[Apeared in Proceedings of The Fifth International Conference on Machine Learning . Ann Arbor, MI: Morgan Kaufmann. (1988).]

The Role of Forgetting in Learning

SHAUL MARKOVITCH
(Shaul_Markovitch@um.cc.umich.edu)
EECS Department, University of Michigan, Ann Arbor, MI 48109 USA
PAUL D. SCOTT
(Paul_David_Scott@um.cc.umich.edu)
Center for Machine Intelligence, 2001 Commonwealth Boulevard, Ann Arbor, MI 48105 USA

Abstract

This paper is a discussion of the relationship between learning and forgetting. An analysis of the economics of learning is carried out and it is argued that knowledge can sometimes have a negative value. A series of experiments involving a program which learns to traverse state spaces is described. It is shown that most of the knowledge acquired is of negative value even though it is correct and was acquired solving similar problems. It is shown that the value of the knowledge depends on what else is known and that random forgetting can sometimes lead to substantial improvements in performance. It is concluded that research into knowledge acquisition should take seriously the possibility that knowledge may sometimes be harmful. The view is taken that learning and forgetting are complementary processes which construct and maintain useful representations of experience.

1. Introduction

Research on machine learning is concerned with the problem of how a system may acquire knowledge that it does not possess. It is therefore not surprising that relatively little attention has been paid to the converse problem: How may a system dispose of knowledge it already possess? This is the phenomenon that is termed *forgetting* when it occurs in humans, and is usually regarded as an unfortunate failure of the memory system. It is our contention that this negative view of forgetting is misplaced and that far from being a shortcoming it is a very useful process which facilitates effective knowledge acquisition. Learning is a process in which an organized representation of experience is constructed (Scott 1983). Forgetting is a process in which parts of that organized representation are rearranged or dismantled. The two processes are thus complementary and the resulting representation is the joint product of both. Mechanisms of forgetting therefore merit study alongside those of acquisition since it is the two together which constitute learning.

Our notion of forgetting is fairly broad. In addition to the obvious mechanism of deletion of items of knowledge it also includes changes in the knowledge structure which render particular items relatively or completely inaccessible. It thus includes processes which weaken memory traces or isolate fragments of a knowledge base. Such changes can be viewed as partial removals with deletion as a limiting case which produces complete removal. In this paper we attempt to explore the role of forgetting in machine learning systems. We begin by discussing the circumstances in which it is better to dispose of an item of knowledge than retain it. Then we describe some experimental work we have done in order to demonstrate that even correct knowledge acquired in the course of solving similar problems can be a disadvantage to a system.

2. The Economics of Learning

The circumstances in which it is to a system's advantage to be able to forget previously acquired knowledge can best be understood by viewing learning and problem solving from an economic perspective. Any form of learning is essentially a form of investment. A system which learns invests a certain amount of resources in acquiring and maintaining some knowledge. It will only be to the system's advantage to learn if the return on that investment is positive. Thus, the cost of acquiring and retaining knowledge must not exceed the value of the resource saving provided by that knowledge. We therefore introduce the notion of the value of knowledge.

2.1. The Value of Knowledge

Let us assume that there is some common currency in which the values of the various costs and benefits associated with the operation of a problem solver may be expressed. Then the value of an item of knowledge may be defined as follows:

Suppose that a problem solver is called upon to solve some problem, and that it determines and executes a solution twice; once using all its current knowledge and once

with one item of knowledge removed. Suppose further that, for each run, P, the cost of solving a problem, and Q, the quality of the solution, are determined. Then the *benefit* achieved by using the problem solver to solve that particular problem may be defined as Q-P. The difference between the two benefits, which may be positive or negative, is the *payoff* of that particular item of knowledge for the solution of that particular problem. Note that the payoff for a particular item of knowledge is likely to depend on the other knowledge the system has at its disposal.

The majority of problem solvers are capable of solving a range of problems. Within this range some problems may occur more frequently than others. Thus one can view the domain of problems which a problem solver is intended to solve as a set of problems together with an associated probability distribution. We will call such a domain the system's *problem space*. If the system attempts to solve a number of problems, chosen from its problem space, it is likely that the payoffs of a given item of knowledge will differ from one problem to another. We therefore define the *value of an item of knowledge* as the expectation value for the payoff which would be achieved in attempting to solve a randomly chosen problem from the problem space.

This definition of value incorporates all the potential benefits an item of knowledge may bring to a system. In particular it reflects both the contribution that the knowledge may produce and the frequency with which that contribution is likely to be made. It also incorporates all the costs associated with retaining the knowledge. Clearly it is to a system's advantage to cease making use of any piece of knowledge with a negative value.

So far we have been concerned with the value of a single item of knowledge. The total value of all the knowledge in the system is also of considerable interest since it should be substantially positive. This is because the system will also have incurred costs during the acquisition of the knowledge. If the total value of the systems retained knowledge does not exceed the cost of acquiring it, the system would be better off if it had not learned at all. Just how positive the total value must be depends on both the cost of acquisition and the number of problem solutions over which that cost can be amortized. However, it is important to note that the total value of all the system's knowledge cannot be obtained by simply summing the values of the individual items of knowledge. This is because the value of a piece of knowledge is a function of whatever other knowledge is present. The value of a set of items of knowledge can be defined in the same way as we have defined the value of an individual item: one considers the benefits produced and cost incurred solving the same problem both with and without the entire set. The value of the whole knowledge base is thus derived by considering the extreme case of solving problems with no knowledge at all.

2.2. Factors affecting the value of knowledge

We now consider some of the attributes of items of knowledge which may influence their value.

2.2.1. Relevance

Knowledge which is of no relevance to future problem solving can never contribute any benefits and hence is typically of negative value.

2.2.2. Correctness

It might seem reasonable to conjecture that relevant knowledge which is correct, that is, knowledge which constitutes an accurate representation of that aspect of the world it purports to represent, would be worth retaining while knowledge which is incorrect would not. The rationale for this conjecture is that correct knowledge will lead to successful performance while incorrect knowledge will lead to performance errors. However, neither of these propositions is always true.

In practice, incorrect knowledge can often contribute to successful performance. An approximate model of the world is necessarily incorrect but may well be better than no model at all. Furthermore, different types of performance error, such as 'misses' and 'false alarms' in a detection task, typically have different associated costs, and an erroneous piece of knowledge which biases the system towards the cheaper type of error may be beneficial. The commonest type of beneficial erroneous knowledge is the inaccurate generalization. For example, the incorrect because over-generalized belief that all grizzly bears are man eaters in all circumstances is certainly better than no knowledge of the possibility that bears can be dangerous.

Less frequently, correct knowledge can be a source of unsuccessful performance. Wilkins (1987) describes a situation in which additional correct knowledge can reduce the frequency with which a medical diagnostic system produces correct diagnoses. Later in this

paper we describe a system in which the acquisition of correct relevant knowledge leads to much slower problem solving.

Hence we conclude that the correctness of an item of knowledge is not a reliable indicator of its value since both correct and incorrect knowledge may be of positive or negative value.

2.2.3. Memory requirements

The most obvious cost associated with retaining knowledge is the cost of the memory space used to store it. How significant this is depends on the cost of memory locations which in turn depends on how scarce they are. In contemporary computer based systems memory is typically a significantly limited resource. On the other hand the storage capacity of the human brain appears to be so vast that it is not clear that shortage of space is the origin of any limitation on human learning. Storage costs are proportional to the time for which the storage is used.

2.2.4. Influence on Search Time

Additional knowledge may either increase or reduce the time needed to solve a problem. The increase arises because there is more to search while a decrease may arise because, in some circumstances, additional knowledge may eliminate part of the search.

An increase can occur in two distinct ways which we term the direct and indirect search time costs. Searching for one item among many by comparing each item in turn with some template specifying the item sought takes a time which is proportional to the total number of items stored. For this reason large memory structures are usually organized by ordering or indexing the items. The effect of such indexing is to reduce the search time to a figure which is, at best, proportional to the logarithm of the number of items stored. Hence the addition of another item to a knowledge base organized in such a manner increases all search times by an amount proportional to log(N+1/N) where N is the number of items already present. This is what we term the *direct search time cost* and is very small for large N. Thus, in a well organized knowledge base, this component of the cost of retaining knowledge is only significant when a large number of new items are being added. This will only occur in systems which are not very selective about the knowledge they acquire. The net effect of this increased time on the cost of problem solving depends on the amount of searching that is involved in solving a problem. For our purposes we are concerned with the expectation value of the total direct search cost for all the searching involved in solving a problem from a given problem space.

The logarithmic cost model for the direct search cost only covers those situations in which a search consists of matching an item against some kind of template. However a very large proportion of programs perform a more sophisticated kind of search in which each item found leads to further possibilities. Common examples are graph searches and backward chaining inference systems. In systems of this type the inclusion of an item of knowledge can produce much larger increases in search time because the search cost is not only that due to the item itself but also the cost of all the other items which the item in question brings into the space that is searched. Such an increase may well be exponential. A common cause of this phenomenon is redundancy which causes a system to explore a number of equivalent solutions to a problem. We call these costs, which accrue through expanding the search space to include items which would not otherwise be considered the *indirect search time cost.*

Some learning systems acquire additional knowledge which incorporates the results of previous searches, thus saving the need to repeat the search and hence reducing the expectation value of search costs. As we will show in the next section, in systems of this nature it is often hard to determine whether the net effect of a new item of knowledge will be an increase or decrease in search time.

3. Learning to Search Graphs

So far our discussion has been very abstract. In order to demonstrate the existence of the phenomenon of negative valued knowledge we have developed a simple program, FUNES[1], which attempts to improve its ability to search graphs by learning.

---

[1]The name was chosen in recognition of the positive view of forgetting taken by Jorge Luis Borges in his short story *Funes the Memorious.*

3.1. The Learning Program

FUNES performs best first searches of arbitrary graphs. For each state FUNES has a list of operators which may be applied in that state. Initially this list contains only *basic operators*: permitted transitions to immediately neighboring states. As searching proceeds the program will discover that other states are accessible via sequences of transitions. FUNES will therefore add additional operators, corresponding to these transition sequences, to its operator lists. These are called *macro operators*. Macro operators can be of any length but FUNES will only acquire macros longer than 2 by combining macros. On subsequent searches the program uses any macro operators associated with a node in the same way as basic operators in the process of node expansion. The intent is that this additional knowledge of the state space will reduce search time. (In this experiment we were not concerned with the quality of the solution which is in fact always close to optimal). Everything that FUNES learns is correct but the amount of additional knowledge that could be acquired in this way is very large. If the state space contains N states and the average number of immediately accessible states is a then FUNES could learn up to $N(N - (a + 1))$ macro operators. In order to provide a bias towards acquiring knowledge of positive value the program only acquired macro operators which formed part of a solution. FUNES is thus a selective rote learner.

3.2. Experimental Procedure

The purpose of our experiments was to investigate the distribution in value across the set of paths learned. An experiment comprised three phases: a learning phase, a knowledge evaluation phase, and a test phase. In the *learning phase* the program solved problems randomly generated from a problem space until it had acquired *K* macro operators. This involved the solution of several hundred problems. We used the number of macro operators learned rather than the number of problems solved as our measure of learning experience because it is easier to compare the effects of forgetting on different runs if the amount learned is always the same. In the *knowledge evaluation phase* the program solved another set of 1000 problems, also randomly generated from the same problem space. During this phase it maintained statistics about how often the various macro operators it had learned were used. As a result of this evaluation each piece of acquired knowledge had an associated value estimate. In the *test phase* batches of 100 randomly generated problems were solved with varying percentages of the macro operators removed. Two conditions were used. In the first the macro operators to be removed were chosen randomly. In the second condition, the macro operators with lowest value estimates were removed. The number of nodes visited was used as a performance measure.

The experiments reported were done with what we term Manhattan spaces. In these the nodes of the state space form a rectangular array. Nodes are only connected to immediate neighbors in the array. If node A is an immediate neighbor of node B then there is a probability *P* that there is a direct transition from A to B. The existence of a transition from B to A is computed independently in the same manner. The value of this type of problem space for our purposes is that it is possible to generate large numbers of distinct problem spaces belonging to the same general class. Furthermore, by varying *P*, it is possible to vary the difficulty of the problem space. In the results reported here, the array was 31 by 31 so there were 961 nodes in the state space. *P* had a value of 0.8. For the results reported we used best first search using the Manhattan distance (sum of horizontal and row displacements) as the heuristic evaluation function. Regions of the problem space were designated as origin and destination zones. Individual problems were generated by randomly selecting a node from the origin zone and a node from the destination zone. The origin zone was a 5 by 15 region along one side of the state space. The destination zone was a similar region along the opposite side, hence the problem space contained 5625 possible problems. *K*, the number of macro operators to be acquired was varied as one of the experimental parameters.

3.3. Experiment 1

The first experiment was designed to demonstrate a somewhat paradoxical result. Learning proceeded until 5000 macros had been acquired and then various percentages were forgotten by random selection. Figure 1 shows the results obtained during the testing phase.

Each point is the mean of 100 trials. A number of features of this graph are worthy of note**.** First, it is clear that the learning was effective. The number of nodes visited when nothing was forgotten was half the number visited before learning began (equivalent to

100% forgotten). Second a further improvement by another factor of 2 was obtained when about 90% of all that had been learned was forgotten. That is the system performed best with only about 10% of what it learned retained. The remaining 90% actually had a detrimental effect. Clearly this is strong evidence for the existence of knowledge of negative value. It should be remembered that all macros learned were correct and all had been part of solutions to earlier problems. However the really striking property of these results is the fact that they were obtained using a strategy of random forgetting. No attempt was made to selectively retain the more valuable knowledge. Consequently we can say that that the system's performance is improved by forgetting 90% of its knowledge even though that knowledge is clearly useful since it improves performance by a factor of 2.

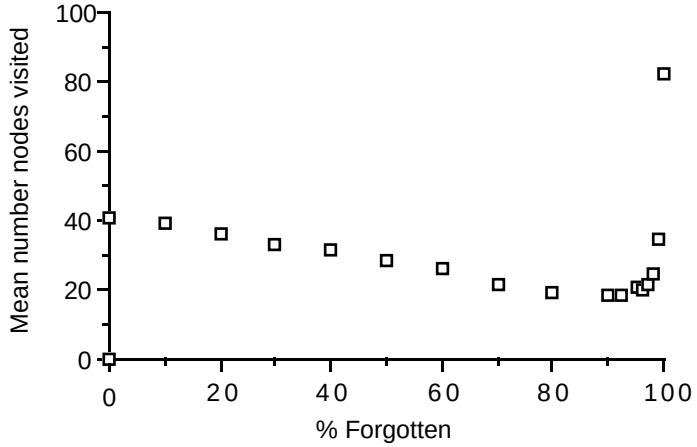

Figure 1. Random forgetting after acquiring 5000 macros

How can these paradoxical results be explained? The answer lies in redundancy. After 5000 training problems the system has learned numerous alternative ways of traversing the space. Most solutions comprise 1 or 2 steps (mean number of nodes expanded with 0% forgetting was 1.58). Because of all the alternative routes the branching factor at each node is large. As it begins to forget this branching factor is steadily reduced without impairing the system's ability to solve problems in 1 or 2 steps. As the branching factor decreases so the number of nodes visited declines and the performance correspondingly improves. Curve fitting showed that the improvement is linear in the region up to 90% forgotten. During this phase each macro deleted has a value of -0.005 nodes visited. However, a point is eventually reached where some of the macros removed are not redundant and the system is forced to engage in searching to remedy their loss. This happens in the present example when about 90% of the macros have been forgotten. At this point the performance declines rapidly. Curve fitting showed that the decline is exponential in the region beyond 95% . During this phase each macro has an average value of +0.247 nodes visited. This is a striking demonstration of the fact that the value of a given piece of knowledge is a function of what other knowledge the system possesses.

3.4. Experiment 2

These results raise an obvious conjecture. If all but a random 500 or so of the items acquired are harmful because of redundancy, would not the system benefit by only learning 500 items in the first place, thus avoiding any need to forget. Our second experiment was designed to test this idea and the results are shown in Figure 2.

The upper curve shows the results obtained using a procedure identical to that of the first experiment except that the training phase was terminated when only 500 items had been acquired. As might be expected, the graph does not show the phase of steady improvement as redundant items are discarded. However, there is an important difference between the curve shown and the latter portion of Figure 1. The best average performance achieved, which occurs when 450 items are retained, is 36.6 nodes visited. This is only half as good as the best performance achieved in the first experiment (18.6 nodes visited when 375 items were retained). So we are forced to conclude that the conjecture that the system could do just as well by avoiding learning so many items is wrong. It is better to learn 5000 items and forget a random 4500 of them than it is to simply learn 500 items.

The explanation for this effect lies in the fact that the two populations of macros acquired are different. The mean length of the macros acquired when 5000 items were

learned was 28.8. In contrast the mean length when only 500 were learned was only 5.1. Since forgetting is random, these means are maintained as items are deleted. The difference arises because the system must necessarily acquire short macros first and only later start putting them together to build longer macros. The longer macros are much more useful since they eliminate far more nodes from the search process. So the system benefits from engaging in more learning and then discarding much of what has been acquired.

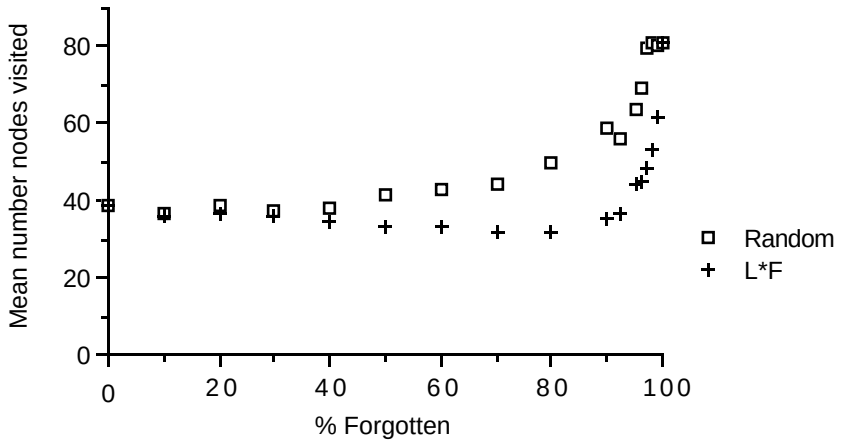

Figure 2. Forgetting after acquiring 500 macros

3.5. Experiment 3

Since the process of forgetting in experiment 2 is discarding the useful longer macros along with the shorter, it might be possible to forget selectively and achieve a performance similar to that achieved by random forgetting from the larger number of items acquired in experiment 1. In order to test this conjecture we repeated the procedure of the second experiment but used the results of the knowledge evaluation phase to discard the apparently less useful items first during the testing phase. The heuristic we used to estimate the value of a macro was the product of the length of the macro and the number of times it was used in solutions in the evaluation phase. This L*F heuristic should identify macros that are both used often and save significant node visits.

The lower curve in Figure 2 shows the results obtained. The results show an improvement over random forgetting. The best performance achieved was 31.5 nodes visited which was achieved when the number of macros retained was 150, which is about 20% better than the optimal value achieved through random forgetting, but is still much worse than the results obtained by random forgetting from 5000 items. The explanation is that in acquiring only 500 items the system had not learned enough to include all the macros it needed to perform well over the entire space. This mode of forgetting did ensure that the harmful redundant macros were discarded but the residual set was not complete.

4. Conclusions

Our graph searching experiments have demonstrated a number of features regarding the value of knowledge many of which are somewhat surprising. First, some knowledge is harmful. This is true even if the knowledge is correct and was acquired in the course of solving problems drawn from the same sample space as that used for testing performance. This result is not an artefact of the particular type of learning we have chosen for our demonstration. Minton(1985) has shown that improved performance can be obtained by selectively discarding knowledge in a system that is based on STRIPS. Minton did not explore the effects of random forgetting or of the extent of training prior to forgetting, but he did show that forgetting heuristics based on frequency of use and value when used could lead to improved performance. (The L*F heuristic we used combines these into a single evaluation). Second, we have shown that the value of a piece of knowledge is highly dependent on what other knowledge the system possesses. This was dramatically demonstrated in experiment 1 which showed that the value of a piece of knowledge could range from positive to negative as a function of how much else was known. We have also shown that surprisingly simple forgetting strategies can lead to better performance. In particular, random deletion of the bulk of a knowledge base led to a substantial improvement in performance. We also showed that in certain circumstances, selective

forgetting of what appeared to be less useful items could lead to slightly better performances than random forgetting.

What are the implications of these results for research on machine learning in general? First the fact that knowledge of negative value is acquired in a system which only learns correct and pertinent items implies that the problem of potentially harmful knowledge should be taken seriously by those constructing knowledge acquisition systems. There are three basic strategies for dealing with negative valued knowledge: ignoring the problem, avoiding the problem by ensuring that only positive valued knowledge is acquired, and forgetting.

Very little of the work on machine learning has directly addressed the issues of harmful knowledge or forgetting. Exceptions include Samuel's (1963) Checker Player, Holland's (1975) Genetic Algorithm, Anderson's (1983) ACT*, and most connectionist systems (Rumelhart & McClelland, 1986). Many systems have either ignored the problem or attempted to ensure that all that was acquired was of positive value. Unfortunately, it is hard to determine the value of knowledge at the time it is acquired. If correctness is a reliable indicator of knowledge value then a cautious approach to acquisition (eg Mitchell's (1977) Version Space algorithm) can be used to obviate the need to forget. However, as our experiments show, in some domains correct knowledge can have negative value. Alternatively, a system can attempt to assess the value of a proposed new piece of knowledge at the time it is created (eg Quinlan 1986) but such evaluations are much harder to devise for systems whose performance involves the more elaborate type of search involved in inference and problem solving. Further discussion of the inherent difficulty of identifying knowledge of negative value will be found in Markovitch & Scott (1988).

References

Anderson,J.R. (1983). *The Architecture of Cognition*, Harvard University Press.

Holland,J.H. (1975). *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press.

Markovitch,S. & Scott,P.D. (1988). *Knowledge Considered Harmful* (Research Paper #030788), Ann Arbor, Michigan, USA: Center for Machine Intelligence.

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

Mitchell,T.M. (1977). Version Spaces: A Candidate Elimination Approach to Rule Learning. In *Proceedings Fifth International Joint Conference on Artificial Intelligence* (pp 305-310). Cambridge, MA: Morgan Kaufmann.

Quinlan,J.R. (1986). *The Effect of Noise on Concept Learning Application* . In Michalski, Carbonell, & Mitchell (1986).

Rumelhart,D.E. & McClelland,J.L. (1986). *Parallel Distributed Processing*. Cambridge,MA: MIT Press.

Samuel,A.L. (1963). Some Studies in Machine Learning Using The Game of Checkers . In E.Feigenbaum & J.Feldman (Eds),*Computers and Thought*, New York, McGraw-Hill.

Scott,P.D. (1983). *Learning: The Construction of A Posteriori Knowledge Structures*. In *Proceedings of the Third National Conference on Artificial Intelligence*. Washington, DC: Morgan Kaufmann.

Wilkins,D.C. (1987). *Apprenticeship Learning Techniques for Knowledge Based Systems.* Doctoral dissertation, EECS Department, University of Michigan, Ann Arbor, MI.