Learning and Representation Change

Jeffrey C. Schlimmer*

Department of Information and Computer Science
University of California, Irvine 92717
schlimmer@ics.uci.edu

Abstract

To remain effective without human interaction, intelligent systems must be able to adapt to their environment. One useful form of adaptation is to incrementally form concepts from examples for the purposes of inference and problem-solving. A number of systems have been constructed for this task, yet their capability is limited by the language used to represent concepts. This paper presents an extension to the concept acquisition system STAGGER that allows it to utilize continuously valued attributes. The combination of methods employed is able to dynamically acquire appropriate representations, thereby minimizing the impact of initial representational bias decisions. Of additional interest is the distinction between the computational flavor of the learning methods, for one is similar to connectionist approaches while the other two are of a more symbolic nature.

I. Introduction

Consider the task of constructing a concept description given a series of examples and non-examples. This has been addressed by a number of learning systems, yet many have limited capability due to inflexibility inherent in the concept representational language. If the language is too restrictive, there will be some concepts which cannot be represented or learned. The restriction imposed by the concept representation language is necessary, though, and without it the learning method could do no better than to guess randomly at the concept's definition (Utgoff & Mitchell, 1982). To alleviate this bind between flexibility and tractability, a system might modify the underlying representational language in some way or another, either increasing the language to accommodate more possible concepts or reducing it to improve the possibility of finding an appropriate concept description.

This paper describes a two part extension to a concept acquisition system called STAGGER (Schlimmer & Granger, 1986). First, a new method is added that discretizes continuously valued attributes. Secondly, by combining this new method with the existing methods that weight and refine a distributed concept description, STAGGER is able to overcome limitations inherent in its initial concept language. After briefly describing some related work, this paper describes each of the three learning methods and then demonstrates their interaction.

*This work was supported by grants ONR N00014-85-K-0854, ONR N00014-84-K-0391, ARI MDA903-85-C-0324, and NSF IST-85-12419.

II. Related Work

A number of researchers have studied the problem of utilizing numerically valued information in symbolic concept learning. For example, Michalski (1983) presents the closing interval generalization rule. It specifies that if two values are found in positive examples, assume that all of the values between them will be. This mapping from continuous to discrete values is similar to the type of approach used by Lebowitz's (1985) UNIMEM which partitions real values in both a generalization and a data-driven manner. In the first, clusters of examples formed on the basis of discrete attributes partition a real range by implicitly grouping the values. The latter, data-driven technique searches a subset of the numeric values present for gaps indicated by the distribution of real values across objects. Quinlan's (1986) ID3 system forms a number of competing pairs of intervals, centered around potential splits in the real-value range. ID3 then considers these intervals as possible decision tree roots by interpreting them as binary valued attributes.

A distinctly different approach for handling numeric information is taken by Bradshaw (1985) in his speech understanding system. Instead of attempting to map the continuous information representing a verbal utterance into some set of symbolic values, his system retains a set of attribute averages for each word concept.

Rendell's (1986) PLS family of concept induction systems incorporate both partitioning and averaging approaches. In some instances concepts are represented as "rectangles" or value ranges, while in other cases concepts appear to be described in terms of their central values.

Along representational lines, Utgoff and Mitchell (1982) were perhaps the first to address the issue of constrictive representational assumptions. In this and subsequent work (Utgoff, 1986) they develop a method which explicitly identifies shortcomings and triggers procedures for relaxing the descriptive language.

STAGGER's numeric learning belongs in the partitioning class, for it divides real values into a dynamically determined number of discrete ones. Perhaps more importantly, interaction between the three learning components of STAGGER alleviates the impact of an ill-fitting initial concept description language. This representational adjustment occurs in a continuous and natural manner; each of the methods assists the others while performing its own task.

III. Three cooperative learning methods

STAGGER uses three interacting learning components and represents concepts as a set of weighted, symbolic description pieces. One of the learning methods adjusts the weights, another adds new Boolean pieces, and the third adds new pieces corresponding to an aggregation of real-values into a few discrete ones. The interaction between these methods may be viewed as a form of representational learning, for they exert influence on each other by changing the substrate from which induction proceeds.

A. Concept representation and matching

Concepts are represented in STAGGER as a set of dually-weighted, symbolic pieces. Each element of the concept description may be a single attribute-value pair, a range of acceptable values for a real-valued attribute, or a Boolean combination. Figure 1 depicts a typical concept description for size=medium & color=red. Each descriptive element is dually weighted to capture positive and negative implication. One weight formalizes the element's sufficiency (solid line), or $matched \Rightarrow example$, and the other represents its necessity (dashed line), or $\neg matched \Rightarrow \neg example$. These weights are based on the logical sufficiency (LS) and logical necessity (LN) measures used in Prospector (Duda, Gaschnig, & Hart, 1979).

$$\begin{aligned} \text{LS} &= \frac{p(matched|example)}{p(matched|\neg example)} \\ \text{LN} &= \frac{p(\neg matched|example)}{p(\neg matched|\neg example)} \end{aligned} \quad (1)$$

LS ranges from zero to infinity and is interpreted in terms of odds.[1] A weight greater than one indicates predictiveness; less than one denotes an element that predicts non-examples. LN has the same range but the opposite interpretation. For both, a weight of one indicates irrelevance.

[1]To convert odds into probability, divide odds by one plus odds.

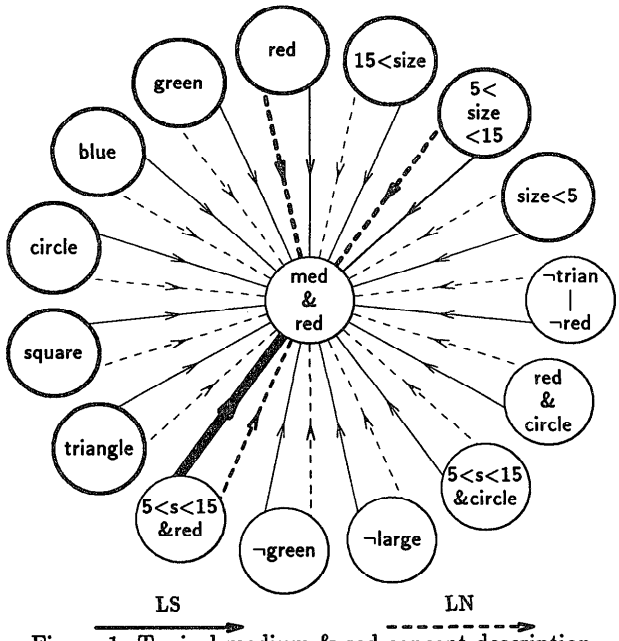

Figure 1: Typical medium & red concept description.

Given a new example, all of the weighted concept elements influence expectation of its identity. Following the mechanism used in (Duda *et al.*, 1979), the prior expectation of a positive example is metered by multiplying in the LS weight of each matching piece and the LN weight of each unmatched one.

$$odds(E|features) = odds(E) \times \prod_{\forall M} \text{LS} \times \prod_{\forall \neg M} \text{LN} \quad (2)$$

The resulting matching score is the odds in favor of a positive example and reflects the degree of match between the concept description and the example. This holistic flavor of matching differs from many machine learning systems in which a single characterization completely influences concept prediction.

B. Modifying element weights

The weights associated with each of the concept description elements are easily adjusted by incrementally counting the number of different matches between an element and examples; these counts are used to compute estimates of the probabilities in Equation 1.

$$\begin{aligned} \text{LS} &= \frac{|matched\&example|/|examples|}{|matched\&\neg example|/|\neg examples|} \\ \text{LN} &= \frac{|\neg matched\&example|/|examples|}{|\neg matched\&\neg example|/|\neg examples|} \end{aligned} \quad (3)$$

Keeping counts of matchings between elements and examples also allows calculating the prior expectation for an example: $odds(example) = |examples|/|\neg examples|$.

By adjusting the weights associated with each of the descriptive elements, STAGGER is behaving as a single layer *connectionist* model. Without hidden units, these models suffer from the same representational limitations that STAGGER does without its Boolean learning method. Both are unable to assign weights to a combination of values, and this severely limits the number of discoverable concepts to only linearly-separable ones. From a representational point of view, the weight method can only form concepts in terms of existing description elements. Instead of including an element for all possible Boolean combinations of the attribute-values, this method begins with the relatively strong bias of only single attribute-value elements.

C. Forming new Boolean combinations

STAGGER selectively adds new elements by beam-searching through the space of all possible Booleans. The initial search frontier is the set of single attribute-value pairs. The Boolean method uses three search operators, specialization, generalization, and inversion to add new elements to the search frontier. The search is limited by proposing a new element only when STAGGER makes an expectation error. This cautiously extends the descriptive power of the weight adjusting process while retaining the constructive properties of a limited representation.

For instance, when a non-example is expected to be a positive example, STAGGER is behaving too inclusively, too generally, and thus a more specific element may be needed. So, STAGGER expands the search frontier by tentatively adding a new AND formed from two elements which are necessary for the concept. The selection of component elements is based on two observations: at least one necessary element is unmatched in a non-example, and necessary elements typically have strong logical necessity weights.

The other type of prediction error also triggers the expansion. A guess that a positive example is negative is overly specific. To correct for this underestimation, search is expanded to include a more general element; a new OR formed from two sufficient elements is tentatively added. Both predictive errors are opportunities to invert poor elements. Further details concerning the Boolean method are documented in (Schlimmer & Granger, 1986).

Though the space of possible Boolean combinations is large, it does not include states for numerically valued attributes. Therefore, STAGGER has a third learning method which extends this space by adding discrete values for real value ranges.

D. Partitioning real-valued attributes

In order to carve up an attribute's real-valued range into a set of discrete intervals, STAGGER retains a simple statistic for a number of potential interval end-points. These end-points are taken from processed examples and, through a beam-search, the best are utilized to naturally break up the range into discrete values. Each new example supplies a value to update these statistics, and in turn this method transforms successive examples into a palatable form for the weight and Boolean learning methods.

Specifically, for each potential end-point, a two by two record is kept of the number of positive and negative examples with values less and greater than this potential end-point. A measure applied to these numbers indicates useful divisions in the value range. By interpreting these divisions as the end points of discrete values, STAGGER maps the real-valued attributes of subsequent instances into discrete values. The utility measure is similar to Equation 2, for it involves the prior odds of each class and a conditional probability ratio similar to LS and LN.

$$U(\textit{end-point}) = \prod_{i=1}^{|\textit{classes}|} odds(class_i) \times \frac{p(class_i| < \textit{e-p})}{p(class_i| > \textit{e-p})} \quad (4)$$

These conditional probabilities may be computed from the number of positive and negative examples with values less and greater than the measured end-point. Since the potential end-points are taken from actual examples, the method is independent of scale considerations and does not entail any assumptions about the range of values. Furthermore, because partitioning is driven by a statistic based on class information, the method is able to uncover effective partitionings even when the values are uniformly distributed across all classes, something a gap finding method (Lebowitz, 1985) is unable to do.

A straightforward strategy for fractioning the real-value range would be to choose the end-point with a maximal utility and thereby divide the range into two discrete values: greater and less. However, for concept learning tasks which require finer distinctions, it would be more effective to divide the range into a number of discrete values. So after applying a local smoothing function, STAGGER chooses the end-points that are locally maximal. These end-points represent pivotal values, for Equation 4 favors those that are predictive of concept indentity. This approach has the advantage that it naturally selects appropriate end-points and an effective number of discrete values.

Having aggregated the real-valued range, attributes with continuous values in successive examples may be transformed into their discrete counterparts. This mapping results in example descriptions that are consistent with the input requirements of both the weight and Boolean learning methods described above. Furthermore, it embodies a type of representational learning, because by partitioning values into ranges, the concept description languages for the other learning methods is expanded.

E. Interactions between the learning methods

STAGGER's three learning methods cooperatively inter-

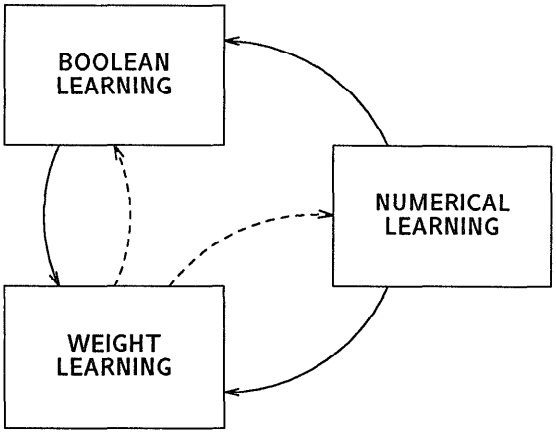

Figure 2: Interaction between STAGGER's three methods.

act as Figure 2 depicts. The Boolean learning method alters the representational base for the weight adjusting method; it is restructuring the input to the weight adjusting method so the latter is able to capture the concept's description. The numeric learning method has this administrative role for both the weight and Boolean learning processes; it rewrites the real-valued attributes into a form suitable for induction by the latter methods. The dependent learning methods also exert influence on the representational processes which counsel them. The Boolean method draws its components from the pool of ranked elements maintained by the weight learner. The weighting method also provides a weak form of feedback for the numeric method: the similarity between the numeric evaluation function (Equation 4) and the matching equation (Equation 2) implicitly ensures that the division of real attributes will be amenable to weighting and matching.

IV. Empirical Performance

The interaction of the three learning methods is perhaps best illustrated by examining their behavior on concept learning tasks. Consider STAGGER's acquisition of a pair of simple object concepts. Each object is describable in terms of its size (real value between 0 and 20), its color (one of 3 discrete values), and shape (3 discrete values).

For the first concept, an object is a positive example if red and between 5 and 15 in size. Optimally, the size attribute should be divided into three ranges: size < 5, $5 <$ size < 15, and $15 <$ size. In each of 10 executions, STAGGER's numeric partitioning method discovers this three-way split, and its Boolean combination technique forms a conjunction combining the middle value of size and the color red. The weight adjusting method further gives this element more influence over matching than any other. The following is typical of the elements formed by the cooperative action of the three learning methods.

$$\text{size} = \text{value}_2 \;\&\; \text{color} = \text{red}$$

$$\text{value}_2 \equiv (5.05 \leq \text{size} \leq 15.01)$$

To illustrate the functioning of the partitioning method, consider the set of potential end-points for this task depicted in Figure 3. Note that the partitioning measure (Equation 4) clearly identifies the two local maxima near 5 and 15 that are used to partition the size attribute into three discrete values.

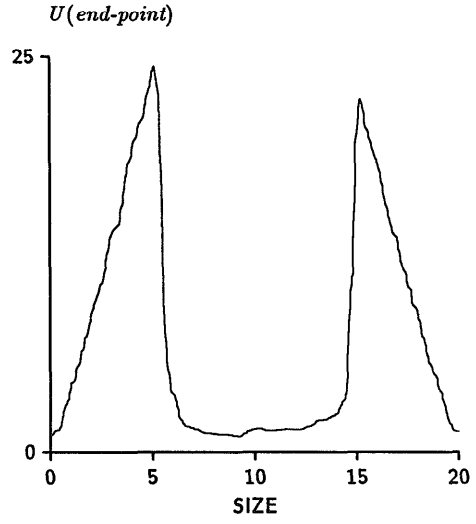

Figure 3: End-points for 5.0 $\leq$size$\leq$ 15.0 & red.

The complete, conjunctive element does not appear suddenly. Though the methods are not explicitly synchronized, they appear to operate in a staged manner as Figure 2 indicates. First, the numerical partitioning method begins to search for a reasonable way to partition real ranges. At the same time, the weight adjusting method searches for appropriate element strengths. The weight of color=red is adjusted at this time, but combination with the size attribute must wait until the numerical method settles down. After processing about 50 examples, the tripartite division of the size attribute is stable, and the weight method is able to assign a strong LN weight to the middle value of the size attribute. After this, the Boolean method combines the size and color elements to form the element depicted above. Weight adjusting finishes the job by giving this element strong LS and LN values.

Regrettably, these three learning methods are not sufficient for all concept learning tasks; there are some concepts for which the numerical method is unable to uncover an effective partitioning. For example, consider the concept of objects that have a size between 5 and 15 or are red but not both. Figure 4 indicates that the numerical method is unable to identify a reasonable partitioning in this case. This limitation also arises if we consider the capabilities of the

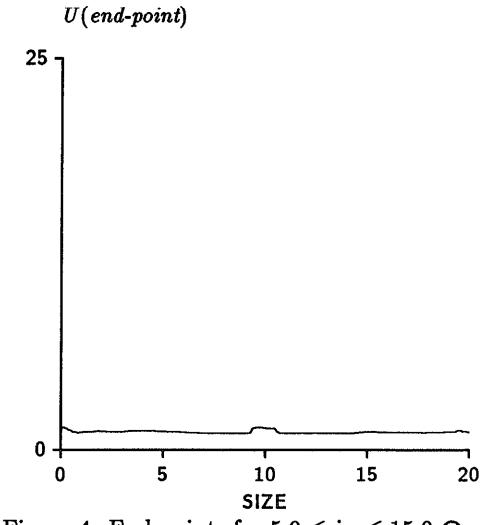

Figure 4: End-points for 5.0 ≤size≤ 15.0 ⊗ red.

weight method used without the other methods; the weight learner alone is only able to describe linearly-separable concepts (which does not include exclusive-or). The Boolean method allows it to overcome this limitation by rewriting its representational language. If the Boolean method could direct the numerical method, then the latter would also be able to move beyond linearly-separable concepts. This is an area for future study.

V. Conclusions and Future Work

This paper describes a three part approach to the task of learning a concept from examples. A connectionist style learning method modifies a simple concept description by changing the weights associated with descriptive elements. A second, symbolic learning method forms Boolean combinations of these descriptive elements and allows the first method to overcome a representational shortcoming. The third learning method divides real-valued attributes into a set of discrete ranges, so other methods are able to construct descriptions of numerical concepts. Overall, the interaction between these three methods is a type of cooperative representational learning. The numeric method changes the bias for both the Boolean and weight method. Similarly, the Boolean method forms new compound elements and thus increases the representational capabilities of the weight adjusting method.

One drawback illustrated in the previous section arises because cooperation between the methods is incomplete. Dynamic feedback is lacking for the numeric method. Though it attempts to form value partitions that allow effective learning by the other methods, it does not use information about the progress of learning at those levels to alter its course of action. Consequently the numerical method can only uncover partitions that can be utilized by the weight method: if a concept involves an exclusive-or of a real-value range, STAGGER is unable to discover it.

Nevertheless, the integrated approach described here serves to identify a source of useful interaction between the representations underlying concept learning and the efficacy of the mechanisms that rely on them. This framework may also demonstrate additional utility as future research integrates additional learning methods and improves existing ones.

Acknowledgements

Thanks to Rick Granger and Michal Young who provided much of the early foundations for this work; to Ross Quinlan for his assistance in formulating the numeric learning method; to Doug Fisher for insight on the interactions between the weight, Boolean, and numeric learning methods; and to the machine learning group at UCI for ready discussion and suggestions.

References

Bradshaw, G. L. (1985). *Learning to recognize speech sounds: A theory and model.* PhD thesis, Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA.

Duda, R., Gaschnig, J., & Hart, P. (1979). Model design in the Prospector consultant system for mineral exploration. In D. Michie (Ed.), *Expert systems in the micro electronic age*, Edinburgh: Edinburgh University Press.

Lebowitz, M. (1985). Categorizing numeric information for generalization. *Cognitive Science, 9*, 285–308.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Rendell, L. (1986). A general framework for induction and a study of selective induction. *Machine Learning, 1*, 317–354.

Schlimmer, J. C., & Granger, R. H., Jr. (1986). Incremental learning from noisy data. *Machine Learning, 1*, 317–354.

Utgoff, P. E., & Mitchell, T. M. (1982). Acquisition of appropriate bias for inductive concept learning. *Proceedings of the National Conference on Artificial Intelligence* (pp. 414–417). Pittsburgh, PA: Morgan Kaufmann.

Utgoff, P. E. (1986). Shift of bias for inductive concept learning. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.