the same data set [4,15]. 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 [5,17].

5. Training BDNNs as Cluster Center Finders

If we train BDNNs with no limitation on the relation between input and output patterns, it will find a representative input vector for each class of output patterns. The vector components will contain the most expected values for that field in the class.

To explore the properties of these vectors, we trained a BDNN to find the representative vector of the class of points in a circle. If these points have an even distribution, BDNN will represent the center of the circle as the cluster center (Figures 3 and 4).

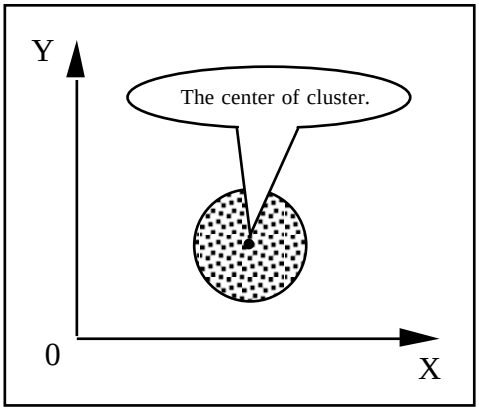

Figure 3. BDNN finds the center of the cluster. The center belongs to the cluster.

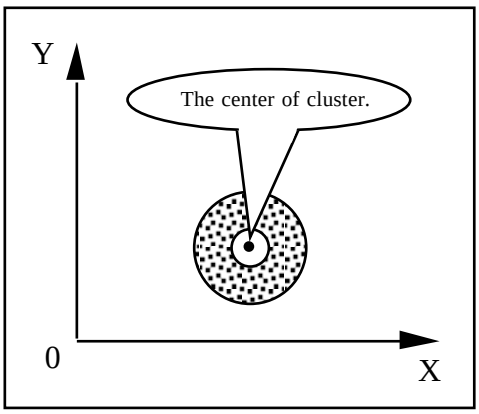

Figure 4. BDNN finds the center of the cluster. The center does not belong to the cluster.

Training a BDNN to find the cluster centers of four classes of SFM produced the four representative vectors or class prototypes shown in Table 3.

Table 3. Center of clusters for SFM prediction extracted by BDNN

Stu. Class	Dist	Cred	Pass	Fail
Course No.	0.37	0.13	0.76	0.56
Stage	0.56	0.51	0.64	0.74
Enrol. Status	0.52	0.65	0.99	1.00
Tutor Group	0.64	0.63	0.72	0.70
Lab. #2	0.82	0.82	0.51	0.36
Tut. Ass.	0.71	0.66	0.52	0.39
Lab. #4	0.91	0.70	0.46	0.25
Assign H1	0.78	0.81	0.73	0.35
Assign H2	0.89	0.75	0.71	0.18
Lab. #7	0.66	0.66	0.39	0.06
Assign. P1	0.98	0.70	0.61	0.22
Assign. F1	0.84	0.36	0.24	0.11
Mid-Exam	0.76	0.58	0.4	0.22
Lab. #10	0.51	0.69	0.47	0.16

A BDNN was also trained to find the class prototypes for the GDP of four classes of countries. Results are shown in Table 4. These tables explore the most expected partial grades of a student for each class and the most expected partial information about each class of GNP. The BDNNs could correctly classify 72% of SFM test data and 62% of GNP test data.

In Table 3, we can find that midterm exam marks are less than the normal standard indicating that it has been rather difficult for all students. Lab. #10 marks for the Distinction class are roughly the same as for the Pass class indicating that time management may have a very important effect in getting best results even in exams. Inconsistent values of the first and second components suggest they are not important variables or their preparation has not been done properly.

From Table 4, we can discover three of the most significant factors in assigning a country to a particular category. These are the variables *Health: Infant Mortality*, *Health: Population per Doctor*, *Education: Illiteracy Rate*, and *Phones per 100 Inhabitants*. We achieved the same results by using the method of sensitivity analysis for the trained neural network.

We found more interesting results in extracting the most significant variables by applying the tree-based

Table 4. Centers of clusters for GNP prediction extracted by BDNN.

Class of GDP	Low GDP	LowMid GDP	UpMid GDP	High GDP
Popul. Urban	0.17	0.18	0.34	0.39
Total Labour	0.51	0.33	0.37	0.27
Agric. Labour	0.50	0.32	0.27	0.09
Infant Mort.	0.59	0.42	0.26	0.12
Health Exp.	0.11	0.18	0.15	0.15
Doctor	0.82	0.60	0.27	0.12
Safe Water	0.08	0.13	0.16	0.18
Illit. Rate	0.63	0.44	0.31	0.26
Educ. Exp.	0.10	0.13	0.16	0.09
AverageFood	0.48	0.46	0.52	0.34
Phone %	0.02	0.04	0.11	0.34
Total Pop.	0.32	0.29	0.27	0.13
Avg Growth	0.09	0.09	0.09	0.10
Density	0.13	0.12	0.13	0.30

algorithm C4.5 [11] on GNP data set. The final tree correctly classified 82% of the training data and 62% of the test data. The tree had only three variables: *Health: Population per Doctor*, then the number of *Phones per 100 Inhabitants* , and if still we can not categorise, we will look at the *Education: Illiteracy Rate* of the country.

6. Discussion

Prototypes are useful when to assign values to components of input patterns for which the value is unknown. We can use the cluster center found by BDNN to assign the most likely value for an unknown value.

Applications of BDNNs in explaining properties of category classification in various science domains will help researchers to improve current techniques. For example, Cardiologists will be able to extract in real time the optimum templates used as standards of normality for segments of different ECG signal records which in turn could be different from a patient to the next.

A prototype is an example with all the essential attributes of good members of a category [12,13]. Neural networks which have been trained by the traditional error back-propagation algorithm can not provide prototypes for the user. We have demonstrated that BDNNs are able to provide such prototypes. These prototypes may then be used to enhance the learning and generalisation ability of neural networks.

According to Rosch, the ability to finding prototypes provide better human memories because there is a tendency among people to abstract such characteristic vectors with exemplars of a concept.

7. Conclusion

We have shown how BDNNs can be trained by a generalised error back-propagation algorithm to provide the capabilities of associative memories and cluster centroid finders. Experimental results of applying BDNNs to artificial data sets and two multivariate real world problems have revealed some of the abilities of BDNNs.

Our method can also be used to enhance other techniques, for example using sensitivity analysis with a BDNN. It is possible to measure the sensitivity of input values to output values as well as that of output values to input values. Our method of training bidirectional neural networks to learn both the forward and reverse tasks at the same time will yield more powerful neural networks. The bidirectional learning aggregate gradient tends to be flatter. We believe this may result in networks less susceptible to noise and providing better generalisation.

It also should be noted that another important advantage of using bidirectional neural networks will be their application in control systems, where the effect of a desireable change in outputs can provide the appropriate change in input values.

We believe that the trained bidirectional neural networks will help us in solving the bottle-neck of acceptance and use of neural networks. This is the need to understand the contents of the black box, hence extracting meaning from the trained networks. Our extracted class prototypes can be used to analyse data sets and abstract the training set for human comprehension. We have also used the class prototypes in designing effective data preparation techniques to reduce the generalisation error.

References

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

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

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

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

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

[6] Hecht-Nielsen, R "Counterpropagation Networks," *Applied Optics*, vol. 26, no. 3, pp. 4979-4984, 1987.

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

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

[9] Kosko, B "Bidirectional Associative Memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-18, pp. 49-60, 1988.

[10] Kosko, B. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prince-Hall, Inc., Englewood Cliffs, NJ, USA, 1992.

[11] Quinlan, JR *C4.5: Programs for Machine Learning*, SanMateo, California: Morgan Kaufmann, 1993.

[12] Rosch, E and Mervis, CB "Family Resemblances: Studies in the structure of Categories," *Cognitive Psychology* 7, pp. 573-605, 1975.

[13] Rosch, E "Principles of categorization," In *Eleanor Rosch and Barbara Lloyd, eds., Cognition and Categorization.* Hillsdale,NJ. pp. 27-48, 1978.

[14] Rumelhart, DE, Hinton, GE and Williams, RJ "Learning internal representations by error propagation," in Rumelhart, DE, McClelland, *"Parallel distributed processing,"* Vol. 1, MIT Press, 1986.

[15] Slade, P and Gedeon, TD "Bimodal Distribution Removal," *Proceedings IWANN International Conference on Neural Networks*, Barcelona, 1993.

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

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

[18] UNCTAD Statistical Pocket Book, United Nations Conference on Trade & Development, New York, 1984.

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