Combining Numeric and Symbolic Learning Techniques

Richard H. Granger, Jr.
Jeffrey C. Schlimmer

Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California
Irvine, California 92717

June 1985
Technical Report #85-27

This research was supported in part by the Office of Naval Research under grant N00014-84-K-0391, the National Science Foundation under grant IST-81-20685, and by the Naval Ocean Systems Center under contract N66001-83-C-0255.

A shorter version of this paper appears in the proceedings of the International Machine Learning workshop, 1985.

Combining Numeric and Symbolic Learning Techniques

Richard H. Granger, Jr.
Jeffrey C. Schlimmer

Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California
Irvine, California 92717

Abstract

Incremental learning from examples in a noisy domain is a difficult problem in Machine Learning. In this paper we divide the task into two subproblems and present a combination of numeric and symbolic approaches that yields robust learning of boolean characterizations. Our method has been implemented in a computer program, and we plot its empirical learning performance in the presence of varying amounts of noise.

1 Introduction

A learning engine in an unconstrained, active environment faces a difficult task, for it must be able to accurately predict the outcomes of its actions. Moreover, there are a large number of features in the environment, the environment is occasionally inconsistent, and the learner must adapt its performance in a continuous manner. This is a task for which neither numeric or symbolic learning methods alone appear to be sufficient. Existing Machine Learning approaches have various shortcomings: either they are non-incremental [e.g., Michalski and Stepp 1983], requiring a large number of instances before learning can occur; they require an explicit theory of the domain [Mitchell 1985] or a benevolent tutor; or as many do [e.g., Winston 1970, Mitchell 1982], they have significant trouble with inconsistent instances or don't learn at all in the presence of noise. Section 4 discusses this related work in more detail.

In this paper we describe an alternative approach for learning from examples that combines numeric and symbolic learning methods. Each characterization under consideration by the program is assigned a measure of effectiveness as a result of a statistical summarization process. Instances are not compared with other instances, but with this summary of past instances. The summary is also used to guide formation of new compound characterizations through the exponential space of possibilities while remaining within reasonable memory limitations. Because the algorithm is incremental and does not rely on a large memory of accrued instances, it is able to track changes in the environment over time and remains extremely robust with respect to noise.

2 Salience assignment

The domain considered in the context of this paper consists of simple, non-relational blocks, a traditional domain for AI systems. Blocks may be described in terms of their size, color and shape (e.g., a large red circle). The proper characterization of the concept to be learned may involve a conjunction across slots

This research was supported in part by the Office of Naval Research under grant N00014-84-K-0391, the National Science Foundation under grant IST-81-20685, and by the Naval Ocean Systems Center under contract N66001-83-C-0255.

A shorter version of this paper will appear in the proceedings of the International Machine Learning workshop 1985.

(e.g., Size[large] and Color[red]) a disjunction of fillers within a slot (Color[red or blue]), a disjunction between slots (Size[large] or Color[red]), or the negation of any of the above.

There are two subproblems an algorithm in this domain must solve: which of the features are the significant ones (the *salience assignment* problem) and which of the possible boolean functions over the features should be formed (the *composition* problem).

The outputs of the salience assignment problem are a classification of each of the characterizations into one of three roles: an accurate predictor of positive instances, a accurate predictor of negative instances or an 'uncorrelated' characterization which does not enable an accurate prediction of either kind.

The inputs for this problem may be divided into one of four logical possibilities: a characterization is present in a positive instance (*prediction*), present in a negative instance (an error of *commission*), absent in a positive instance (an error of *omission*), or absent in a negative instance (*non-prediction*). The first and last possibilities indicate a positively predictive characterization. Errors of commission and omission indicate a negative or uncorrelated cue.

	Positive instance	Negative instance
Feature present	++ Prediction	+− Error of Commission
Feature absent	−+ Error of Omission	−− Non-prediction

Table 1: Possible combinations of features and instances

A constraint from psychology helps identify an appropriate algorithm. Specifically, animal experiments [Rescorla 1968] clearly identify a necessary condition for learning an association between two events E1 and E2: the likelihood of E2 must be greater following E1 than it is without E1. Formally, $p(E2|E1) > p(E2|\overline{E1})$. This is called the law of *contingency*. This principle is in contrast to approaches which simply strengthen an association whenever E1 and E2 are paired together and weaken it when they are not (e.g., when Color[red] (E1) is tagged as a positive or negative instance (E2)).

Bayesian statistics (see e.g., Duda et. al. [1979]) provide similar formulae for the calculation of two values in inductive logic: *Logical Sufficiency* (LS), which indicates the extent to which the presence of one event predicts another particular event; and, reciprocally, *Logical Necessity* (LN), which represents the extent to which the *absence* of an event *decreases* expectation or prediction of the second event. Our algorithm makes use of an incremental method of calculating approximations of these two values, via a simple formula composed of precisely the four possible categories of pairwise feature occurrences given above.

$$\mathrm{LS} = \frac{s(n+o)}{o(s+c)} \qquad\qquad \mathrm{LN} = \frac{c(n+o)}{n(s+c)}$$

where s is the count of successful predictions, c is errors of commission, o is errors of omission, and n denotes non-predictions.

Our algorithm utilizes these approximations of LS and LN to assign a role to each of the characterizations in an incremental manner. The role assigned is based on the interpretation of the LS and LN values. LS and LN values range from 0 to ∞, with high LSs corresponding to a characterization strongly

Propose	When error of commission
Not[A]	LN(A) >> 1, A present
And[A,B]	LN(A) << 1, A present LN(B) << 1, B absent
	When error of omission
Or[A,B]	LS(A) >> 1, A present LS(B) >> 1, B absent

Table 2: Composition proposers

predictive of positive instances and very low LSs corresponding to the case where the characterization implies a *negative* instance. When the value of LS is approximately 1 then the characterization is considered uncorrelated. LN values are interpreted reciprocally. That is, high LN values indicate a necessary characterization for a negative instance; LN value about equal to 1, a lack of correlation; and LN values much less than 1, the necessity of a characterization for a positive instance.

3 Composition

Learning in a complex domain necessitates noting useful combinations of characterizations. For instance, a conjunction of the size and color may indicate a positive instance while neither the size nor the color alone do.

Our algorithm for the composition problem is failure driven and, upon an error, it compares an instance with the classifications of characterizations provided by the salience assignment algorithm in the process of composing new characterizations.[1] When the algorithm makes an error of omission, it is behaving as if the characterizations as a whole are too specific; as a result, a new characterization is formed which is the disjunction of the most significant characterizations (measured by sufficiency, i.e., LS). Upon an error of commission (behaving too generally) a new characterization may be formed which is the conjunction of two significant characterizations. The measure of necessity, or LN, is used to determine the best candidates. Negations are also proposed following an error of commission.

3.1 Limiting the growth of memory

This algorithm does not continue to form new characterizations without bound. Two mechanisms serve to limit this growth. The first is simply that new characterizations are only introduced following a failure.

[1]Given the specific algorithm presented above, this amounts to comparing an instance against a statistical summary of the previous instances. This is in contrast to approaches which compare instances with other instances (see e.g., [Langley 1983]), or instances with symbolic generalizations of other instances (e.g., [Mitchell 1982]). Utilizing this methodology allows the algorithm to propose disjunctions and retain tolerance to noise.

In an environment without inconsistent instances this alone can be quite effective.

Secondly, this algorithm utilizes *competition* between a newly introduced characterization and its components. When a new characterization outperforms its components, the latter are deactivated. Measuring the predictiveness of each characterization is done by comparing LS and LN values. When a new characterization is introduced it is assigned an LS *threshold* if it is a conjunction or negation and an LN threshold if it is a disjunction.[2] These thresholds are set at the maximum of the LS values of the components or the minimum of the LN values for conjunction/negation and disjunction respectively. The original characterizations are deactivated when the significance of the new characterization exceeds its threshold. Until this time the new characterization cannot be combined with others.

Utilizing this mechanism has the additional advantage that this algorithm can (1) correct some erroneous characterization formations and (2) follow changes in the true characterization of the concept (i.e., if the environment changes). When a characterization falls below its threshold, the characterizations that led to its formation are reactivated and now compete with the ineffective characterization. If the ineffective characterization fails to perform well, it is deactivated and the reactivated characterizations are free to form new combinations.

3.2 Run time output

The preceeding algorithms were encoded in a program written in Franz Lisp on a VAX-11/750 under Unix. In the following transcript the concept to be characterized is (Size[small] and Color[red]). Annotations are separated from actual output by semicolons.

⋮

```
Instance features: Size[small], Color[red], Shape[square].
    strongly expecting a SMALL-AND-RED (odds = 166.88 >> 1).
Positive instance of SMALL-AND-RED.          ; This is a successful
    updating contingency                     ;  classification.
      marking successes
      marking omissions

      ; Here is a list of the initial characterizations
      ;  the program begins with.  The characterization
      ;  small ? ? is considered satisfied if the size
      ;  is small.  Each characterization has counts
      ;  which summarize the instances seen thus far.
      ;  Note that no compositions of characterizations
      ;  have yet been proposed.

Characterizations of SMALL-AND-RED:
-----------------------------------------------------
Size, Color, Shape | ++  +-  -+  --     ls   ||   ln   |
-----------------------------------------------------+
    small ? ?      | 5   1   1   1     1.67  ||  0.33  |
-----------------------------------------------------+
```

[2] An LS threshold is chosen for conjunctions since a more restrictive clause is *falseness-preserving* [Michalski 1983] and thus LN is guaranteed to be as good. A similar argument follows for disjunction.

```
    medium ? ?     | 1   1   5   1     0.6   ||   3.0  |
-------------------------------------------------------+
    large ? ?      | 1   1   5   1     0.6   ||   3.0  |
-------------------------------------------------------+
    ? blue ?       | 1   1   5   1     0.6   ||   3.0  |
-------------------------------------------------------+
    ? green ?      | 1   1   5   1     0.6   ||   3.0  |
-------------------------------------------------------+
    ? red ?        | 5   1   1   1     1.67  ||  0.33  |
-------------------------------------------------------+
    ? ? circle     | 1   1   5   1     0.6   ||   3.0  |
-------------------------------------------------------+
    ? ? square     | 3   1   3   1     1.0   ||   1.0  |
-------------------------------------------------------+
    ? ? triangle   | 3   1   3   1     1.0   ||   1.0  |
-------------------------------------------------------

Instance features: Size[small], Color[green], Shape[triangle].
    strongly expecting a SMALL-AND-RED (odds = 26.78 >> 1).
Negative instance of SMALL-AND-RED.          ; This is an error of
    updating contingency                     ;  commission.
      marking commissions
      marking non-predictions
    introducing new characterizing cue:      ; small is present and has
      small red ?                            ;  an LN value below 1; red
    introducing new characterizing cue:      ;  is absent, LN below 1.
      ? (blue v red) ?

                                             ⋮

Instance features: Size[medium], Color[green], Shape[circle].
    not expecting a SMALL-AND-RED (odds = 0.03 << 1).
Negative instance of SMALL-AND-RED.          ; When nothing happens, none
                                             ;  of the learning
                                             ;  mechanisms are updated.

                                             ⋮

Instance features: Size[small], Color[blue], Shape[square].
    strongly expecting a SMALL-AND-RED (odds = 4.56 >> 1).
Negative instance of SMALL-AND-RED.          ; This is another error
    updating contingency                     ;  of commission.
      marking commissions
      marking non-predictions
    establishing characterizing cue:         ; small and red has proved
      small red ?                            ;  to be more predictive than
    deactivating characterizing cue:         ;  either small or red alone.
      small ? ?                              ;  The program removes each
    deactivating characterizing cue:         ;  of these characterizations
      ? red ?                                ;  in order to keep memory
    introducing new characterizing cue:      ;  within reasonable bounds.
      ? (green v red) ?
```

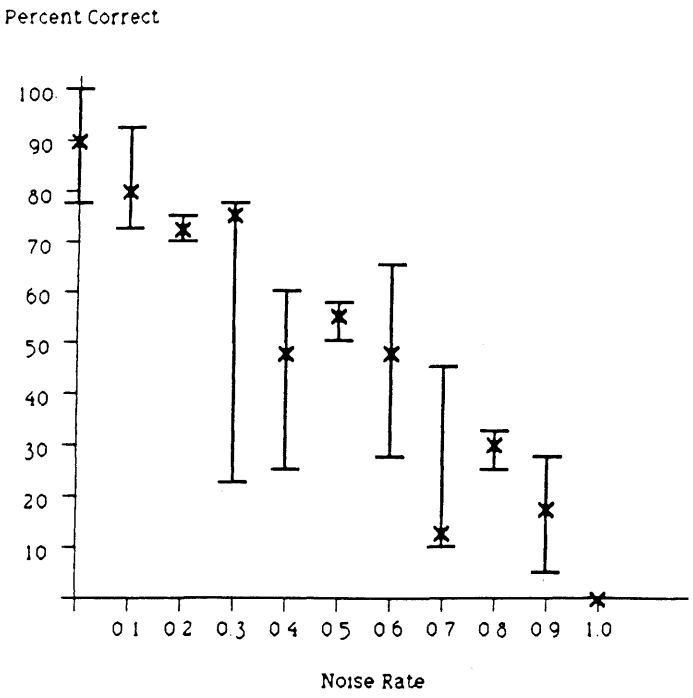

Figure 1: Performance as a function of noise.

During a given instance confidence is calculated by multiplying together the LS values of each characterization that is present and the LN values of each characterization that is absent. This confidence measure is then interpreted in terms of odds: much less than 1 indicates that a negative instance is expected; about 1 indicates uncertainty; much greater than 1 indicates that a positive instance is expected.

In the example above, Size[small] has an LN value below 1, as does Color[red], and an error of commission occurs where Size[small] is present and Color[red] is absent; the program hypothesizes that Size[small] *and* Color[red] together might be a better predictor than either alone.

3.3 Program performance

Real-world environments invariably entail some degree of noise, so a learning algorithm must be able to tolerate inconsistent instances. We have been pleasantly surprised by the performance of our program. Figure 1 depicts the performance of our program when trained for the concept Color[red *or* blue] with various rates of inconsistency (the ratio of incorrectly identified instances to all instances). As the rate of inconsistency approaches 0.2, the performance falls toward a chance level of 50%. As one would expect, inconsistency rates in excess of 0.5 cause the program to acquire the opposite of the concept and perform at less than a chance level.[3]

The program's tolerance of inconsistent instances is partly due to the smoothness of the salience

[3]The performance curves for disjunction and conjunction between slots are similar.

assignment function. However, we found that the robustness depends critically on the introduction of composed characterizations. With no noisy instances, the program performs well with the composition algorithm disabled, but when even a small number of inconsistent instances are introduced, performance falls off precipitously unless combination characterizations are proposed. This is due to the higher predictive value of a combination of characterizations as compared to its components and correspondingly smaller influence of inconsistent instances on that value. Methods which rely on an implicit encoding of the compound characterizations will not perform as well in presence of noise as those that use an explicit, composed encoding.

4 Related work

There has been a significant amount of work done in Machine Learning on the task of learning from examples. Numeric techniques (e.g. [Samuel 1963]) tend to be tolerant with respect to noise but have some difficulty exploring the exponential composition space of characterizations. Michalski and Stepp [1983] address the larger task of conceptual clustering and deal effectively with the space of characterizations and noise. However, their algorithm is non-incremental and processes the instances in a 'batch' manner.

Symbolic techniques, on the other hand, have been used successfully by a number AI learning programs though they fail to tolerate noise [Mitchell 1982] and may require specific instances to maximize learning [Winston 1970]. Recent analytic approaches minimize the number of instances required to accurately characterize a concept [Mitchell, Mahadevan, and Steinberg 1985], yet they rely heavily on a nearly complete domain theory, which restricts the applicability of this work only to domains and problems which offer pre-built-in domain theories.

Cognitively oriented Machine Learning researchers have focussed on developing psychologically plausible models of learning. In the ACT* family of programs [Anderson 1983], the basic associational unit, the production rule, is strengthened when it is reinvented and when it is activated through the spread of activation in memory. Rules are weakened by negative feedback. Though this scheme for assigning rule strength may be empirically valuable, it does not appear to be consistent with basic psychological data concerning contingency (see Section 2).

IPP [Lebowitz 1983] learns while parsing English newspaper stories. Features present in stories are used to form groups of predictive features which may be used to aid further parsing. These feature groups are strengthened (or weakened) by a prespecified amount given positive (or negative) instances. The program improves its parsing of story texts, but again this method is inconsistent with psychological contingency data. Secondly, as Lebowitz notes, IPP has some difficulty when subset of features in a group are predictive while any single one isn't. Lacking a representation for explicit boolean functions prohibits IPP from assigning predictiveness only to a subset of features.

5 Conclusions: Constraints and heuristics from cognition

Our work on the problems of learning addressed in this paper arose from our twin concerns with accurately modeling human and animal learning behavior, and generating useful AI learning algorithms. We began by examining a number of results from the literature on animal learning psychology. An

experimental result of specific interest gave a precise quantification of a necessary condition for animal learning [Rescorla 1968] called *contingency*. Pursuing the implications of this class of experiments, we found that a large number of algorithms currently in use in AI systems which are based on a simple 'strengthening and weakening' scheme (strengthen when it works, weaken when it doesn't) are clearly inconsistent with these experimental data.

Although this is a valuable observation for cognitive modelling efforts, it wasn't immediately obvious what impact this would have on researchers in Machine Learning. However, this is an area in which results from AI and Cognitive Science converge: analysis of the relevant experimental data led us to a highly robust algorithm for the salience assignment task [Granger, Schlimmer and Young 1985]; one that it is doubtful we would have otherwise considered.

We plan to continue to investigate algorithms that account for existing psychological results and incorporate those that yield effective solutions into our ongoing work on Machine Learning.

6 References

Anderson, John R. 1983. *The Architecture of Cognition.* Cambridge: Harvard University Press.

Duda, R. O., Gaschnig, J. G., Hart, P. 1979. Model design in the Prospector consultant system for mineral exploration. In *Expert Systems in the Micro-electronic Age*, D. Michie (ed). Edinburgh: Edinburgh University Press.

Granger, R.H., Schlimmer, J.C. and Young, M.T. 1985. *Contingency and latency in associative learning: computational, algorithmic and implementational analyses.* In *Brain Structures, Learning and Memory*, Eds. Davis, J., Wegman, E. and Newburg, R. (To appear). Also appears as: Computer Science Department Technical Report #85-10. University of California, Irvine.

Langley, P.W. 1983. Learning effective search heuristics. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 419–421.

Lebowitz, M. 1983. Generalization from natural language text. *Cognitive Science*, **7**, 1–40.

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

Michalski, R.S. and Stepp, R.E. 1983. Learning From Observation: Conceptual Clustering. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Palo Alto: Tioga Publishing Company.

Mitchell, T.M. 1982. Generalization as search. *Artificial Intelligence*, **18**, 203–226.

Mitchell, T.M., Mahadevan, S. and Steinberg, L.I. 1985. *LEAP: A Learning Apprentice for VLSI Design.* Rutgers Technical Report LCSR-TR-64, Rutgers University, New Brunswick, NJ. To appear in *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Los Angeles, California.

Rescorla, R. 1968. Probability of shock in the presence and absence of CS in fear conditioning. *J. Comparative and Physiological Psychology*, **66**, 1–5.

Samuel, A.L. 1963. Some studies in machine learning using the game of checkers. In Feigenbaum, Edward A. and Feldman, Julian, eds., *Computers and Thought.* New York: McGraw-Hill.

Winston, Patrick H. 1970. Learning structural descriptions from examples. Technical Report AI-TR-231, Massachusetts Institute of Technology.