Parity: The Problem that Won't Go Away

Chris Thornton
Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QN
UK
Email: Chris.Thornton@cogs.susx.ac.uk
WWW: http://www.cogs.susx.ac.uk/users/christ/index.html
Tel: (44)1273 678856

December 20, 1995

Abstract

It is well-known that certain learning methods (e.g., the perceptron learning algorithm) cannot acquire complete, parity mappings. But it is often overlooked that state-of-the-art learning methods such as C4.5 and backpropagation cannot generalise from *incomplete* parity mappings. The failure of such methods to generalise on parity mappings is sometimes dismissed as uninteresting on the grounds that it is 'impossible' to generalise over such mappings, or on the grounds that parity problems are mathematical constructs having little to do with real-world learning. However, this paper argues that such a dismissal is unwarranted. It shows that parity mappings are hard to learn because they are statistically neutral and that statistical neutrality is a property which we should expect to encounter frequently in real-world contexts. It also shows that the generalization failure on parity mappings occurs even when large, minimally incomplete mappings are used for training purposes, i.e., when claims about the impossibility of generalization are particularly suspect.

1 Introduction

The parity rule is easily stated (e.g., 'the output value is true if and only if an odd number of input values are true') but it is surprisingly hard to learn by conventional methods. The reason is related to the fact that parity mappings are statistically neutral. The probability (i.e., frequency) of seeing some particular input value mapped onto some particular output in a parity mapping always turns out to be the chance value of 0.5. This means that it is impossible to build successful rules which focus on particular input values: any successful rule must attend to *all* the input values in order to get the answer right in all cases.

For the machine learning researcher, the significance of the parity problem has traditionally been its effectiveness as a small but challenging benchmark problem. However, I will argue that this view underestimates the fundamental nature and wider implications of the problem. I claim that the parity problem is at the heart of a major class of hard learning problems and that our understanding of the way in which it can be solved will thus be a key component in our understanding of learning.

The paper divides up into three main sections. The next section (section two) analyses the statistical basis of the parity problem and clarifies its relationship with the wider class of statistically neutral problems. Section three presents a task analysis of learning which leads to a basic distinction between hard and easy learning problems. Section four shows how the class of hard learning problems are statistically neutral in principle but not in practice and demonstrates how incidental, statistical effects can be exploited by standard learning methods. Section five is a discussion and summary.

2 Statistical properties of the parity problem

In a parity problem we have a number of boolean input variables and one boolean output variable. The input/output rule states that the output should be true just in case an *odd* number of input values are true.[1] If there are just two input variables the problem is known as 'Exclusive-OR' (or **XOR**) since it is effectively the rule that either of the inputs can be true, but not both.

It is well known that parity problems are statistically *neutral* [1]. This means that all conditional output probabilities exhibited by a parity mapping have 'chance' values, i.e., that no input/output associations exist. Consider the 3-bit (i.e., 3-input) parity problem, which can be written as a training set (using 1=true, 0=false) as follows.

x_1	x_2	x_3		y_1
1	1	1	$\Longrightarrow$	1
1	1	0	$\Longrightarrow$	0
1	0	1	$\Longrightarrow$	0
1	0	0	$\Longrightarrow$	1
0	1	1	$\Longrightarrow$	0
0	1	0	$\Longrightarrow$	1
0	0	1	$\Longrightarrow$	1
0	0	0	$\Longrightarrow$	0

In Table 1 we see the unconditional and conditional probabilities for all input-variable instantiations. Note that all the probabilities are exactly the chance value for a boolean value, namely 0.5. This is a *necessary* consequence of the nature of the input/output rule. The frequencies with which we see each of the two possible outputs when we put an input variable into a fixed state must always be equal since there will be just as many cases of the unfixed variables which produce parity as non-parity. Thus the output probabilities conditional on any particular input variable instantiation will always be at the chance level. The statistical neutrality of the parity problem means that it cannot be solved by statistical methods: any process of searching for dependencies between specific input and output variables is of no benefit because such associations simply do not exist. Thus, the performance of any learning method which exploits such processes is necessarily compromised on a parity problem. (This is of course what makes the parity problem a challenging benchmark.) But interestingly, it turns out that parity is not the *only* type of problem for which statistical neutrality is guaranteed. Any problem that can be converted into a modulus-addition problem is guaranteed to be statistically neutral provided that the number of possible values for any given input variable is equal to, or an exact multiple of the number of possible output values. To show this we argue backwards from observations about the statistically neutral training set.

[1] Arguably, since the rule tests for an odd number rather than an even number, the problem should be called the *dis*parity problem.

C	$P(y_1 = 0\|C)$	$P(y_1 = 1\|C)$
	0.5	0.5
$x_3 = 0$	0.5	0.5
$x_2 = 0$	0.5	0.5
$x_1 = 0$	0.5	0.5
$x_3 = 1$	0.5	0.5
$x_2 = 1$	0.5	0.5
$x_1 = 1$	0.5	0.5

Table 1: Conditional output probabilities in parity mapping.

If the training set for some learning problem is statistically neutral then all the conditional output probabilities must be at their chance levels. This tells us that in the training set we will see each value of an input variable X appearing with each output value an equal number of times (i.e., that the number of possible values of X must be equal to, or a multiple of the number of possible outputs). We could therefore map the N values of our input variable X onto integers in the range $0...N-1$ and the M output values onto integers in the range $0...M-1$. Moreover, since the only constraint is that each input value must associate with each output value an equal number of times, we could do this in such a way as to ensure that the output value is always the modulus to base M of the value of X.

If we do this to each input variable in turn, using a fixed mapping of the output values, we end up with a purely numeric version of the training set. Then, 'incrementing' the integer value of any variable (i.e., switching attention to a case in the training set showing the next highest value of the variable) always has the effect of 'incrementing' the output value. The training set therefore instantiates a modulus-addition rule. The general conclusion is that statistically neutral mappings can always be translated into modulus-addition problems provided that the cardinality of the set of output values is a factor of the cardinalities of *all* the input-value sets.[2]

2.1 A statistically neutral, non-parity problem

Consider the 'consumer problem' whose target mapping is shown below. This is a straightforward learning problem with a relatively obvious input/output rule. However, we can translate it into a modulus-addition 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. Under this translation, the requirement that the input-set cardinalities are equal to, or a multiple of the output-set cardinality is met so we know that the problem is *necessarily* statistically neutral.

[2] Parity is modulus-addition with modulus 2.

x_1	x_2	x_3		y_1
person	consumes	heat	$\Longrightarrow$	yes
person	consumes	electricity	$\Longrightarrow$	no
person	consumes	moisture	$\Longrightarrow$	yes
person	consumes	silicon	$\Longrightarrow$	no
person	dislikes	heat	$\Longrightarrow$	no
person	dislikes	electricity	$\Longrightarrow$	yes
person	dislikes	moisture	$\Longrightarrow$	no
person	dislikes	silicon	$\Longrightarrow$	yes
computer	consumes	heat	$\Longrightarrow$	no
computer	consumes	electricity	$\Longrightarrow$	yes
computer	consumes	moisture	$\Longrightarrow$	no
computer	consumes	silicon	$\Longrightarrow$	yes
computer	dislikes	heat	$\Longrightarrow$	yes
computer	dislikes	electricity	$\Longrightarrow$	no
computer	dislikes	moisture	$\Longrightarrow$	yes
computer	dislikes	silicon	$\Longrightarrow$	no

We can confirm 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.)

C	$P(y_1 = \text{no}\|C)$	$P(y_1 = \text{yes}\|C)$
	0.5	0.5
x_1 =person	0.5	0.5
x_1 =computer	0.5	0.5
x_2 =consumes	0.5	0.5
x_2 =dislikes	0.5	0.5
x_3 =electricity	0.5	0.5
x_3 =heat	0.5	0.5
x_3 =silicon	0.5	0.5
x_3 =moisture	0.5	0.5

Table 2: Conditional output probabilities in consumer mapping.

2.2 Performance of learning algorithms on neutral mappings

The performance of learning methods that rely on the exploitation of statistical effects is necessarily compromised on statistically neutral problems. Learning methods that rely *solely* on the exploitation of statistical effects produce worst-case performance on such problems. Algorithms in the CART family [2] are a case in point. ID3 [3,4] for example, constructs a decision tree by recursively partitioning the training set until every pair in a given partition maps onto the same output value. At each stage of the process, a new partitioning is constructed by dividing up the cases in an existing partition according to which value they have on the variable whose values are most strongly correlated (within the partition) with specific output values. This has the effect of maximizing the output-value uniformity of new partitions and thus minimising (subject to horizon effects) the total number of hyper-rectangular

partitions required in order to achieve full uniformity. The algorithm is thus guided only by statistical effects in the training data.

The implication is that ID3 should produce worst-case performance on parity problems and on the consumer problem. In fact, it is easy to confirm that it will *necessarily* do so. Since any statistically neutral problem is equivalent to a modulus-addition problem we know that, in the consumer problem, every variable value must be associated with equal numbers of each output value. This means that no variable value is of any use at all in the initial, splitting-up of cases and each, successive, split produces the same, non-uniform situation at a finer-grained level. [5] The result is that ID3 necessarily builds a 'lookup table' for the consumer problem, i.e., a decision tree that captures no generalisations whatsoever and which has one leaf node per case in the training set. The decision tree actually produced in shown in Figure 1.

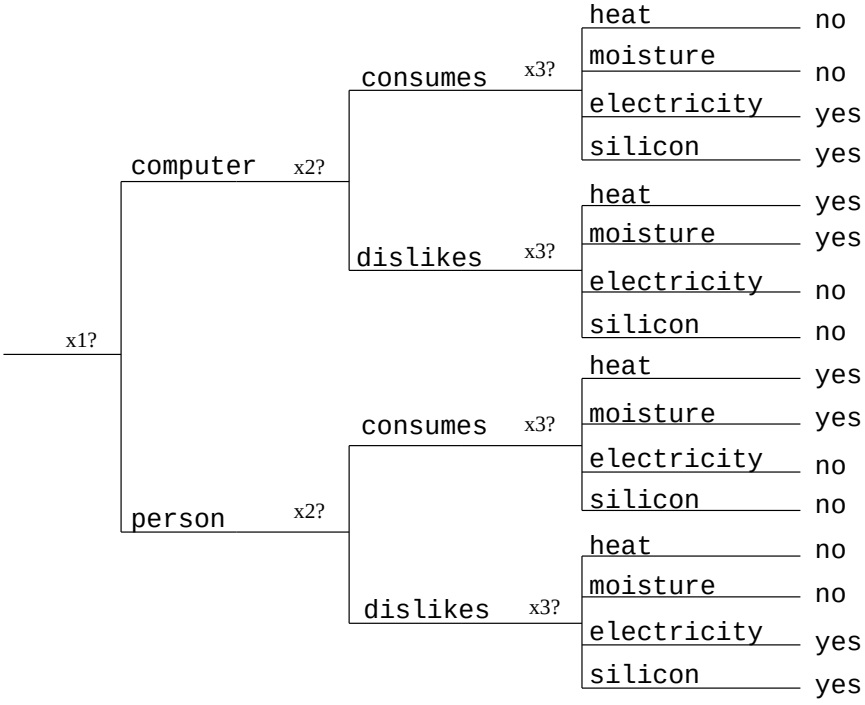

Figure 1: Decision tree produced by ID3 on consumer problem.

2.3 Acquiring complete, neutral mappings with backpropagation

Of course not all learning algorithms rely solely on the exploitation of statistical effects. Thus, not all algorithms produce worst-case performance on neutral mappings. The backpropagation algorithm [6] for example, had been widely touted as having the ability to exploit structural and/or relational properties of the target mapping [7]. However, even with backpropagation, the damaging effects of neutrality can be readily discerned.

These are demonstrated in the algorithm's *generalization* performance on neutral mappings. Typically, when testing backpropagation on neutral mappings (e.g., parity problems), researchers have included

the *entire* mapping in the training data, and have thus not tested the algorithm's ability to generalise to unseen cases.[3] However, if we test backpropagation's ability to generalise to one unseen case in, say, the 4-bit parity mapping (i.e., we present 15 of the 16 cases as training data, and test generalization on the one remaining case), then the results are unambiguously poor.

In an exhaustive empirical analysis, backpropagation was tested for its ability to generalise to one, randomly selected unseen case in the 4-bit parity mapping. In this analysis a standard, two-layer, (strictly) feed-forward network was used with the number of hidden units being varied between 3 and 80. Data were collected for 20 successful runs (i.e., achievement of negligible error on the training data) with each architecture. The learning rate was 0.2 and the momentum value was 0.9.

The results are summarised in Figure 2. This shows the post-training mean error for seens and unseens averaged over the 20 successful training runs which were performed in each architecture. The basic error value used here is simply the average difference between the target output and actual output produced. The graph shows negligible mean error for seen cases due to the fact that data were only

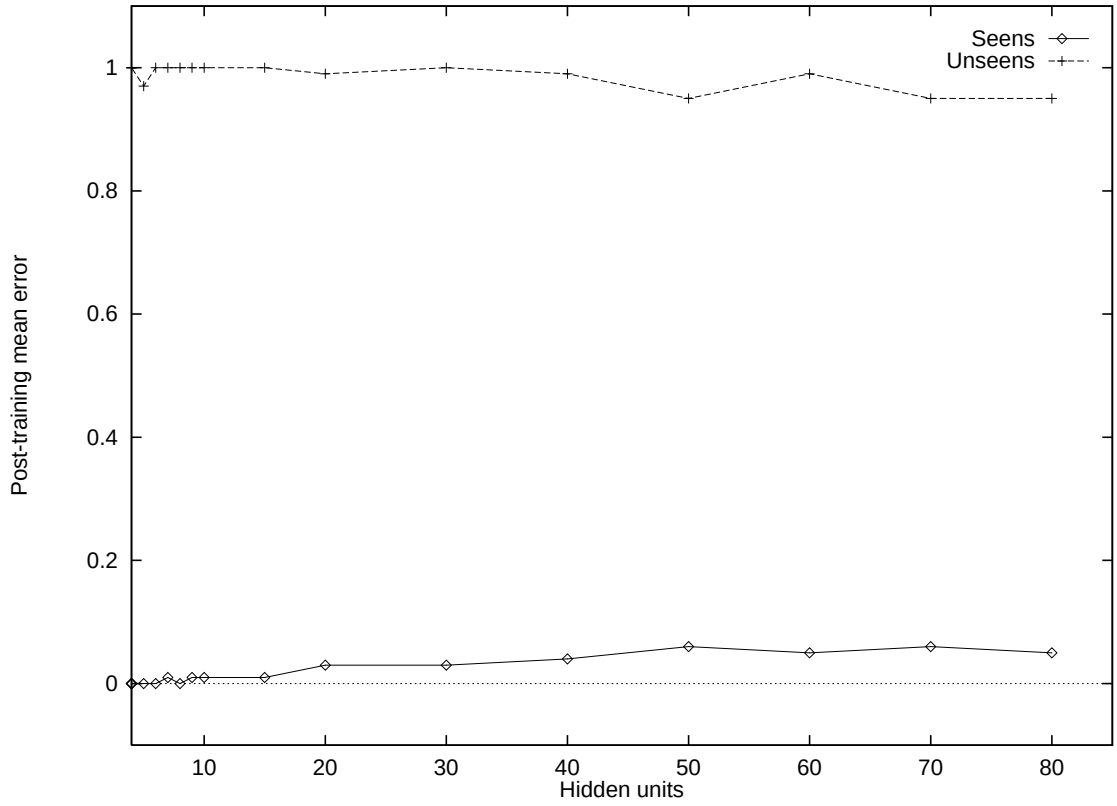

Figure 2: Post-training mean-error curves for parity generalization.

recorded for successful runs. More interestingly, it shows that the mean error on the unseen case is very

[3] Why this has become the established convention is not entirely clear. It may be to do with an unduly weak appreciation of the fact that generalization is the key goal in learning. Alternatively, it may be due to the fact that attention has typically focussed on the acquisition of small parity mappings involving two or three inputs, in which the retention of even a single case seems absurd.

poor for all architectures used, i.e., no generalization is achieved. (The fact that the generalization here is significantly worse than chance is explained below.) For purposes of comparison we carried out an identical analysis of backpropagation's generalization performance on the consumer problem (holding back one case as an unseen) and obtained qualitatively similar results, see Figure 3.

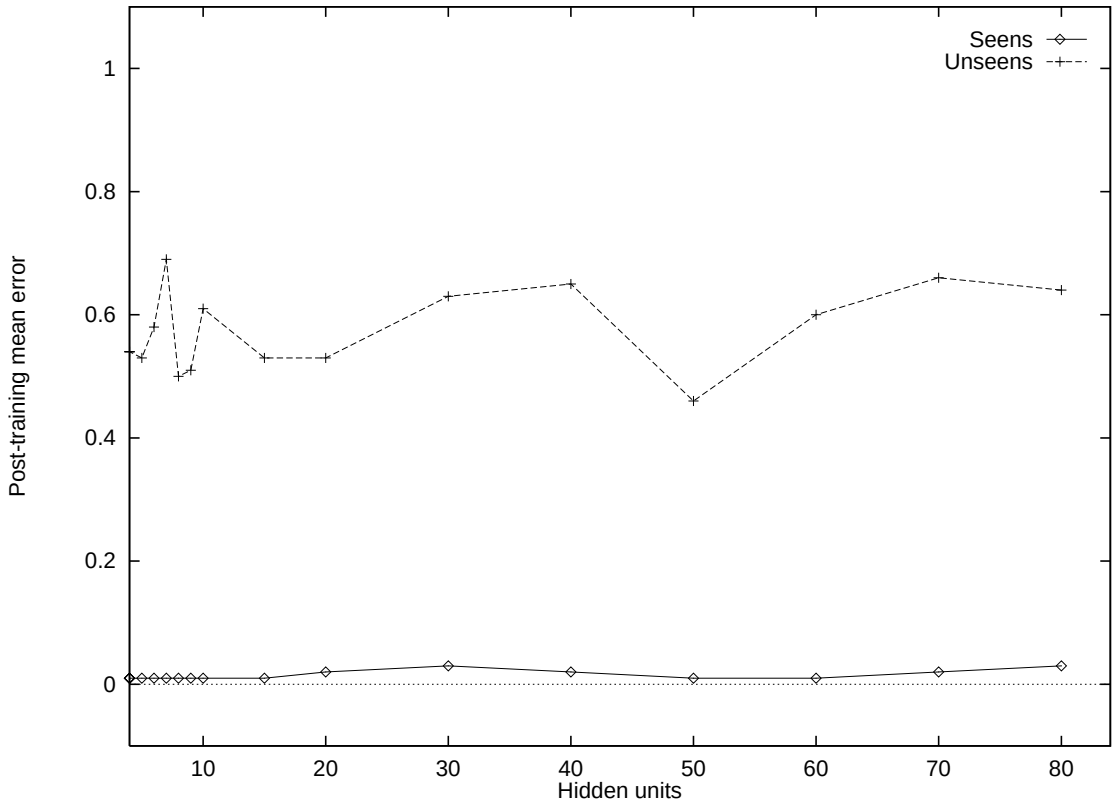

Figure 3: Post-training mean-error curves for consumer generalization.

2.4 Why algorithms fail to generalise over neutral mappings

The dynamics of the backpropagation process are complex. Explaining its generalization failure on neutral mappings is thus not straightforward. The simplest hypothesis may be that, despite its manifest success in the acquisition of small, complete parity mappings [8] backpropagation relies primarily on the exploitation of statistical effects and is thus unable to deal properly with neutral mappings. There are several arguments in favour of this idea.

First, the backpropagation learning algorithm is a generalization of the least-mean-squares algorithm [9] (and perceptron learning algorithm [10]) which is effectively an iterative method for deriving statistical input/output correlations. Thus the backpropagation learning method is rooted in a method for exploiting statistical effects. Second, the generalization performance observed in the 4-bit parity tests tended to be much *worse* than chance. This result is explained if the algorithm is primarily relying

on statistical effects since the effect that is created when we delete one case from a parity mapping is a correlation between input cases one Hamming unit away from the deleted case and the complement of the output for those cases (i.e., the 'wrong' output). Thus if the algorithm exploits input/output correlations then it will tend to always generalise *incorrectly* from the minimally incomplete parity mapping. The fact that it does do so tends to confirm the hypothesis that backpropagation primarily exploits statistical correlations.

3 Relational problems are approximately neutral

It is sometimes argued that parity mappings are artificial, mathematical constructs and that we should therefore not be too concerned if we find that our learning methods fail to generalise over them. Parity mappings are hard to generalise because they are statistically neutral. But neutrality, or approximate neutrality is actually a common property of challenging learning problems. In fact it should be obvious that any learning problem with a complex, *relational* input/output rule (i.e., a rule which tests for a relationship among the inputs) will have an approximately nearly neutral training set.

If the input/output rule is relational then we do not expect to see any associations between specific input values and specific output values showing up in the training set. There is an association; but it involves a relationship among the inputs. Thus, we should expect that any relational learning problem will have a neutral target mapping. Unfortunately, the truth of the matter is less clear-cut. Relational rules, in fact, do not guarantee neutrality. The way in which a particular relational rule is encoded in a set of input/output examples can create 'incidental' statistical effects. The training set shown in Figure 4 illustrates this.

x_1	x_2		y_1
0	1	$\Longrightarrow$	1
4	1	$\Longrightarrow$	3
2	0	$\Longrightarrow$	2
2	2	$\Longrightarrow$	0
2	4	$\Longrightarrow$	2
4	0	$\Longrightarrow$	4
1	3	$\Longrightarrow$	2
0	0	$\Longrightarrow$	0
2	1	$\Longrightarrow$	1
3	3	$\Longrightarrow$	0
3	2	$\Longrightarrow$	1
2	3	$\Longrightarrow$	1
1	0	$\Longrightarrow$	1

The input/output rule here is that the output is equal to the absolute difference between the input values. The rule is clearly relational and thus we might expect that the training set will be statistically neutral. However, the conditional output probabilities shown in Table 3 reveals the existence of many values which deviate markedly from the chance-level (see, for instance, the value of $P(y_1 = 1|x_2 = 1)$).

We see, then, that the way in which a relational rule is embodied in a training set can produce incidental statistical effects, i.e., spurious associations. Even where we known that the relational rule we are dealing with yields a fully neutral target mapping, we still cannot be sure that a training set derived

C	$P(y_1 = 4\|C)$	$P(y_1 = 0\|C)$	$P(y_1 = 2\|C)$	$P(y_1 = 3\|C)$	$P(y_1 = 1\|C)$
	0.08	0.23	0.23	0.08	0.38
$x_1 = 2$	0.0	0.2	0.4	0.0	0.4
$x_2 = 0$	0.25	0.25	0.25	0.0	0.25
$x_2 = 3$	0.0	0.33	0.33	0.0	0.33
$x_2 = 1$	0.0	0.0	0.0	0.33	0.67
$x_1 = 4$	0.5	0.0	0.0	0.5	0.0
$x_1 = 3$	0.0	0.5	0.0	0.0	0.5
$x_2 = 2$	0.0	0.5	0.0	0.0	0.5
$x_1 = 1$	0.0	0.0	0.5	0.0	0.5
$x_1 = 0$	0.0	0.5	0.0	0.0	0.5
$x_2 = 4$	0.0	0.0	1.0	0.0	0.0

Table 3: Conditional output probabilities from differencing problem.

from the mapping will be neutral. In fact, if the training set is a proper subset of the target mapping then it is guaranteed *not* to be neutral.

We can illustrate this using the consumer problem. The full target mapping for this problem is equivalent to a modulus addition problem (which satisfies the relevant cardinality constraints) so we known that it is necessarily statistically neutral. However, when we derive a training set by selecting a subset of the cases from the target mapping we necessarily introduce incidental effects. Let us say we split the problem up into the following training set

x_2	x_2	x_3		y_1
person	consumes	heat	$\Longrightarrow$	yes
person	consumes	electricity	$\Longrightarrow$	no
person	consumes	moisture	$\Longrightarrow$	yes
person	consumes	silicon	$\Longrightarrow$	no
person	dislikes	heat	$\Longrightarrow$	no
person	dislikes	electricity	$\Longrightarrow$	yes
computer	consumes	heat	$\Longrightarrow$	no
computer	consumes	electricity	$\Longrightarrow$	yes
computer	consumes	moisture	$\Longrightarrow$	no
computer	consumes	silicon	$\Longrightarrow$	yes
computer	dislikes	moisture	$\Longrightarrow$	yes
computer	dislikes	silicon	$\Longrightarrow$	no

with the following held back as test cases.

x_1	x_2	x_3		y_1
person	dislikes	moisture	$\Longrightarrow$	no
person	dislikes	silicon	$\Longrightarrow$	yes
computer	dislikes	heat	$\Longrightarrow$	yes
computer	dislikes	electricity	$\Longrightarrow$	no

The table of conditional probabilities derived from the training set now includes several non-chance

values (see Table 4).

C	$P(y_1 = \text{no}\|C)$	$P(y_1 = \text{yes}\|C)$
	0.5	0.5
x_2 =consumes	0.5	0.5
x_1 =person	0.5	0.5
x_1 =computer	0.5	0.5
x_2 =dislikes	0.5	0.5
x_3 =electricity	0.33	0.67
x_3 =heat	0.67	0.33
x_3 =silicon	0.67	0.33
x_3 =moisture	0.33	0.67

Table 4: Conditional output probabilities from partial consumer mapping.

Thus, relational rules may produce training sets showing at least two types of incidental statistical effect. However, there is a distinction to be made between incidental effects which are created by the encoding procedure and incidental effects which are created through the extraction of some of the cases. The latter form of incidental effects exist solely within the residual cases (i.e., the seen cases). They do not exist — except by chance — within the deleted cases (the unseen cases). Thus, incidental effects created this way do no support generalization. Conversely, incidental effects which are created as artefacts of the original encoding of the problem do exist throughout the target mapping and thus do support generalization.

The importance of the distinction can be demonstrated by running ID3 on the training set for the consumer problem shown above. ID3 is able to exploit the incidental effects in the production of a non-degenerate decision tree, see Figure 4. However, because the effects are non-generalising, the generalization performance turns out to be as bad as it could be. The decision tree actually produces the wrong output in all four cases (i.e., it yields 100% generalization error).

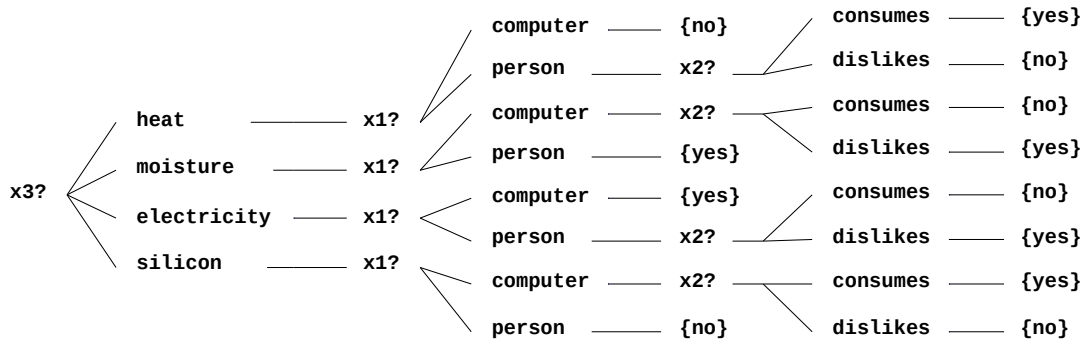

Figure 4: Decision tree produced by ID3 from partial consumer mapping.

3.1 Sparse codings amplify incidental effects

The fact that generalising incidental effects can be created through the encoding of the underlying, relational rule means that we can sometimes turn a 'hard' learning problem into an 'easier' problem simply by applying an encoding which maximizes the strength and range of generalising incidental effects. A simple approach involves using a sparse coding in which each input variable records the presence or absence of one *value* from the original encoding. The effect of this is to re-instantiate the input information in a higher-dimensional input space based on purely binary (i.e., boolean) dimensions. In such a space, the number of ways of achieving a given partitioning of points using linear discriminations is typically greatly increased. The performance of algorithms (such as backpropagation) which rely on the creation of such discriminations is thus enhanced.

Backpropagation, in fact, reliably succeeds in generalising to a single unseen case in the consumer problem, provided that a sparse coding of the problem is used. Moreover, it can do so utilising just two hidden units. The fact that it can do this, even though it remains unable to generalise to an unseen case in a sparse coding of a 4-bit parity problem, is attributable to the fact that one of the input variables in the consumer problem takes *four* values. The sparse coding of the problem expresses the value of this variable in terms of four, binary variables. The dimensionality of the original variable is thus quadrupled rather than doubled.

3.2 Backpropagation and the MONKS2 problem

The analysis of backpropagation's performance on the sparse-coded consumer problem may help to explain certain aspects of the performance results reported by Thrun et al. for the MONKS2 problem [11]. In this problem, the input/output rule is that 'exactly two of the six [input] attributes have their *first* value'. This problem is 'similar to parity problems. It combines different attributes in a way which makes it complicated to describe in DNF or CNF' [11]. The task analysis prompts us to restate this in terms of the statistical/relational distinction. The fact that the input/output rule refers to a relational property of the input values means that we should not expect to see correlations between specific input values and specific outputs and thus should not expect the problem to be solvable through exploitation of statistical effects, such as might be done via a search for a simple DNF definition.[4]

Backpropagation's performance on parity generalisation leads us to expect that the algorithm should perform badly on the MONKS2 problem. But in fact the generalization performance reported by Thrun is 100% good. To understand this apparent paradox we need to bear in mind that that in applying backpropagation to this problem, Thrun recoded the problem into a sparse form, thereby inflating the input space from its original, six-dimensional form to a 17-dimensional, binary-valued space. Even algorithms such as ID3, which use axis-parallel, linear discriminations are assisted by this sort of recoding. In fact, if we apply the standard ID3 algorithm to MONKS2 represented in a sparse binary coding, the generalization performance of the algorithm on the fixed unseen cases jumps by nearly 10% from the level of 67.9% reported in [11]. This case provides us with a useful reminder of the fact that we need to exercise extreme caution in interpreting performance claims in machine learning studies.

[4] Note that a simple DNF definition can be produced if and only if there exist a number of conditional output probabilities with a value of 1.

4 Summary and discussion

The general aim of the paper has been to show that the parity problem is more than just a challenging benchmark problem for empirical learning methods. Once its grounding in statistical neutrality is understood, it reveals itself as a conceptual landmark in our understanding of hard learning. I have shown that parity problems are within the class of statistically neutral problems and that any problem which is interpretable as a modulus-addition problem (with the restriction that the input value cardinalities must be equal to, or a multiple of the output cardinality) is also within the class. In other words, I have shown that the class of neutral problems is not restricted to the domain of parity problems.

I have presented empirical data demonstrating how some learning methods perform on neutral problems. In particular, I presented data relating to ID3, which can be shown analytically to rely *solely* on the exploitation of statistical effects and thus to be incapable, in principle, of dealing with neutral problems. I demonstrated that, if we present the complete target mapping for the (statistically neutral) consumer problem, ID3 produces what is in effect a lookup table and that when we present a particular partial mapping with four cases held back as unseens, ID3 produces a tree which yields 100% *in*correct generalization.

I also presented data relating to backpropagation which, arguably, is biased towards the exploitation of statistical effects. I showed, that the generalization performance of backpropagation on 4-bit parity problems (with one unseen) and on the consumer problem (with one unseen) was, in an extensive empirical study, no better than chance.

In the latter part of the paper I have shown how relational problems — which we might expect *a priori* to be statistically neutral — typically exhibit 'incidental' statistical effects. I have shown how these effects are produced in two different ways. First they may be produced by the way in which the problem is *encoded*. Second they may be produced through the extraction of elements of the mapping for use as unseens. The distinction is crucial since it is only the former type of effect which permeates the entire mapping and thus supports generalization.

I have noted that the generalization performance achieved by some methods can be improved when a neutral problem is recoded into a form which maximizes the prevalence and intensity of the right sort of incidental effect. And I have shown that this can be straightforwardly achieved by giving a problem a *sparse* coding. It was noted, in passing, that when the consumer problem is put into a sparse coding, backpropagation is able to generalise correctly to a single unseen using a network of only two hidden units.

To emphasise the point about sparse coding I presented some data relating to the MONKS2 problem which featured in the recent international learning competition [11]. I noted that this problem is inherently relational and thus statistically neutral (modulo incidental effects), and that the reported level of generalization achieved by backpropagation (100% correct) therefore comes as a surprise. However, the generalization performance can be explained by noting that the researcher who carried out the tests with backpropagation recoded the problem in a sparse form, thus increasing the dimensionality of the input space from six to 17 dimensions. As I have said, this recoding is of benefit to any method which relies — as backpropagation does — on the utilisation of spatial discriminations in the input space. In fact, the recoding turns out to be of benefit even to methods which are restricted to using axis-parallel discriminations. As an illustration of this, I noted that the generalization performance of the standard ID3 algorithm [3] turns out to be improve its generalization performance on MONKS2 by nearly 10% as a result of using a sparse coding. The generalization performance increases from from 67.9% [11] to 77%.

5 Acknowledgements

Many of the ideas in this paper were developed in collaboration with Jim Stone.

References

[1] 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.

[2] Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). *Classification and Regression Trees.* Wadsworth.

[3] Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1* (pp. 81-106).

[4] Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* San Mateo, California: Morgan Kaufmann.

[5] Wnek, J. and Michalski, R. (1994). Discovering representation space transformations for learning concept descriptions combining DNF and m-of-n rules. *Proceedings of ML-COLT'94.*

[6] Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning representations by back-propagating errors. *Nature, 323* (pp. 533-6).

[7] Beale, R. and Jackson, T. (1990). *Neural Computing: An Introduction.* Adam Hilger.

[8] Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning internal representations by error propagation. 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. 318-362). Cambridge, Mass.: MIT Press.

[9] Thornton, C. (1992). *Techniques in Computational Learning: An Introduction.* London: Chapman & Hall.

[10] Minsky, M. and Papert, S. (1988). *Perceptrons: An Introduction to Computational Geometry* (expanded edn). Cambridge, Mass.: MIT Press.

[11] Thrun, S., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fisher, D., Fahlman, S., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J. and Zhang, J. (1991). The MONK's problems - a performance comparison of different learning algorithms. CMU-CS-91-197, School of Computer Science, Carnegie-Mellon University.