Knowledge Considered Harmful

Shaul Markovitch
&
Paul Scott

Research Paper #030788

Center for Machine Intelligence
2001, Commonwealth Blvd
Ann Arbor, Michigan 48105

1. Introduction

The dominant theme of AI research for the last two decades has been the central role of knowledge. As Buchanan and Feigenbaum (1982) put it "*the power of an intelligent program to perform its task well depends primarily on the quantity and quality of knowledge it has about that task*". Hence there is a widely held view that it is always beneficial for a program to be given additional knowledge about its task domain. In this paper we consider how far this view is correct. In particular we ask whether additional knowledge can actually be detrimental and if so under what circumstances. We also consider what form the detrimental effects may take and examine some of the possible remedies. We shall only be concerned with additional knowledge which is correct. The problems arising from the acquisition of incorrect knowledge are of great importance but they fall outside the scope of this paper.

Suppose that we have a knowledge-based problem solving system that is capable of solving problems in some task domain. Suppose further that we provide it with additional correct knowledge about that domain. There are two kinds of detrimental effect which the additional knowledge might produce. It could impair the actual process of solving problems by requiring more resources, such as time or memory space, to produce the same solutions. It could also lead to the production of poorer solutions, the extreme case being solutions that are wrong. We will consider each of these kinds of detrimental effect in turn.

2. Knowledge That Impairs Problem Solving

The reason that knowledge can sometimes have a deleterious effect on problem solving performance is that there are costs as well as benefits associated with adding knowledge. These

costs take various forms. The most obvious is the additional memory requirement. The significance of this depends on how close to its memory limits the program is operating. Of greater importance for many systems is the effect on search time. Adding knowledge to a system is likely to increase the size of the space that must be searched during problem solving. Hence adding knowledge to a system may slow it down. The real value of a piece of knowledge to a system is the difference between the benefits and the costs. If the costs exceed the benefits then the knowledge is harmful.

2.1. Irrelevant Knowledge

The simplest example of knowledge whose net value is negative is knowledge which is irrelevant; that is, knowledge which, though correct, can never be used in the process of solving problems drawn from the task domain of the problem solver Such knowledge produces no benefits to the system and yet has costs. Hence its value is negative.

In principle, irrelevant knowledge could be identified by solving all the problems in the task domain and determining if the knowledge was ever used. In practice the relevance must be estimated from the solutions to a sample set of problems. Learning systems typically attempt to eliminate irrelevant knowledge in one or both of two ways. The first is to ensure that the knowledge has some relevance by only acquiring knowledge which has made a contribution to solving a problem drawn from the task domain. The second is to monitor the use of retained knowledge and discard that which does not make a significant positive contribution. Such monitoring may involve simply noting the frequency with which a piece of knowledge is used or it may also consider the magnitude of the contribution that is made.

2.2. Learning to Search State Spaces

It might reasonably be conjectured that the addition of knowledge to a system would not impair the process of problem solving if steps were taken to ensure that the knowledge added to a system was both correct and relevant. We have run a series of experiments with a simple learning program, FUNES, to demonstrate that this conjecture is wrong.

Our work builds on that of Minton (1985), who observed that systems which learn by storing successful solutions (or partial solutions) to problems gradually become "swamped" by acquired knowledge. His program, MORRIS, was written to assess whether the problem could be eliminated or reduced by retaining only those solutions which satisfy his proposed criteria of usefulness. Our concern is with investigating the nature and extent of the "swamping" rather than with testing potential remedies.

FUNES (Markovitch & Scott, 1988) solves problems by performing best first search of a state space representation. Learning takes the form of retaining sequences of state transitions which have been encountered in the course of solving problems in a training set. These transition sequences are called macros. During subsequent problem solving, macros are used in the same way as simple transitions in the search for solutions. Only macros which have formed part of a solution during training are retained. Hence all the additional knowledge the system acquires is relevant

.

An experiment comprises two phases: a learning phase followed by a forgetting phase. During the learning phase the system solves randomly generated problems and retains all sequences which form part of solutions as macros. During the forgetting phase, in which no further learning takes place,

macros are removed *at random*. At intervals during both the learning and forgetting phases performance of the system is evaluated by giving it a sample of 100 randomly generated problems. The mean number of nodes visited during the process of solving the problem is used as a performance metric. No learning or forgetting occurs during evaluation.

The results of a typical experiment are shown in Figure 1. In this case the learning phase was continued until 3000 macros had been acquired.

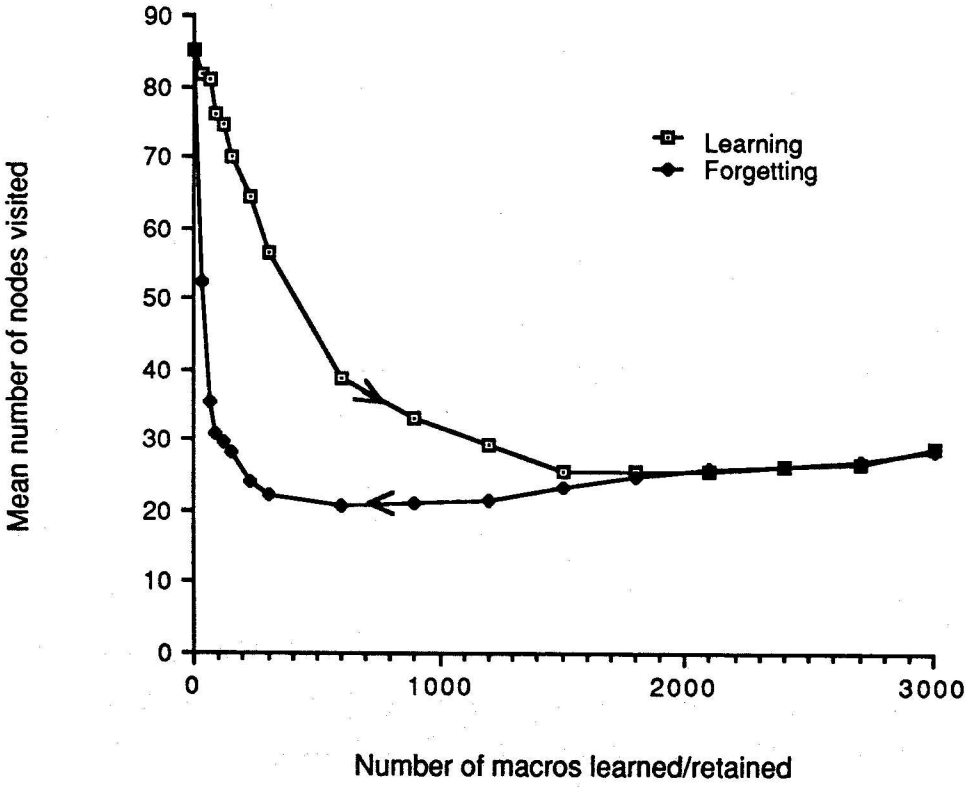

Figure 1

During the initial portion of the learning phase performance continues to improve until it reaches an optimal level of 25.6 nodes visited after about 2000 macros have been learned. Beyond this point the performance slowly but steadily deteriorates. This is the "swamping" effect , noted by Minton (1985), in which newly acquired macros increase the search space without

contributing compensating benefits. During the initial portion of the forgetting phase the performance level retraces the learning curve. Thus the performance slowly improves. However, once the forgetting curve reaches the optimal value achieved during learning the curves separate as the forgetting curve continues to show steady improvement right down to an optimal value of 20.8 at a point where only about 500 macros are retained. This is about 20% better than the optimal performance achieved during the learning phase and about twice as good as the performance with the same number of macros during the learning phase. Beyond this optimal point in the forgetting phase performance deteriorates rapidly to reach the initial performance level when all the acquired macros have been discarded.

Experiments with shorter and longer learning phases produce essentially similar results. In experiments with longer learning phases the swamping continues monotonically during learning and the forgetting curve retraces the learning curve down to its optimal point and then continues on to its own optimal value. In shorter learning phases, in which the optimal point is not reached, the forgetting curve shows immediate improvement which continues to an optimal value which is poorer than that achieved for longer learning phases.

The most striking feature of these results is that substantial performance improvements are achieved by *random* removal of knowledge. This finding appears to be extremely paradoxical because it implies that it does not matter what is forgotten. When the number of macros retained gets down to about 500 an optimum performance level is reached. Beyond this point performance deteriorates rapidly. Thus learning 2000 macros and discarding a random 75% of them leads to very much better performance than simply learning 500.

The explanation of this paradoxical result lies in the fact that most of the knowledge acquired is redundant. After the system has acquired 2000 macros it has numerous alternative ways of traversing the space and hence the branching factor at each node is large. Because of the redundancy drastic pruning is possible without any deleterious affect on performance. Thus the random forgetting eliminates redundant paths and hence the search space is reduced. Only when the random deletion starts to remove non-redundant paths is there a deterioration in performance.

2.3. Redundant Knowledge

This examples shows that redundant knowledge can have a deleterious effect on performance. Redundant knowledge differs from irrelevant knowledge in two ways, each of which makes it a more difficult problem to deal with.

First, while irrelevant knowledge is at best harmless, redundant knowledge is often very useful. All the additional knowledge learned by FUNES is redundant because the state transitions which make up any macro were already known at the start of learning. The acquired knowledge does not permit any additional solutions to be found but is useful because it speeds problem solving. A large number of learning programs only acquire knowledge which is, in this sense, redundant. Examples include STRIPS (Fikes, Hart & Nilsson, 1972), Korf's (1983) macro-operators, SOAR (Laird, Rosenbloom & Newell, 1986), EBL (Dejong & Mooney, 1986) and EBG (Mitchell, Keller & Kedar-Cabelli, 1986). A particular piece of knowledge may speed the solution of some problems while retarding the solution of others. Hence deciding whether a piece of redundant knowledge is worthwhile involves considering its effects on both increasing and reducing search time on a population of problems. The problem therefore is not

simply a matter of eliminating redundancy but rather of eliminating harmful redundancy.

The second difference is that while the irrelevance of a piece of knowledge for a given task domain is a property of that knowledge alone, redundancy depends on the entire set of knowledge that a system has. One cannot speak of an isolated piece of knowledge as being redundant. It only makes sense to say either a piece of knowledge is redundant in the context of certain other knowledge or, equivalently, an entire corpus of knowledge exhibits redundancy.

Harmful redundancy is clearly undesirable. Unfortunately, identifying and eliminating it is an inherently hard problem. Suppose we have some set, K, of pieces of knowledge from which we wish to eliminate the harmful redundancy. The elements of K may be rules, propositions, or whatever else is the basic module of the representation system. Let us define Cost(x) as being the expectation value of the cost of solving a problem from a task domain using the knowledge set x. Then eliminating all harmful redundancy can be viewed as finding the subset of K with minimal cost. That is of finding $K_{min} \subseteq K$ such that

$$\forall s \subseteq K \, [\, Cost(K_{min}) \leq Cost(s)]$$

In general this minimization will be of exponential complexity because finding the minimum will involve evaluating the cost function for all subsets of K.

It might appear that the complexity of the problem can be reduced by considering only the effect of adding a new piece of knowledge k to a knowledge base K to make a new knowledge base K'. Assuming Cost(K) is known it is then only necessary to evaluate Cost(K') in order to determine whether adding k will be beneficial. Unfortunately this approach suffers from two disadvantages. First, there may be some subset of K' other than K

which has a lower cost than K' so this method does not guarantee that the knowledge base is optimal (in the sense that it could not be improved by removing some of the knowledge). Second the method is one of hill-climbing and requires that each piece of knowledge added should reduce the cost. Hence there may be global minima which are inaccessible from the current knowledge state of the system.

It should also be noted that the computational effort required to identify redundancies depends also on what is involved in evaluating the cost function. This is required to return the expectation value for the cost of solving a problem drawn from the task domain. Typically it will involve solving a representative sample of problems and thus the effort required will depend on both the effort needed to solve a single problem and the number of examples needed to form a representative sample of the task domain.

Hence we can conclude that the problem of eliminating harmful redundancies is inherently difficult. This conclusion is in no way confined to the particular problem space in which FUNES operates. Redundancy can occur in all but the most trivial representations and in many cases it is beneficial. Since eliminating harmful redundancy algorithmically is too expensive a procedure to be of practical value, heuristics must be sought which eliminate significant amounts with much less effort. Minton's (1985) studies with MORRIS show that the heuristics he proposed led to improved performance but our own experiments with FUNES raise the question of whether Minton's heuristics are significantly better than random forgetting which is itself very successful. (We have also investigated the effects of alternatives to random forgetting. An account of these studies together with a

more detailed account of the program form the subject of Markovitch & Scott (1988)).

Both our experience with FUNES and our discussion of the inherent complexity of finding optimal subsets of knowledge suggest that better performance may be achieved by acquiring a large body of knowledge and then discarding part of it than by attempting to retain only worthwhile knowledge as it is acquired. In view of the current interest in incremental learning (Schlimmer & Fisher, 1986) this is an interesting conclusion since it suggests there may be limits to the quality of representation that can be achieved in this way.

3. Knowledge That Leads To Poor Solutions

The examples considered in the preceding section all involve poorer performance in the sense that more resources are required to find a solution. In some circumstances additional correct knowledge may lead to poorer performance in the sense that the quality of solutions produced is lower.

Wilkins (Wilkins & Buchanan, 1986; Wilkins, 1987) has described the occurrence of this phenomenon in rule based systems which use the certainty factor representation of uncertainty originally introduced in MYCIN (Buchanan & Shortliffe, 1984). In such a system correct uncertain rules will sometimes contribute evidence towards incorrect conclusions. The frequency with which incorrect conclusions are drawn will depend upon both the other rules present in the system and the distribution of problems in the task domain. Hence in some circumstances the addition of a correct new rule to the system will result in an increase in the number of mistakes the system makes. Wilkins describes such knowledge bases as *sociopathic*.

Sociopathic knowledge has much in common with the problem of harmful redundancy. The detrimental effects of a piece

of knowledge depend upon what other knowledge is present in the system. Hence it makes more sense to speak of a knowledge base exhibiting sociopathy than to say a particular piece of knowledge is sociopathic. As with harmful redundancy, the fact that sociopathy arises from the interactions between various pieces of knowledge means that the problem of identifying and eliminating it is computationally hard. Wilkins has shown that the general problem of finding that subset of a sociopathic MYCIN type rule base which gives optimal performance is NP-Complete. As with harmful redundancy, the elimination of sociopathy must be accomplished by heuristic methods.

Wilkins also argues that the problem of sociopathy implies that traditional incremental methods of improving a knowledge base are inadequate. This is the counterpart of our conclusion that the problem of reducing harmful redundancy may place limits on the quality of representation which can be achieved incrementally.

It is difficult to assess the generality of Wilkin's findings. In Wilkins (1987) it is claimed without proof that the same problems occur in all representations of uncertainty such as fuzzy sets and Dempster-Shafer belief functions. No mention is made of Bayesian inference. In fact the problem would not occur in a Bayesian representation because in that system it is not possible to draw invalid conclusions from correct premises. More generally the problem of sociopathy can only arise in representation systems whose rules of inference allow the deduction of incorrect conclusions.

This might suggest that the solution to this problem is to use a representation in which sociopathy was impossible. Unfortunately this is not a practical conclusion. Such representation schemes (of which the Bayesian representation of uncertainty is a good example) require far more knowledge than is

generally available. Systems like MYCIN's certainty factor representation are essentially schemes to permit useful inferences to be made without the need for very large quantities of knowledge. The price paid is the potential for error.

It seems plausible to conjecture the phenomenon of sociopathy is inherent in any representation scheme which allows incorrect inference to be drawn and hence will arise in any practical representation scheme which allows useful inferences to be made about non-trivial real world problems.

4. Conclusions

In this paper we have shown that in some circumstances correct knowledge can be harmful. We have identified three types of knowledge which impair either problem solving performance or the quality of solutions. Irrelevant knowledge and harmfully redundant knowledge have a deleterious effect on performance while sociopathic knowledge leads to poorer solutions.

Harmful redundancy and sociopathy have some important features in common. Each is a property of an entire knowledge base rather than of a particular piece of knowledge. Consequently procedures to minimize them are inherently complex since any such algorithm must consider all possible subsets of the knowledge base. Both of them impose limits on the quality of a knowledge base that may be acquired or learned by purely incremental methods.

4.1. Implications for machine learning

These conclusions have a number of implications for the development of machine learning systems. Perhaps the most important is that the fact that knowledge can be harmful means that those developing learning systems must consider whether their systems actually do acquire harmful knowledge. Suppose for

example that the results displayed in the initial phase of the learning curve in Figure 1 had been obtained from a new learning system. The designer would doubtless have been pleased with a learning curve showing monotonic improvement leading to a performance about three times as good as that obtained without learning. However, as the rest of Figure 1 shows it is possible to achieve a significantly better performance using only a quarter as much knowledge. Hence it is well worthwhile for those developing learning systems to investigate whether their systems carry a significant burden of harmful redundancy.

There are two basic strategies which a learning system might use in order to avoid an accumulation of harmful knowledge: selective acquisition (ie avoiding acquiring knowledge which is harmful), and forgetting (ie detecting harmful knowledge and eliminating it). Unfortunately the conclusion that minimizing harmful knowledge is computationally hard presents serious problems for both these strategies. Selective acquisition suffers from the further disadvantage that it is basically a hill climbing method. Hence it is unlikely to be feasible to build systems which never acquire harmful knowledge and heuristic methods must be used to reduce it to acceptable levels. It seems likely that a combination of both selective acquisition and forgetting will be the most efficient method of achieving this goal. (For further discussion of forgetting strategies see Markovitch & Scott (1988)).

5. References

Buchanan,B.G. & Feigenbaum,E.A. , 1982, Foreward to *Knowledge Based Systems in Artificial Intelligence*, R.Davis & D.B.Lenat, McGraw-Hill

Buchanan,B.G. & Shortliffe,E.H. , 1984, *Rule-Based Expert Systems* , Addison-Wesley

Dejong,G. & Mooney,R. , 1986 , *Explanation-based Learning: An Alternative View* , Machine Learning 1 pp 145-176

Fikes,R.E., Hart,P.E. & Nilsson,N.J. , 1972 , *Learning and Executing Generalized Robot Plans* , Artificial Intelligence **3** pp 251-288

Korf,R.E., 1983, *Learning to Solve Problems by Searching for Macro-Operators*, Pitman, Marshfield, Mass.

Laird,J.E., Rosenbloom,P.S. & Newell,A. , 1986 , *Chunking in Soar: The Anatomy of a General Learning Mechanism* , Machine Learning 1 pp 11-46

Markovitch,S. & Scott,P.D., 1988 *The Role of Forgetting in Learning*, To appear in Proceedings of Fifth International Conference on Machine Learning, June 1988

Minton,S. , 1985 , *Selectively Generalizing Plans for Problem Solving* , Proceedings Ninth International Joint Conference on Artificial Intelligence, Los Angeles, CA. pp 596-599

Mitchell,T.M., Keller,R.M. & Kedar-Cabelli,S.T. , 1986 , *Explanation-based Generalization: A Unifying View* , Machine Learning 1 pp 47-80

Schlimmer,J.C. & Fisher,D. , 1986 , *A Case Study of Incremental Concept Learning* , Proc. AAAI-86, Philadelphia, August 1986

Wilkins,D.C. & Buchanan,B.G. , 1986, *On Debugging Rule Sets When Reasoning Under Uncertainty*, Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence, Philadelphia, pp 448-454.

Wilkins,D.C. , 1987, *Apprenticeship Learning Techniques for Knowledge Based Systems*, University of Michigan doctoral dissertation, November 1987.