Trading Spaces: Computation, Representation and the Limits of Learning*

Andy Clark and Chris Thornton†

Cognitive and Computing Sciences

University of Sussex

Brighton

BN1 9QH

U.K.

E-mail: Andy.Clark@cogs.sussex.ac.uk

Chris.Thornton@cogs.sussex.ac.uk

June 15, 1993

*Research on this paper was partly supported by a Senior Research Leave fellowship granted by the Joint Council (SERC/MRC/ESRC) Cognitive Science Human Computer Interaction Initiative to one of the authors (Clark). Thanks to the Initiative for that support.

Abstract

The increasing level of attention being given to incremental learning is, we argue, fully justified. Although some view the process as something akin to an 'efficiency hack', we argue that it is, in fact, a key cognitive process. The argument is based on a statistical observation. Learning, for most purposes, is all about acquiring the ability to generate appropriate outputs from given inputs, i.e., it is all about acquiring the ability to predict — or, in general, give a probability to — specific bindings of output variables. Such predictions can be justified in two quite different ways by available training data (i.e., input/output examples). They can be justified *directly*, in virtue of being in 1-to-1 correspondence with probabilities (i.e., frequencies) directly observed in the training data. Or they can be justified *indirectly*, in virtue of being in 1-to-1 correspondence with probabilities observed in some re-coding of the data. Thus, where learning is driven by supplied training data, it must exploit some combination of these two types of justification.

Since the space of *indirectly* observed probabilities is grounded in the space of possible re-codings (i.e., applicable Turing machines), searching through it is intractable. But we should not infer from this that learning does not (and cannot) make use of indirectly observed probabilities. As we show, learning problems whose solution necessarily entails exploiting such probabilities (or *type-2* problems, as we call them) seem to be the norm in many realistic learning scenarios. We are thus left in need of an

[†]The order of names is arbitrary.

explanation as to how such problems can be solved. Incremental learning, it seems to us, provides the answer.

We argue that what is accomplished in incremental learning is the systematic breaking-down of type-2 learning problems into problems that can be solved by exploiting directly *observed* probabilities (i.e., frequencies), or *type-1* problems as we call them. With each decomposition, a new, originally implicit, property of the input data is 'exposed' and a new virtual input variable thus created. Gradually the computational cost of the original (type-2) problem is traded-off against representational structure. The final solution is a representational structure which circumvents the intractably long computation entailed in a non-incremental approach.

The body of the paper provides the fine detail of this account. It also shows how the process of incremental learning provides a means of gleaning maximal benefit from any (perhaps initially fortuitously achieved) re-coding of the input data. A variety of techniques currently being studied as aids to connectionist learning (Elman (1991), Jacobs, Jordan and Barto (1991), Jacobs, Jordan, Nowlan and Hinton (1991)) are shown to fulfil just such a role. Our central claim is that incremental learning is not just one more item in the cognizer's toolkit: instead, it is the a priori essential mainstay of any learning device which needs to solve realistic learning problems in realistic time scales. The trick is always the same; to maximize the role of any achieved re-coding (representation) so as to minimize the space of subsequent search. The most distinctive features of human cognition — language and culture — may themselves be viewed

as adaptations enabling this representation/computation trade-off to be pursued on an even grander scale.

Keywords

Learning, connectionism, statistics

Introduction: The Limits of Learning

The potency of connectionist learning techniques for extracting statistical regularities from bodies of training data is widely appreciated. Less widely appreciated, however, is the fact that such statistical regularities come in two quite different forms, only one of which is 'transparent' to unbiased algorithmic processes. Statistical regularities may be described in different ways. However, we can always construe a regularity in terms of probability effects, i.e., in terms of some set of significant probabilities[1] applying to, and perhaps conditional on, aspects of the training data. But such probabilities can be derived from (i.e., justified in terms of) training data in two different ways. They can be justified *directly*, in virtue of being in 1-to-1 correspondence with probabilities directly observed in the training data. Or they can be justified *indirectly*, in virtue of being in 1-to-1 correspondence with probabilities observed in some re-coding of

[1] We use this phrase to denote probabilities not attributable to chance effects.

the data. Thus any process for regularity-extraction must attend to two different sources. The search space for regularities of the directly-observed variety is large but not, in general, intractably large. The search space for regularities of the indirectly-observed variety, on the other hand, is *infinitely* large, since it is grounded in the space of possible re-codings of the input data, i.e. the space of applicable Turing machines. Thus, on complexity grounds, we classify one form of regularity as being relatively 'transparent' and the other as 'opaque'.

When confronted with problems that involve extracting regularities of the opaque variety, connectionist learning algorithms such as standard backpropagation are unreliable. Parity generalization problems provide the obvious example. If one presents standard backpropagation (Rumelhart, Hinton and Williams, 1986b) with a complete parity mapping — allowing the algorithm to make use of a sufficiently rich internal architecture — then the learning will almost certainly succeed in achieving perfect performance. However, if one presents only a plausible training set (i.e., some large proportion of the complete mapping) then success is not assured. For example, if one presents 12 of the possible 16 cases from the 3-bit parity mapping, backpropagation typically learns those 12 cases rather rapidly and rather well (see Section 1.4 below). But it usually learns them in such a way that the four unseen cases are not handled correctly; i.e. it fails to generalize to the full mapping. The reason for this is fairly straightforward. With parity mappings, the significant probability effects are tied not to input values but to relations between them. (The conditional probabilities

for all combinations of explicit values are necessarily at chance levels.) This means (for reasons that are explained in detail below) that solving the learning problem involves deriving an appropriate re-coding; i.e., it means extracting regularity of the opaque variety. The problem cannot be solved by extracting the transparent form of regularity — for the simple reason that it is not present.

The fact that connectionist learning algorithms (such as backpropagation) may fail on parity generalization problems shows that their capacities for extracting regularities of the less accessible form are limited to some degree. (Note that the problem here is *all* to do with the ability of such systems to learn *generalizable* solutions: the fact that a non-generalizable solution can be *learned*, or that a generalizable solution can be *represented* is not what's at issue.) Possible responses to the problem are (1) to blame connectionist learning algorithms and hope to find something better or (2) to deny that most of the problems solved by the human brain require the sort of re-coding that opaque regularity-extraction may require (i.e., to claim that parity-generalization is a pathological case). We reject (1) on the grounds that the problem is not, as far as we can tell, a mere artifact of connectionist learning; instead it is a problem which will arise for any learning algorithm which operates without specific inbuilt knowledge about a target domain. And we reject (2) because the problem of re-coding seems to arise even in relation to apparently simple 'robotics-style' tasks (i.e. in manifestly non-pathological cases — see Section 1.6 below).

The *correct* response to the problem, we argue, is to reconsider the difficulties in the light of an accumulating body of work, e.g., (Elman, 1991) which highlights the role of representational trajectories in connectionist learning. A representational trajectory is just a connected path through a temporally or spatially extended sequence of different learning tasks. By following such a trajectory a single intractable problem can be reduced to an incremental sequence of different and individually tractable problems. The solution to each sub-problem yields, in effect, a re-coded representational base. Such re-codings progressively reduce the complexity of the task of learning the target mapping until it becomes tractable. Otherwise put, each re-coding allows us to avoid a quantity of computational search by providing a more suggestive set of 'virtual inputs'. We thus trade computation off against representation.

While this strategy is not unfamiliar (it is just a kind of 'temporal homuncularism' and has been the subject of a well-known investigation by Elman, 1991), we believe that it must constitute not just a clever solution to one or two specific problems but a fundamental feature of the cognitive process. Without such a strategy, even apparently simple learning problems may be effectively unsolvable.

Having emphasized the foundational role which the understanding of representational trajectories must play in cognitive science, we go on to argue that a wide variety of superficially distinct ploys and mechanisms can be fruitfully un-

derstood in these terms. Such ploys and mechanisms range from simple evolved filters and feature-detectors all the way to complex cases of analogical reasoning. The goal, in every case, is to systematically re-configure a body of input data so that computationally primitive learning routines can find some target mapping, i.e. to trade representation against computation. A topical moral is that it is virtually impossible, in such cases, to avoid talk of internal representations as the products of each stage of re-configuring. Pace the more radical apologists of Artificial Life, even quite simple robotics tasks seem to demand analysis in terms of achieved internal representations.

The paper ends with a look at two major worries which the stress on representational trajectories raises. First, how efficiently can such trajectories be discovered and maintained? And second, does the intractability of learning in cases where no such viable trajectory exists imply some major limitation on the class of knowable truths about the universe? Once we allow that the answer to the latter question is 'Yes', the former reveals itself as perhaps less problematic than it at first appears.

The blow-by-blow strategy is as follows. We begin (section 1) by distinguishing the two kinds of statistical regularity. The simplest kind, which we call 'observed' regularities, consist in pronounced frequency effects (i.e., probabilities) in the original data, i.e., features of a full frequency matrix (see below) that are not attributable to chance. The more complex kind, which we call 'derived'

regularities, consist in frequency effects found via some re-coding of the original data. We argue that learning problems which are based on this more complex kind of regularity are precisely those on which empirical learning algorithms typically produce poor performance.

We go on (section 2) to show how a sample problem (that manifestly does involve the extraction of the inaccessible form of regularity) can be solved in an incremental manner, i.e., be continually breaking the learning task 'down'. This idea (already familiar from Elman's (1991) discussion) can now be understood in terms of a fully general distinction (between 'observed' and 'derived' regularities) firmly grounded in the basic statistical nature of a target mapping.

Such a statistical lens quickly reveals the unexpected ubiquity of the problem: the learning tasks faced by even simple organisms are riddled with mappings involving derived regularities. If (as it appears) such mappings force us to exploit incremental learning strategies, such learning must pervade biological cognition. In section 3 we go on to show how analogy and a surprisingly large range of other ploys and mechanisms can be understood as implementations of the incremental re-coding strategy. Even so, it may still be questioned whether there must, in addition, operate some as-yet-undiscovered learning algorithm exquisitely tailored to the discovery of useful learning trajectories. Section 5 discusses this question and notes one possible source of misleading intuitions. Section 6 is a brief conclusion.

1 Statistical properties of training sets

Let us begin by making some general observations about the statistical properties of training sets (i.e. sets of training examples). Consider the training set shown below. This is based on two input variables (x1 and x2) and one output variable (x3). There are six training pairs in all. The pairs are laid out with one pair per line. An arrow separates the 'input vector' of the pair from the 'output vector'. 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       x3


1   2  -->   1

2   2  -->   0

3   2  -->   1

3   1  -->   0

2   1  -->   1

1   1  -->   0
```

In this training set we can observe a number of instantiation n-tuples, henceforth called *cases*. First-order cases are instantiation 1-tuples. An example is <x1=3>, i.e., the binding of the variable x1 to the value 3, as exhibited in the third pair. Other examples include <x3=0> and <x2=2>. Second-order cases are

instantiation 2-tuples. An example is <x1=3, x2=1>. This case is observed in the fourth line of the training set. A second-order case from the second line of the training set is <x3=0, x1=2>. Since there are only three variables in all there is exactly one third-order case for each member of the training set.

Given a particular case, we can compute the frequency (i.e., probability) with which it appears in the training set. The frequencies for all first and second-order cases in the training data above are shown in Table 1. Note that the frequencies for the third-order cases (i.e., the cases that specify values for all three variables) are degenerate. Assuming there is no duplication in the training data, each third-order case occurs exactly once. Thus its frequency is necessarily 1/n where n is the size of the training set.

1.1 Conditional frequencies

The frequencies shown in Table 1 are *unconditional* frequencies. We can also derive *conditional frequencies*.[2] These are frequencies that exist with respect to a particular constraint over variable instantiations. In Table 2 we see the frequencies for particular instantiations of the output variable (x3) given possible constraints on other variables. The column headed 'Fr' shows the absolute frequencies for the constraints. The column headed 'Fr: x3=0' shows the conditional frequency with which x3=0 when the relevant constraint applies. The

[2] These can be construed as Bayesian probabilities (Duda and Hart, 1973).

Case	Fr
	1
x2=2	0.5
x2=1	0.5
x3=1	0.5
x3=0	0.5
x1=3	0.33
x1=2	0.33
x1=1	0.33
x2=2, x3=1	0.33
x2=1, x3=0	0.33
x1=3, x2=2	0.17
x1=2, x2=2	0.17
x1=1, x2=2	0.17
x1=3, x2=1	0.17
x1=3, x3=1	0.17
x1=2, x2=1	0.17
x1=2, x3=1	0.17
x2=1, x3=1	0.17
x1=3, x3=0	0.17
x1=1, x3=1	0.17

Table 1:

column headed 'Fr: x3=1' does the same thing for the case x3=1.

By the argument used previously, the 2nd-order conditional frequencies here are of no interest since there is necessarily exactly one occurrence of each 2nd-order case of the constrained variables.

1.2 Type-1 versus type-2 frequencies

A clear distinction must be made between cases (such as those considered above) that can be observed *directly* in the training data, and cases that can only be observed *indirectly*. For our purposes, a case can be observed indirectly if it can be observed directly in some systematic *re-coding* of the original data. What this means is that an instantiation n-tuple that occurs in some re-coding of the original data, is considered to be a case that is 'observed indirectly' in the *original* data. We will call frequencies for directly observed cases type-1 frequencies. We will call frequencies for indirectly observed cases type-2 or *derived* frequencies.

The difference between the two types of frequency can be illustrated by re-coding our original training set. Imagine that we re-code the inputs (from above) by substituting — in each training pair — the original input variables with a single variable whose value is just the difference between the original variables. This gives us a set of derived pairs as shown in Figure 1 (the value of x4 here is the difference between the values of x1 and x2). The frequencies we *directly* observe

Constraint	Fr	Fr: x3=0	Fr: x3=1
	1	0.5	0.5
x2=2	0.5	0.33	0.67
x2=1	0.5	0.67	0.33
x1=3	0.33	0.5	0.5
x1=2	0.33	0.5	0.5
x1=1	0.33	0.5	0.5

Table 2:

```
Original pairs                Derived pairs (x4 = |x1-x2|)

x1  x2      x3                x4      x3

1   2  -->  1                 1  -->  1
2   2  -->  0                 0  -->  0
3   2  -->  1                 1  -->  1
3   1  -->  0                 2  -->  0
2   1  -->  1                 1  -->  1
1   1  -->  0                 0  -->  0
```

Figure 1: Recoding of training set.

in this derived training set are type-2 (derived) frequencies with respect to the original training data. However, they are still frequencies. Thus we can derive tables of conditional and unconditional frequency statistics in the usual way. The unconditional frequencies for the derived training set are shown in Table 3.

The conditional frequencies for instantiations of x3 given instantiations of x4 are shown in Table 4.

1.3 Classes of regularity

For the purposes of learning the statistical regularities in the training data are of central importance. But our perception of these is grounded in our assumptions concerning *chance*. The 'chance-level' for a frequency is simply the frequency (or in general range of frequencies) that we expect to see given purely random effects. Where we observe a frequency that diverges markedly from what we assume to be its chance level(s) we necessarily believe that non-random processes are at work. For present purposes we will assume that the chance-level for an instantiation of a random variable capable of taking n values is precisely 1/n, no more and no less. This is a 'least-conservative' approach since we are assuming that any frequency that diverges from 1/n, no matter by how small an amount, is to count as a non-chance-level frequency. (Since we aim to make a purely theoretical point this lack of conservatism is of no consequence.)

Constraint	Fr
	1
x3=0	0.5
x3=1	0.5
x4=1	0.5
x4=0	0.33
x4=2	0.17
x4=1 + x3=1	0.5
x4=0 + x3=0	0.33
x4=2 + x3=0	0.17

Table 3:

Constraint	Fr	Fr: x3=0	Fr: x3=1
	1	0.5	0.5
x4=0	0.33	1.0	0.0
x4=2	0.17	1.0	0.0
x4=1	0.5	0.0	1.0

Table 4:

A non-chance-level frequency is caused (by definition) by a non-random effect and thus forms a statistical regularity. Since we have distinguished two types of frequency effect, we can distinguish two types of regularity.

- **Type-1 regularity**: divergence from chance-levels in type-1 frequencies.
- **Type-2 regularity**: divergence from chance-levels in type-2 frequencies.

To place this in a concrete setting, consider the example training sets shown above. The output variable x3 is a binary variable. Thus the frequency for either of its two possible instantiations is exactly 0.5. When we look at the type-1 conditional frequencies for the training data we see that most of the values are at or close to their chance-level of 0.5. However, when we derive the relevant type-2 conditional frequencies (after re-coding in the suggested way) we obtain a frequency table in which *every* value diverges *maximally* from its chance level. Intuitively, then, the training set can be classified as exhibiting more type-2 regularity than type-1.[3]

The frequency effects brought to light by the re-coding translate naturally into a completely general input/output rule. The table of type-2 conditional frequencies makes it obvious that x3=1 if and only if x4=1. From this we trivially obtain the input/output rule 'x3=1 if x4=1; otherwise x3=0.' Thus we see how the re-coding effectively brings the regularity underlying the training set to the

[3] The question of how type-1 regularity should be measured formally is addressed below.

surface. Once this has happened it is a straightforward matter for a learning algorithm to exploit it. Recognizing the strong, mutual interdependence between learning and regularity leads us to distinguish three classes of learning problem.

- **Pure type-1 learning problems**: problems that involve exploiting type-1 regularities only,
- **Pure type-2 learning problems**: problems that involve exploiting type-2 regularities only,[4] and
- **Hybrid problems**: problems that involve exploiting some mixture of both types.

1.4 Parity problems are pure type-2

The distinction between type-1 and type-2 problems is nicely illustrated by the parity problems (cf. Rumelhart, Hinton and Williams, 1986a). Complete parity mappings show no type-1 regularity at all. Their observed frequencies are always *exactly* at their chance levels. (Hinton and Sejnowski, 1986) The input/output rule for a parity mapping is simply that the output should be 1 (or true) just

[4] A special case of the type-2 problem occurs when the relevant re-coding can be performed by deriving simple probability effects from *subsets* of the training data. The complexity of this variant exceeds that of the type-1 problem due to the additional cost of exploring possible partitions of the training data. However, it is not as great as that for the unrestricted type-2 case since there is no necessity to explore any part of Turing-machine space.

in case the input vector contains an odd number of 1s (or, in general, an odd number of odd values). The complete mapping for the third-order, binary-valued parity problem (i.e., 3-bit parity) is as follows.

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

Every single first and second-order conditional frequency for this mapping (for values of the output variable x4) is at its chance level of 0.5. And, in fact, the frequency statistics for parity mappings are *always* like this. If we are dealing with n-bit parity then the highest order, non-degenerate frequencies are the (n-1)th-order frequencies. Given binary variables we will necessarily find exactly two occurrences of each (n-1)th-order case in the training set, and these two cases will necessarily show a different value for the variable excluded from the case. Thus the conditional frequencies for the case in question will be

evenly distributed between the two output cases and the conditional frequencies for instantiations of the output variable will always be identical. If they are identical, they must be precisely at their chance level. Thus, parity problems are always pure type-2.[5]

1.5 Complexity implications

Distinguishing between type-1 and type-2 problems helps to shed light on the complexity implications of different learning scenarios. Type-2 regularities are non-chance frequencies for cases observed in some *re-coding* of the original data. Thus, points in the space of type-2 regularities correspond to possible data re-codings, i.e., possible computational devices capable of processing those original data. The space of possible type-1 regularities, on the other hand, is made up of the set of all frequencies (conditional and unconditional) for the problem. Suffice it to say that the former space is, in general, infinitely large, while the latter is small, relatively speaking. Thus, other things being equal, type-1 problems should be easier to solve than type-2 problems.

The *practical* consequences of this are hard to determine. It seems to be the case that so-called 'real world' machine learning problems are almost never pure type-2. Thus, they can usually be solved by techniques that do not resort to exploring possible data re-codings. Even learning problems that are intrinsically type-2

[5] Arguably, all problems that are pure type-2 are quasi-parity problems.

(i.e., which are constructed on the basis of an input/output rule that implicitly invokes a re-coding step) may well exhibit 'spurious' type-1 regularity.

The example training set used above illustrates this. The problem is 'intrinsically type-2' since the input/output rule used to construct the pairs assumes the re-coding step of converting the original input variables to their difference. And yet the type-1 frequencies show some marked, non-chance values (see the frequencies for the cases <x2=1> and <x2=2>). These would be straightforwardly exploited by processes that perform no re-coding whatsoever, e.g., learning algorithms such as the perceptron learning algorithm (Minsky and Papert, 1988) or Quinlan's ID3 (1986).

Even where intrinsically type-2 problems show very little spurious type-1 regularity they may still be solved by sophisticated learning algorithms such as backpropagation (Rumelhart, Hinton and Williams, 1986b), cascade-correlation (Fahlman and Lebiere, 1990) or copycat (Hofstadter, 1984).[6] It is, after all, well known that backpropagation can solve problems based on parity, symmetry or 'shift' relationships and that all these typically involve the algorithm deriving what can be thought of as an internal re-coding scheme.

However, we should not overestimate the generality of such methods. All of them introduce restrictive assumptions about the nature of the type-2 regularity to be

[6] This latter is not usually presented as a learning algorithm. However it can certainly be construed as such.

discovered. Backpropagation for example effectively assumes that the required re-coding can be expressed in terms of the user-fixed architecture of semi-linear transfer functions, and that it can be discovered by the gradient descent method embodied in the learning algorithm. If the assumption is invalid, the learning necessarily fails.

This may help to explain why backpropagation often fails to solve low-order parity problems when they are presented as generalization problems (i.e., when some cases are held back for testing purposes). The graph shown in Figure 2 was produced from an empirical survey that involved running standard backpropagation (Rumelhart, Hinton and Williams, 1986b) on 4-bit parity generalization problems (with four, randomly selected cases used as unseens) using a wide range of internal architectures. All the curves in the upper half of the graph are error profiles[7] for the testing set of four cases. All the curves in the lower half of the graph are error profiles for the training set. There are 32 pairs of curves in all although many of them are bunched together in two clumps at the far left of the graph. Rather obviously, generalization over the testing cases was never observed to improve much beyond the chance level in any of the runs recorded. But the point to note is that the training-set error profiles typically go to zero rather rapidly (usually within 1000 epochs). This tells us that the generalization

[7] The error measure is the average difference between actual and target activations. For these experiments we used standard learning parameters; i.e., a learning rate of 0.5 and a momentum of 0.9.

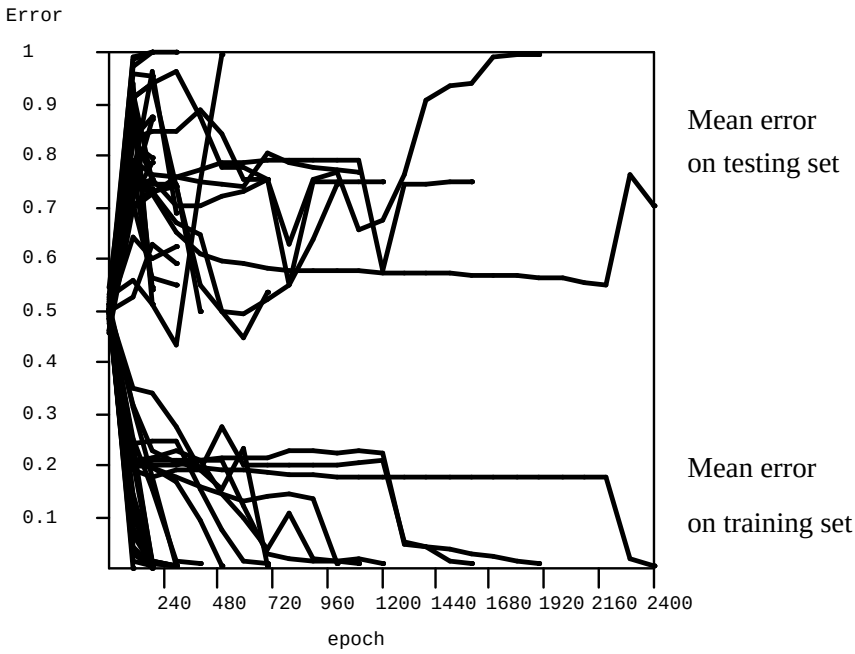

Figure 2: Generalization performance on 4-bit parity.

failure occurs in the context of perfectly 'successful' learning, i.e., perfect acquisition of the training cases. This is a particularly concrete sort of generalization failure since it cannot be overcome by increasing the amount of training or by changing parameters. Once a supervised algorithm has learned the training cases perfectly, generalization grinds to a halt. As far as the algorithm 'knows', it is *already* producing perfect performance.

1.6 Is parity a pathological case?

The parity problem is often thought of as a kind of pathological, unrealistic case. But it would be a mistake to conclude that all type-2 problems are therefore pathological and unrealistic. It is fairly easy to show that, in principle, any problem that is based on a relational input/output rule (i.e. a rule that implicitly takes account of relative rather than absolute input values) may create no type-1 effects whatsoever.

A (supervised) learning problem is always defined in terms of a target input/output mapping. In all reasonable problems, the mapping is based on an input/output 'rule' of some sort: it is the task of the learning to discover this rule and represent it in such a way as to enable unseen inputs to be mapped onto their correct outputs. In the simplest case, the rule refers (perhaps implicitly) to particular input-variable values. If it does so, we will expect to see correlations between particular input values and particular output values. In other words,

the rule will tend to have a 'representation' in the form of pronounced, type-1, frequency effects.

But of course the rule may not refer — even implicitly — to particular input values. It may refer to *relationships* between input values. The parity rule is an obvious example. The parity rule does not 'care' about explicit input values. It only cares whether there is a particular relationship among them. And in this case, where the rule only takes account of input-value relationships, it will *not* have the effect of producing type-1 correlations in the training set. The rule has nothing to do with explicit values, so how could it?

On purely a priori grounds, then, we will expect to see training sets based on 'relational' input/output rules exhibiting low-levels of type-1 regularity. The obvious rider to this is that we will necessarily expect to see such training sets exhibiting *high* levels of type-2 regularity. If the input/output rule is based on a relational effect, then any re-coding which effectively captures the value of the relationship in a single variable value (as our differencing encoding did above) will automatically produce a strong type-1 effect, and thus a strong type-2 effect with respect to the original data.

Now, we may be able to dismiss parity problems as a rare and pathological cases. But we certainly cannot thus dismiss *all* problems based on relational input/output rules. In practice such problems seem to arise with considerable frequency. 'Visual' learning problems, for example, involving pattern-recognition

in a visual field are *necessarily* of this type — unless of course the patterns in question are 'normalized' as to remove all rotational, translational and scale variation.[8]

Even very primitive robotics-style learning problems may turn out to involve learning relational rules. A case in point is the 'conditional-approach' behaviour investigated by (Thornton, forthcoming). In this behaviour an animat having a radially-arranged set of range-finding sensors must approach any small object appearing in its sensor field but stand clear of any large object. It turns out that the only way the animat can achieve this behaviour is by learning to calculate the ratio between the apparent width and apparent closeness of the object in the sensor field. Thus, the underlying input/output rule for the behaviour is grounded in a relational effect. The result (as per expectation) is that conventional learning algorithms produce rather poor generalization performance when used as training methods for this behaviour. But the fact that the behaviour examined is so manifestly primitive leads us to believe that the acquisition of relational input/output rules is something that will concern even very primitive cognitive mechanisms at the earliest stages of development.

[8] When attempting to recognize a pattern it is not the point coordinates that matter but the relations between them.

2 Incremental solutions

Given the likely ubiquity of learning problems involving relational (type-2) regularities how should we react to the ease with which natural systems appear to deal with them? One possibility is that there exists a more powerful, as-yet-undiscovered class of learning algorithms capable of performing type-2 search in an individual lifetime. Alternatively, we might conclude that nature's achievement is somehow to exploit forms of learning which involve only type-1 search in ways which somehow cumulatively lead to the solution of type-2 problems. Given our discussion of the statistical roots of the difficulty of type-2 search, we suspect that no conceivable individual learning algorithm will be able reliably to negotiate such spaces in biologically realistic time-spans. We will therefore investigate a version of the second response in which type-2 problems are reduced to incremental complexes of type-1 problems.

We can illustrate the basic idea using an example in which we imagine a type-2 learning problem being solved by a learner with rather limited computational abilities. The example is based on the training set shown below. This uses 21 variables and these are shown in the usual fashion starting with the first variable (called x1) on the far left and finishing with last (called x21) on the far right. (Only the integer parts of names for variables between x2 and x21 are shown). At the very end of each line there is a comment in square brackets.

```
x1 2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21
```

```
1  2  1  1  1  4  10 3  11 4  3  4  4  4  5  4  6  4  7  4  0 [three_of_a_kind beaten_by flush]
2  2  3  3  4  3  5  3  6  2  10 1  4  2  9  4  12 3  4  1  1 [straight beats pair]
12 2  12 1  7  2  11 2  2  3  4  3  4  3  4  1  4  4  10 2  0 [pair beaten_by four_of_a_kind]
3  4  3  2  3  1  12 3  8  4  11 1  11 4  10 2  10 3  1  2  1 [three_of_a_kind beats two_pairs]
7  2  6  2  4  3  3  3  10 3  10 4  10 4  10 3  4  2  3  1  0 [nothing beaten_by three_of_a_kind]
6  4  7  4  8  4  9  4  10 4  4  4  4  4  11 3  8  3  3  3  1 [flush beats pair]
3  4  3  2  3  1  6  2  9  3  6  3  6  1  10 2  10 2  10 3  0 [three_of_a_kind beaten_by full_house]
3  3  3  1  3  4  12 3  5  2  12 1  7  1  8  4  8  4  10 3  1 [three_of_a_kind beats pair]
5  4  5  2  5  1  10 1  11 4  3  4  4  1  5  2  6  4  7  3  0 [three_of_a_kind beaten_by straight]
4  3  5  4  6  1  7  3  8  1  10 1  10 1  12 1  12 2  12 2  1 [straight beats full_house]
2  4  2  1  2  4  11 4  11 2  6  4  6  3  6  1  6  2  8  2  0 [full_house beaten_by four_of_a_kind]
1  4  2  1  3  4  4  2  5  1  8  3  8  1  6  2  9  2  11 2  1 [straight beats pair]
8  3  8  3  8  1  8  2  11 2  5  3  6  3  7  3  8  3  9  3  0 [four_of_a_kind beaten_by flush]
3  3  4  3  5  3  6  3  7  3  11 4  11 1  9  3  9  3  6  2  1 [flush beats two_pairs]
4  3  4  4  4  3  4  1  4  2  5  2  5  2  5  3  3  1  3  4  0 [nothing beaten_by full_house]
3  1  3  1  1  1  1  3  1  1  3  3  3  1  11 3  2  3  4  4  1 [full_house beats pair]
5  4  8  3  2  2  3  4  4  2  12 4  12 1  9  3  9  1  9  4  0 [nothing beaten_by full_house]
11 3  11 1  11 2  11 4  5  4  8  3  8  3  11 2  11 3  11 1  1 [four_of_a_kind beats full_house]
10 4  10 1  5  2  5  3  5  1  10 4  10 3  10 4  10 3  9  1  0 [full_house beaten_by four_of_a_kind]
4  2  5  1  6  1  7  1  8  4  1  3  1  4  6  2  6  2  3  4  1 [straight beats two_pairs]
2  2  2  4  12 3  12 2  4  1  4  4  4  4  6  1  6  4  6  2  0 [two_pairs beaten_by full_house]
```

The input variables (x1-x20) decompose into two groups of ten, each of which represents a 'hand' in a poker game. Variables from x1 to x10 represent 'hand A' and variables from x11 to x20 represent 'hand B'. x1 represents the number value of the first card in hand A; variable x2 represents the suit value of the

first card in hand A. Variables x3 and x4 represent the number and suit values of the second card in hand A, and so on. Cards in hand B are represented in a similar fashion by the variables beginning with x11.

Adopting the convention that 1 stands for Diamonds, 2 for Hearts, 3 for Spades and 4 for Clubs, the first half of the first input vector represents (working left to right) the ace of Hearts, the ace of Diamonds, the ace of Clubs, the Jack of Spades and the Queen of Clubs. Now, in all cases where x21 = 1, the rank value of hand A is higher than that of hand B.[9] The value of x21 thus shows whether hand A beats hand B.

The input/output rule here is hand-oriented and thus primarily grounded in relational effects. This suggests that we will find little type-1 regularity within the training set. Certainly, there will be a certain amount of spurious, type-1 regularity. We might, for instance, see a slightly raised probability of seeing x21=1 for lower values of the lower number-value variables, since lower values are more easily accommodated within high-rank hands (such as straights) whose higher values must occupy variables with higher subscripts. However, the effect of this should be small. In general, we would expect the observed frequencies to be close to their chance values.

To uncover the regularity in this training set we need to re-code the input

[9]Each input vector actually shows a sorted hand. A straight is thus always shown with its lowest card instantiating the first pair of variables in the hand.

vectors. In the simplest case, this involves applying 'hand-evaluation' functions to the relevant sub-ranges of input values and 'evaluation-comparison' functions to the values thus produced. But what happens if the re-coding is to be carried out by an agent (e.g. a learner) that is limited in its computational properties? What if we assume that it is *unable* to carry out such sophisticated operations as 'hand-comparison' and 'evaluation-comparison'?

It is tempting to say that the regularity in question then becomes *inaccessible* to the learner. But this is certainly an exaggeration. The learner might have access to functions that could be combined together somehow so as to bring the regularity in question to light. In this case, the regularity would be accessible but discoverable via a different route (i.e. trajectory).

Imagine, for the sake of argument, that the learner has access to a function that computes the equality of an arbitrary number of inputs and a function that computes the difference of two values. Let us also imagine that the learner is able to feed arbitrary constant values into these functions as they are applied. Given these computational properties, is there a re-coding of the Poker training set that brings out the underlying regularity?

Figure 3 illustrates one possibility. It shows a re-coding that involves several applications of both the accessible functions. Applications of the 'equals' function correspond to nodes marked with an 'eq'. Applications of the 'difference' function correspond to nodes marked with 'diff'. Inputs for these applications arrive

via the relevant branches. They may be the outputs of function applications or, at the lowermost level, values of original dataset variables. The numbers labeling the leaf nodes of the tree are training-set variable numbers. Thus the leftmost application of the difference function takes values of the first and third variables (x1 and x3) and feeds an output into the leftmost application of the equals function.

This re-coding does indeed bring to light one aspect of the underlying regularity, i.e., it does produce a final value that exhibits strong frequency effects. Within the structure we see a series of applications of the 'diff' function to successive values of the hand A variables. If hand A is a straight, the values produced by all these applications will be 1. The values flow on up to become — together with a constant value 1 — the inputs for an application of the 'equals' function. If hand A is a straight the output of this application will be 1; otherwise it will be 0. Thus at this level we effectively have a feature detector for 'straights'.

On the other side of the tree we see a pair of applications of the 'equals' function to four of the number values of hand B. Both applications effectively test to see whether two specific cards have identical number values. If both tests succeed and produce a 1 output, then the 'equals' function into which they both feed will also produce a 1. Thus the right-hand side of the tree effectively tests for the presence of 'two-pairs' in hand B.

Putting it all together, the topmost 'equals' produces a 1 only if hand A is a

straight and hand B is two-pairs. According to the rules of Poker, a straight beats two-pairs. So we would expect values of this final output to be correlated strongly with values of the output variable x21. Thus we find that the re-coding brings out one aspect of the underlying regularity.

This example shows that a particular, type-2 regularity can be reified via different routes, i.e., via different re-codings. This does not prove that type-2 regularities are themselves subjective; but it *does* prove that the interpretation that reifies the regularity *is*. This is an intriguing result.

A learner subject to the rather profound computational limitations applied in the example must resort to constructing a *particular* re-coding. As suggested above, this re-coding might plausibly be based on the implementation of what are, in effect, feature detectors for abstract objects, e.g. a feature detector for 'straights', a feature detector for 'two-pairs' and — exploiting these — a further detector for situations in which a straight beats two pairs. Once the learner has constructed these detectors, the relevant objects effectively move inside the learner's sensory universe. The functionality involved in their detection then becomes hidden within a new source of (virtual) input data.

The picture that is beginning to emerge, then, is one in which learners using relatively primitive computational resources are driven to construct relatively rich representational structures. In doing so, they are exploiting a kind of divide-and-conquer approach. Each time a new feature detector is constructed from

the limited function base, some of the computation that underpins the relevant regularity is 'picked off'. Eventually, a level of representation is reached at which the computation can be performed using one of the basic functions.

It is important that the sense of 'representation' in play here is not as rich as, say, that of Fodor and Pylyshyn (1988). Fodor and Pylyshyn seek to show that connectionist knowledge-encoding is compromised by the (putative) lack of a systematic symbol *system* with quasi-linguistic syntax. By contrast, the representations which figure in our representation/computation trade-off need amount only (but significantly) to acquired abilities of feature-detection. In arguing (section 3 below) that it will be essential to ensure the wider availability of such acquired feature-detection skills outside the original problem-solving context, we are in effect seeking to outline a kind of connectionist analogue to explicit representation, classically conceived. But it is an analogue shorn of the unnecessary baggage of the Language of Thought hypothesis, and defined specifically with learning in mind. Explicit representation, in this usage, consists in re-coding strategies becoming *globally* available, i.e., available as a means of searching for type-2 regularities in any new domain we encounter. We therefore claim that the tradeoff between computation and representation is one which affects learning mechanisms at the most basic and primitive level. Given modest limits on computational functionality, even apparently simple regularity-exploitation tasks are likely to necessitate the construction of rich re-coding structures.

The moral of the story so far is that type-2 learning problems can indeed be cracked using familiar statistically-based search methods, but only by reducing the type-2 problem specification to an incremental sequence of type-1 problems. The repeated re-formulations achieved by type-1 learning produce feature detectors which transform a body of gross input data into a form relative to which the target regularity is of the statistically tractable (type-1) kind. The importance of this general kind of manoeuvre (using incremental learning to reduce the search space for complex mappings) has been highlighted in a wide variety of recent work in connectionist learning (e.g. Elman, 1991; Jacobs, Jordan and Barto, 1991).

What our discussion adds to this emerging consensus is, first, a principled account of the range of cases in which such a strategy is, as far as we can see, compulsory, viz. all cases in which a given mapping is type-2 relative to a body of gross training data. And second, a sense of the surprising ubiquity of such cases; the problem bites not just in relation to e.g. learning the complex hierarchical structures of a grammar (Elman, 1991) but also in much more primitive contexts. In fact; as we have seen, the problem is likely to raise its head whenever learning involves attending to relations between input variables. Given this quite unexpected ubiquity, it seems that the ability to learn incrementally is a fundamental key to cognitive success.

3 From Feature Detection to Analogical Reason

Incremental learning has loomed large in several recent discussions of connectionist learning. Two especially revealing cases are Elman (1991) and Jacobs, Jordan and Barto (1991). Elman studied a grammar acquisition problem in which a simple recurrent net was required to learn a grammar involving features such as verb-subject number agreement and long distance (cross-clausal) dependencies . He discovered that ordinary backpropagation learning was unable to prompt a net to acquire knowledge of the grammar. But success could be achieved in either of two ways. First, it was possible successfully to train a net if the training data was divided into graded batches beginning with simple sentences and progressing to more complex (multi-clausal) ones. Second, success could be achieved by providing the network with a limited initial window of recurrency (re-setting the context units to 0.5 after every 3rd/4th word) which was allowed to increase as training progressed. In the latter case there was no need to batch the training data as the restricted initial memory span in effect filtered out the mappings involving cross-clausal dependencies and allowed in only the simpler constructions: the data was thus 'automatically sorted'. It is clear that the two techniques are functionally equivalent and that the reason that they are needed is, as Elman comments, that

> If the domain is of sufficient complexity, and if there are abundant 'false solutions', then the opportunities for failure are great. What

> is required is some way to artificially constrain the solution space to just that region which contains the true solution. (Elman, 1991, p.8)

By 'false solutions' Elman means the extraction of the wrong regularities, i.e. finding spurious type-1 regularities which will fail to determine successful performance on unseen cases. Both of Elman's solution techniques force the net to learn certain basic mappings first (e.g. verb/subject number agreement). Once this knowledge is in place, the more complex mapping-tasks (e.g. agreement across an embedded clause) alter in statistical character. Instead of searching the explosive space of possible relations between input variables, the net has been alerted (by the simpler cases) to a specific relation (agreement) which characterizes the domain. A type-2 learning problem is thus reduced to an incremental sequence of type-1's.

The *type* of incremental learning which can be supported by an Elman-style regime is, however,somewhat limited. It is limited insofar as the proposed strategies will succeed only when a problem (a target, generalizable, input-output mapping) has what we may term a *conservative* decomposition. A body of training data presents a conservatively decomposable problem if the early knowledge needed to reduce the search space for subsequent learning is discoverable by first focussing on some subset of that same overall body of training data. In our terms this amounts to solving the overall problem by first pur-

suing the type 1 statistics of a fragment or fragments of the overall training data and then using the feature detection skills thus acquired to reduce subsequent search; i.e., the overall problem is solved by the devious (temporally structured) exploitation of type-1 learning focussed initially on subsets of the overall training data.

This assumes, however, that the specific feature detectors needed to reduce subsequent search are discoverable just by looking at fragments of the training data appropriate to the final task. But not all learning problems (not even all those prone to a *kind* of incremental solution) will fall into this class. Instead, some target mappings may be learnable only if the search space is controlled by the exploitation of feature detectors which could *not* be acquired by focussing on subsets of the training data specifying the target mapping.

As an example, consider once again the conditional approach problem briefly described at the end of section 1. Successful learning here depends on the presence of two feature detectors: one for closeness and one for apparent width. The crucial feature which the net must later identify to learn the target mapping is actual width — a ratio between these two lower level features. But no amount of exposure to the data which embodies the conditional approach mapping will prompt the net to develop these two lower level feature detectors. Instead, they must be developed as a result of the early attempts of the system to learn to perform other kinds of task. In such cases the target problem has only what

we can call an *extended* incremental solution. The type 2 task (conditional approach) *can* be learned from the given data as long as the two prior feature detectors are present. But these feature detectors will not themselves arise merely as a result of exposure to that data, nor as a result of exposure to any subset of it. In these extended cases the right learning trajectory over time will indeed ensure the acquisition of the target skill. But the trajectory involves the importation of other feature detection skills acquired as a result of attempts to solve different problems.

Conservative incremental learning is thus self-sufficient in a way which extended incremental learning is not. Elman's grammar acquisition case is conservatively incremental and hence is well suited to a treatment involving a single network and a single (but usefully fragmented) body of training data. Extended incremental learning, by contrast, seems to call for a more complex, modular processing economy. Such a broader notion of incrementality is readily illustrated. Jacobs, Jordan and Barto (1991) describe a system which comprises a variety of architecturally distinct sub-nets. These sub-nets compete to be allowed to learn to represent a given input pattern. Whichever net, early on in the training, gives the output closest to the target, is allowed to learn that pattern. In the trained-up system. a gating network selects which sub-net should come into play to yield the output for a given input.

Such modular systems of sub-networks afford yet another means of reducing a type-2 mapping task to a set of type-1's. Thus in a task such as multiple-speaker vowel recognition (Jacobs, Jordan, Nowlan and Hinton, 1991) a modular system can avoid the intractable task of finding a single position in weight space capable of solving the problem for all the voices and instead tackle a set of more tractable ones, viz. one sub-net learns to identify vowels in children's voices, another in men's and another in women's. (See also Churchland and Sejnowski, 1992, p.130.)

Spatial modularization is also the key to the flexible re-use of the valuable products of early learning. Given the vital role we have assigned to incremental learning, it is clear that it would be beneficial if an organism were able to re-deploy individual parts of a problem-solving trajectory. One way to allow this is to ensure that each step along the way to a solution is individually available for use in relation to other (subsequently encountered) problems. Otherwise put, it is beneficial to ensure that a detector for some property P is not inextricably embedded into the solution to a more complex problem, since P may be just the property or sensitivity which would render some other subsequently encountered, problem tractable. Assigning specific tasks to specific modules allows for the future re-use of a trained-up module in relation to some other overall task (see Jacobs, Jordan and Barto, 1991). This kind of approach is thus capable of dealing with cases in which an incremental solution is non-conservative with respect to overall input-output mappings (as it allows us to invoke specific

feature-detecting modules in wholly new overall problem solving contexts.)

Other ways of exploiting spatially extended cascades of processing devices are close to hand. Thus imagine a device which, in place of the developing short term memory component in Elman's simulation, had a pre-processor whose task (or effect) was to filter the input data, effectively allowing only simple sentences (short ones) to proceed downstream. The device could be gated so that once the error signal in a down-stream network was low, the filter was by-passed. Here a simple gated cascade of processing devices promotes the kind of early learning needed to reduce the statistical complexity of the latter task to a manageable level. More generally still, any cascade of processors in which the upstream devices sort, filter or re-code incoming data holds out the promise of promoting successful learning in just the way imagined.

An illustrative example of an upstream filtering effect is provided by Johnson and Morton's (1991) speculations about face recognition. The idea (pursued in Karmiloff-Smith, 1992) is that evolution may have provided us not with an innate face-recognition module as such, but rather with an upstream filter (pre-processor) which allows through only inputs whose visual profile is roughly that of three high contrast blobs in the positions of eyes and mouth. Such a filtering device (a feature detector) will ensure that a downstream module 'sees' only data involving a very high statistical preponderance of human faces. Specialization for human face-recognition (Karmiloff-Smith, 1992) will rapidly follow. Thus

even if the statistical presence of human faces in the gross visual inputs to the system is low (relative to e.g. walls and buildings) the effective statistics (further down the processing cascade) may be quite different.

It is also worth distinguishing the provision of (mere) filtering devices from the provision, or development, of more complex early re-codings. As Kim Plunkett (personal communication) has pointed out, some pre-processing devices may do much more than filter an input corpus. They may instead respond to an input corpus by creating a wholly new corpus of data for further processing. For example, an input sentence may be re-coded as [verb/subject]. In this way the similarity space defined by the bare inputs is itself altered. Two items which are very alike considered as bare inputs may be re-coded so as to be very different, and vice versa (see e.g. Plunkett and Marchman, 1991). In both cases (re-coding and mere filtering) the upshot is a change in the statistical profile presented downstream. But in the re-coding case the change is both more radical and in a sense more intelligent.

The sheer variety of ploys and mechanisms available to reduce type-2 complexity to manageable levels is now becoming visible. The common thread uniting a variety of recent developments in connectionist learning is that they all add, in some way, a kind of transformation factor which reconfigures the statistical task involved in learning a given mapping. Evolved pre-processors, acquired re-codings, the careful management of training, the provision of an initially

limited short term memory and techniques of modular decomposition all play the same basic role; a role forced on them by the bedrock intractability of type-2 problems to brute statistically-based search. The most advanced and intelligent incarnation of the transformation strategy has yet to be mentioned, however. That incarnation is analogical reasoning itself.

Analogical reasoning, we suggest, is fruitfully viewed as a device which enables a learner to solve problems which are statistically intractable (type-2) and which themselves lack a natural decomposition into type-1 fragments. The trick of analogical reasoning is to use the natural decomposition of one problem as part of a representational trajectory whose terminus is a solution to the problem in hand. This is closely akin to the re-use of existing modularized knowledge. To clarify this idea, it is useful to draw on a recent discussion by Paul Churchland (Churchland, forthcoming) in which he develops a problem concerning connectionist learning and unobservables. In the light of our previous discussion, it will become apparent that Churchland may have misconceived the precise nature of the problem, while nonetheless spotting a powerful solution!

The problem is presented like this: connectionist learning is known to involve a kind of curve-fitting procedure in which the parameters of a polynomial are gradually adjusted so as to 'pull' the defined curve over the known data points. Churchland depicts this curve-fitting as the search for a set of categories which enable the net to accommodate the data in a powerful and generalizable way.

Such categories, he goes on to argue, must, however, be in some sense given (either explicitly or implicitly) in the body of input data. A category counts as explicitly given if the input coding isolates those very items (e.g. the category 'verb' is explicitly given if the input involves a variable which stands for verbs). And a category is implicitly given if the training data allows the net to induce such a category, although no individual variable coded for it in the input (e.g. NETtalk's induction of a partition corresponding to the noun/verb distinction on the basis of a corpus of unmarked sentences, see (Sejnowski and Rosenberg, 1987)).

Churchland's worry is that in some cases the category needed to make systematic sense of a body of data may not be present in that body of data — not even implicitly. Suppose that what is present (implicitly or explicitly), is conceived of as, in a broad sense, being observationally available. The question then becomes how a net can learn a category which would (a) make sense of a body of data but (b) is not observationally available in the data. The problematic case is thus described as one in which:

> ... a functionally essential category is simply not present in the input vectors at all, not even implicitly. Here the network must fail to learn if there is no functional relation that binds the observationally available categories in the absence of the hidden category. A net cannot learn a function that does not exist. (Churchland, forthcoming,

p.19)

The problem is pressing insofar as the course of human knowledge has involved the repeated positing of categories which (according to Churchland) are not given (implicitly or explicitly) in the data. Instead, we 'reach behind the appearances' to posit atoms, electrons, electromagnetic waves etc. etc.. How is this possible if learning consists (as the connectionist suggests) in a curve-fitting procedure defined over observables?

Churchland's answer is fascinating and suggestive. But before we consider it, it is worth pausing to question the way the problem has just been set up.

On the one hand, Churchland wants to recognize the ability of connectionist learning to induce new variables and representations. This is what happens when a net fixes on a regularity which was not explicitly marked in the input data. On the other hand, he sees that very often connectionist learning will be unable to discover a given regularity precisely because a specific variable is missing. To balance these two observations, he appeals to the notion of what information is actually present (implicitly or explicitly) in the input data. But on what grounds do we decide what information is or is not thus present? Churchland's underlying criteria looks to be this: if a curve-fitting procedure applied to that set of data points can unearth the regularity, it is counted as present (observable) in the data, and otherwise not. But this is hardly explanatory. What we want to know is why, given that the mapping is learnable if the data

is viewed through the lens of a specific category, is a network sometimes able to induce knowledge of the essential category and sometimes not? The distinction between type-1 and type-2 regularities provides the answer.

When the statistical regularity captured by a given category is type-1 relative to a training set, the kind of search undertaken by standard learning algorithms can be relied on to discover it. When the regularity is type-2, that kind of search will fail. What is needed in the latter case is some means of reducing the problem presented by the input data to that of extracting a type-1 regularity. The problem is not well understood by asking what is in some elusive sense in the training data. Rather, the issue concerns what can plausibly be got out of the training data given a certain manner of searching the space of potential relations between input items.

Thus (re)construed, Churchland's general problem concerns not just those rare cases where essential data is genuinely absent but also (and more commonly) cases where the raw data is available but it is not presented in a form amenable to type-1 (curve-fitting) learning. Such a construal seems well suited to Churchland's actual interest, which is in cases in which what the scientist needs is not more data as such, but simply to posit a variable which enables her to discern a regularity in that body of data.

Despite this partial misconception of the nature of the problem, Churchland fixes on what is surely one powerful solution, viz. he notes the possibility of

what he terms conceptual redeployment. Conceptual redeployment is a process in which a set of categories successfully induced to facilitate success in one domain is imported to another. Such importation (promoted by e.g. the chance juxtaposition of the intractable problem with a reference to the other, successfully-theorized, domain) allows the data from the problematic domain to be systematically reconceptualized via the lens of the imported categories. Thus, to follow Churchland's example, we may induce the categories of wave phenomena to make sense of observable input data concerning liquid behaviour. But once these categories (wavelength, velocity, frequency, etc.) are available, they may be redeployed to make sense of bodies of data from other domains (sound, light, etc.). Churchland comments that:

> While the conceptual prototypes of wave phenomena could not have been learned in either of these comparatively opaque domains, they were certainly capable of being redeployed there... (Churchland, forthcoming, p.25)

Without the transforming lens of the feature detectors for wave phenomena, the bodies of data concerning sound and light presented intractable problems of search. What Churchland has described, we believe, is a representational trajectory in which more tractable bodies of data (the water data) yield knowledge of variables (or even just feature detectors) whose use in pre-processing transforms the sound and light data into a form in which the target regularities are closer

to the surface and hence prone to succumb to the kind of search methods we actually have available. Churchland tends to depict conceptual redeployment (aka analogical reasoning) as a strategy in which no process of 'curve-fitting learning' occurs. But it is not clear that this is so. We claim instead that the imported categories/features reconfigure the data so that the curve-fitting is relatively trivial. Achieved representation is thus traded against intractable computation. But computation is not altogether abandoned.

4 Evolution, Creativity and Cognitive Closure

The single most important key to cognitive success, we have argued, lies in the use of a variety of diverse methods by which to transform intractable type-2 learning problems into temporally or spatially extended sequences of problems of an easier statistical stripe. Such methods may range from the genetically determined provision of simple feature detectors to the analogical redeployment of achieved knowledge. There is, however, a snag.

The snag is that all the methods available depict the solution of type-2 learning problems as in a certain sense fortuitous. By this we mean that it looks to be impossible, if our analysis is correct, to actively take a type-2 problem and work backwards to unveil a successful problem-solving trajectory. Type-2 problems get solved, it seems, only when we find ourselves in a position to solve them. We cannot put ourselves in a position to solve them. We cannot put ourselves

into that position unless the problem has already been solved and we are merely recapitulating an achieved decomposition.

There are two possible responses to this observation. We might use it to motivate a rejection of the claim that all possible learning algorithms will flounder in the face of unreconstructed type-2 regularity. For it might seem that human beings typically can non-fortuitously solve type-2 problems. Or we might use it to motivate a thesis concerning the shape of the space of non-fortuitously solvable problems, and thus suffer the consequence that certain genuine regularities in the world may be such as to be unlearnable by any non-chance method.

Despite its initial plausibility, we are inclined to reject the claim that human beings can non-fortuitously solve type-2 problems. True, we can (and do) deliberately set out to solve type-2 problems. And true, we often succeed. But the solution may still be fortuitous in the relevant sense. Thus it may be that a lucky juxtaposition of ideas prompts the exportation of an existing set of categories to a new domain, as in Churchland's scenarios. Or it may even be that we can search (perhaps even intelligently search) through our various bodies of knowledge in the hope of finding such a companion domain. But in either case, success or failure is determined by our chance possession of such a body of knowledge.

Similarly, we may be lucky enough to acquire bodies of early knowledge which suitably (as in Elman's staged learning) reduce the search space for subsequent

learning. Or we may be lucky enough to be genetically provided with filters — pre-processors which transform the gross input data into a manageable form. What is lacking in all cases, however, is any general purpose type-2 learning algorithm, i.e. an algorithm capable of either (a) taking a body of data which is organized around a type-2 regularity and inducing the feature detectors, categories or early knowledge which will render the mapping learnable or (b) just inducing the solution directly, without the benefit of the intervening stages. Perhaps such a learning algorithm is on the horizon, but we doubt it. (Recall that the full search space for type-2 regularities is a superspace of the space of applicable Turing machines.)

If type-2 learning is, in this sense, always a matter of good fortune, just how serious a problem is this? Three observations suggest that it is not so serious as it may at first appear.

First observation: biological evolution is naturally incremental. The relevant facts here are well known; the evolutionary process constructs extended pathways through a kind of adaptive space. Each step along such a pathway involves a change to the physical structure of an organism. And each such change must yield a successful organism. One result of this pressure is that evolutionary change tends to be both conservative (existing and successful structures are preserved) and incremental (new structures are added on, in a progressive manner). Of course there are exceptions. But the general trend (see e.g. Dawkins, 1986;

Jacob, 1977; Ridley, 1985) of 'preservation and incrementation' means that biological selection is ideally suited as a way of easing organisms deeper and deeper into the space of type-2 problems. Thus a creature with an evolved feature detector for a property (say, a relation between input features) P is a candidate for incremental adaption in which a feature detector for a new complex property (say, a relation between P and some other input features) is constructed. The upshot is that evolutionary search is especially well-suited to scouting the space of incremental solutions: a space which, we have argued, represents the accessible face of (otherwise) intractable type-2 learning problems.

As an aside, we note that this may help capture part of what is special about evolutionary learning. For in a sense, individual connectionist learning and evolutionary change are no different: both face the problem of searching a space of potential changes and the statistical difficulty of finding the changes which would yield a solution to a type-2 problem are the same in each case. Both, indeed, perform a kind of gradient descent learning in which changes which minimize failure messages are preserved, But one difference is that the sequence of changes selected by evolution is constrained to be a sequence in which most early solutions are preserved and built upon; whereas connectionist learning is highly prone to totally abandoning early solutions as a search progresses (see e.g. French (1991) on 'catastrophic forgetting'). Evolution may be best conceived as a kind of learning akin to connectionist (gradient descent) learning but applied to whole complexes of sub-networks and with an added cost factor

whose effect is to promote the preservation of the structure of the successful sub-nets of previous generations. The complex problems which will succumb most gracefully to such a system are, of course, those problems which decompose into a set of tractable sub-problems.

Second observation: evolution flexibly co-constructs problems and solutions. The point here is that it is easy to overestimate the statistical difficulty of an evolutionary search which has terminated in a solution to a type-2 problem (such as conditional-approach). One source of such overestimates is a mistaken tendency to depict evolution as facing a specific problem and needing to search for a solution. Thus in the case of e.g. supervised connectionist learning, the system is given the task of learning a specific mapping M and must search the space defined by its architecture and inputs for a solution. But the evolutionary case is rather different. Any 'solution' which yields a surviving and reproducing being will do. If evolution then hits on the solution to a type-2 problem, we should not imagine that it had to perform a search whose unique terminus was that solution. If that solution is instead just one out of a multitude of 'good evolutionary moves', the odds change dramatically. Thus if, in a space of 1,000 lottery tickets there is a unique point 28 which represents success, the chances of drawing a successful ticket at random are small. But if, in a space of 1,000 tickets, 100 are winners (albeit of different prizes) the chances of success increase.

Another (related) source of overestimates is our tendency to think of problem

solving rather than problem construction. Evolutionary pressure selects niche and organism together i.e. it simultaneously defines a problem space and organisms adapted to it. It may thus select problems which are satisfactorily solvable given the limited individual learning abilities of organisms. In the terms of our earlier discussion, there might indeed be selection for an organism-environment pairing which provides a nicely staged series of problem-solving tasks and hence aids incremental learning![10]

It is interesting to note that both of the above observations (that evolution is natural incremental and conservative, and that it co-constructs problems and solutions) may have echoes in individual development. Annette Karmiloff-Smith (1979, 1992) has pioneered the idea of individual development as involving an endogenously (or partly endogenously) driven process in which existing problem solutions are conservatively re-described. Re-description consists in a more abstract, unifying re-coding which enables more flexible use of the information specified in the solution. For example, re-coding a push-down stack of operations for playing a song on the piano as a random access list, thus enabling you to begin midway or play jazz variations.

Despite a lack of concrete mechanisms (but see Clark and Karmiloff-Smith

[10] In fact, some recent simulation work in which a connectionist net X is allowed to co-evolve alongside another net Y which sends it training data can be fruitfully conceived as an exploration of the effects of the co-evolution of an environment (net X) and a gradient descent learning device (net Y), see Nolfi and Parisi (1991) and discussion in Clark (forthcoming).

(forthcoming), Clark (forthcoming) for some suggestions) the idea is attractive. For endogenous pressure to re-code is precisely self-generated pressure to explore continuously the space of incremental problem solutions without commitment to the solution of any specific problem. Each such re-coding may just happen to reduce a problem that was previously type-2 (and hence effectively outside the scope of individual learning) to a tractable type-1 incarnation. The learner will thus be engaged in a kind of continuous search for new problems insofar as each re-coding changes the shape of the space defined by the inputs and hence opens up new cognitive horizons. An individual, endogenously specified tendency to engage in representational redescription would thus amount to a natural injunction to persistently pull as much as possible into the space negotiable by our on-line weak type-1 learning methods. With direct task-driven exploration of type-2 spaces out of the question, evolution bestows on the individual a generic drive to code and re-code and re-re-code. Once again, we are trading spaces — using achieved representation to reduce the complexity of computation.

Third (and final) observation: language and culture are just what weak learning devices need — they enable us to trade achieved representation against computation on a cosmic scale. Public language may be seen as a ploy which enables us to preserve the fruits of one generation's or one individual's explorations at the type-1/type-2 periphery and thus quickly to bring others to the same point in representational space. Otherwise put, we can now have learning trajectories

which criss-cross individuals and outstrip human lifetimes. In addition, we can (by grace of such cultural institutions as schooling) easily re-create, time and again, the kind of learning trajectory which leads to the solution of key complex problems. In these ways, the occasional fruits of good fortune (the discovery of a powerful input re-coding (a concept) or a potent sequence of training items) can be preserved and used as the representational base-line of the next generation's mature explorations. Language and culture thus enable us to trade achieved representation in any member of the species, past or present, against computation for all posterity. Given the necessarily fortuitous nature of the search for new representations, this is an advantage whose significance cannot be exaggerated.

It is interesting to compare this vision (of language and culture as a means of preserving representational achievements and extending representational trajectories) with that of Dennett (forthcoming). Dennett usefully identifies a weak type of learning which he calls ABC learning and defines as the 'foundational animal capacity to be gradually trained by an environment'. He depicts standard connectionist learning as falling into this class and asks what leads us to outstrip the achievements of such ABC-learners. His answer is: the augmentation of such learning by the symbol structures of public language. (See also Dennett (1991) p.190, 220, 298-302.) An augmentation which is itself, he conjectures, the probable root of our abilities of representational redescription.

We think Dennett is almost right. He is right to depict language as a key factor in our abilities to frequently and repeatedly appear to exceed the bounds of ABC (or, as we would have it, type-1) learning. Yet in a very real sense there is, we believe, no other type of learning to be had. What looks like type-2 learning is in fact the occasional re-formulation of a type-2 problem in terms which reduce it to type-1. Language, we suggest, simply enables us to preserve and build on such reductions, insofar as the key to each reduction is an achieved re-representation of a body of data. But language is not, we think, the root of such re-representations. Instead, such re-representations must be discovered essentially by chance (perhaps aided by an endogenous, though undirected, drive to continuously seek re-coding of existing knowledge) in either individual or species learning. Language is a preserver of chance representational discoveries and of useful learning trajectories. It also doubtless feeds and augments the trick of analogical reasoning which we discussed earlier. Language-users will thus indeed steer a profoundly deeper course into the type-2 problem space than anyone else, but for reasons which are, we suspect, a little more pedestrian than Dennett imagines.

The three observations just rehearsed combine, we hope, to make our cognitive situation seem a little less bleak. True, we are type-1 learning devices inhabiting a world populated by much more complex regularities. But by trading representation against computation, in both individual and species time, we nonetheless make significant inroads into the type-2 space. That said, we must still live with

two depressing prospects. The first is that our image of true creativity threatens to become somewhat emaciated. For the important conceptual innovations are just those which bring what was previously an intractable type-2 learning problem down into the space of problems solvable by type-1 methods. These innovations turn problems which previously could not have been solved (in the very real sense of being intractable to type-1 learning) into ones which can be solved. We thus give a precise meaning to the distinction which Boden, in her (1990) investigation of creativity, makes between ideas which as it happens did not occur before and ideas which, in some real but elusive sense could not have occurred before (see Boden, 1990; p.31-41.) But alas, on our account, there is no intelligent means of searching for the re-codings which turn a previously type-2 problem into a type-1 format. Hard work, the constant juxtaposition of ideas in the hope of analogical 'trajectory hopping', the blind endogenous exploration of re-codings and sheer good fortune (genetic or otherwise) just about exhaust the possibilities. Intelligent reduction of the new categories needed to suck hitherto intractable problems into our cognitive ambit is, regretfully, just not to be had.

The second depressing prospect is that large tracts of genuine regularity in the universe threaten to be forever unknowable to us, and probably to any learning creature. For we can discover deeper (type-2) regularities only when such regularities lie on a convenient trajectory of achieved representations. But there is, we suppose, no reason to believe that all the interesting truths about the universe will lie on such trajectories. If a regularity is type-2 and there

is no natural sequence of problem solutions relative to whose representational products it reduces to type-1, then it is effectively unlearnable. We are thus quite dramatically cognitively closed (to use the terminology of McGinn (1989)) to phenomena of a certain well-defined type. Of course, we do not actively perceive this closure, as our universe is conceptualized by a body of achieved representations, and we continue to solve the problems defined in those terms. If nature itself then looks systematic and incremental to us, it may be because we are systematic and incremental learning devices who are cognitively closed to truths which lie outwith cumulative problem-solving trajectories.

To sum up, we have scouted both good news and bad. The good news is that despite the unexpected weakness of currently conceivable (type-1) learning methods, there exist a variety of ploys which enable us to solve type-2 problems. What all these ploys have in common is that they trade achieved representation against prohibitive computational search. The bad news is that there is no general, intelligent way of seeking just those re-codings which would transform a specific type-2 learning task so as to present a tractable type-1 regularity. Instead, the re-coding manoevre will save us only in those cases in which the required redescriptions are to be encountered along some natural problem solving trajectory. The boundaries of the genuinely *learnable* thus probably fall well short of the boundaries of what is in principle *representable* by a computing device such as the brain.

Conclusions: Probing the Unobservable

It is widely understood that the 'difficulty' of a particular computation varies according to how the input data are presented. With the data presented one way, achieving particular computational effects may require elaborate and expensive processing. With the data presented differently, it may be possible to achieve the same effects much more straightforwardly. (For a classic discussion, see Marr, 1982, p.21.) What is less well understood is the effect of this computation/representation trade-off within the learning paradigms. We have suggested that existing learning algorithms tend to rely predominantly (and in some cases exclusively) on the extraction of a specific type of regularity from a body of input data. This type of regularity lies close to the surface of the training data, in the form of pronounced frequency effects and is thus fairly straightforwardly extracted by a variety of strictly empirical methods. Some appearances to the contrary, the extraction of type-1 regularities is really all we currently know how to achieve — and no wonder, once the size of the search space for the other form is appreciated.

The extraction of the more opaque type-2 regularities is not, however, an advanced or exotic requirement. Even in the domain of apparently simple 'artificial life' style robotics behaviours, type-2 regularities are quickly significant. Unlike Dennett (forthcoming), we do not believe that it is only language-users who regularly go beyond type-1 (or, as he says, ABC) learning. What, then, is the

trick by which inherently weak learning devices regularly solve apparently tough type-2 problems?

The trick, we suggested, is to trade representation against computation; to use luck, genetic evolution and individual type-1 learning to yield representations which re-code the input sample and hence gradually transform the nature of the learning task. Being computationally weak, we compensate by valuing representational richness and constructing long problem solving trajectories in which a cascade of coding and re-coding is the essential prerequisite to effective learning.

Since no intelligent method of searching for the specific representations needed to render a target mapping learnable exists, good fortune must play a major role in our successful forays. But this reliance on good fortune begins to look less counter-intuitive once we see (a) that evolution is itself a naturally incremental species-level learning device, (b) that problems and solutions co-evolve (so we never really seek a solution to a specific problem) and (c) that language and culture provide us (the most successful explorers of type-2 problem space) with an invaluable means both of preserving and building on each and every fortuitous representational movement, and of recapitulating successful learning trajectories once they are achieved.

It may be useful, finally, to cast our results in familiar (but not previously well-defined) terms. Our main distinction is between two forms of regularity: type-1 regularity which takes the form of non-chance frequency effects in the original

data, and type-2 regularity which takes the form of non-chance frequency effects in some re-coding of the original data. Type-2 regularities are tractably discoverable only if the input data is systematically re-coded so as to highlight certain properties. Each such re-coding can be seen as effectively altering what is observable to the system in question. The basic task of higher cognition is thus to progressively expand an organism's 'observable' universe so as to suck in as many useful, previously type-2, regularities as possible. Such expansion always involves seeking and exploiting re-codings of the input data (representations) which re-shape the search space for other interesting regularities. This process is both effectively blind (unintelligent) and highly incremental. It results in a cascade of re-codings most reminiscent of Karmiloff-Smith's hypothesis of repeated 'representational redescriptions'. What we now see is why such a process is rapidly forced on us, courtesy of the surprisingly weak nature of achievable (type-1) learning.

It is no surprise, then, that incremental learning, in a variety of forms, has loomed so large in recent published attempts to expand the horizons of connectionist knowledge acquisition. Despite important individual differences, a common thread links all such treatments. What they all add to standard connectionist learning, is an increased opportunity to discover transformation factors: processing episodes which re-configure the statistical task involved in learning a given mapping. Modularization, incremental memory expansion, batched and sequenced training and the use of highly multi-layered architectures are all

means to the achievement of this common, statistically intelligible end. And the underlying trick is always the same: to maximise the role of achieved representation, and thus minimize the space of subsequent search. This now familiar routine is, as far as can we tell, obligatory. The computationally weak will inherit the earth, just as long as they are representationally rich enough to afford it.

Acknowledgements

Thanks to Bruce Katz, Noel Sharkey and Inman Harvey for useful comments on the text.

References

[1] Boden, M. (1990). *The Creative Mind: Myths and Mechanisms.* London: Weidenfeld and Niicolson.

[2] Churchland, P. (forthcoming). Learning and conceptual change: the view from the neurons. In A. Clark and P. Millican (Eds.), *Essays in honour of Alan Turing Vol.II: Concepts, Connectionism and Folk Psychology.* Oxford: Oxford University Press.

[3] Churchland, P. and Sejnowski, T. (1992). *The Computational Brain.* Cambridge, Ma.: M.I.T./Bradford Books.

[4] Clark, A. (forthcoming). *Associative Engines: Connectionism, Concepts and Representational Change.* MIT/Bradford.

[5] Clark, A. and Karmiloff-Smith, A. (forthcoming). The cognizer's innards: a psychological and philosophical perspective on the development of thought. *Mind and Language.*

[6] Dawkins, R. (1986). *The Blind Watchmaker.* Longman, England.

[7] Dennett, D. (1991). *Consciousness Explained.* Boston, Ma.: Little, Brown Co.

[8] Dennett, D. (forthcoming). *Learning and labelling: commentary on Clark and Karmiloff-Smith, 'The Cognizer's Innards' in Mind and Language.*

[9] Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* New York: Wiley.

[10] Elman, J. (1991). Incremental learning or the importance of starting small. Technical Report 9101. Center for Research in Language, University of California, San Diego.

[11] Fahlman, S. and Lebiere, C. (1990). *The Cascade-Correlation Learning Architecture.* CMU-CS-90-100, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.

[12] Fodor, J. and Pylyshyn, Z. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition, 28* (pp. 3-71).

[13] French, R. (1991). Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. CRCC Technical Report 51, University of Indiana, Bloomington, Indiana., 47408.

[14] 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.

[15] Hofstadter, D. (1984). The copycat project. A.I. Memo 755, Masachusetts Institute of Technology.

[16] Jacob, F. (1977). Evolution and tinkering. *Science, 196*, No. 4295 (pp. 1161-1166).

[17] Jacobs, R., Jordan, M. and Barto, A. (1991). Task decomposition through competition in a modular connectionist architecture: the what and where visual tasks. *Cognitive Science, 15* (pp. 219-250).

[18] Jacobs, R., Jordan, M., Nowlan, S. and Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation, 3* (pp. 79-87).

[19] Johnson, M. and Morton, J. (1991). *Biology and Cognitive Development: The Case of Face Recognition.* Oxford: Blackwell.

[20] Karmiloff-Smith, A. (1979). *A Functional Approach to Child Language.* London: Cambridge University Press.

[21] Karmiloff-Smith, A. (1992). Nature, nurture and PDP: preposterous development postulates?. In A. Clark (Ed.), *Connection Science, special issue on Philosophical Issues in Connectionist Modelling, 4*, No. 3 and 4 (pp. 253-270).

[22] Marr, D. (1982). *Vision.* New York: W.H. Freeman.

[23] McGinn, C. (1989). Can we solve the mind-body problem?. *Mind, 98* (pp. 349-366).

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

[25] Nolfi, S. and Parisi, D. (1991). Auto-teaching: networks that develop their own teaching input. Institute of Psychology. C.N.R. Rome. Technical Report PCIA91-03.

[26] Plunkett, K. and Marchman, V. (1991). U-shaped learning and frequency effects in a multi-layered perceptron: implications for child language acquisition. *Cognition, 38* (pp. 1-60).

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

[28] Ridley, M. (1985). *The problems of evolution.* Oxford: Oxford University Press.

[29] Rumelhart, D., Hinton, G. and Williams, R. (1986a). 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.

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

[31] Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce english text. *Complex Systems, 1* (pp. 145-68).

[32] Thornton, C. (forthcoming). *The Statistics of Conditional Approach.*

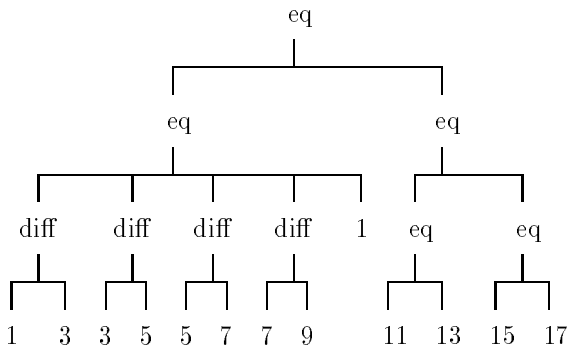

Figure 3: