A Simple Symbolic Algorithm for Incremental Concept Acquisition*

Craig S. Miller and John E. Laird
Artificial Intelligence Laboratory
The University of Michigan
1101 Beal Ave.
Ann Arbor, Michigan 48109-2110
USA

Phone: (313) 763-9074
Fax: (313) 763-1260
Email: cmiller@engin.umich.edu

January 13, 1992

Abstract

We present SCA, a symbolic algorithm for concept acquisition. SCA is an incremental concept learner that maintains a distributed set of rules for predicting category names. While the design is very simple, it deviates from previous approaches in machine learning. For learning, starting with general rules, SCA incrementally acquires more specific rules. We empirically demonstrate its unusual behaviors, as well as showing its potential performance as a concept learner. Comparisons with other machine learning systems are made.

1 Introduction

In this paper, we explore the learning behavior and performance of a new incremental supervised learning algorithm. Previously we have presented our approach [Miller and Laird, 1991] as a model of human category acquisition designed within the context of Soar, a unified theory of cognition [Newell, 1990]. Here, however, we present the SCA (Symbolic Concept Acquisition) algorithm separate from the Soar context, as a novel algorithm with interesting learning behaviors that distinguish it from previous approaches in machine learning. Furthermore, we demonstrate SCA's promise as a powerful learning approach in terms of accuracy, noise tolerance and learning rate.

Traditionally, supervised learning algorithms in machine learning have taken three approaches to representing concepts. The first approach seeks a compact symbolic representation that covers the concept's positive examples while excluding negative examples. Searching for a parsimonious concept representation as an approach is defended by Blumer *et al.* (1987) and underlies the working principles of many supervised learning systems. These include decision trees [Quinlan, 1986], which search for the smallest tree that covers the training examples, and the STAR methodology [Michalski, 1983], which seeks the simplest logical formula as represented in disjunctive normal form.

The second approach saves classification examples instead of maintaining a compact explicit concept definition. In categorizing a new example, the algorithm seeks a similar, previously classified example. This approach places a burden on the system to correctly index the appropriate pre-classified example in

*This research was sponsored by a grant from the Nissan Corp. and grant NCC2-517 from NASA Ames.

memory. Example systems include a nearest neighbor method [Aha, 1989], and COBWEB [Fisher, 1987], which indexes instances through a concept hierarchy constructed by an unsupervised clustering strategy. Also, in contrast to the first approach, both of these systems use probabilities or weights which quantify a matching metric by which search for the best exemplar is guided.

The third approach also does not keep an explicit concept definition. It differs from the second group, however, in that it does not maintain the integrity of individual training examples. The primary examples of this approach are the artificial neural network models. Concept representations are distributed across a network of excitory and inhibitory connections. Learning occurs by incrementally adjusting weights in order to minimize the prediction error on training examples. Evolutionary systems similarly alter their conceptual representation as they search for the representation that minimizes prediction error.

The approach we describe here does not conform to any of these three approaches. SCA is a symbolic rule-based system, but it does not try to maintain a compact explicit concept definition. Neither does it keep whole training instances for future classification. Instead, it incrementally acquires prediction rules. For a particular concept, there may be many. At first, SCA learns very general rules, but as learning progresses, more specific rules are acquired. For category prediction, SCA searches for the most specific prediction rules first before accessing more general ones.

In the next sections, we describe how SCA incrementally acquires prediction rules. Then, we show some of SCA's learning behaviors that result from its design. Finally, we make some empirical comparisons with other established concept acquisition systems in order to demonstrate SCA's effectiveness as an approach to concept acquisition.

2 The Algorithm

The current implementation of SCA accepts training instances described in terms of symbolic attributes and values. For example, a training instance may be described as follows:

```
[category:ball  shape:spherical, color:blue, texture:smooth]
```

SCA incrementally acquires new prediction rules based on training instances and previously acquires prediction rules.

Before we look into how SCA acquires its rules, let us explain how SCA predicts the category name for an instance description when the description does not include the category name. Through learning, SCA acquires a large set of prediction rules. Some are very general (for the following examples, we have omitted the attribute names):

```
[spherical] --> predict category:ball
[spherical] --> predict category:globe
```

Others are more specific:

```
[spherical, red] --> predict category:ball
[spherical, blue] --> predict category:globe
[spherical, red, smooth] --> predict category:ball
```

SCA makes a prediction by searching for the most specific rule that matches the instance description. In particular, the process takes the instance description and then checks if there is a rule that matches all of its features. If there isn't any, it then removes a feature from the instance description and checks if there are any matches on all of the remaining features. This process of removing a feature and then checking for a match continues until either a prediction rule matches or until there are no features left. In the latter case, no prediction can be made until more prediction rules are learned. If several competing rules match at the same time, a guess for one of the predictions must be made.

The performance of the system hinges on the ability to remove the least relevant attributes first, especially when only the more general rules can match. We defer this issue to the next section where we discuss a scheme that controls the search through the prediction rule space.

Here is an example with the above rules and the instance description [spherical, red, fuzzy, large]. SCA makes its prediction by first checking whether there is a rule that matches all four of these features. Since there isn't any, one feature is removed, say large, leaving the description [spherical, red, fuzzy]. Still there is no match. Fuzzy is removed, leaving [spherical, red]. Now, a rule matches, predicting 'ball.'

For learning, SCA accepts a training instance description, with the category label, and processes it by trying to make a prediction as described above. This time, however, the search does not necessarily stop with the first-matched rule. Instead, search continues until a matching-prediction rule makes the correct prediction, or until no features are left in the instance description. With a match, the system has thus discovered prior experience that supports the current training instance. The training instance now serves as new knowledge for adding an additional rule. The conditions of the new rule include all of the features that matched (or no features if no match occurred) *plus the feature that was last removed before the search stopped.* The prediction of the new rule is simply the correct category given by the training instance.

Let us use the training instance [ball: spherical, blue, fuzzy, small] as an example of how a new rule is acquired. First, the description [spherical, blue, fuzzy, small] is processed in search for a category prediction. 'Small' is removed and then 'fuzzy' is removed. At this point, the description [spherical, blue] matches a prediction rule. However, this rule predicts 'globe'–the wrong category. Search continues by removing 'blue'. Finally, the description [spherical] matches a correct rule and search stops. A new rule is constructed and added to memory:

```
[spherical, blue] --> predict category:ball
```

With the acquisition of this new rule, there are now two competing rules with these attributes at this level of specificity. Should both of these rules match during performance, a guess is required in order to make a prediction. Typically, however, the acquisition of this new rule is merely an intermediate step towards the acquisition of still more specific ones. Eventually, with enough training instances, sufficiently specific rules are acquired so that conflicts among them become quite rare.

The choice of adding only one rule with only one more feature in the condition represents a compromise between previously obtained knowledge and the knowledge implicit in the newly presented training instance. Should the previously obtained knowledge be incomplete, *i.e.* the prediction conditions do not include all relevant features, the addition of a new rule with an extra feature in the condition helps complete the system's knowledge. On the other hand, should the knowledge implicit in the training example be irrelevant (due to spurious correlations) or incorrect (due to noise), the addition of only one more rule with only one additional conditional feature allows room for error recovery with the acquisition of new knowledge. Error recovery automatically occurs with the presentation of additional training instances which incrementally lead to rules more specific than the incorrect ones. By searching through the rule space from the most specific rule to the more general ones, the first matched rule is typically the correct rule.

We have yet to explore intermediate alternatives such as adding a rule with two additional features or adding several different rules each with one additional feature. The performance of these alternate strategies are still open to further research. However, our empirical studies have ruled out the extreme alternative, which adds rules of all levels of specificity for each presented training instance. Our findings indicate that this approach learns at the same rate (with respect to the number of training instances) but with significantly worse accuracy.

Over time, the system acquires many rules. While SCA's approach is memory intensive, it does not make exceeding demands on rule indexation in terms of computational time. Because all of the description's features must match for a rule to apply, a simple hashing scheme can be used for rule indexation. This allows a match to occur in nearly constant time regardless of how many rules are currently in memory.

3 Search control

As noted earlier, the effectiveness of this approach depends critically on which features are removed from the instance description. Ideally, all the relevant attributes should be in the conditions of the acquired rules. Furthermore, the irrelevant features should be removed first so that rules with relevant features in their conditions can be successfully indexed. On the other hand, if relevant attributes are removed from the description before a match occurs, the quality of the category prediction will certainly suffer. In addition,

if the selection of attributes during training trials is erratic, the learning rate of more specific rules will suffer. An inconsistent choice of attribute removal will fail to index specific rules previously acquired with a different order of feature removal. Thus, the ideal search heuristic should be as stable as possible from training trial to training trial.

Despite SCA's dependency on a reasonable selection heuristic, SCA's performance depreciates only in terms of learning rate as the choice of attribute removal suffers. In the case where irrelevant attributes are kept in the instance description at the expense of having relevant ones removed, SCA will still be able to make good predictions once specific enough rules have been learned that include both the irrelevant and the relevant rules. Likewise, erratic selection of attributes will slow the progression of how fast specific rules are learned. However, after enough training instances, specific rules are still acquired.

An in depth analysis of attribute selection heuristics for our system is beyond the scope of this paper. Instead, we have chosen the relatively simple entropy measure as used in ID3 [Quinlan, 1986]. This roughly respects our conditions for a good heuristic and supplies an immediate means in which we can empirically evaluate SCA's learning behavior and performance. With this, we can project a lower bound on SCA's learning potential while still constraining our system to that of an incremental learner.

A choice must be made on whether the entropy should be calculated for each attribute/value pair, with the selection based upon the attribute's value, or the entropy measure for an attribute's values averaged, with the selection made by attribute. Initial studies suggest this choice has a large impact upon the system's performance. Our preliminary findings suggest the by-attribute scheme works well when the dataset only uses two categories, whereas the by-value approach proves far more successful when many categories are learned. For now, we choose to use both of these methods as appropriate. Further research should address this issue and we anticipate a more unified approach to selecting features.

4 System behavior and empirical evaluation

As SCA learns, it gradually acquires more specific rules. Figure 1 shows how more specific rules are matched through the course of learning. The x-axis specifies the number of encountered training instances. After intervals of twenty training instances, the system made predictions on a set of test instances. The y-axis specifies the average specificity, in terms of features matched, of the indexed prediction rules for the test set[1]. Figure 2 shows the relationship between rule specificity and prediction correctness. Using data from the same run of Figure 1, the prediction accuracy was calculated according to rule specificity. The x-axis specifies the specificity of the indexed prediction rules. The y-axis specifies the accuracy. Roughly, the more specific the rule is, the more likely the prediction will be correct. This observation helps understand SCA's external properties which we now explain.

SCA has three behavioral consequences that are independent of the attribute selection heuristic choice. They are 1) learning often improves over multiple exposures of the same training data, 2) computational expense lessens through the course of learning and 3) the algorithm has a built-in tolerance for noise.

Often one pass through a training set is sufficient for SCA to acquire rules specific enough to make reasonably accurate predictions. Typically, however, running it over the same dataset continues to improve performance. Figure 3, shows the effect on accuracy by exposing a small training dataset multiple times. The x-axis specifies the number of times the training set was presented to SCA. The left side of the y-axis specifies the prediction accuracy on a larger test set. The graph shows how accuracy improved after a second exposure to the same dataset. The improvement is due to the system's having learned some more specific rules containing relevant features not previously included from the first training cycle.

The same figure also shows SCA's computational runtime (including training and testing) in relation to the number of exposures. The right side of the y-axis specifies the total runtime including training and testing in terms of a machine-internal time unit. Most of the runtime can be accounted for by the constant overhead required in calculating the entropy measure. Nevertheless, runtime slightly decreases with learning. Recall that SCA searches for specific rules first before it tries to match a more general rule. Runtime is thus inversely proportional to the specificity of the matched rule. If specific rules can be found in memory, response time is faster. After many training trials, more specific rules have been acquired, causing runtime to be faster.

[1] All of the datasets we used can be obtained from the University of California-Irvine dataset depository.

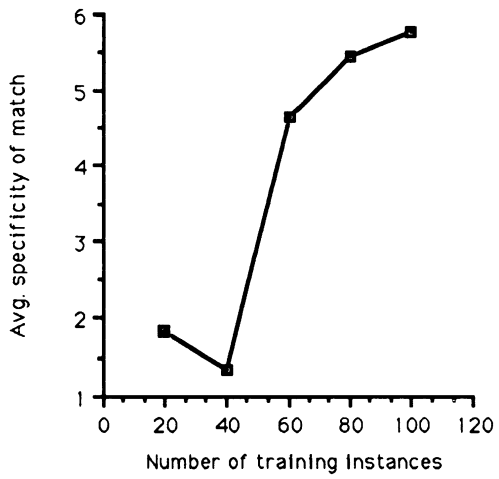

Figure 1: Relationship of Avg. rule specificity to number of training instances (cong. voting dataset).

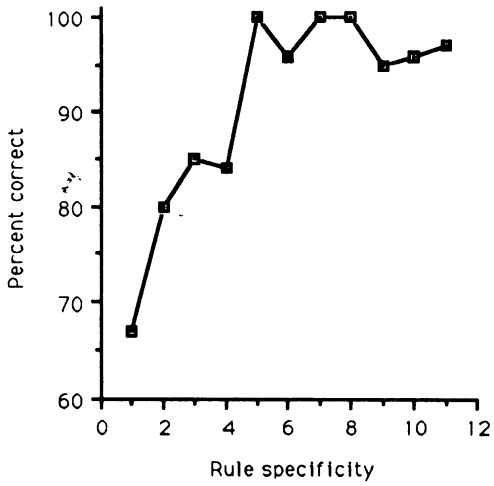

Figure 2: Relationship of accuracy to rule specificity (cong. voting dataset).

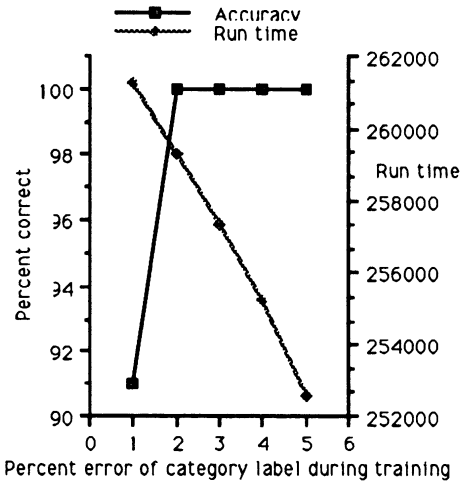

Figure 3: Performance and runtime improvement over multiple trials (mushroom dataset).

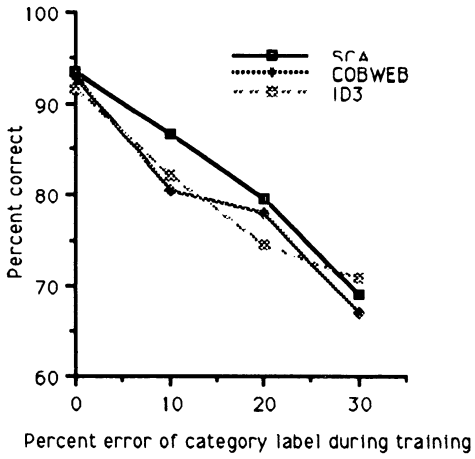

Figure 4: Effect of category noise during training (cong. voting dataset).

Name	Training set size	Testing set size	Number of attributes	Number of categories
Mushrooms	50	200	22	2
Soybeans	290	340	35	15
Cong. Voting	235	200	16	2

Table 1: Description of datasets

Figure 4 shows how SCA is tolerant of noise. For this task, during training, the wrong category was given with the instance description a given percent of the time, specified by the x-axis. The y-axis shows the prediction accuracy on a separate test set. For comparison purposes, the same task was run with ID3[2] and COBWEB[3]. In understanding SCA's resilience to noise, it can thought of as a competition model where conflicting rules compete in making the category prediction. Because SCA searches for the most specific rule first, it is the most specific rule which wins the competition. Presumably, during training, the system encounters more correct instances than incorrect ones. The correct rules thus contribute to the learning of rules that are more specific than prediction rules formed from incorrect instances. Typically, correct predictions result despite there being many incorrect rules (rendered benign by their lack of specificity) in memory.

SCA's accuracy and learning rate can also be compared with COBWEB and ID3. Table 1 describes the datasets we used in testing accuracy. Figure 5 shows the accuracy of the three systems on these datasets. For the mushroom data, we limited the size of the training set to 50 instances because of the dataset is relatively easy. For the other two datasets, the prediction accuracies approach asymptotic performance. The prediction accuracy is the performance on the test set. Figure 6 examines the learning rates of these systems on the congressional voting dataset. The x-axis specifies the number of training instances given to the systems. The y-axis specifies the prediction accuracy on the separate test set after each training installment. Certainly, SCA is the slowest learner of the three systems that we compare here, although it eventually achieves the superior accuracy. This also explains why SCA did the poorest with the mushroom data in which case it had hardly reached its asymptote in prediction accuracy (which is 100

It is important to realize that both COBWEB and SCA are incremental learners whereas, ID3 is not. For all of our studies, ID3 had the added benefit of being able to access any of its training instances at any time.

5 Conclusion and future work

We have shown how SCA differs from previous work in supervised learning. While the rules it acquires are entirely symbolic, it does not try to maintain a compact concept representation, nor does it explicitly save whole training examples. Furthermore, because of its design, the learning behavior elicits interesting properties, including improved response time and resilience to noise. While it has not been our goal to demonstrate how SCA is superior to any other concept acquisition system, it is clear that SCA is competitive in asymptotic prediction accuracy, though it requires more training instances to attain asymptotic behavior. Furthermore, SCA's efficient method of indexing prediction rules should prove critical in scaling up to larger tasks requiring many prediction rules.

In this paper, we have neglected the role of the search-control heuristic used with SCA and have instead emphasized the behavior resulting from the symbolic-rule acquisition aspect of the system. There remains much research studying search heuristics which could improve system performance. We have also refrained from discussing any properties that validate SCA as a psychological model of human category learning. While Miller and Laird (1991) discuss some of these including typicality effects and extension errors, more recent studies indicate that SCA successfully replicates many other phenomena.

[2]We used a version of ID3 that built a binary tree, implemented by John Paxton. Fayyad (1991) reports that a binary tree is superior to the standard implementation.

[3]The implementation we used was COBWEB/3, obtained from NASA Ames Research Center.

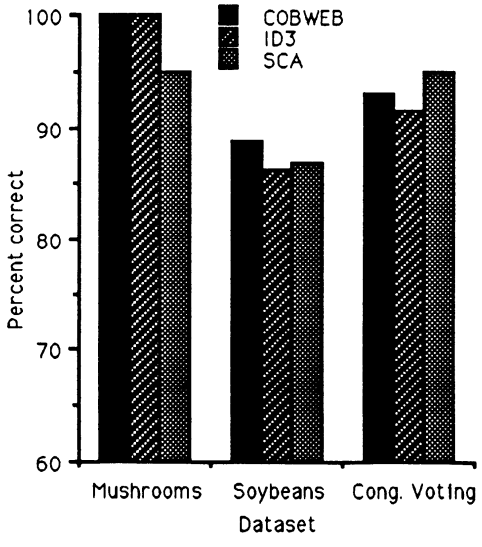

Figure 5: Prediction accuracy.

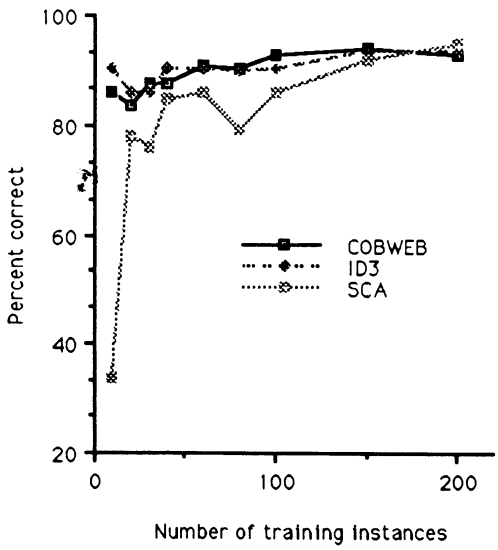

Figure 6: Learning rate (cong. voting dataset).

Acknowledgments

We would like to thank Allen Newell and Paul Rosenbloom for their helpful comments on earlier drafts of this paper.

References

[Aha, 1989] D. W. Aha. Incremental, instance-based learning of independent and graded concept descriptions. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 387–391, 1989.

[Blumer *et al.*, 1987] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 23:377–380, 1987.

[Fayyad, 1991] U. M. Fayyad. *On the induction of decision trees for multiple concept learning.* PhD thesis, The University of Michigan, 1991.

[Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[Michalski, 1983] R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.

[Miller and Laird, 1991] C. S. Miller and J. E. Laird. A constraint-motivated model of concept formation. In *The 13th Annual Conference of the Cognitive Science Society*, 1991.

[Newell, 1990] A. Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA, 1990.

[Quinlan, 1986] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.