

Can Artificial Neural Networks Discover Useful Regularities?

JV Stone and CJ Thornton*

Cognitive and Computing Sciences, University of Sussex.

May 21, 2003

Abstract

It is argued that the success of artificial neural networks (ANNs) to date has depended almost exclusively upon the judicious choice of input representations which effectively recode high-order input parameters as low-order parameters. It is demonstrated that, where such a recoding is required but not provided, BP ANNs fail to generalize. Of course, BP ANNs can utilize the spurious low order statistics associated with almost any ‘natural’ problem. However, these provide partial information about the underlying mapping, and therefore do not, in general, permit generalization. In order to obtain generalization it is necessary to hand-craft inputs so that parameters which were coded as relations between input components are coded by a single input component. Thus the responsibility for enabling an ANN to learn a given task lies ultimately with the designer of the ANN. It is argued that this is an undesirable state of affairs, in a field where it is widely accepted that the process of learning supposedly obviates the need to hand-craft computational models.

Introduction

The early promise of ANNs for complex learning tasks has not been fulfilled. Instances of success have relied heavily upon appropriate recoding of input data, or on the use of a task-specific, often elaborate, architecture. In both cases, the system designer has had to apply expert knowledge in order that the task could be learned by an ANN. Rather than applying heuristics directly as rules in an expert system, connectionists implement heuristics in an implicit manner via the application of expert knowledge to the input coding and/or the system architecture.

Backpropagation (BP) ANNs are universal function approximators. In principle, there is no input/output mapping they cannot approximate. In practice, existing learning methods such as BP can only acquire certain mappings. We

*JV Stone is a joint member of Biological Sciences and Cognitive and Computing Sciences

begin by demonstrating that BP ANNs cannot generalize over parity mappings, even if all but one case from the training data is provided. We hypothesize that this is because parity mappings are statistically neutral (see definition below), and show that the same result is obtained for another, neutral problem (the likelihood problem). We then show that the effects of neutrality can be circumvented in several ways, some of which provide effective generalization, and some of which do not.

Generalization On Parity Mappings

It is widely accepted that any learning problem can be defined in terms of a mapping from an input space to an output space. Given a finite training set which is a sample of input/output pairs from this mapping, the learner's objective is to map *any* input to its associated output.

Consider the 4-bit parity problem. This can be written as a training set as follows:

x1	x2	x3	x4		y1
1	1	1	1	-->	0
1	1	1	0	-->	1
1	1	0	1	-->	1
1	1	0	0	-->	0
1	0	1	1	-->	1
1	0	1	0	-->	0
1	0	0	1	-->	0
1	0	0	0	-->	1
0	1	1	1	-->	1
0	1	1	0	-->	0
0	1	0	1	-->	0
0	1	0	0	-->	1
0	0	1	1	-->	0
0	0	1	0	-->	1
0	0	0	1	-->	1
0	0	0	0	-->	0

This data set is *statistically neutral*. That is, the value of any single input component x_i provides no information about the value of the output y . For a binary data set, this implies that the conditional probability $p(y = 1|x_i = 1) = 0.5$ for every input component x_i .

Using a feedforward ANN, the BP method can learn this data set. However, if the ANN is only given 15 of the 16 pairs then it does not learn to solve the parity problem. That is, when presented with the 16th, previously 'unseen' input, it invariably generates an incorrect output. This is because BP relies only on the low order statistical structure of the training set, rather than any inferential mechanisms which might otherwise permit it to 'guess' the correct output.

We have carried out an exhaustive empirical analysis of the performance of a BP ANN with three unit layers. The particular learning algorithm uses conjugate gradients [1] to minimize the standard error function E . The number n of hidden units was varied between three and thirty. For each n , the results from 10 successful training runs were obtained. In gathering these results, the proportion of failed runs was typically around 3%. The required number of conjugate gradient line-searches was typically around 20, where each line search requires two complete passes through the training set.

On each training run, a randomly chosen data pair was removed from the data set, and used to test the ANN after it had learned the other 15 data pairs. The ANN was then trained on the set of 15 pairs. Due to the the fact that we used an ANN with units with a \tanh function, we used a (functionally equivalent) set in which each 0 was replaced by -1 . Runs were terminated once negligible mean error on the training cases had been achieved or after 50,000 iterations.

The results are summarized in Figure 1. This shows the mean error for the ‘seen’ items in the incomplete training set and for the remaining, unseen input, for 10 training runs. The error is defined as $|\sigma|$. (This is a more intuitive measure of the performance of the ANN than the more conventional RMS error). Clearly, generalization beyond the incomplete training set failed. In every run, the output associated with the single test item was incorrect, that is, $|\sigma| > 1$.

Why does BP perform so badly on this problem? One hypothesis is that BP relies on low order statistical information extracted from the training set [Thornton, Measuring, Forthcoming]. The poor generalization is then explained, since parity problems are known to exhibit *no* low order statistical regularities whatsoever [2]. This result is partially demonstrated by Table 1, which plots out all the conditional output probabilities for first-order input cases (i.e., single, input-variable instantiations). Note that all the probabilities are at their chance level of 0.5.

C	P(C)	P(y1=1 C)	P(y1=0 C)
	1	0.5	0.5
x4=1	0.5	0.5	0.5
x3=1	0.5	0.5	0.5
x2=1	0.5	0.5	0.5
x1=1	0.5	0.5	0.5
x4=0	0.5	0.5	0.5
x3=0	0.5	0.5	0.5
x2=0	0.5	0.5	0.5
x1=0	0.5	0.5	0.5

Table 1:

It is sometimes argued that BP’s generalization failure on parity problems is of no consequence because the parity mapping is an artificial, mathematical

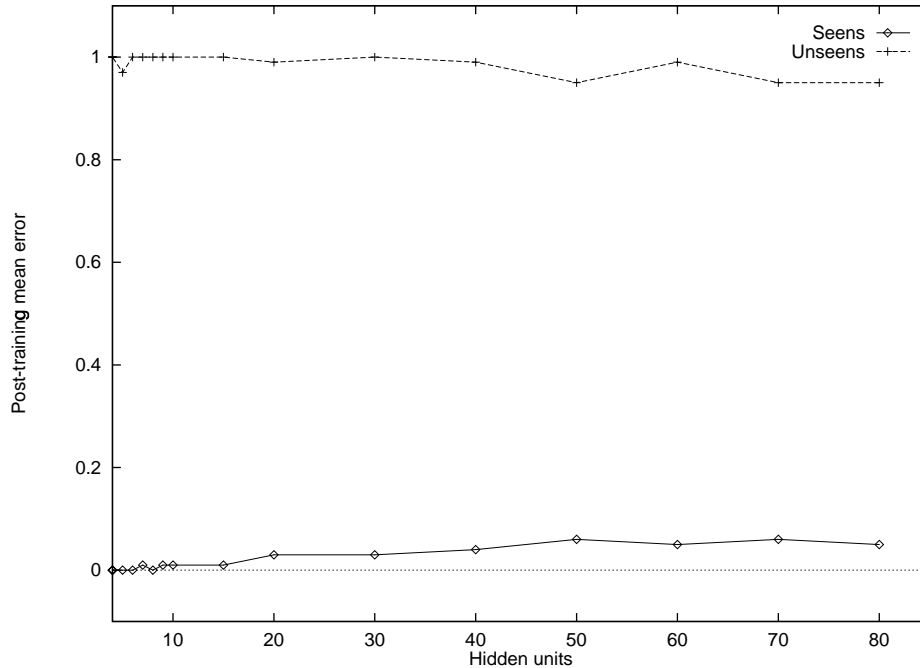


Figure 1:

construct which lacks a realistic input/output rule. However, problems which are statistically neutral are not *necessarily* parity problems.

A Statistically Neutral, Non-Parity Problem

The ‘likelihood problem’, whose training set is shown below, is not a parity problem. It has an intuitively obvious input/output rule. However, we can translate it into a numerical problem with the following substitutions: person/0, computer/1, consumes/0, dislikes/1, heat/0, electricity/1, moisture/2, silicon/3, yes/0, no/1.

```

person consumes heat           -> yes
person consumes electricity     -> no
person consumes moisture       -> yes
person consumes silicon        -> no
person dislikes heat           -> no
person dislikes electricity     -> yes
person dislikes moisture       -> no
person dislikes silicon        -> yes
computer consumes heat         -> no

```

```

computer consumes electricity -> yes
computer consumes moisture   -> no
computer consumes silicon    -> yes
computer dislikes heat       -> yes
computer dislikes electricity -> no
computer dislikes moisture   -> yes
computer dislikes silicon    -> no

```

We can establish the neutrality of this training set empirically by tabulating the relevant conditional probabilities. (Table 2 shows the complete set of probabilities which have a first or zeroth-order condition.) Note that, in contrast to parity problems, the expected value of input component values are *not* at chance-level. However, this does not affect the statistical neutrality of the problem because, in supervised learning, it is only the conditional output probabilities that are of relevance. If we apply BP learning to the likelihood problem,

C	P(C)	P(y1=unlikely C)	P(y1=likely C)
	1	0.5	0.5
x1=person	0.5	0.5	0.5
x2=dislikes	0.5	0.5	0.5
x2=consumes	0.5	0.5	0.5
x1=computer	0.5	0.5	0.5
x3=electricity	0.25	0.5	0.5
x3=heat	0.25	0.5	0.5
x3=silicon	0.25	0.5	0.5
x3=moisture	0.25	0.5	0.5

Table 2:

keeping back one example as an unseen case, we find once again that the generalization performance is typically worse than chance, see Figure 2. This tends to confirm the hypothesis that BP relies primarily on statistical information extracted from the example set.

Generalization and Statistical Neutrality

A statistically neutral mapping has no correlation between input values and output values. Thus, if the mapping has any input/output rule at all, it cannot be contingent upon states of individual input variables. It *must* be based on relational states between those variables. In other words, statistically neutral mappings have *relational* input/output rules.¹

¹We define a problem as relational if the output value depends upon relations between input variables, and not upon any individual input variable.

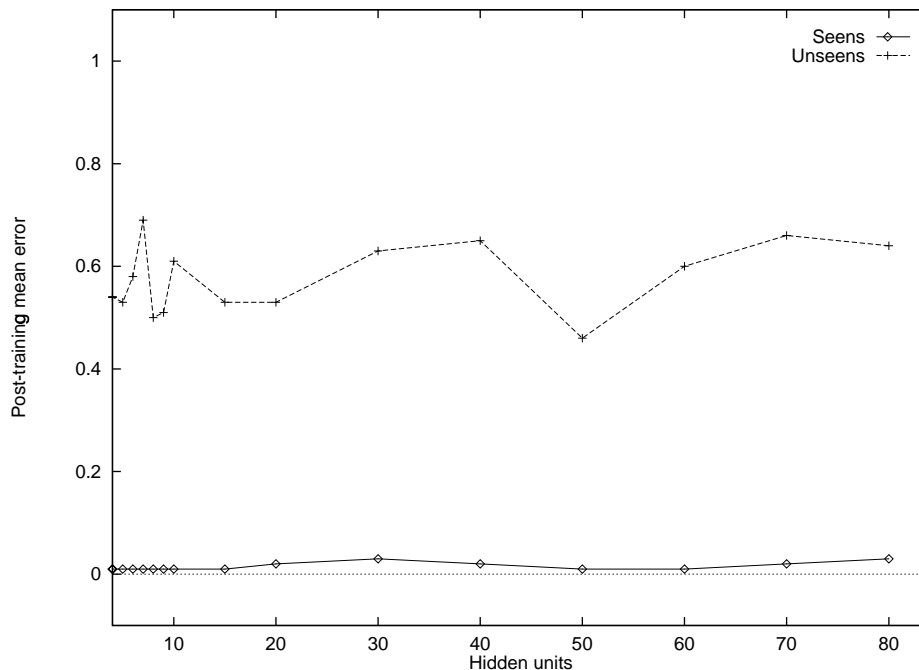


Figure 2:

In principle, this result is devastating for algorithms which rely on low order statistical information. Such algorithms fail to generalize on relational input/output rules. However, in practice, the situation may not be quite as bad as it seems.

If we take a statistically neutral data set and delete one or more of the pairs (as we do in setting up a generalization problem) then the residual mapping usually exhibits *spurious* statistical effects. These effects are produced by deleting certain elements of the ‘full’ (underlying) mapping. Thus, these spurious effects are not exhibited by the deleted cases. Therefore any algorithm which makes use of them cannot, in general, generalize to the deleted data pairs. This is essentially what happens when we apply BP to parity problems.

A BP ANN can learn parity problems if it is given the ‘full’ data set. However, anecdotal evidence suggests that learning time increases rapidly as the number of input units increases. This is consistent with systematic studies of the time complexity of BP ANNs on other problems [3]. Scaling problems can be alleviated by recoding the inputs so that the output is specified by low order statistical correlations between input and output. This can be achieved in two ways.

The simplest approach is to produce a *sparse coding* of the problem, e.g., to recode the problem so that each *value* of each original variable has its own unique

binary variable. Any given input is now specified by setting a single binary input variable to 1. All other inputs sets this variable to 0. The effect of this encoding is that most input variables have the value 0 *most of the time*. A system of low order statistical effects is thereby created. This can be readily exploited by a statistically oriented generalizer such as BP. The results of learning on a sparse coding of the likelihood likelihood problem are shown in Figure 3. In contrast to the results with parity and the ‘dense’ coding of this problem, generalization is fairly good. Alternatively, we can introduce a *hand-crafted* new input variable

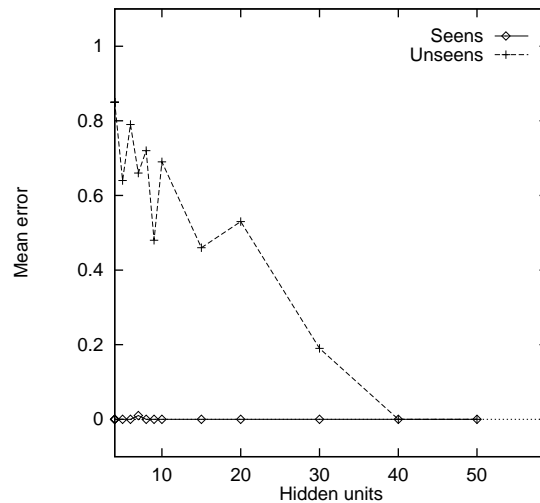


Figure 3:

(or variables) which captures the relational effect upon which the mapping is based. This re-expresses the relational effect so that informative correlations between individual input/output components arise. For example, in the case of a parity problem, if we introduce a variable which explicitly contains a count of the number of 1’s appearing in the input then the new, revised mapping will show strong statistical regularities, i.e., strong, non-chance conditional probabilities connecting the states of the count variable with states of the output variable.

A critical difference between the sparse coding and hand-crafted re-codings is that the former generates spurious correlations between input and output components, whereas the latter produces correlations which reflects the underlying mapping. It follows that generalization performance with sparse codings cannot, in general, be completely accurate, whereas generalization obtained with hand-crafted inputs can.

Conclusion

We conjecture that many ‘natural’ problems represent sparse codings of some underlying, relational mapping. If true, this suggests that the ability of BP ANNs to solve such problems is more apparent than real, because BP ANNs rely upon informative, but spurious, correlations between input and output variables. Making use of these spurious correlations allows BP ANNs to appear as if they are solving some specified problem. However, in principle, such correlations do not permit the underlying mapping (from which the training set were derived) to be recovered by any ANN.

References

- [1] Williams, P. (1991). A marquardt algorithm for choosing the step-size in backpropagation learning with conjugate gradients. Cognitive Science Research Paper CSRP 229, University of Sussex.
- [2] Hinton, G. and Sejnowski, T. (1986). Learning and relearning in boltzmann machines. In D. Rumelhart, J. McClelland and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Vols I and II* (pp. 282-317). Cambridge, Mass.: MIT Press.
- [3] Stone, J. and Lister, R. (1994). On the relative time complexities of standard and conjugate gradient back-propagation. *Proceedings of the IEEE Int. Conf. Neural Networks*, (Orlando) (pp. 84-87).