A Hybrid Bidirectional Network

Tamás D. Gedeon

School of Information Technology

Murdoch University, Perth Australia

tom@it.murdoch.edu.au

Abstract:
Bidirectionally trained neural networks would be very useful in many circumstances. Often, we have data available for a prediction problem, but prediction of properties for unknown or new situations is only part of the story. In many cases we know the effect we wish to achieve on the output, but what we do not know is how to modify the inputs to achieve this goal. A basic problem in this area is the inversion of many to one mappings. Our work is based on the popular backpropagation neural network to predict the GDP of developing countries. These networks are integrated with a Self-Organising Map to allow the inversion of many to one mappings.

Keywords: neural networks, backpropagation, self-organising map, bidirectional learning

1 Bidirectional Backprogration

1.1 Introduction

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. Bidirectional associative memories [1, 2] and the bidirectional version of counterpropagation networks [3] 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.

1.2 Normal backpropagation training

This network has no backward, lateral or layer-skipping connections. All processing neurons have a bias which is implemented as an extra input which is always on. Note that following the usual (unfortunate) convention, input neurons are drawn, however these are not processing neurons, merely switch boxes distributing the single input x_i to the hidden neurons.

The standard algorithm is as follows:

1. Initialise weights (including bias weights) to small random values.
2. Present input X = , desired output D.
3. Calculate output Y = :

   hidden activations

   output activations

   where
4. Adapt weights recursively, starting with the output layer and working backwards towards the input layer:

   For the output neurons:

   For the hidden neurons:
5. Repeat from step 2 until finished.

1.3 Reverse direction training

We have applied the error back-propagation technique [4] 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 are used normally, along with biases. Note that here input neurons are processing neurons when used in the reverse direction.

When training in the reverse direction, different biases are used as shown. This would be the case for multiple hidden layers also.

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.

Figure 1: Bidirectional network

2 Data

We use two data sets to demonstrate our techniques. These are a small synthetic data set, and a complex real world data set. These are described below.

2.1 dh46

This data set consists of 16 patterns, being all combinations of 4 binary elements. The single output is on when 3 out of 4 binary elements are on. This data set is of the class of data sets such as *xor* which are beyond the power of single perceptrons, and require a hidden layer trained by algorithms such as the back-propagation training algorithm described in §1.2. At the same time, the data set is complex enough that in a 4-6-1 back-propagation network, it shows a range of behaviour from immediate learning, learning after some time with little visible improvement (training plateau), and never learning (stuck on a plateau / local minimum). For historical reasons this is called the dh46 pattern set.

2.2 gdp

The gdp data set is from the United Nations Conference on Trade and Development selected economic and social indicators of developing countries [5]. The data set includes the following 14 indicators: % population urban; size of total labour force; % of labour force in agriculture; % infant mortality; % public health expenditure; population per doctor; % access to water; illiteracy %; % public education expenditure; food production per capita; number of phones per capita; total population; average growth of population; density of population. The task is of course to predict the GDP of each country.

3 Interactive competition model

After training a back-propagation network, using as the input to the network the economic and social factors for a particular developing country, the network can quite accurately predict the GDP (given the quality of the data due to the difficulty of collection). But how is the network's conclusion useful?

3.1 Background

In modelling the relationship between GDP and the selected socio-economic indicators, the real objective is not to predict GDP accurately as it can be measured directly, but to answer questions relating to change such as "If we wanted a higher GDP, how should (for example) general access to safe drinking water change," or (more likely) "given the economic and social factors we are committed to politically, what factors can we change to achieve an effect on GDP or health or ..." and so on.

A predictive model such as the trained feed-forward network can be forced to this end by perturbing the input values into the network and examining the results. This is very inefficient, given the large number of inputs even in this relatively simple case.

The causal index of each input to the feed-forward network to the GDP output value provides a significance measure of that input's relationship to the output, and can be used as a weight in a simple interactive neural network model. That is, the rate of change of the output when the input is in the vicinity of its actual value in a pattern represents the significance in the context of the actual example.

3.2 The model

The interactive activation and competition (iac) model consists of a collection of processing units organised into some number of competitive pools. There are excitatory connections among units in different pools and inhibitory connections between units within the same pool. The excitatory connections between pools are generally bi-directional, thereby making the processing interactive in the sense that processing in each pool both influences and is influenced by processing in other pools. Within a pool, the inhibitory connections run from each unit in the pool to every other unit in the pool implementing a kind of competition among the units such that the unit or units in the pool that receive the strongest activation tend to drive down the activation of the other units [6].

The network is operated by turning on a number of units and allowing the network to cycle until state has been reached. It is also possible to clamp some of the unit on to serve as a constant stimulus to the network.

Figure 2 is a simplified diagram of an iac network, with three sample input pools. Values within pools are sub-ranges of the input data to the back-propagation net [7]. Weights between pools are bi-directional, set to the sum of causal indexes for each input / output sub-range pair. Note that not all sub ranges need to be indexed.

Figure 2: Simplified iac diagram showing pools

3.3 Results

Testing the interactive network in its response to change, for an example we clamp the gdp6 unit on, being two GDP sub-ranges higher than the actual GDP. All of the input values are clamped on except for the % public health expenditure, and % public education expenditure. After the stable state of the network is reached, the values of the variables which were not clamped provide the solution.

Validation of the solution was done by using the trained feed-forward network, by constructing a new input pattern consisting of the old values of the clamped variables, and the resulting values of the unclamped variables. The test is whether the value of GDP predicted by the feed-forward network based on the new pattern is similar to the value the GDP was clamped to in the interactive network. In the case of the prediction, the result of the new pattern is a GDP value of 0.47 which is an increase in the GDP of about 2% which is not insignificant. This confirms both the increase we have tried to produce, and at the same time the difficulty of increasing the GDP with only the flexibility allowed by the few variables left unclamped. This suggests there are one or more countries in the list which have similar relationships between their public health and education expenditures and GDP. There is, however, a component of this approach which is still too simplistic, that of clamping values. The above strategy also assumes that, for example, the illiteracy rate will not be affected by the reduced public education expenditure, since the value was clamped.

4 Hybrid SOM model

4.1 Introduction

The key notion in solving the inversion of many to one (many to many) mappings is to recognise the different categories on input which may give rise to a particular output. For this purpose we will use a self-organising map, described below.

4.1.1 SOM algorithm

1. Initialise weights to small random values:

   N inputs and *M* outputs (*N*M* weights), set neighbourhood to large size

2. Present input

3. Compute distance from input to all nodes:

   distance d_j from input pattern to node *j* is

   where x_i is component *i* of input *X*

   w_{ij} is weight from input *i* to node *j*

4. Select output node with minimum distance from input – j^*

5. Update weights to j^* <u>and its neighbours</u>

   for , ,

6. Decrease neighbourhood size (*NE),* and decrease gain term (η)

7. Continue from step 2.

4.3 The hybrid model

The graphical description of the model is below, in Figure 3, with only a few connections shown for clarity. The model consists of a standard back-propagation trained feedforward neural network, a single dimension self-organising map (SOM), the output of which is used to create the training set for a single layer back-propagation (or perceptron) network. This latter network uses (selected instances of) the activation values of the first network as its training input and produces the appropriate original input as its output. For use on unknown patterns, the output value is input to the SOM, mapped to the relevant SOM categories, and the appropriate SOM exemplars are used as inputs to the single layer to produce outputs. The process is explained and justified in detail in subsequent sections.

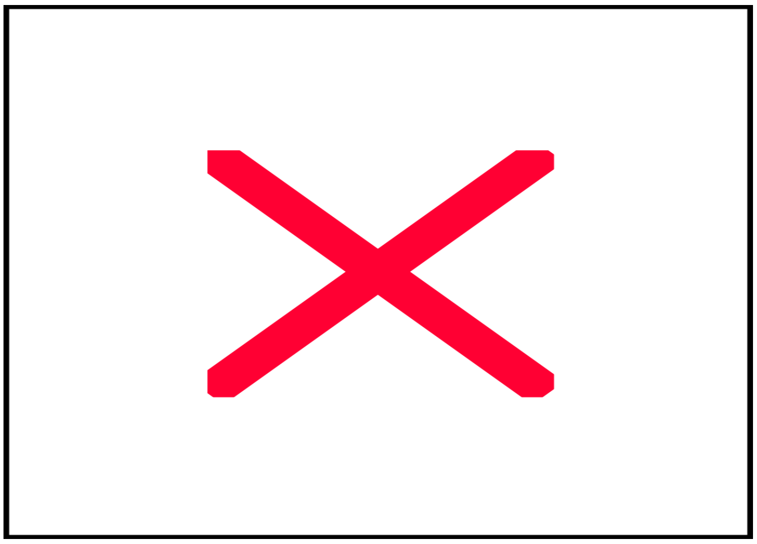
Figure 3: The hybrid model during training

4.3.1 Forward training on bp

The original network is trained using standard back-propagation as described earlier, and can be used normally for prediction of outputs based on the usual inputs. An issue to resolve, however, is how to determine when to stop training, as we do not want to reserve some training patterns as a test set, for obvious reasons.

In previous work [8], we have examined the activations of hidden neurons during back-propagation training. There is a clear indication of learning taking place while there is little or no change in the total sum of squares (tss) error measure. The point at which the tss falls, and there is substantial diminishing of the changes in behaviour of the hidden units indicates the point at which to cease training. The curves for the tss and hidden unit behaviour (represented as angular differences in their behavious vectors in the state space) for both the *dh46* and *gdp* examples are below.

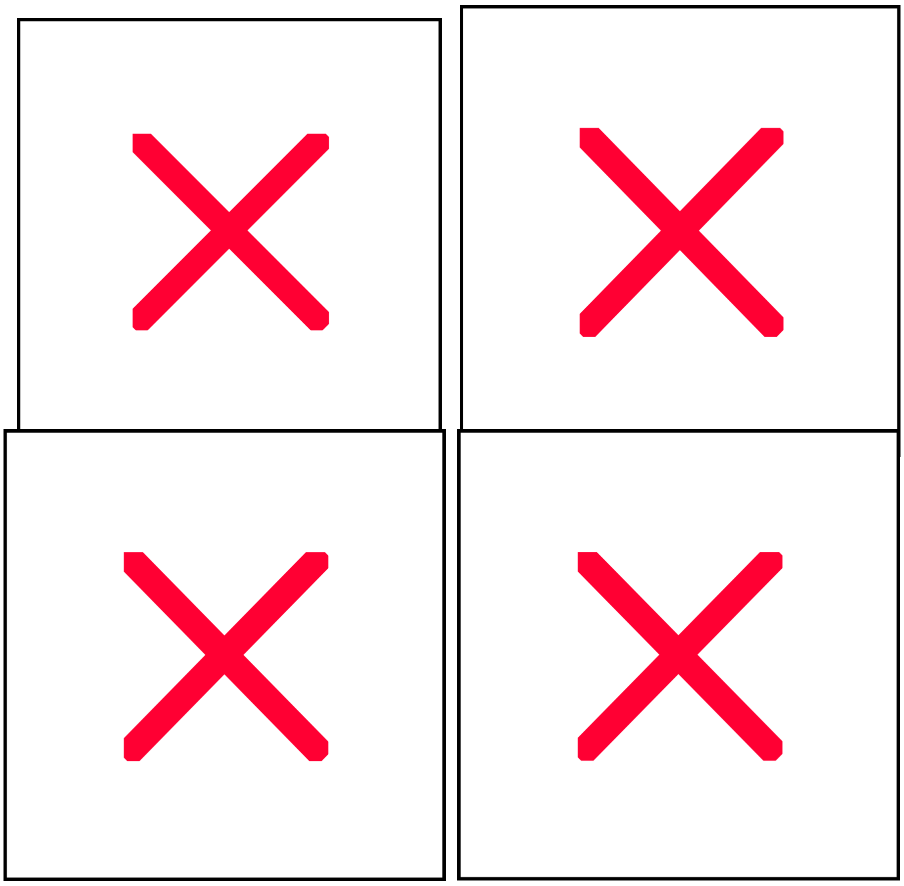
Figure 4: Graphs of Total sum of squares and Neuron behaviour for *dh46* and *gdp*

From the *dh46* graph on the total sum of squares, it is plausible to assume that the network has learnt at about 1500 epochs. This is clearly confirmed by the graph of neuron behaviour, where the lines smooth out after that point.

For the *gdp* graph on total sum of squares, there is no obvious point which is different. The graph on neuron behaviour, however, indicates that sometime around 750 to 800 epochs was the appropriate point to cease training.

4.3.2 Input into SOM

The activation values produced by each hidden neuron in the original back-propagation network are used as input for the SOM. These activation values were used to create the neuron behaviour graphs, and used to determine when to stop training the back-propagation network. In this case, the activation values from every tenth epoch were used. The SOM was trained using the algorithm described earlier, with an initial gain term of 0.9, initial neighbourhood of 30%, and 16 SOM neurons for both data sets. The SOM results for the *dh46* and *gdp* data sets are described below.

4.3.3 SOM results - dh46

For each of the 16 patterns (labelled *p00* to *p15*), the second row shows the index of the nearest SOM unit (ie its category). The third row shows the 4 patterns which satisfy the '3 out of 4' condition.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	2	5	3	4	4	**8**	1	6	6	**13**	4	**11**	**9**	15
							1				1		1	1	

The four sets of patterns which have a 1 in the desired output in the original training set have been identified by a contiguous subset of the SOM indices. There is substantial redundancy in the indices for many of the remaining patterns. Note that the SOM did not have direct access to the original training set, only to the activations of neurons being trained on that data set. Thus, for example, the activations of the hidden neurons every ten epochs for a total of 2000 epochs on pattern *p07* had clearly contained some indication of the significance of this pattern and that it had some relationship to the patterns *p11*, *p13*, and *p14*. It is clear that *p15* is also somewhat related, being the only pattern with 4 out of 4 being on, and hence being somewhat more difficult to distinguish from the correct patterns.

For comparison, the SOM was also run on the training pattern set. Clearly, there is only one set, which is repeated, to ensure the same total number of presentations of inputs to the SOM have taken place. The results were:

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
12	7	13	8	15	6	14	**1**	11	10	12	**9**	16	**5**	**15**	3
							1				1		1	1	

There is no pattern to the 4 four on patterns, clearly the SOM on hidden neuron activations was discovering information which it is not able to discover directly from the original training set. This validates our use of the hidden neuron activations as the input to the SOM.

4.3.4 SOM results - gdp

There are 143 patterns in the original *gdp* training set, so we can not readily show the pattern in the same way. Also, the target output is not symbolic and as easy to interpret as the *dh46* data set.

Some qualitative comparisons are possible. The developing countries most familiar to many in the West are the richest developing country. A test of the success of our technique would be whether these countries are sorted together.

The SOM grouped the following countries in category 16: Brunei, Qatar, and United Arab Emirates; and in category 4: Chile, Colombia, Fiji, Hong Kong, Malta, Mauritius, Panama, Saudi Arabia, Singapore, Uruguay, and Venezuela. While these are surprises in the latter group, the three richest are grouped together, and clearly Hong Kong and Singapore we would have expected to end up in the same group.

Again we compare with the result of running the SOM on the original data set. The results are less good. The three richest countries are in three different groups. The second grouping identified above partially survives, with 3 different countries. The changes would be hard to justify, for example Saudi Arabia is now clustered with Ghana.

5 Conclusion

We have introduced our technique for training a back-propagation network in the reverse direction by the use of a Self-Organising Map. This hybrid structure bidirectional neural network solves the previously identified problems of inverting many to one or many to many mappings. The steps in the process have been individually justified. The ability of such networks to provide suggestions for modification of input parameters to achieve the desired results will be useful in a number of application areas.

6 References

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

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

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

[4] Gedeon, T.D. "Stochastic bidirectional training," *IEEE International Conference on System Man and Cybernetics (SMC'98)*, session on Semantics and Logic, San Diego, pp. 1968-1971, 1998.

[5] *UNCTAD Statistical Pocket Book*, United Nations, New York, 1984.

[6] McClelland, J.L. and Rumelhart, D.E. *Explorations in Parallel Distributed Processing*, MIT Press, 1988.

[7] Gedeon, T.D. and Good, R.P. "Interactive modelling of a neural network model of GDP," *Proceedings International Conference on Modelling and Simulation*, Perth, pp. 355-360, 1993.

[8] Gedeon, T.D. and Harris, D. "Hidden Units in a Plateau," *Proceedings 1st International Conference on Intelligent Systems, Singapore*, pp. 391-395, 1992.