

Statistical Biases in Backpropagation Learning

Chris Thornton

Cognitive and Computing Sciences
University of Sussex
Brighton
BN1 9QH
UK

Email: Christopher.Thornton@firenet.uk.com

Tel: (44)1273 678856

May 21, 2003

Keywords: Cognitive Science, Pattern recognition

Abstract

The paper investigates the statistical effects which may need to be exploited in supervised learning. It notes that these effects can be classified according to their conditionality and their order and proposes that learning algorithms will typically have some form of bias towards particular classes of effect. It presents the results of an empirical study of the statistical bias of backpropagation. The study involved applying the algorithm to a wide range of learning problems using a variety of different internal architectures. The results of the study revealed that backpropagation has a very specific bias in the general direction of statistical rather than relational effects. The paper shows how the existence of this bias effectively constitutes a weakness in the algorithm's ability to discount noise.

1 Introduction: learning by the capture of statistical effects

The process of learning is conveniently conceptualized in terms of the acquisition of a target input/output mapping. To have any chance of success the learner requires some source of feedback regarding the mapping to be acquired. In the much studied supervised learning scenario, this feedback takes the form of a set of examples taken from the target mapping. [1] The learner's aim is to arrive at the point at which it is able to map any input taken from the mapping onto its associated output. In more general terms, the learner's aim is to be able to

give a high probability to the correct output for an arbitrary input taken from the mapping.

If the learner is to have any chance of achieving this goal, the feedback it receives must contain information which justifies the assigning of particular probabilities to particular outputs. Thus we see that supervised learning is essentially the process of discovering and exploiting such justifications. To understand the nature of the process we need to analyze the ways in which supervisory feedback can provide justifications for assignments of particular probabilities to particular outputs.

There are two main cases to consider. Supervisory feedback (i.e., sets of examples) can justify probability assignments either *directly* or *indirectly*. Direct justification is provided if the probabilities in question are observed directly within the training examples, as statistical effects. They are justified indirectly if they cannot be observed directly but can be systematically derived from those examples; or — which amounts to the same thing — if they can be observed in data which are derived from the original data. [2].

In this paper I will be concerned only with the direct (i.e., statistical) form of justification, and with the ways in which it is exploited by the backpropagation learning algorithm (but see [3]). The nature of this form of justification can be illustrated with an example. Consider the following training set. This is based on two input variables (x_1 and x_2) and one output variable (y_1). There are six training examples in all. They are laid out with one example per line. An arrow separates the input part of the example from the output part. The values of the two input variables appear on the left of the arrow. The value of the output variable appears on the right.

x1	x2		y1
1	3	⇒	1
2	1	⇒	0
3	2	⇒	1
3	1	⇒	0
1	2	⇒	1
3	1	⇒	0

A wide variety of probabilities can be observed directly in these training examples, some of which relate specifically to the output variable (and are thus relevant for a supervised learner). To begin with we have the probabilities for first-order cases (i.e., instantiations of a single variable). These are shown in Table 1. The ‘C’ column shows the case in question and the ‘P(C)’ column shows the observed probability of that case. We can also observe the probabilities of many second-order cases — cases involving the instantiation of two variables. These are shown in Table 2.

The probabilities for the third-order cases (i.e., the cases that specify values for all three variables) are, of course, degenerate. Assuming there is no duplication in the training data, each third-order case occurs exactly once. Thus its probability is necessarily $1/n$ where n is the size of the training set.

C	$P(C)$
	1
$x1 = 3$	0.5
$x2 = 1$	0.5
$y1 = 1$	0.5
$y1 = 0$	0.5
$x2 = 2$	0.33
$x1 = 1$	0.33
$x2 = 3$	0.17
$x1 = 2$	0.17

Table 1:

C	$P(C)$
$x2 = 1, y1 = 0$	0.5
$x1 = 3, x2 = 1$	0.33
$x2 = 2, y1 = 1$	0.33
$x1 = 1, y1 = 1$	0.33
$x1 = 3, y1 = 0$	0.33
$x1 = 1, x2 = 3$	0.17
$x1 = 3, x2 = 2$	0.17
$x2 = 3, y1 = 1$	0.17
$x1 = 3, y1 = 1$	0.17
$x1 = 1, x2 = 2$	0.17
$x1 = 2, x2 = 1$	0.17

Table 2:

The probabilities introduced so far are all unconditional. A variety of conditional probabilities can also be observed. For example, we can observe the conditional probability of observing a particular instantiation of the output variable for given first-order cases of the input variables. These probabilities are shown in Table 3. By the argument used previously, the second-order conditional probabilities here are degenerate since there is necessarily exactly one occurrence of each second-order case of the constrained variables.

As mentioned above, in the supervised learning scenario, it is the probabilities affecting the output variable(s) which are of interest. Thus, Table 3 in conjunction with the table listing the first-order unconditional probabilities for the output variable, provide an exhaustive enumeration of all directly observed, probability-assignment justifications (henceforth just ‘statistical effects’). A quick perusal of the two tables shows that the probabilities for possible values of $y1$, conditional on values of $x2$ and $x1$ are extreme. These might form the underlying justification for the summary rule: $y1 = 1$ if $x2 = 1$ or $x2 = 3$;

C	$P(C)$	$P(y1 = 0 C)$	$P(y1 = 1 C)$
	1	0.5	0.5
$x1 = 3$	0.5	0.67	0.33
$x2 = 1$	0.5	1.0	0.0
$x2 = 2$	0.33	0.0	1.0
$x1 = 1$	0.33	0.0	1.0
$x2 = 3$	0.17	0.0	1.0
$x1 = 2$	0.17	1.0	0.0

Table 3:

otherwise $y1 = 0$.

2 Statistically-oriented learning methods

By definition, directly-observed justifications for probability assignments can be discovered more readily than indirectly-observed justifications. It is to be expected therefore that general-purpose learning algorithms such as ID3 [4] and Backpropagation [5] will be able to exploit them most effectively. However, it is also to be expected that such learning algorithms will exhibit some form of bias or predisposition to deal more effectively with statistical effects of particular types. For example, we might find that a particular learning algorithm exploits first-order effects very easily but is effectively insensitive to higher-order effects.

The main aim of the present paper is to present the results of an empirical study which sought to evaluate the statistical bias of the backpropagation algorithm. The study showed that although backpropagation exploits all statistical effects quite effectively it often deals with higher-order effects better than with low order effects.

3 The study

The study involved training a standard backpropagation implementation (the PDP package of [6] was used) on a variety of artificial learning problems. The solution of each artificial learning problem was based on a statistical effect of a particular order. The results showed clearly that the generalization performance of backpropagation varies monotonically with the order of the underlying statistical effect. Best generalization performance was consistently produced on n th-order problems, where n was the number of input units used and thus the maximum order of statistical effect that could be represented.

All the artificial learning problems were classification problems. They involved learning to correctly allocate an input to one of five classes (to produce an output correctly classifying the input as a member of one of five different input classes). The classification rule — the solution to the learning problem

— was, in all cases, prototype-based. That is to say, the five input classes were defined in terms of five input prototypes. To obtain solutions based on low-order effects, I used prototypes defined over *subspaces* of the input space. Thus, for a solution based on an m th-order effect, I used a prototype defined over an m -dimensional subspace of the input space.

Note that, in the present context, a ‘prototype’ is just a set of input-variable instantiations. Low-order prototypes are instantiations for a small subset of input variables. High-order prototypes are instantiations for a large subset of input variables. An example of a prototype, i.e., an instance of the input class defined by the prototype, is derived simply by generating an input vector which ‘matches’ the prototype. An input vector forms a good match to a prototype if the values it shows for the prototype variables are close to (within 1 standard deviation of) the instantiations specified by the prototype. Thus, to generate an example of a prototype we construct a vector which features small random variations of the prototypical-variable instantiations and purely random values elsewhere. To generate a complete training set we repeatedly cycle through our N prototypes generating prototype examples and appending, as target output, the appropriate class ‘label’.

All the data shown is based on experiments with ‘minimal’ backpropagation architectures. These were strictly layered, feed-forward, networks with one layer of hidden containing just enough units to guarantee perfect acquisition of the *training* cases.¹ All the networks had eight input units and one output unit. Thus all the problems presented involved producing an ‘output classification’ expressed as a single activation value. In practice, the artificial problems involved making 5-way classifications and this effectively necessitated using output values in the set $\{0.0, 0.25, 0.5, 0.75, 1.0\}$. Having networks of eight input units allowed statistical effects from first-order up to eight-order to be tested.

To minimize interference effects (i.e., ambiguities) between prototype definitions, the same input variables were always used for the specification of all five, n th-order prototypes. Thus all first-order prototypes were defined in terms of a single instantiation of a single input variable. All second-order prototypes were defined in terms of a instantiations of two particular input variables, and so on.

4 The results

The main results of the study are summarized by the graph shown in Figure 1. This shows the mean generalization performance (expressed in terms of mean-difference error on the testing set) for learning problems based on statistical effects of different orders. Note how the generalization performance improves monotonically with the order of the relevant effect. The general implication is that backpropagation is biased towards higher order statistical effects. In some of the runs performed, overtraining was observed but in general this did not seem to be a major feature of the learning. A typical error curve for a run on a low-order training problem is shown in Figure 2. Note the rapid descent of the

¹The precise number of units in each case was determined by trial and error.

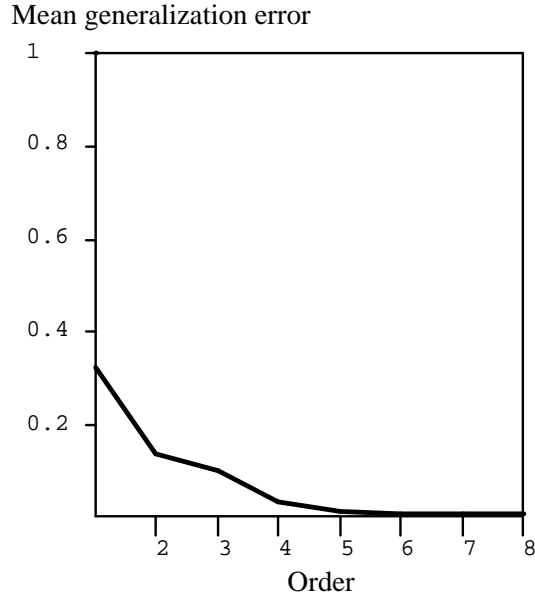


Figure 1:

error curve for the training set and the much more gradual descent of the error curve for the testing cases.

5 Explaining the results

At first sight, the results of the study seem somewhat paradoxical. In some sense, high-order statistical effects are more ‘complex’ than low order effects since they involve the specification for larger numbers of values. Therefore, one might expect a learning algorithm to deal more readily with low-order effects. On the other hand, exploiting low-order statistical effects involves ‘factoring out’ all those input variables which are not involved in the statistical effect, and whose values are therefore pure noise.

As it turns out, it is this ‘factoring out’ aspect of the low-order exploitation task which appears to be backpropagation’s weakness. The problem generator used in the study instantiates prototypes working upwards through the input dimensions. Thus all the first-order prototypes are instantiated over input variable 1. This input variable corresponds to the leftmost, lowest numbered input unit in the input layer of the network. In a first-order problem, the instantiations for all input variables bar the first one are effectively pure noise. Thus, in a backpropagation solution for such a problem, we would hope to see all other variables factored out of the internal representation. In other words, we would hope to see zero weights on all the connections from all input units except input

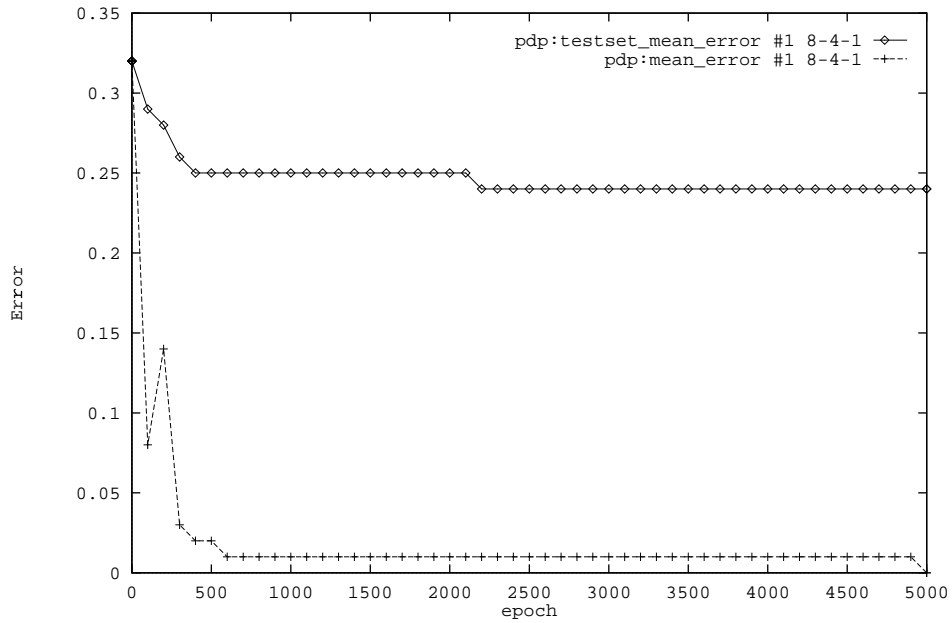
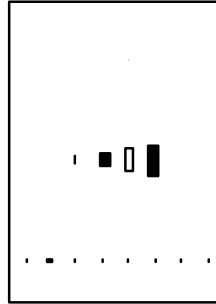
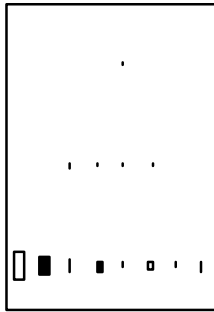


Figure 2:

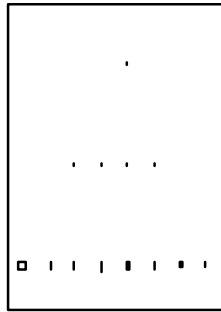
unit number 1. However, when we come to examine the Hinton diagram [7] for non-input nodes in the the network (see Figure 3), we certainly do not see this effect. In fact what we see is that the learning has produced quite pronounced positive and negative weightings for connections from higher-numbered input units.



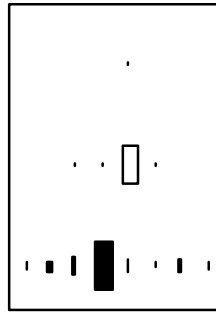
13



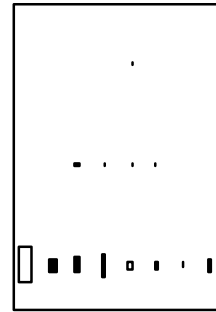
9



10



11



12

In a sense, this is only to be expected. Backpropagation is relying on the assumption that no part of the input space will contain pure noise. For the low-order problems, this assumption is invalid. And thus we see a quite clear deterioration in generalization performance for problems of this type.

6 Summary

The paper has discussed the purely statistical effects which form the most readily-obtained justifications for solutions to supervised learning problems. It has observed that such effects can be classified according to their conditionality and their order. It seems plausible that learning algorithms able to exploit statistical effects will have some form of bias towards particular types of effect. In the case of backpropagation it appears that this bias takes the form of a strong predisposition towards higher-order effects. This, somewhat counter-intuitive result may be explicable in terms of backpropagation's inability to properly discount input variables whose instantiations are randomly acquired and thus form pure noise.

References

- [1] Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. New York: Wiley.
- [2] Thornton, C. (1993). Supervised learning of conditional approach: a case study. CSRP 308, Cognitive and Computing Sciences, University of Sussex, UK.
- [3] Clark, A. and Thornton, C. (1993). Trading spaces: computation, representation and the limits of learning. Cognitive Science Research Paper 291, Brighton BN1 9QH: University of Sussex (Price:1.50).
- [4] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1 (pp. 81-106).
- [5] Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323 (pp. 533-6).
- [6] McClelland, J. and Rumelhart, D. (1988). *Explorations in Parallel Distributed Processing: A handbook of Models, Programs, and Exercises*. Cambridge, Mass.: MIT Press.
- [7] 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.