Manually Excising Soar Chunks During Long-term Learning

William G. Kennedy
George Mason University
wkennedy@gmu.edu

Introduction

The Soar system (Laird, Rosenbloom, and Newel 1986) learns by creating chunks for solved subgoals. It then retains all learned chunks unless excised manually (Laird et al. 1993). However, production users of Soar generally turn learning off, as in (Rosenbloom et al. 1994). Preliminary investigations into Soar's use of chunks found as many as half of the chunks were never used. Because the random removal of learned rules was demonstrated to improve the performance of another problem solver (Markovitch and Scott1988), the removal of chunks automatically in Soar is likely to be valuable.

As a prelude to modifying, tests have been conducted using the manual excise function of Soar to test the value of the new functionality.

Purpose

The purpose of this experiment was to manually test the effects of excising unused chunks in Soar during long-term learning.

Method

The approach was to compare performance of an existing Soar system during long-term learning with and without the excising of low-use chunks. The Soar system used was the classifier, Symbolic Concept Acquisition (SCA) (Miller 1993).

SCA is a relatively simple model of human concept acquisition. It is a supervised, symbolic learner that has reproduced some of the characteristics of human concept acquisition found in psychological literature.

From each training example of a concept, SCA learns one new classification rule consisting of a random subset of the training example's feature-values pairs. The first chunk learned contains the smallest, subset of the training examples' feature-values, therefore most general rule. With experience, the system's new rules contain more and more feature-values, becoming more and more specific. When making a prediction, the system prefers the more specific rules and does not learn a new rule.

The classification problem source was king+rook versus king+pawn where the pawn is one square away from "queening." The data was obtained from the University of California, Irvine's collection of machine learning problem data (Murphy 1994). This chess data contains 3,196 instances of two class problems (White-can-win and White-cannot-win) each with 36 attributes. The class distribution is 52 percent White-can-win and 48 percent White-cannot-win.

This chess data was chosen because the relatively large number of attributes and available instances from which to create training examples and testing problems supports relatively long-term problem solving. Long-term problem solving is necessary to test the value of excising chunks. However, the chess endgame classification problem was expected to be beyond the capability of SCA to successfully learn.

The experimental method was to conduct training with 10 sequential sets of 50 examples with testing using 100 problems between each training set for a cumulative total of 500 training examples and 1000 testing problems. Problems for both training and testing were taken at random with replacement from the database of 3,196 instances. The difference between a training example and a testing problem is whether or not the classification category is provided as input. The base case retained all chunks learned and the test case involved excising low-value chunks after the learning set and before the testing. Chunks were considered of low value if they had not been used at all by the end of the training set after the one in which it was learned, i.e., after a minimum of 50 problems and maximum of 100 problems since the chunk was learned. Chunk use during testing was not considered in evaluation of the chunks. Analysis of the trace of Soar's learning identified the number of times each chunk was used. Chunks that were not used at all during the subsequent learning set were excised manually before the next testing set. The pattern was a learning set followed by excising unused chunks followed by a testing set

followed by the next learning set.

Results

The results will be discussed in the areas of learning and chunk use, performance in terms of accuracy, and performance in terms of use of resources.

Learning and Chunk Use

As explained above, SCA always learns one chunk for each training example. Learning consists of adding one more characteristic to the current rule's set, provided it does not invalidate the deduction. If the same training example is repeated, more and more specific chunks are learned. In both the base case and the test case, the number of chunks learned was equal to the number of training examples.

Analysis of chunk use in subsequent training examples showed an increasing number of unused chunks. Figure 1 shows the number of chunks meeting the excising criteria. Figure 2 shows that with manual excising of the low-value chunks, learning was trending toward producing no valuable chunks.

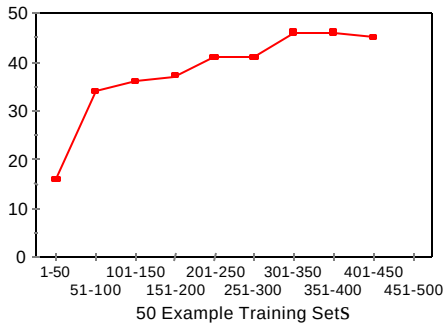

Figure 1 Chunks Meeting Excise Criteria

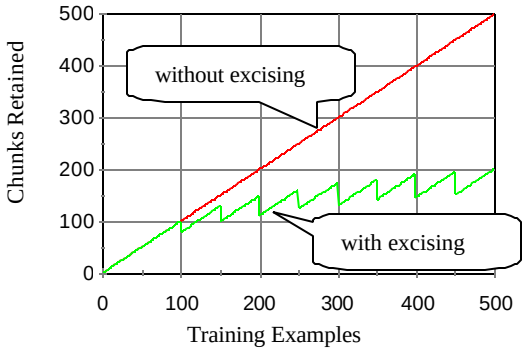

Figure 2 Chunks Retained

After each training set, a different, randomly selected testing set of 100 classification problems was run. Accuracy of the classifications is presented in Figure 3.

Figure 3 Accuracy With & Without Excising

Performance in Terms of Resources

The use of resources is presented in two ways: during learning and during testing. The following resource variables were indistinguishable between learning and testing runs:

total number of production firings,
number of chunks fired,
number of decision cycles,
number of working memory changes,
and maximum working memory size.

However, the CPU time to either learn or solve the classification problems grew over problem solving experience. During the 500 learning examples, the plots of CPU time with and without chunks were indistinguishable. The Figure 4 shows the variations in raw CPU times.

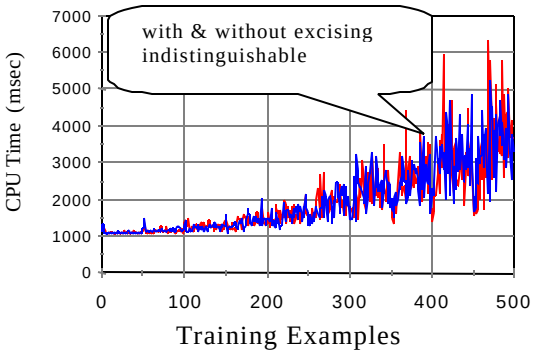

Figure 4 CPU Time During Learning

CPU time during the 100 test problems was separable using a moving average with a width of 10.

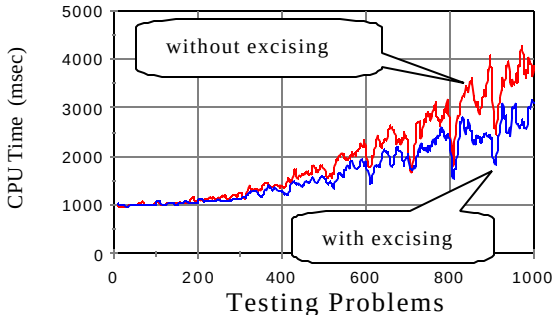

Figure 5 CPU Time During Testing

Analysis

The experiment was able to demonstrate an observable impact of the excising of low-value chunks. The 500 training examples led to the learning of 500 chunks without excising and less than 200 with excising. Although the only resource impact was in CPU time, the difference in the time was becoming very significant, nearly 1 second per problem or approximately 25 percent savings.

The memory involved with the retention of chunks was not monitored but would be expected to show the savings associated with maintaining only 200 of the 500 learned chunks.

Conclusion

This experiment has shown potential to justify adding the ability to excise chunks under program control to the Soar system.

References

Blake, C., Keogh, E. & Merz, C. J., 1998. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/ MLRepository.html], Irvine, CA: Department of Information and Computer Science, University of California.

Laird, J.E., Rosenbloom, P.S., and Newel, A, 1986. Chunking in Soar: The anatomy of a general learning mechanism, *Machine Learning* 1:1, 11-46.

Laird, J. E., Congdon, C. B., Altman, E., and Doorenbos, R., 1993. Soar Users Manual, Version 6, The Soar Group, School of Computer Science, Carnegie Mellon Univ.

Markovitch, S., and Scott, P. D., 1988. The role of forgetting in learning," *Proceedings of the Fifth International Conference on Machine Learning*, 459-465, Morgan Kaufmann Publishers.

Miller, C. S., 1993. Modeling Concept Acquisition in the Context of a Unified Theory of Cognition, Ph.D. diss., The University of Michigan.

Rosenbloom, P.S., Johnson, W.L., Jones, R.M., Koss, F., Laird, J.L., Lehman, J.F., Rubinoff, R., Schwamb, K.B. and Tambe, M., 1994. Intelligent Automated Agents for Tactical Air Simulation: A Progress Report, *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Orlando, FL, 69-78.