

Measuring the Difficulty of Specific Learning Problems

Chris Thornton

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

Email: Christopher.Thornton@firenet.uk.com

Tel: (44)1273 678856

May 21, 2003

Abstract

Existing complexity measures from contemporary learning theory cannot be conveniently applied to specific learning problems (e.g., training sets). Moreover, they are typically non-generic, i.e., they necessitate making assumptions about the way in which the learner will operate. The lack of a satisfactory, generic complexity measure for learning problems poses difficulties for researchers in various areas; the present paper puts forward an idea which may help to alleviate these. It shows that supervised learning problems fall into two, generic, complexity classes only one of which is associated with computational tractability. By determining which class a particular problem belongs to, we can thus effectively evaluate its degree of generic difficulty.

1 Introduction

Recent progress in computational learning has produced a wide variety of learning models and methods. Some of these are associated with the field of connectionism, e.g. backpropagation [1]; others are associated with work on evolutionary and genetic methods [2]; still others are associated with work in the field of machine learning, e.g. C4.5 [3], Foil [4] and Golem [5]. Given the wide range of methods on offer, it is important for researchers to adopt a principled approach to the task of matching up methods with learning problems. Unfortunately, there are still major obstacles to be overcome.

A central question in any attempt to find a solution to a particular learning problem is ‘how hard is the problem?’ But although our theoretical understanding of computational learning has increased rapidly in the last decade, there is still no generic way to evaluate the difficulty of an arbitrary learning problem. Recent results in computational learning theory [6] extend earlier work by Valiant on PAC learning [7, 8] and rely in particular on the concept of VC dimension [9]. Work in this area has tended to concentrate on distribution-free (i.e., problem-independent) analysis although some recent work has given attention to distribution-specific analysis [10]. However, even results in distribution-specific analysis do not address the issue of the complexity of a specified learning problem. As things stand, then, there is no general method to evaluate the difficulty of a specific learning problem without making assumptions about which learning algorithm will be used.

This theoretical lacuna may seem, at first sight, to be inevitable. Any learning problem can only be solved by a given method with a given bias. This seems to tell us that (1) the idea of a general solution to a learning problem is nonsensical and (2) that a generic method of difficulty is therefore out of the question. The conclusion seems to be reinforced by Wittgenstein’s well-known claim to the effect that any mathematical sequence has an arbitrary number of generative rules. [11] It is also mutually-reinforcing with the observation that solving a learning problem is very much the same task as finding the ‘best’ (i.e. smallest) computer program which reproduces a particular dataset. This task is known to be NP-hard [12, 13,14,15]. So, once again, the implication seems to be that ideal learning is impossible and that generic measurement of learning difficulty is a non-starter.

In the face of the apparent impossibility of a generic measure of learning-problem complexity, practitioners who find themselves having to estimate complexity must resort to informal and heuristic methods of evaluation. A long-standing complexity-estimation heuristic is simply to determine whether or not the learning problem in question involves the capturing of relations. As has been known for some time [16] problems which involve relational effects are, in general, harder to solve than problems which do not. Another, reliable heuristic approach involves determining the order of statistical effect which underpins the solution to the problem [17]. The idea here is that higher-order effects are harder to exploit.

Are these heuristics unfounded? If so, we need to ask why researchers find them relatively reliable. If they are not unfounded then we need to determine precisely what their true foundation is. If the foundation turns out to be theoretically secure we need to reexamine our assumptions with respect to the impossibility of a generic measure of learning-problem complexity. This, in a nutshell, is the aim of the paper.

I will begin by noting that the problem of learning is essentially the problem of generalization. (Learning problems which do not involve generalization are effectively storage problems.) Thus, solving a learning problem must involve exploiting whatever generalization information is contained within the feedback available to the learner. By taxonomizing such information into a set of effects

and by estimating the complexity of exploiting each distinct effect we can move closer to a generic definition of learning-problem complexity. Turning the definition into an operational measure turns out to be less than straightforward. But I will show that there are approximate methods that can be profitably employed.

2 A taxonomy of justifications

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

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

There are two main cases to consider. Supervisory feedback (i.e., sets of examples) can justify probability assignments either *directly* or *indirectly*. Direct justification is provided if the probabilities in question are observed directly within the training examples (in the form of frequencies). They are justified indirectly if, though not observed directly, they can nevertheless be derived from those examples; or if they can be observed in data which are derived from the original data (which amounts to the same thing).

The distinction between the direct and indirect forms of justification can be illustrated with an example. Consider the following training set. This is based on two input variables (x1 and x2) and one output variable (y1). There are six training examples in all. They are laid out with one example per line. An arrow separates the input part of the example from the output part. The values of the two input variables appear on the left of the arrow. The value of the output variable appears on the right.

x1	x2		y1
1	2	-->	1
2	2	-->	0

¹The mapping may be one-to-many, e.g., in the case where the mapping effectively implements a behaviour which can be achieved in a variety of ways.

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

A wide variety of probabilities can be observed directly in these training examples. To begin with we have the probabilities for first-order cases (i.e., instantiations of a single variable). These are shown in Table 1. The ‘C’ column shows the case in question and the ‘P(C)’ column shows the observed probability of that case. We can also observe the probabilities of many second-order cases

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

Table 1:

— cases involving the instantiation of two variables. These are shown in Table 2. Note that the probabilities for the third-order cases (i.e., the cases that specify

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

Table 2:

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 probability is necessarily $1/n$ where n is the size of the training set.

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

C	P(C)	P(y1=0 C)	P(y1=1 C)
	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 3:

In the supervised learning scenario, it is the conditional and unconditional probabilities relating to the output variable(s) which are of interest. Thus, Table 3 in conjunction with the table listing the first-order unconditional probabilities for the output variable, provide an exhaustive enumeration of all directly observed, probability-assignment justifications. A quick perusal of the two tables shows that none of the output-variable probabilities are particularly extreme. Thus the training examples do not contain directly observable justification for strong output predictions (i.e., predictions made with a probability close to 1).

What, then, of the indirectly observed justifications? Imagine that we recode our training examples 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 below (the value of x4 here is the difference between the values of x1 and x2).

Original pairs			Derived pairs (x4 = x1-x2)	
x1	x2	y1	x4	y1
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

The probabilities we *directly* observe in these derived training data are indirectly observed probabilities with respect to the original training data. However,

they are still probabilities. Thus we can derive tables of conditional and unconditional probabilities in the usual way. The unconditional and conditional output probabilities for the derived training set are shown in Table 4. Note

C	P(C)	P(y1=0 C)	P(y1=1 C)
	1	0.5	0.5
x4=1	0.5	0.0	1.0
x4=0	0.33	1.0	0.0
x4=2	0.17	1.0	0.0

Table 4:

how the recoding we applied to the training examples has produced a training set in which we observe a number of extreme probabilities relating to the output variable $y1$. The recoding thus provides us with indirect justification for predicting $y1=0$ with a probability of 1, if the difference between the input variables is 1. It also provides us with indirect justification for predicting $y1=1$ with a probability of 1, if the difference between the input variables is either 2 or 0. In short, we have indirect justification for the output rule ‘ $y1=1$ if $x4=1$; otherwise $y1=0$ ’.

3 Complexity implications for type-1 problems

As I noted above, the aim in learning is to be able to identify target outputs with high probability. We have now seen how the justifications for such assignments contained within the learner feedback (i.e., training examples) can be classified as directly or indirectly observed. Discovering direct forms involves deriving probability statistics. Discovering indirect forms involves (1) deriving a recoding of the training examples and (2) deriving probability statistics within the recoded data.

From this we can draw a preliminary conclusion regarding the generic complexity of learning problems. Finding a solution to a particular learning problem necessarily entails discovering some combination of two forms of justification. The number of direct (henceforth ‘type-1’) justifications is exponentially related to the number of variables and values involved in the training examples. Thus it may be large but it is necessarily finite.

The number of indirect (henceforth ‘type-2’) justifications is the number of direct justifications (derivable from the relevant recoded data) plus the number of possible recodings of the data. The number of possible recodings is simply the number of distinct Turing machines we can apply to those data. There are infinitely many of these. Thus the space of indirect justifications is infinitely large.

The task of discovering justifications is naturally viewed as a search. In the case of direct justifications the search takes place in a finite space while in the case of indirect justifications the search takes place in a space that is infinitely

large. Thus, other things being equal, learning problems which necessitate the discovery of indirect justifications are ‘harder’ than problems which do not do so. This is the first and most fundamental observation to be made regarding generic learning-problem complexity.

Unfortunately, translating the observation into a computable measure of learning-problem complexity is not straightforward. The difficulty of a learning problem with a solution based on type-1 justifications (henceforth a ‘type-1 problem’) can be measured in terms of the difficulty of discovering the relevant type-1 justifications. This is a function of (1) the number of justifications involved and (2) the size of the space which has to be ‘searched’.² It is a straightforward matter to calculate the size of the space. But counting the number of type-1 justifications required involves working out what they are. Thus, measuring the degree of difficulty of a type-1 learning problem requires *solving* that problem (i.e., finding its type-1 solution).

This is not necessarily a disadvantage. Even when the search for type-1 effects is executed exhaustively, it may still be tractable. But, in fact, by using learning algorithms which perform a close approximation of the type-1 discovery task, we can effectively measure type-1 complexity much more efficiently than is done via exhaustive search.

The well-known ID3 algorithm [18] is perhaps the most obvious candidate for our purposes here.³ This supervised algorithm builds a solution by recursively splitting the training cases into subsets until the point is reached where each subset is associated with a unique output. The splits are arrived at by dividing up the relevant set of cases according to their value on a particular input variable. Target outputs are generated by finding which output is associated with the leaf node accessed by the relevant input.

The efficiency of the algorithm derives from the fact that, at each state, a split is selected which maximizes the output uniformity (minimises the entropy) of the subsets produced. In the initial case — where a split is sought for the complete training set — the process is effectively identifying the input variable which correlates most strongly with the outputs. This variable is the one which participates in the most pronounced conditional probability effects with the output variable(s). ID3’s initial aim, therefore, is to identify the input variable which participates in the most informative set of first-order, type-1 justifications. In cases where these justifications provide a satisfactory solution to the problem, the algorithm is guaranteed to find the ideal type-1 solution.

If the type-1 solution involves higher-order probability effects then ID3 is not guaranteed to find the ideal type-1 solution. But in many cases this does not diminish the algorithm’s suitability as an approximate type-1 discovery method. Supervised learning problems are typically prepared so as to ensure statistical independence of input variables. If this property is obtained, then higher-order justifications will not exist and any type-1 solution will necessarily be the first-order solution identified by ID3.

²Of course, there is no commitment in this to the idea that the learning method actually carries out a search.

³See also the latest variant of ID3 called C4.5 [3].

Various other learning algorithms exist which provide effective methods for accessing type-1 justifications. Such methods are often biased towards first-order justifications (cf. the Least-Mean-Squares [19] and Perceptron [20] methods) and are thus subject to the same reservations — and recommendations — as the ID3 algorithm. Methods related to backpropagation [1], which do not have such an obvious first-order bias, can potentially be used to find higher-order type-1 solutions (see further discussion below).⁴

4 Complexity implications for type-2 problems

Type-2 justifications are probability effects which are only brought to light via a recoding of the original training data. (Intuitively, they are effects which are discovered via an ‘analytic’ process.) The fact that there are typically infinitely many possible recodings that might be applied to any given training set means that measuring the difficulty of a type-2 problem (a problem whose solution is based purely on type-2 effects) is impossible unless we set constraints on the nature of the recoding that can be applied.

Before looking at what this means, I will make some preliminary observations about the nature of type-2 effects. Recall that a type-1 effect is an output probability which is either unconditional or conditional on the absolute state of one or more input variables. A type-2 effect is an output probability which is associated in some way with the states of input variables. But it cannot depend in any way on the *absolute* states of input variables since this would make it a type-1 effect. Thus it must be associated with the *relative* states of input variables. Putting it more simply, a type-2 effect must be based on a relational property of the input variables.⁵

This provides us with a rule-of-thumb for deciding whether a learning problem has a type-1 or a type-2 solution. If the problem is based on a relational input/output rule, then probability effects affecting the output variable(s) cannot be based on absolute values of the input variables. Thus the solution can be expected to be found in type-2 effects. If the problem is based on a non-relational input/output rule, then the probability effects applying to the output variable must be based on absolute values of the input variables. Thus the solution can be expected to be found in type-1 effects.

Of course, in reality, nothing is this simple! Learning problems which are based on relational input/output rules typically have training sets which exhibit ‘spurious’ type-1 effects. The training set shown above is a case in point. The

⁴The Perceptron learning algorithm is, of course, a method which can only be successfully applied to linearly separable problems, i.e., discrimination problems which can be solved by identifying a linear hyperplane separating the relevant classes. Where the hyperplane is aligned with one of the axes of the input space, there will be marked correlations between values of the corresponding input variable and the output variable, and thus pronounced type-1 effects. However, where the hyperplane is unaligned such effects may disappear.

⁵In fact this is not quite true. The argument stated allows it to be based on a non-relational property of a single input variable provided that that property effectively masks any type-1 effect.

input/output rule is relational and yet the probability analysis reveals that the training set does exhibit type-1 effects, albeit rather weak, affecting the output variable.

Training sets for ‘parity problems’ provide the extreme case here. They are, of course, guaranteed to be free of all type-1 effects, provided that they are exhaustive (i.e., include the entire target mapping). The input and output variables in a parity problem have binary values. The target output is 1 (i.e., true) if the input contains an odd number of 1s; otherwise it is 0. A training set for a parity problem based on three input variables is shown below.

x1	x2	x3		y1
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

Type-1 probabilities for an exhaustive parity training set (such as the one shown) are always precisely at the chance level. The probabilities shown in Table 5 are for the 3-bit parity training set shown above. That this state of affairs is a necessary outcome is easily shown. The nature of the parity problem guarantees that when we look at an (n-m)th-order case ($n > m > 0$), one half of the possible states of the other variables will be associated with an output of 1 and the other half will be associated with an output of 0. Thus the conditional probabilities associated with any chosen case must be at the chance (50/50) level. This shows us that all non-degenerate type-1 probabilities for parity mappings will always be at the chance level. The unconditional output probabilities are at the chance level by definition. Thus all the type-1 output probabilities are at the chance level.

Exhaustive training sets for parity problems then, show no type-1 effects whatsoever.⁶ All other problems are likely to show type-1 effects. This includes parity problems with non-exhaustive training sets together with all those problems which are based on relational input/output rules and which therefore should *not* show type-1 effects in principle. From the engineering point of view, the news is relatively welcome. It implies that problems in the ‘difficult’ type-2 class may often be solved by exploiting readily accessible type-1 effects. From the theoretical point of view, however, the news is not so good. It means that the presence of type-2 effects cannot be determined simply by establishing the *absence* of type-1 effects.

⁶In fact recent work by the author has shown that *all* problems which can be converted into modulus-addition problems show no type-1 effects.

C	P(C)	P(y1=0 C)	P(y1=1 C)
	1	0.5	0.5
x3=0	0.5	0.5	0.5
x2=0	0.5	0.5	0.5
x1=0	0.5	0.5	0.5
x3=1	0.5	0.5	0.5
x2=1	0.5	0.5	0.5
x1=1	0.5	0.5	0.5
x2=1, x3=1	0.25	0.5	0.5
x1=1, x3=1	0.25	0.5	0.5
x1=1, x2=1	0.25	0.5	0.5
x1=0, x3=1	0.25	0.5	0.5
x1=0, x2=1	0.25	0.5	0.5
x2=0, x3=1	0.25	0.5	0.5
x1=1, x3=0	0.25	0.5	0.5
x1=1, x2=0	0.25	0.5	0.5
x2=1, x3=0	0.25	0.5	0.5
x1=0, x3=0	0.25	0.5	0.5
x1=0, x2=0	0.25	0.5	0.5
x2=0, x3=0	0.25	0.5	0.5

Table 5:

But let us leave this problem on one side and focus on the more fundamental issue of how to measure the difficulty of discovering type-2 effects. A type-2 effect is a probability effect observed in a recoding of the training set. As we noted above, the search space for type-2 effects is potentially infinite since there are infinitely many recodings that might be applied to the training data. However, the complexity of discovering a particular type-2 effect is simply the amount of searching that has to be done to find the relevant recoding. To estimate the complexity of a type-2 problem we therefore have to establish how much searching has to be done to find the recoding(s) which encapsulate the relevant probability effects.

The full search space for type-2 effects is infinitely large. Thus all we can say with respect to an unguided search for type-2 effects is that it may take an infinitely long time. To get beyond this rather unhelpful conclusion we have to make assumptions about the way in which the search is carried out. The assumptions we make will clearly depend on contextual issues, e.g., the reason why we are interested in learning in the first place. However, an attractively general assumption is that all realistic learners will face resource limitations and will thus choose to ‘search’ the space of recodings in such a way as to consider simpler recodings first. Under this assumption, the complexity of discovering a particular type-2 effect is simply the number of recodings which (1) do not reify the effect in question but (2) are simpler than the recoding which does.

With this assumption in place, we are in a better position to derive measurements of type-2 complexity. In some cases we can find a type-2 solution using straightforward search and thus measure difficulty in terms of the amount of search actually performed. Alternatively, we can press existing learning algorithms into service as ‘approximate’ type-2 discovery mechanisms and derive complexity measures directly from their performance statistics.

Unfortunately, the choice of candidate mechanisms is much smaller in the type-2 case than it is in the type-1 case. If we leave aside those mechanisms which are based on domain-specific search [21, 22, 23, 24], we are left with a fairly small field, the main contenders in which appear to be the well-known connectionist learning algorithms such as backpropagation [1] and cascade-correlation [25]. However, the jury is still very much out on the degree to which such algorithms are capable of discovering and exploiting type-2 (i.e., relational) effects [26].

The empirical evidence, then, is somewhat contradictory. For example, it is well known that backpropagation can solve parity problems such as exclusive-or, the notoriously ‘difficult’ problem that could not be solved by perceptron learning [20]. As I showed earlier, exhaustive training sets for parity problems are completely lacking in type-1 effects. Thus if parity problems have a solution it must be based on type-2 effects (which is no surprise since the parity rule is based on a relational property of the input variables) and backpropagation’s successful performance on parity problems would seem to reflect a general ability to discover and exploit relational effects.

But the picture is clouded by the fact that backpropagation reliably *fails* to solve parity problems when they are presented as learning problems, i.e., generalization problems presented using non-exhaustive training sets. The graph shown in Figure 1 was produced from an empirical study that involved running standard backpropagation [1] on 4-bit parity generalization problems (with four cases used as unseens) using a wide range of internal architectures. All the curves in the upper half of the graph are error profiles⁷ 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. Generalization over the testing cases was never observed to improve much beyond the chance level in any of the runs recorded and in fact tended to deteriorate steadily throughout training. Thus there was no over-training effect in which an initial improvement in generalization was followed by a subsequent deterioration. More significantly, the training-set error profiles go to zero rather rapidly. This tells us that the generalization failure occurs in the context of perfectly ‘successful’ learning, i.e., perfect acquisition of the training cases. This, then, is a particularly concrete sort of learning failure since it cannot be overcome by increasing the amount of training or by interrupting the learning at the observed onset of over-training.

Backpropagation’s performance on parity problems thus does not provide unambiguous evidence in favour of the view that the algorithm reliably discov-

⁷The error measure used was the standard RMS (root-mean-squared) error.

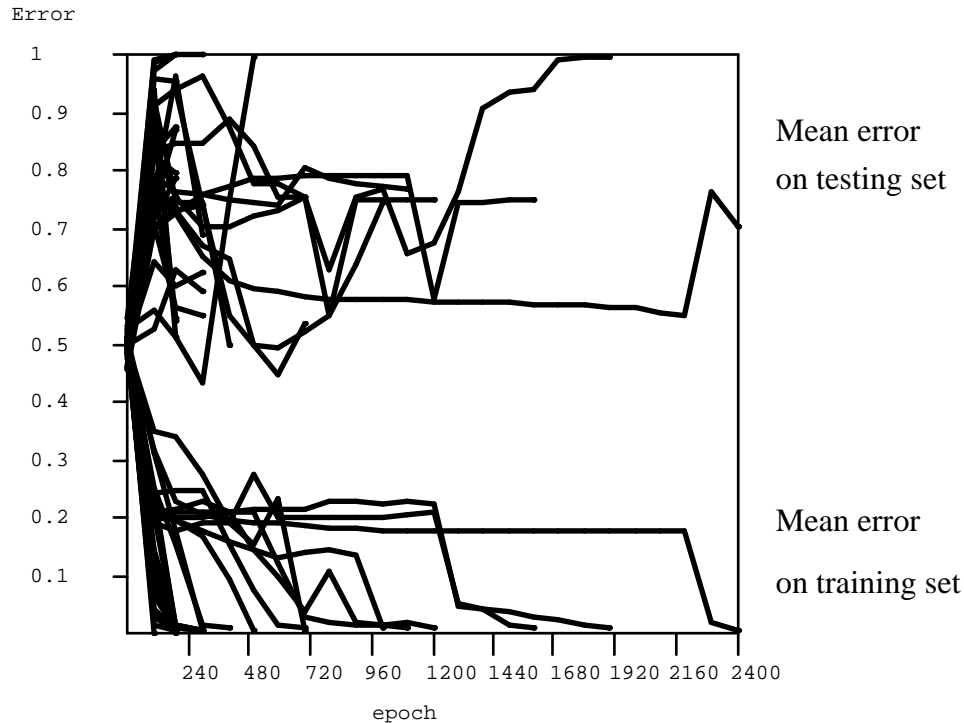


Figure 1:

ers relational effects. There is clearly much still to be discovered about such connectionist procedures. But in the meantime it may be unwise to place too much reliance on the algorithm's performance as a guide to the type-2 complexity of learning problems.

5 Discussion

Contemporary learning theory provides no method by which the generic difficulty of specific learning problems can be measured. Those complexity measures which do exist necessitate making assumptions about the way in which the learner (i.e., the solution method) will operate and effectively disallow the application of the relevant measure to a specific learning problem (training set). This situation tends to militate against the taking of a principled approach to the application of learning methods.

The present paper has shown that supervised learning problems can be conceptualized in terms of the exploitation of probability-assignment justifications, and that these can be classified according to whether they are directly observed or indirectly observed. Directly observed justifications exist in a potentially

large but finite space. Indirectly observed justifications exist in an infinitely large space. This enables us to distinguish two basic complexity classes: the class of type-1 problems (problems whose solutions is comprised of type-1 effects) and the class of type-2 problems (problems whose solution is comprised of type-2 effects). This fundamental distinction between a computationally tractable class of problems and a computationally intractable class of problems is the main contribution of the paper.

I have shown that computing the generic complexity of a learning problem does indeed seem to involve *solving* that problem, whether it be a type-1 problem or a type-2 problem. This is somewhat unwelcome news for the applications engineer for whom an analytic measure of problem complexity would be most useful. The only sweetener is the fact that solving type-1 problems turns out to be a computationally tractable operation which is fairly well approximated by a variety of existing learning algorithms.

Type-2 learning remains problematic in all respects. The difficulty of unearthing a particular type-2 effect is essentially the amount of searching that must be done to find the recoding which brings that effect to light (as a type-1 effect). But this search cost will obviously vary depending on the nature of the search mechanism and the way in which it works through the space of recodings. Were it possible to identify the ‘minimal’ program for a given problem, and to arrange all programs into a ‘simplicity’ ordering, then we would be able to locate the position of any recoding program within the ordering and thus determine the minimal amount of searching that would be necessitated to find it. However, the identification of minimal programs is known to be an NP-hard task and there is at present no satisfactory method for putting programs into a simplicity ordering. Thus the prospects for a generic measure of type-2 complexity seem bleak.

References

- [1] Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323 (pp. 533-6).
- [2] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
- [3] Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. San Mateo, California: Morgan Kaufmann.
- [4] Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, 5 (pp. 239-266).
- [5] Feng, C. and Muggleton, S. (1992). Towards inductive generalization in higher-order logic. In D. Sleeman and P. Edwards (Eds.), *Proceedings of the Ninth International Workshop on Machine Learning (ML92)* (pp. 154-162). San Mateo, California: Morgan Kaufmann.

- [6] Kearns, M. (1990). *The Computational Complexity of Machine Learning*. The MIT Press.
- [7] Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27 (pp. 1134-42).
- [8] Valiant, L. (1985). Learning disjunctions of conjunctions. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 560-566). Los Altos: Morgan Kaufmann.
- [9] Vapnik, V. and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theor. Probab. Appl.*, 16, No. 2 (pp. 264-280).
- [10] Linial, N., Mansour, Y. and Nisan, N. (1988). Constant depth circuits, fourier transform and learnability. *Proceedings of the 30th I.E.E.E. Symposium on Computational Learning Theory* (pp. 56-68). Morgan Kaufmann Publishers.
- [11] Wittgenstein, L. (1958). *Philosophical Investigations*. Oxford: Basil Blackwell.
- [12] Chaitin, G. (1987). *Algorithmic Information Theory*. Cambridge: Cambridge University Press.
- [13] Chaitin, G. (1966). On the length of programs for computing finite binary sequences. *Journal of The Association of Computing Machinery*, 13 (pp. 547-569).
- [14] Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Prob. Inf. Trans.*, 1 (pp. 1-7).
- [15] Solomonoff, R. (1964). A formal theory of inductive inference i and II. *Information and Control*, 7 (pp. 1-22 and 224-254).
- [16] Dietterich, T., London, B., Clarkson, K. and Dromey, G. (1982). Learning and inductive inference. In P. Cohen and E. Feigenbaum (Eds.), *The Handbook of Artificial Intelligence: Vol III*. Los Altos: Kaufmann.
- [17] 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.
- [18] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1 (pp. 81-106).
- [19] Hinton, G. (1989). Connectionist learning procedures. *Artificial Intelligence*, 40 (pp. 185-234).

- [20] Minsky, M. and Papert, S. (1988). *Perceptrons: An Introduction to Computational Geometry* (expanded edn). Cambridge, Mass.: MIT Press.
- [21] Langley, P., Simon, H., Bradshaw, G. and Zytkow, J. (1987). *Scientific Discovery: Computational Explorations of the Creative Processes*. Cambridge, Mass.: MIT Press.
- [22] Lenat, D. (1983). Theory formation by heuristic search; the nature of heuristics II: background and examples. *Artificial Intelligence*, 21 (pp. 31-59).
- [23] Lenat, D. (1983). EURISKO: a program that learns new heuristics and domain concepts: the nature of heuristics III: program design and results. *Artificial Intelligence*, 21 (pp. 61-98).
- [24] Hofstadter, D. (1984). The copycat project. A.I. Memo 755, Massachusetts Institute of Technology.
- [25] Fahlman, S. and Lebiere, C. (1990). The cascade-correlation learning architecture. In D.S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2* (pp. 524-532.). Morgan Kaufmann Publishers, Los Altos CA.
- [26] Clark, A. and Thornton, C. (1993). Trading spaces: computation, representation and the limits of learning. Cognitive Science Research Paper 291, Brighton BN1 9QH: University of Sussex (Price:1.50).