Machine Learning 1: 317–354, 1986
© 1986 Kluwer Academic Publishers, Boston – Manufactured in The Netherlands

Incremental Learning from Noisy Data

JEFFREY C. SCHLIMMER (SCHLIMMER@ICS.UCI.EDU)
RICHARD H. GRANGER, JR. (GRANGER@ICS.UCI.EDU)
Irvine Computational Intelligence Project, Department of Information and Computer Science, University of California, Irvine, CA 92717, U.S.A.

(Received March 5, 1986)
(Revised May 2, 1986)

Key words: learning from examples, contingency, systematic noise, concept drift, constructive induction

Abstract. Induction of a concept description given noisy instances is difficult and is further exacerbated when the concepts may change over time. This paper presents a solution which has been guided by psychological and mathematical results. The method is based on a distributed concept description which is composed of a set of weighted, symbolic characterizations. Two learning processes incrementally modify this description. One adjusts the characterization weights and another creates new characterizations. The latter process is described in terms of a search through the space of possibilities and is shown to require linear space with respect to the number of attribute-value pairs in the description language. The method utilizes previously acquired concept definitions in subsequent learning by adding an attribute for each learned concept to instance descriptions. A program called STAGGER fully embodies this method, and this paper reports on a number of empirical analyses of its performance. Since understanding the relationships between a new learning method and existing ones can be difficult, this paper first reviews a framework for discussing machine learning systems and then describes STAGGER in that framework.

1. Introduction

The ability to adapt to the environment is an essential quality for any intelligent mechanism. For domains in which learners have extensive previous knowledge, such as electronics, it is appropriate to view learning as being heavily guided by that prior knowledge. However, in domains in which there are no high-quality theories, such as weather or financial prediction, some fundamental methods must be used to guide learning. This paper investigates a bottom-up learning technique which does not rely on a strong domain theory. The specific class of learning tasks addressed fall under the term *concept attainment*, in which there is an *a priori* division of the world into at least two categories. What must be learned is a description useful for predicting the category of a previously unseen instance. For example, in weather prediction, the categories might be rain and no rain. Given a series of weather instances, the problem is to learn which combinations of features allow predicting whether there will be rain or not. An example description might be 'low barometer and either high winds or low temperature plus high humidity.'

This general task of concept attainment (or 'learning from examples') has been widely studied by machine learning researchers; however, several important dimensions of the task have been simplified in the past. For example, few researchers have studied the deleterious effects of errors in the descriptions of instances. In complex domains, it is rarely the case that even the best hypotheses will be 100% accurate; sometimes high wind and low barometer readings lead to rain, and sometimes they do not. Hence the learner must be able to tolerate *noise*. Furthermore, most concept attainment work has assumed that the concepts to be defined are stable over time. Frequently, however, a derived description of a useful concept is disrupted by some change which requires its revision. Consider the fox who must revise his model of prey as summer brings out brown in a rabbit's coat and winter removes it. Similarly, after a volcanic eruption, feature combinations that were once predictive of rain may now fail to be. Furthermore, the learner must be able to distinguish between noise and concept change. At any given prediction failure, the question arises as to whether this failure is simply due to noise, or whether it is indicative that the concept is beginning to drift.

This paper presents a learning method which draws from results in the fields of psychology and mathematics. The heart of the method is based on a distributed concept representation which is composed of a set of dually weighted, symbolic characterizations. Modification of the concept description occurs at two levels: adjustment of the weights and generation of new Boolean characterizations. This latter process constructs more general, more specific, and inverted versions of existing elements of the concept description in order to classify concept instances more effectively. Previous learning is utilized in subsequent concept attainment tasks by adding an attribute for each previously acquired concept to the instance description.

The resulting method is incremental and is capable of learning in the presence of noise and drift. This work has suggested a series of psychological studies as well as interesting mathematical proofs. A program called STAGGER fully embodies this method, and this paper presents a number of empirical studies of its performance. In an attempt to describe these results clearly, the following section reviews a basic component framework for models of learning. The remainder of the paper gives a detailed description of STAGGER in terms of this framework, presents a number of empirical results, and then reviews the processes employed in related systems.

1.1 Components of learning from experience

Though methods in machine learning are remarkably similar in many respects, the mapping between individual techniques can be confusing. Terminology in the field is not yet well defined or stable, and methods are often described in terms of only one specific domain. It is often difficult to determine to what extent some new proposed method is similar to existing ones, or how general an aspect of a method might be across domains. Recently, Easterlin (1985) and Langley and Carbonell (in press) have proposed components of the task of learning from experience; their goal is to clearly describe distinct portions of the general learning task so that each specific learning method may be properly compared with others. Collectively the authors suggest seven component processes:

1. **Clustering**: deciding which objects to group extensionally into a class (e.g., at a zoo, separating animals into dangerous and docile groupings).
2. **Initialization**: forming initial intensional characterizations of the clustered groups of objects (e.g., referring to 'all furry animals' rather than enumerating them).
3. **Projection**: matching intensional descriptions against subsequent experience (e.g., guessing whether a new animal is dangerous or docile).
4. **Evaluation**: determining the effectiveness of clusterings and characterizations in capturing experience (e.g., measuring how accurately 'furry' describes dangerous animals).
5. **Refinement**: modifying a clustering or characterization to improve descriptions (e.g., regrouping animals into carnivores and herbivores).
6. **Aggregation**: deciding which elements of instance descriptions are objects (e.g., focusing on the whole animal rather than considering each limb as a separate object).
7. **Storage**: saving learned concepts so that they may be utilized effectively (e.g., saving 'furry' and other relevant characterizations for matching, but failing to save irrelevant descriptions like 'salty').

This is not intended as an exhaustive list of learning components, but for the purposes of this paper it is a useful starting point for dividing the overall task of concept formation into intuitive parts.[1]

2. Overview of STAGGER

In terms of these components, STAGGER assumes a solution to the clustering task, since the world is predivided for it into positive and negative classes (e.g., predictors of rain versus predictors of no rain). This assumption is common to all learning from examples or concept attainment systems. STAGGER's input consists of individual descriptions of instances in the world, along with labels indicating their class.

[1] In addition to the learning from examples or concept attainment task, the tasks of learning search heuristics, analytic learning, conceptual clustering, and others may also be described in this framework. We refer the reader to Easterlin (1985) and Langley and Carbonell (in press) for more examples.

STAGGER's initial concept description is a collection of the simplest possible features (e.g., input features: temperature, barometer reading, humidity). Each of these simple characterizations has an initially unbiased pair of 'weights.' The projection process matches this distributed concept representation against new instances. For each characterization, one of its weights is used if it is matched in this instance, and the other weight is used if it is unmatched. Weights are adjusted by the evaluation process which keeps track of the number of times each characterization is matched or unmatched in examples and nonexamples. The refinement process unifies this distributed concept representation by combining individual elements to form more complex Boolean characterizations (e.g., low temperature and either low barometer or high humidity).

The major component processes in STAGGER are initialization, projection, evaluation, and refinement. This paper first describes these component processes in STAGGER and then presents a series of examples to illustrate their operation.

2.1 Initialization

Initialization is the process of forming initial intensional descriptions of clustered groups of objects. In concept attainment, this description is used to predict the category of previously unseen instances. For domains containing noise, a description will not be a perfect predictor of a category. Rather, the description must represent the extent to which particular combinations of features are thought to be in the class of positive instances (e.g., rain predictors). In STAGGER, a concept description is a set of numerically weighted characterizations. The individual description elements are represented by Boolean functions of attribute-values, and each is dually weighted to indicate its predictiveness. This representation includes conjunction, disjunction, and negation.

The initial concept description is the set of all single attribute value pairs. These are initially assigned a pair of unbiased weights. During the projection process of matching a given instance against this concept description, each of the individual elements are examined and one of their weights influences the cumulative prediction made. One weight indicates the predictive value of an element when it is matched, and the other indicates its predictive value when unmatched. For pedagogical simplicity, consider the initial concept description for a domain of objects that can be described by **size** $\in$ {**small, medium, large**}, **color** $\in$ {**red, blue, green**} and **shape** $\in$ {**circle, square, triangle**}. The initial characterizations are:

Characterization	Weights	
size = small	1	1
size = medium	1	1
size = large	1	1
color = red	1	1
.		.
.		.
.		.
shape = triangle	1	1

In this case there are nine initial characterizations. These simple characterizations are combined as necessary to form more general or more specific concept description elements by the search operators described in Section 2.4. An individual element is represented as a disjunctive list of conjunctive clauses; the conjunctive forms are represented by a list of the acceptable values for each attribute. For example, a characterization matching any object which is either small and red or is not a square would be represented as: **(size = small *and* color = red) *or* shape = (circle *or* triangle)**. Negation of a specific attribute's value is expressed by the disjunction of all of the possible values except the one negated; negating more complex characterizations is done by applying DeMorgan's theorem. Note that this representation is Boolean complete, and is therefore sufficient to express disjunction within an attribute and disjunction between attributes. However, it cannot represent *relations* such as 'two objects with the same color.'

2.2 Projection

Projection is the process of matching intensional descriptions against subsequent experience. In STAGGER, projection matches each characterization element in the concept description against a new instance. If an individual element is matched in this instance, one of its weights is used to increase expectation of a positive instance. If the element is unmatched, the other weight. is used to decrease expectation of a positive instance. This collective expectation is used to decide whether some new instance is positive or negative. This approach differs from most traditional machine learning systems in which a single characterization completely influences concept prediction.

STAGGER utilizes Bayesian formulae to weight each characterization. These formulae were originally derived in work done for the Prospector mineral exploration system (Duda, Gasching, & Hart, 1979). The first of the two formulae, logical sufficiency (LS), approximates the degree to which the presence of a feature (F) increases expectation of an outcome (O).

$$LS = \frac{p(F|O)}{p(F|\neg O)} \tag{1}$$

LS ranges from zero to positive infinity and is interpreted in terms of odds. Odds in favor of an outcome may be easily converted to probability by noting that the probability of the outcomes is $p = \frac{odds}{1+odds}$. An LS value less than unity (approaching zero) indicates a negative correlation, unity indicates independence, and a value greater than unity indicates a positive correlation.

Logical necessity (LN) is the second formula. It approximates the degree to which the *absence* of a feature *decreases* expectation of an outcome, and it is defined as:

$$LN = \frac{p(\neg F \mid O)}{p(\neg F \mid \neg O)} \tag{2}$$

LN also takes on values from zero to positive infinity. However, an LN value less than unity indicates a positive correlation because the absence of the feature predicts an absence of the outcome; a value greater than unity indicates a negative correlation. For both LS and LN, unity indicates that the feature is irrelevant to the outcome.[2]

Projection computes an expectation of class membership by multiplying the prior odds of a positive instance and the LS weights of all matched characterizations with the LN weights of unmatched characterizations.

$$Odds(positive \mid instance) = Odds(positive) \times \prod_{\forall matched} LS \times \prod_{\forall unmatched} LN \tag{3}$$

The resulting number represents the odds in favor of a positive instance. A value much greater than unity indicates confidence that this instance is positive; a value less than unity indicates it is negative, and a value near unity indicates uncertainty.

This process is quite similar to the weighted featural sum computation of the general processing algorithm given by Smith and Medin (1981); that is, category membership is calculated by summing the weighted values of each feature that matches a specific instance. Both of these calculations have the desirable property that the prototypicality of a new instance is computed. An instance with a very high projection score is highly prototypical; one with a socre slightly greater than unity is less so. Though this work does not attempt to duplicate various typicality effects (Smith & Medin, 1981), the projection process does exhibit them.

STAGGER's projection process matches a new instance against all previously acquired concept descriptions, generating an expectation for each concept attainment task; that is, it attempts to recognize as many concepts as it can in a single instance. For example, a park green could be recognized as both a picnic area

[2] It can be shown that $LS = 1$ if and only if $LN = 1$. Furthermore, when $LS > 1$ it will be the case that $LN < 1$ and vice versa. However, in general, $LS \neq 1/LN$. For example, if $p(F|O) = 3/10$ and $p(F|\neg O) = 1/10$ then $LS = 3$ and $LN = 7/9$.

and a spot for a ballgame if STAGGER has previously acquired both of those concept definitions. Expectation of one concept may further be used as a predictive characterization for another. Section 3.4 describes how STAGGER augments an instance description by projecting over previously attained concepts. In that example, learning about a simple concept for chess endgames eases learning about a more complex one.

In addition to representing concepts in a distributed manner and using Bayesian measures to compute a prototypical expectation, STAGGER incrementally modifies both the weights associated with individual characterizations and the structure of the characterizations themselves. These two latter abilities allow STAGGER to adapt its concept description to better reflect the concept.

2.3 Evaluation

Evaluation is the process of determining the effectiveness of the internal concept descriptions. In STAGGER, each characterization in the concept description is continually evaluated by adjusting its weights. This evaluation reflects psychological findings in learning and is based on the number of times each characterization has succeeded and failed as a predictor.

2.3.1 Psychological data

The Bayesian measures used by STAGGER to evaluate characterizations are influenced by results in the field of animal learning. In classical conditioning, animals learn to associate a novel stimulus with an unpleasant stimulus even when the two stimuli are not perfectly paired. This corresponds directly to a concept attainment task with noise: the trials are the instances, the unpleasant stimulus represents the class information of each instance (positive or negative instance), and the presence of various other stimuli constitutes the instance description. Psychologists discovered in the late 1960s that, for a rat to associate a novel stimulus (NS) with an unpleasant stimulus (US), the likelihood of the unpleasant stimulus in the presence of the novel stimulus must be greater than the likelihood of the unpleasant stimulus in the absence of the novel stimulus (Rescorla, 1968). Stated in mathematical terms, the relation $p(\mathrm{US}|\mathrm{NS}) > p(\mathrm{US}|\neg\mathrm{NS})$ must hold for learning to occur.[3] This characteristic of learning in classical conditioning is known as *contingency*; the situations covered by the formula can be broken into four possible cases.

- When the novel stimulus and the unpleasant stimulus always occur together, $p(\mathrm{US}|\mathrm{NS}) = 1 > p(\mathrm{US}|\neg\mathrm{NS}) = 0$ and thus the inequality holds. This

[3] This is a bit of an oversimplification. Animals learn that the novel stimulus and unpleasant stimulus are *irrelevant* to each other if $p(\mathrm{US}|\mathrm{NS}) = p(\mathrm{US}|\neg\mathrm{NS})$ and learn that the novel stimulus predicts the *absence* of the unpleasant stimulus if $p(\mathrm{US}|\mathrm{NS}) < p(\mathrm{US}|\neg\mathrm{NS})$.

corresponds to a noise-free situation because the novel stimulus perfectly predicts the unpleasant stimulus; neither occurs separately.

- Moreover, if the unpleasant stimulus always follows the novel stimulus, but the novel stimulus sometimes occurs separately, $1 > p(\text{US}|\text{NS}) > p(\text{US}|\neg\text{NS}) = 0$. This case corresponds to *partial reinforcement* since the novel stimulus is only partially reinforced by the unpleasant stimulus. Extensive testing indicates that learning also occurs in this case (e.g., Fitzgerald, 1963).
- Furthermore, if the novel stimulus always leads to the unpleasant stimulus, but the unpleasant stimulus sometimes occurs alone, then $p(\text{US}|\text{NS}) = 1 > p(\text{US}|\neg\text{NS}) > 0$. Here the novel stimulus always leads to the unpleasant stimulus, but so may other cues. We have recently termed this case *partial warning* (Granger & Schlimmer, in press) and are currently investigating human and animal learning in this condition. Our hypothesis is that learning occurs in this case as well.
- In the fourth possibility, the novel stimulus and unpleasant stimulus sometimes occur together and sometimes occur separately. The inequality may not hold, for $1 > p(\text{US}|\text{NS})\ ?\ p(\text{US}|\neg\text{NS}) > 0$. Learning fails to occur if the two stimuli are unpaired even a small number of times (Rescorla, 1968).

This differential tolerance to isolated stimuli has been replicated extensively with different animal subjects (Gamzu & Williams, 1971) and in human experiments (Wasserman, Chatlosh, & Neunaber, 1983).

2.3.2 Calculating contingency

The above four types of conditioning situations may also be described from a machine learning point of view. Consider the possible situations that may arise when matching a concept description element against an instance. The characterization in question may either be satisfied by the current instance features or not. Similarly, the instance may be either positive or negative. Following the terminology used by Bruner, Goodnow, and Austin (1956), a positive instance is positive evidence which may either *confirm* the predictiveness of a characterization (if it is matched in this instance) or *infirm* the characterization's predictiveness (if it is unmatched). Similarly, a negative instance is negative evidence which either confirms an unmatched element or infirms a matched one. Table 1 summarizes these possibilities. Contingency theory states that, in humans and animals, learning is impaired in situations containing both positive and negative infirming evidence. In situations containing only positive or only negative infirming evidence, learning proceeds unhindered. Note that a positive infirming instance may also be termed an *error of omission* (since the characterization was omitted), while a negative infirming instance is sometimes termed an *error of commission*. However, in the context of this paper we find it useful to distinguish between the *prediction* made by the program and the *match* of some characterization. Thus, an error of commission is an occasion when the program predicts a positive instance that is in fact negative.

Table 1. Possible situations in matching a characterization to an instance

Instance	Characterization	
	Matched	Unmatched
Positive	Confirming (C_P)	Infirming (I_P)
Negative	Infirming (I_N)	Confirming (C_N)

Negative infirming evidence is the simple presence of a characterization in a negative instance; no prediction is involved.

The Bayesian weighting measures *LS* and *LN* may be easily calculated for each characterization by keeping counts of the possible situations listed in Table 1. The formulae for *LS* and *LN* in terms of the counts of each situation are:

$$LS = \frac{C_P(I_N + C_N)}{I_N(C_P + I_P)} \qquad LN = \frac{I_P(I_N + C_N)}{C_N(C_P + I_P)} \tag{4}$$

Derivations of these formulae appear in Appendix A.

The prior odds of a positive instance is also part of the projection process which matches the distributed concept description onto an instance. This may also be easily computed by dividing the sum of positive evidence for any characterization by the sum of the negative evidence for that same characterization: prior odds = $(C_P + I_P)/(I_N + C_N)$.

Though these measures only bear a surface similarity to the contingency inequality (i.e., $p(\text{US}|\text{NS}) > p(\text{US}|\neg\text{NS})$), as Section 3.2.1 describes, *LS* and *LN* predict the same learning in the four situations enumerated above.

If STAGGER limited its learning to adjustment of the characterization weights via evaluation, the distributed concept representation would be sufficient to describe accurately the class of 'linearly separable' concepts (Hampson & Kibler, 1983). This class is rather small, however. For instance, Exclusive-Or is not linearly separable. In this respect STAGGER is similar to *connectionist* models of learning when those models do not have any 'hidden' units. The purpose of the hidden, internal units is to allow the encoding of more complicated concepts. The refinement process of STAGGER serves an analogous purpose: individual characterizations are combined into more complex Boolean functions.

2.4 Refinement

The process of refinement modifies learned concepts to improve their effectiveness as measured by evaluation. In STAGGER, the elements of the distributed concept description are modified by specialization, generalization, and inversion. Modifications are triggered by errors in projection and are heuristically guided by the

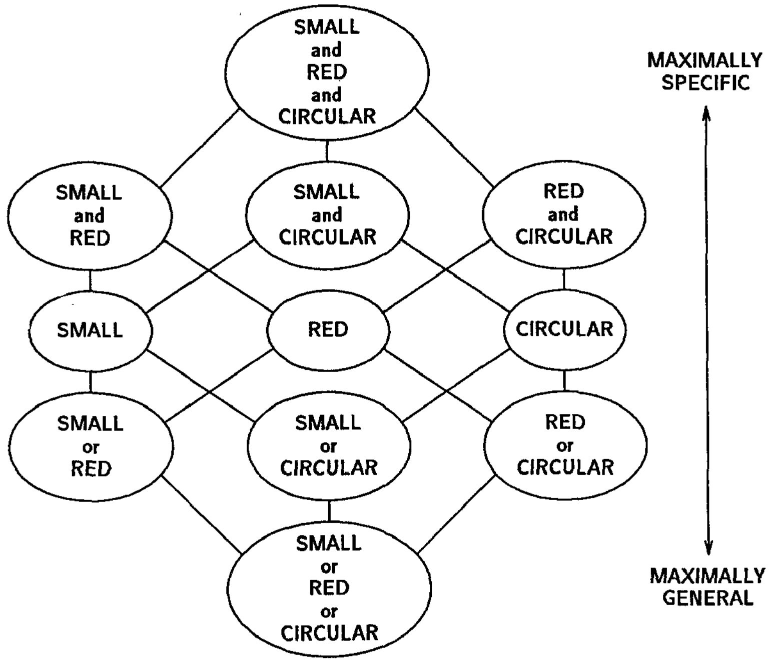

Figure 1. Partial characterization search space.

Bayesian evaluation measures. Only a frontier of effective characterizations is maintained and, over time, refinement condenses the distributed concept description into a unified characterization. Backtracking is employed if necessary to recover from ineffective refinement.

2.4.1 Search space and operators

STAGGER searches through a space of possible characterizations as it refines its initial distributed representation of the concept into a unified, accurate one. Each possible boolean characterization of attribute values may be viewed as a node in the space of all such functions. Figure 1 depicts a small portion of this space over the simple domain described earlier (each ellipse represents a Boolean function). Any two of the possible Boolean functions are partially ordered along a dimension of generality (Mitchell, 1982). Nodes higher in Figure 1 are more *general* while more *specific* nodes are below.

STAGGER's initial concept description consists of the simple characterizations in the middle of Figure 1. Notice that this space is more than twice the size of that typically searched by a conjunction-only method like version spaces (Mitchell, 1982). Another interesting difference is that the version space method searches its space of characterizations from both sides toward the middle; STAGGER beam-searches (Lowerre & Reddy, 1980) from the simplest points in the middle outward toward both boundaries. In order to limit memory requirements, only a frontier of highly ranked characterizations is retained; hence, the search is linear with respect to the number of attribute-value pairs in the description language.

STAGGER's three search operators correspond to specializing, generalizing, or inverting characterizations. To make a concept description element more specific, search proceeds down a path which is a conjunction of two good elements currently in the search frontier. Complex characterizations are conjoined by first pairwise conjoining each clause of the disjuncts and then disjoining all of the results (see Section 2.1). Conversely, to make a more general element, search proceeds to a new disjunction. Lastly, a poorly scoring characterization may be negated; this does not raise or lower its degree of generality. Complex negation is accomplished via application of DeMorgan's theorem.

2.4.2 Operator preconditions

The conjunction, disjunction, and negation operators are not applied exhaustively; search is limited by proposing new elements only when STAGGER makes an error in projection. Furthermore, the type of error heuristically guides the direction of search. When a negative instance is predicted to be positive (an error of commission), projection is behaving in too general a manner. Thus search is expanded toward a more specific characterization, and a new conjunction is proposed from two good elements. On the other hand, a guess that a positive instance is negative (an error of omission) is overly specific; search is expanded to include a more general characterization by proposing a new disjunction. Either type of projection error also causes STAGGER to expand search by proposing the negation of a poor characterization. Table 2 summarizes these search direction heuristics.

The criteria for selecting good candidate elements differ between operators. STAGGER follows a two-step process of choosing good component characterizations. One set of heuristics *nominates* potential candidates, and a second set *elects* the crucial ones for inclusion in new characterizations.

The nomination heuristic specifies alternative groups of characterizations from which to form compounds, depending on the search operator and the type of prediction error made. After STAGGER has made an error of commission, concept description elements matched in this negative instance may be partially necessary, but are clearly not sufficient. Some elements must have suggested (via

Table 2. Search direction heuristic

Projection	Actually	Error type	Search direction	Boolean function
Positive	Negative	Commission	Specialize	AND[c1, c2]
Negative	Positive	Omission	Generalize	OR[c1, c2]
—	—	Either	Invert	NOT[c]

Table 3. Nomination heuristic

Error type	Function	Characterization nomination
	AND[c1, c2]	Matched, Unmatched
Commission	OR[c1, c2]	Unmatched, Unmatched
	NOT[c]	Matched
	AND[c1, c2]	Matched, Matched
Omission	OR[c1, c2]	Matched, Unmatched
	NOT[c]	Unmatched

the projection process) that this instance was likely to be positive, but because this instance was negative, some necessary element was unmatched. Conjunction combines two necessary elements, so matched characterizations are nominated along with unmatched ones. If a disjunction is formed, elements which are unmatched in this nonexample are nominated since disjunction combines two sufficient characterizations, and no sufficient characterizations were present. Negation is used to invert characterizations which predict nonexamples. Its component is nominated from those characterizations which are matched in this nonexample.

In an error of omission (an unpredicted example), all necessary characterizations must have been present, but some sufficient ones may have been missing. Again, conjunction combines necessary characterizations, so matched elements are nominated. Disjunctions are constructed from a known sufficient characterization (one matched) and another potentially sufficient (one unmatched). Nominating unmatched elements takes into account that STAGGER failed to predict an example: some characterization it considered important was missing. A new negation is nominated from the unmatched characterizations. Table 3 summarizes STAGGER's nomination heuristic.

The nomination heuristic indicates potentially valuable candidates for conjunction, disjunction, and negation. However, more characterizations are nominated than are needed by the search operator. The election heuristic further narrows the possible candidates by electing the crucial elements from those nominated. A Bayesian evaluation measure elects the most predictive of the nominated characterizations. Consider a situation leading STAGGER to propose appropriately a new conjunction. For example, the familiar concept *father*: a parent and a male. The two characterizations (parent and male) are always present in a positive instance (father) though they sometimes occur alone (a brother is male). This is negative infirming evidence (refer to Table 1). As Section 3.2.1 illustrates, *LN* tolerates negative infirming evidence and therefore elects criterial elements for conjunctions. In practice, a conjunction would be formed from the lowest *LN* element matched in the nonexample and the lowest *LN* unmatched concept description element. By a similar argument, the converse Bayesian measure, *LS*,

Table 4. Election heuristic

Function	Election measure
AND[c1, c2]	$LN(ci) \ll 1$
OR[c1, c2]	$LS(ci) \gg 1$
NOT[c]	$LN(c) \gg 1$ or $LS(c) \ll 1$

elects high scoring characterizations to be used in forming new, disjunctive characterizations. New negated characterizations are elected equally by both Bayesian measures. Table 4 summarizes this second step candidate election heuristic.

2.4.3 Pruning and backtracking

New characterizations are introduced into the search frontier in a generate-and-test manner. The search operator heuristics generate new characterizations which are then either pruned from the frontier or established as part of it. If an established concept description element becomes ineffective, search backtracks and the established characterization is pruned.

A characterization is immediately pruned unless (a) all of its sponsoring components are part of the established search frontier and (b) at least one of the components is not sponsoring any other characterizations. This effectively limits the size of the search frontier to at most twice the number of attribute-value pairs, because each expanding characterization requires at least one uncommitted sponsor from the among the established elements.

For a new characterization to avoid being pruned and become established as part of the search frontier, it must be more effective than its sponsoring components. At the time of its introduction, a threshold based on the Bayesian values of the components is stored with the new characterization. If the new element surpasses this threshold, it is established as part of the search frontier and the components are pruned. The interim performance of a new characterization is assessed by examining recent changes in its Bayesian values (e.g., the last 10). These changes are averaged and, if this average is very small (e.g., below 0.1), the element appears to be reaching an asymptote. If this new characterization is still below the designated threshold, it is pruned.

The threshold for a new, conjunctive characterization is based on the *LN* values of its sponsoring components. Because the new, more specific element matches less often than its sponsoring components do, it is guaranteed to have the same or less negative infirming evidence. Therefore, if the new characterization also has less positive infirming evidence, it is more effective than its components. *LN* is the appropriate measure of competition since it *sensitively* measures positive infirming evidence (see Section 3.2.1). By similar reasoning, the competition measure for a

Table 5. Pruning heuristic

Boolean function	Threshold	Prune	Establish	Impeach
AND [c1, c2]	T = min{*LN*(c1), *LN*(c2)}	$\not< T$	$< T$	$> T$
OR [c1, c2]	T = max{*LS*(c1), *LS*(c2)}	$\not> T$	$> T$	$< T$
NOT [c]	T = 1/*LN*(c)	$\not< T$	$< T$	$> T$
	T = 1/*LS*(c)	$\not> T$	$> T$	$< T$

more general characterization is *LS*. Inverted characterizations compete by exceeding the mathematical inverse of the selection measure used. For example, a negation proposed from a small *LS* score would need to surpass 1/*LS* to avoid being pruned.

When *STAGGER* advances the search frontier inappropriately, it may perform the functional equivalent of backtracking by reversing the direction of search (see Table 2). However, since the search space behind the frontier has been pruned, simply using the search operators described above may not be sufficient to recover the appropriate characterization. For this reason, when a new concept description element is established and its sponsoring components are pruned, those components are saved with the new characterization. If the Bayesian evaluation functions indicate that the new characterization is currently performing worse than when it was established, it is *impeached*. The saved components are reactivated and compete as the failing element did before. This amounts to chronological backtracking because moves through the search space are retracted in the opposite order from which they were proposed. Table 5 summarizes the pruning heuristic.

STAGGER searches from simple toward complicated descriptions, stopping when the characterizations accurately describe the concept. The concept description elements are as simple as possible, and this preference for parsimony is a major source of *bias*. The three search operators employed by STAGGER are triggered by projection errors and they use nomination and election heuristics to generate new characterizations. These new parts of the concept description are then either pruned out or added to the concept description frontier according to their performance as assessed by the evaluation measures. If a characterization fails to perform as well as it has in the past, backtracking is triggered to unwind the search. Overall, the beam-search nature of refinement in STAGGER allows modification of the distributed concept representation while remaining within a reasonable memory size.

2.5 Summary

STAGGER forms an initial concept description from a weighted set of attribute-

Table 6. Pseudo code for the STAGGER program

```
(defun stagger (concepts instance)
  (prog (concept odds guess)
        (if (is-a-new concept) then
           (setq concept (initial-characterizations))
           (grow-new-concept concept concepts)
        (setq instance (aggregate-over concepts instance))
        (setq odds (project-over instance))
        (setq guess
              (if (> odds 1) then 'positive
               elseif (< odds 1) then 'negative
               else (random))
        (evaluate-all (characterizations-for concepts))
        (if (guess-agrees-with instance guess) then nil
         elseif (error-is-a 'commission) then
           (propose-a-conjunction instance concepts)
           (propose-a-negation instance concepts)
         elseif (error-is-an 'omission) then
           (propose-a-disjunction instance concepts)
           (propose-a-negation instance concepts)
        (prune-ineffective-characterizations concepts)))
```

value characterizations. These characterizations collectively influence prediction of the class of subsequent instances. Evaluation serves to adjust the weights of the distributed description while refinement consolidates it. In Table 6, pseudo code for the top level of STAGGER is listed to clarify further the functioning of the system. New concept attainment tasks have their search frontier initialized to the singletons of each attribute-value pair. The description of the instance is then augmented with attributes projected from previously acquired concepts. The matching process of projection yields a prototypicality score for this instance which is used to predict its class. An evaluation process then ranks each competing characterization according to its effectiveness at predicting the class correctly. If a mistake was made in classifying the instance, characterization refining operators take a single step in the search toward more accurate characterizations. Points along the search frontier which prove ineffective are pruned.

3. Examples of STAGGER at work

STAGGER is a fully implemented Franz Lisp program. We have tested its performance in a number of domains, including simple weather prediction, animal classical conditioning, and chess endgame classification. The following examples demonstrate the ability of STAGGER to tolerate noisy instances, track changing concepts over time, and utilize previous learning to aid subsequent learning.

Table 7. Sample weather instances (after Quinlan, 1985)

Attributes				Instance
Outlook	Temperature	Humidity	Windy	
Sunny	Hot	High	True	Negative
Sunny	Hot	High	False	Negative
Sunny	Mild	High	False	Negative
Sunny	Mild	Normal	True	Positive
Sunny	Cool	Normal	False	Positive
Overcast	Hot	High	False	Positive
Overcast	Hot	Normal	False	Positive
Overcast	Mild	High	True	Positive
Overcast	Cool	Normal	True	Positive
Rain	Mild	High	True	Negative
Rain	Mild	High	False	Postive
Rain	Mild	Normal	False	Positive
Rain	Cool	Normal	True	Negative
Rain	Cool	Normal	False	Positive

3.1 Classifying weather

STAGGER is well suited for the domain of weather prediction. Classifying situations that lead to tornadoes, hurricanes, or droughts requires tolerating occasionally misleading data. Furthermore, previously effective predictors of weather events may be nullified by global changes in the world weather situation, such as a volcanic eruption. As an example, consider the simple set of 14 weather instances in Table 7 (after Quinlan, 1985):

The characterization that STAGGER eventually forms for this set of instances is **(humidity = normal *and* windy = false) *or* outlook = overcast**. This characterization accurately covers 12 of the 14 instances. The remaining two instances are correctly classified by the distributed concept description as a whole.

Consider an intermediate state, before any modifications have been made to the distributed concept description. After processing a few initial instances, the concept description looks like Table 8. Note that the evidence counts are initialized to 10 and, after being incremented, they have decayed to 99% of their value.

Given a new instance in which **outlook = rain, temperature = cool, humidity = normal**, and **wind = true**, this concept description is projected by multiplying the *LS* and *LN* weights of the individual characterizations; the *LS* weights of each matched characterization are multiplied together with the *LN* weights of each unmatched characterization and the prior odds. The weights used for this instance are italicized in Table 8. The prior odds are approximated by dividing the number of times instances were positive ($C_P + I_P$) by the number of time instances were negative ($C_N + I_N$). The expected odds in favor of this instance being positive are 1.42 to 1.

However, this particular instance is negative. The evaluation process in

Table 8. Intermediate weather concept description

Characterization	C_P	C_N	I_P	I_N	*LS*	*LN*
outlook = sunny	10.48	10.78	15.35	12.99	0.74	*1.31*
outlook = overcast	12.56	14.49	13.28	10.00	1.19	*0.87*
outlook = rain	11.75	12.99	14.17	10.78	*1.00*	1.00
temperature = cool	11.36	13.51	14.48	10.78	*0.99*	1.01
temperature = mild	12.58	13.43	13.26	10.57	1.11	*0.92*
temperature = hot	11.32	11.55	15.07	12.45	0.83	*1.19*
humidity = normal	14.19	13.51	11.64	10.78	*1.24*	0.81
humidity = high	11.64	10.78	14.19	13.51	0.81	*1.24*
windy = true	10.65	12.00	15.18	11.77	*0.83*	1.16
windy = false	15.18	11.77	10.65	12.00	1.16	*0.83*

Prior odds = 1.09 $\Pi_{LS} = 1.02$ $\Pi_{LD} = 1.28$
Odds = 1.42

STAGGER first updates the counts of negative confirming and infirming evidence for each characterization. For instance, the count of negative infirming evidence for the characterization **outlook = rain** is incremented because this characterization was matched in this negative instance. Conversely, the negative confirming count is incremented for **outlook = overcast** since this characterization was unmatched.

This error of commission also triggers the refinement process to search for a more specific characterization. The lowest *LN* characterization matched in this negative instance is conjoined with the lowest *LS* characterization unmatched in this nonexample. In this case the new characterization **humidity = normal *and* windy = false** is added to the concept description. The threshold that this new characterization must surpass to become an established part of the search frontier (and avoid being pruned) is the minimum of the *LN* values of the two sponsoring characterizations, or min {0.81, 0.83} = 0.81. The inversion operator also constructs a new characterization which is the negation of the matched, poorly scoring characterization **windy** = **true**. Since **windy** = **false** is already a part of the concept description frontier, the component **windy** = **true** is immediately pruned.

Over time, STAGGER continues to apply its refinement search operators as it makes errors in projection. The evaluation process provides a measurement of candidates for refinement as well as indicating appropriate occasions for pruning. An accurate concept description is a carefully weighted set of potentially complex characterizations. In this case, a single characterization does 86% of the work of categorization and the distributed representation handles the remaining 14%.

3.2 Tolerating noisy data

Like weather prediction, real-world concept attainment tasks will inevitably entail inaccuracies in the description of instances. These descriptions may be subject to either *random* or *systematic* variation. An example of random noise would be a temperature sensor which is accurate to within 10% of its operating range. It may read too high on one occasion and too low on another; the *direction* of its error is random. Only a few authors have dealt with this possibility. The most notable is Quinlan (1983), who identifies several potential sources of random errors: faulty measurement, ill-defined thresholds (e.g., when is a person 'tall'?), and subjective interpretations of a multitude of inputs (e.g., what criteria are used when describing a person as 'athletic'?).

However, it may often be the case that errors in description are the result of a systematic variation. For example, a smoke detector may give a false alarm though it does not fail to detect real fires. Or a rain gauge may leak and sometimes read lower, but never higher, than it should. The errors of these latter two instruments are *systematically* of one type (e.g., only to low for the rain gauge), though they may occur with an unpredictable frequency. Systematic noise is defined as the occurrence over instances of positive infirming events or negative infirming events, but not both.

When random variation occurs, a learning system may attempt to extract the 'real' concept from the extraneous instance information, up to a point. When all of the instances are subject to random variation, there is no distinction between the concept and the noise itself. When there is systematic variation (i.e., only one type of infirming evidence), however, it is possible to tolerate a much higher level of variation. This section first explains the ability of *LS* and *LN* to tolerate systematic variations better than random variations and then depicts the overall ability of STAGGER differentially to tolerate systematic versus random noise.

3.2.1 Evaluation of noisy data

Empirically analyzing the behavior of the *LS* and *LN* measures indicates that they

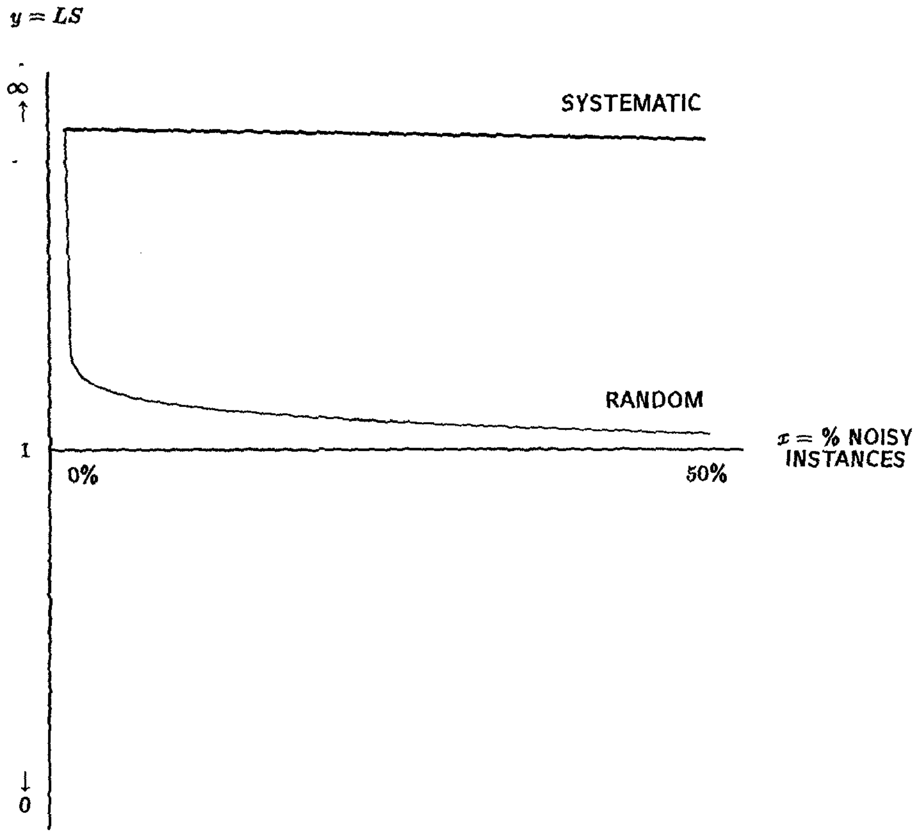

Figure 2. Relative ratings of systematic versus random variations.

differentiate between systematic and random variations in instance descriptions, rating the former highly and the latter poorly. Figure 2 plots the relative ratings of systematic and random variation situations.

In Figures 2, 3, and 4, the x-axis denotes the percentage of instances that were subject to potential variation. The scale ranges from a noise-free situation (on the right) to one in which half of the instances were randomly identified as positive or negative (on the left). The y-axis is plotted on a log scale and denotes the magnitude of LS for a single feature. The y-scale ranges from maximal correlation (at the top), to irrelevance (the midpoint), back to maximal correlation (at the bottom). The further a point is from the midpoint the more relevant it is.

For both of the cases depicted in Figure 2, half of the instances were positive and contained the feature in question; the other half were negative instances and did

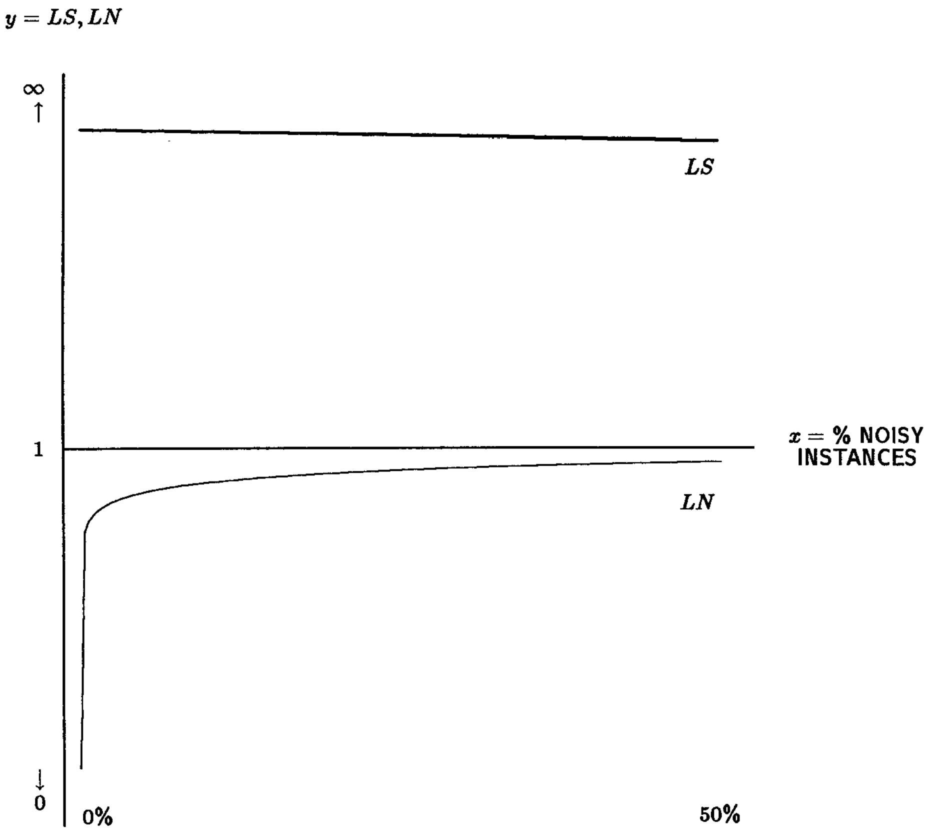

Figure 3. Positive infirming.

not contain the feature.[4] The systematic variation case (upper, heavy line) was generated by varying the percentage that a negative instance might be erroneous; the random variation case (lower, lighter line) was generated by varying the percentage that *either* type of instance might be erroneous.

Introducing randomness in only one type of instance adds only one type of infirming evidence. In the systematic case depicted in Figure 2, mutating negative instances into positive ones introduces positive infirming evidence. The faulty rain gauge falls into this case, since it sometimes leaks and reads too low (positive infirming evidence), but it never indicates precipitation unless it has been raining

[4] Figures 2, 3, and 4 assume that there are the same number of positive and negative instances. This assumption is relatively harmless, since the shape and magnitude of the curves remain constant as the ratio between the two is varied.

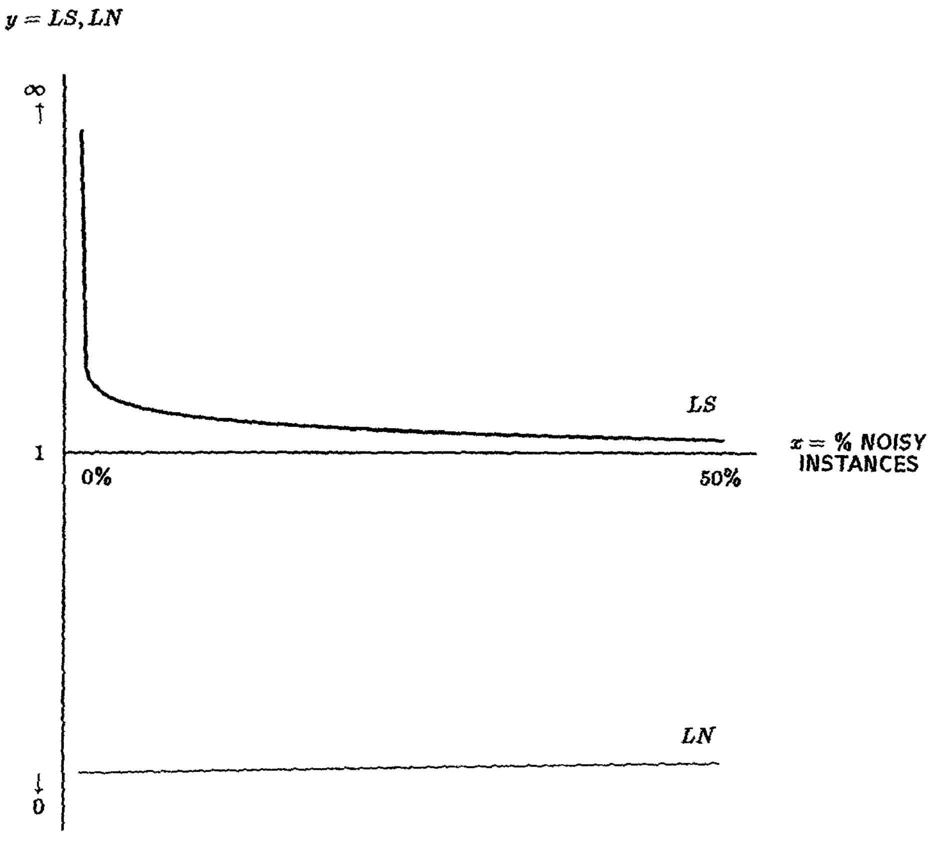

Figure 4. Negative infirming.

(no negative infirming evidence).

Adding randomness to either type of instance, however, introduces both types of infirming evidence. The random variation case (lower, light line) contains a combination of positive infirming evidence (some negative instances were identified as positive) and negative infirming evidence (some positive instances were identified as negative). A temperature gauge which sometimes reads too low and sometimes reads too high behaves in a similar manner. Notice that subjecting an instance to a random identification process is not the same as always replacing the class information of an instance a given percentage of the time for, in the latter case, a noise level of 100% would result only in an inversion, but no loss, of information.

The *LS* measure tolerates positive infirming evidence, but ranks positive and negative infirming situations poorly. However, *LS* also ranks negative infirming evidence situations poorly. *LN* provides the opposite characteristics, for it tolerates negative infirming situations but not a mixture of the two types. Figure 3 and 4 depict the evaluations yielded by *LS* and *LN* under positive infirming and negative infirming situations, respectively (recall that unlike *LS*, *LN* values less than unity indicate correlation).

The use of *LN* as an election heuristic for conjunctions and *LS* for disjunctions is based on the above analysis. Characterizations which should be combined into a conjunction accrue negative infirming evidence; *LN* tolerates this type of systematic variation. Similarly, *LS* tolerates positive infirming evidence which potential disjuncts accumulate.

The tolerance of these measures to either positive or negative infirming evidence, but not both, corresponds closely to the findings of contingency experiments. Section 2.3.1 identifies four situations corresponding to types of trial presentation conditions. In the no noise case, the partial reinforcement case, and in the partial warning case, learning occurs in all but the noisiest of situations. Each of these conditions contains at most one of the two types of infirming evidence, and thus the Bayesian measures presented here will indicate relevance between potential predictors and outcomes. In the fourth case, however, which includes unpaired novel and unpleasant stimuli, even small amounts of infirming evidence inhibit learning. This phenomenon is mirrored by the intolerance of *LS* and *LN* to a mixture of positive and negative infirming evidence.

An interesting effect of these evaluation functions is that they may lead the projection processes in STAGGER to make instance classification errors. For example, if a particular attribute-value pair is subjected to a large degree of systematic, negative infirming evidence (always present in positive instances but sometimes present in negative instances) it will be ranked highly (Figure 4). When this attribute-value pair is present in an instance description, projection is likely to make a positive prediction; a feature with negative infirming evidence will lead to errors of commission. In this case, the performance of STAGGER is violating the *consistency* condition (Michalski, 1983) since it is failing to exclude all negative instances from the positive class. Animal subjects behave similarly in analogous partial reinforcement situations (Fitzgerald, 1963). Conversely, if an attribute-value pair is subject to systematic positive infirming evidence, STAGGER's errors of omission will be violating the *completeness* condition (i.e., failing to include all positive instances in the positive class). This models the *partial warning* condition recently proposed by Granger and Schlimmer (in press). An attribute-value pair with random variation, however, will be ranked poorly. STAGGER will not be misled to classify instances incorrectly.

3.2.2 Program performance with noisy data

The tolerance of STAGGER's evaluation measures to noise is also reflected in the system's overall performance. Figure 5 depicts STAGGER's performance after noisy training for a simple conjunctive concept. The upper, heavy line portrays average performance when the training cases were subjected to varying percentages of negative infirming, systematic noise. Even at the 50% noise level when all of the

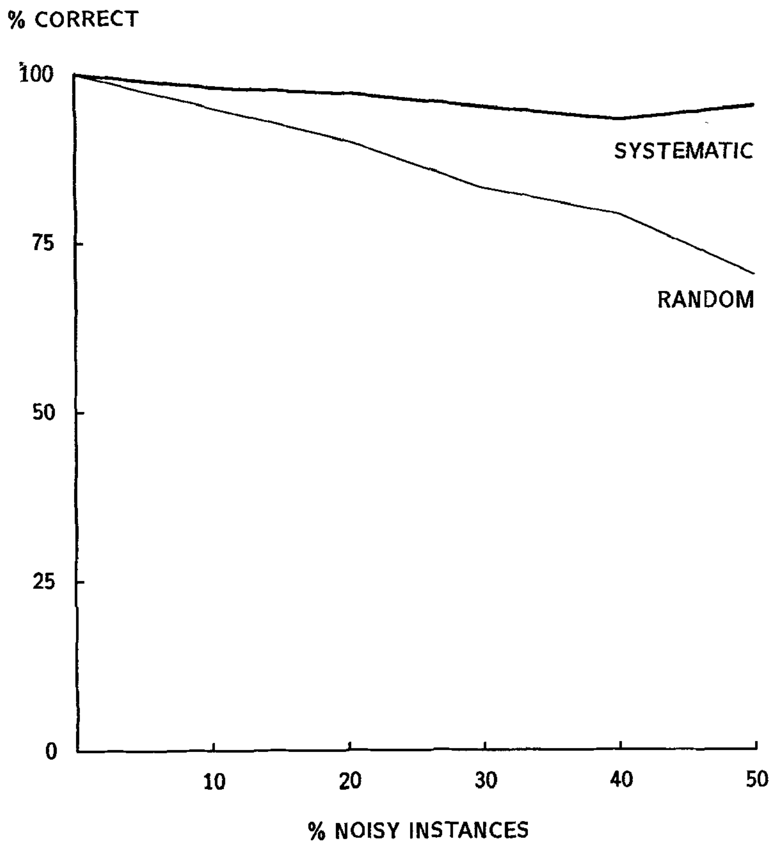

Figure 5. STAGGER's differential noise tolerance.

positive instances are randomly designated as examples or nonexamples, STAGGER is still able to utilize the information present in the negative instances to form an accurate description of the concept. In the random variation case depicted by the lower, light line, both positive and negative examples were subject to random identification. STAGGER's performance in this case is still reasonable, but is severely degraded when compared to either the noise-free case or the systematic noise case.

3.3 Tracking changes in a concept definition

An essential feature of a learning mechanism is the ability to respond to changes in the environment. For instance, a fox learns to look for a changed coat color in his prey as the seasons change. One factor to consider when addressing the issues of

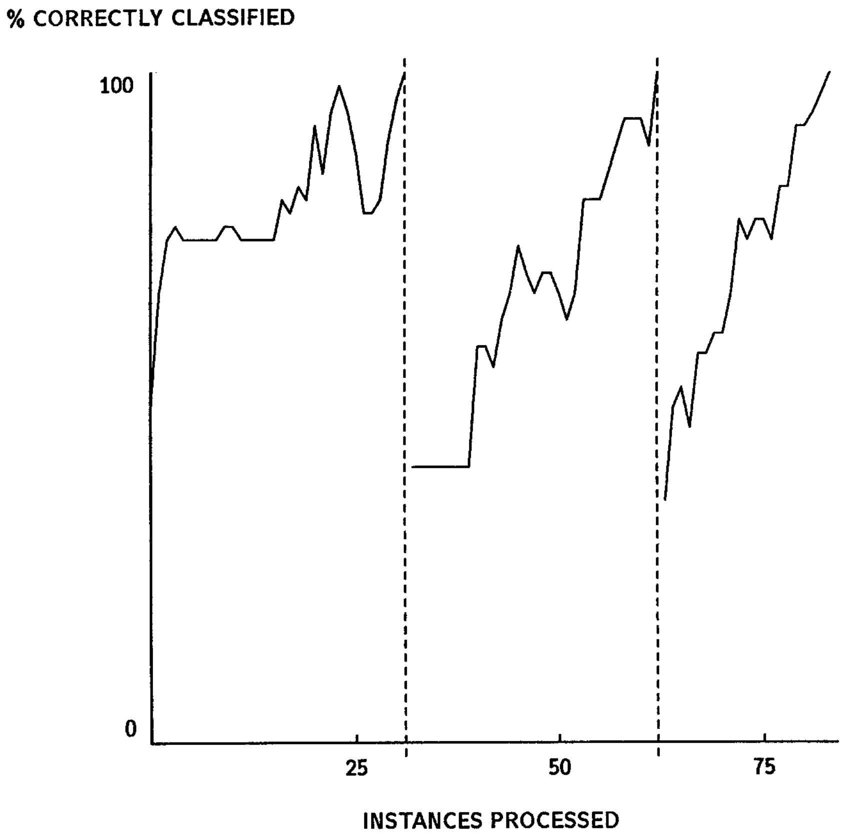

Figure 6. Tracking concept drift.

changing concepts is distinguishing between randomness and genuine change. Secondly, the amount of previous learning about a given concept should adversely affect relearning of a new definition for that concept. The adage 'It's hard to teach an old dog new tricks' summarizes a main finding of a class of studies in animal learning (e.g., Siegel & Domjan, 1971). In short, these studies indicate that the ease of revising a learned concept definition is inversely proportional to the amount of previous training.

STAGGER addresses the noise versus change issue through the use of its evaluation function. When the impeachment heuristic (Table 5) indicates that a characterization is currently below its threshold, backtracking is triggered. A conjunction is typically impeached when it gathers positive infirming evidence; this indicates that it is too specific. A disjunction is impeached when it collects negative infirming evidence and is indicated as being too general. Figure 6 depicts the performance of STAGGER on three successive definitions for the same concept:

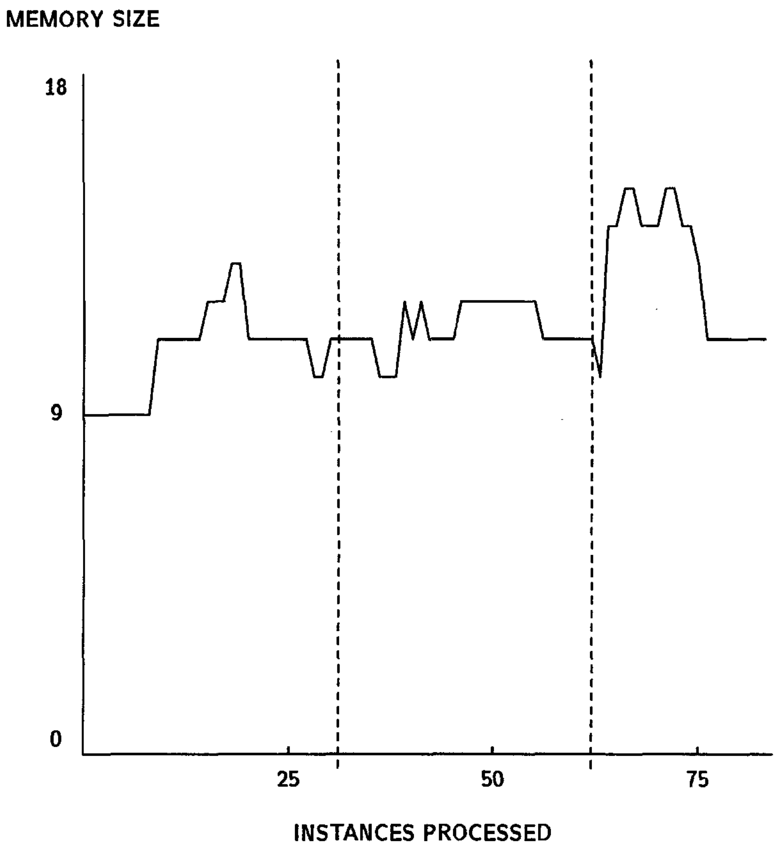

Figure 7. Search .frontier size.

(1) **size = small** *and* **color = red**, (2) **color = green** *or* **shape = circular**, *and* (3) **size = (medium** or **large)**.

The dashed vertical lines indicate when the definition of the concept was changed. Notice how performance falls immediately following the change because the previously acquired definition was not sufficient to characterize new changed instances. Initial sensitivity to specific instances is the main cause for the bumpiness of the curves. A secondary cause arises when effective clauses are pruned out as the search frontier advances (e.g., when **small** and **red** are pruned out for **small** *and* **red**). In each of the three cases, STAGGER formed the explicit, symbolic representation of the concept's definition and evaluated it as the best among those on the search frontier. The size of the search frontier (the number of characterizations in the distributed concept description) at each step is displayed in Figure 7. As effective characterizations are established, their sponsoring components are

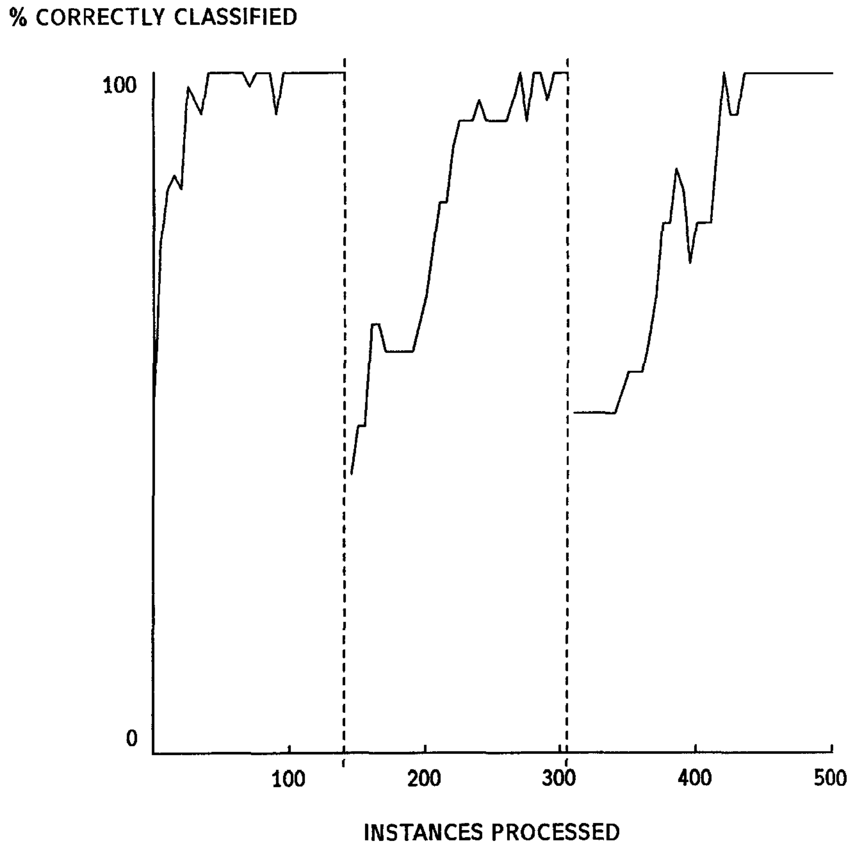

Figure 8. Tracking concept drift given extensive training.

pruned and the size of the search frontier is reduced. Notice how backtracking is triggered after each concept change, resulting in a sharp climb in the size of the frontier as previously effective characterizations are impeached.

Because STAGGER retains counts of situations types, it is in effect keeping an abbreviated history of the effectiveness of each characterization. This allows the program to model the effects of previous learning at a gross level. Contrast Figure 8 which depicts the program performance with more than four times the amount of training for each concept than depicted in Figure 6. Notice that the the horizontal scale in Figure 8 is compressed by one-fifth and recovery learning is considerably quicker in the minimal training case.

Requiring a learner to follow changes in the environment over time raises an old issue: the sensitivity of various learning methods to the order of instances. In general, immunity to the order of presentation is a desirable quality in a learning method. However, consider two orderings of instances for the single concept of the fox's prey. If all of the white rabbits precede all of the brown ones, then the appropriate behavior is to describe the prey as white and then change that description to brown. The learner could form a disjunction, but that would be less suitable and might be inaccurate as well. On the other hand, if white rabbits and brown ones are intermixed (domestic and wild rabbits, for example), then it would be appropriate to form the disjunction white *or* brown. Learning methods will be more widely applicable if they form a disjunctive characterization given a mixed presentation ordering, and if in a sequenced presentation they form and then modify a characterization.

3.4 An example from chess

Some domains are amenable to knowledge-intensive learning techniques since they are well-understood environments. Chess is a classic example because the function of each of the pieces may be presented from the outset, and search techniques can be applied to discover the outcome of any given situation. However, even the most agile player benefits from the ability to quickly recognize a board as an instance of 'situations lost in two moves.' In general, descriptions of piece positions are also additive in chess; the ability to recognize a particular type of board control may aid in quickly recognizing more abstract situations. Given piece positions and an indication whether these are an example of a particular situation, STAGGER forms a characterization which may be used to identify other similar boards. Furthermore, by adding an attribute with a true or false value for each learned concept, STAGGER uses learning about previous classes of board positions to speed learning about subsequent situations.

As an illustrative problem, consider the task of determining whether a particular board is safe for black's knight (Quinlan, 1979). The specific problem pits a black king and black knight against a white king and white rook. The initial configuration is neither a checkmate nor a stalemate position. Furthermore, the black knight is *pinned* in each instance, meaning that it cannot be moved without placing the black king in check (an illegal move). An instance is identified as 'safe' if black can move the king to defend its knight, leaving white unable to capture the knight without forfeiting its rook. Figure 9 depicts a sample unsafe board configuration meeting these constraints.

Boards preserving these constraints are randomly generated and then described in terms of the following three simple types of attributes:

- Distance in king moves — allowing only legal king moves, how many would it take to move from the square of one piece to the square of another? This attribute has the values, 1, 2, or > 2. There are six attributes of this type resulting from a pairwise combination of each of four pieces.
- Board relationship — whether a pair of pieces are on the same rank or file, the same diagonal, or otherwise related. There are again six attributes of this

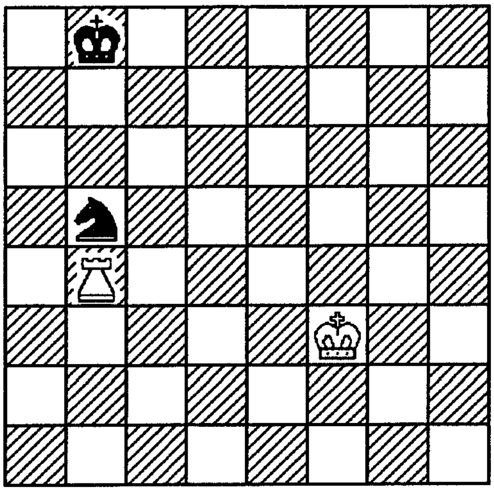

Figure 9. Example of an unsafe, pinned black knight.

type.

- Square type — whether a piece is on a corner square, on an edge, or in the open section of the board. There are four attributes of this type.

There are a total of 16 attributes, each with three values. Although there are $6^3 \times 4^3 \times 6^3 = 2{,}985{,}984$ objects possible, an exhaustive enumeration of the actual 95,480 distinct boards corresponding to a knight pin indicates that there are only 3,259 actual objects in terms of these attributes. This compression is due to the natural constraints which arise in chess; for instance, the black knight cannot be on a corner square since it must be pinned between the black king and the white rook.

Quinlan (1979) included an additional six true or false attributes when he presented this task to his ID3 program. Of these, five are always true or always false for this endgame task. They were useful for other endgame tasks Quinlan investigated. The remaining attribute is more abstract than those listed above. If there is a legal move which leaves the black king next to the knight then the attribute **black king can move adjacent to knight** is true.

Randomly generated boards (69% of the instances were positive) were described in terms of the 17 attributes listed above and then presented to STAGGER sequentially.[5] The most salient characterization that STAGGER formed for the concept **safe** is shown below.[6]

[5] Inclusion of the additional constant attributes used by Quinlan did not affect the quality of any of the results presented here.

[6] The last condition is actually the disjunction **king moves from white king to knight** = (**two** *or* >**two**). We interpret it as a negation since it resulted from an application of the inversion operator described in Section 2.4.

black king can move adjacent to knight is true *and* king moves from white king to knight $\neq 1$

This characterization is relatively simple, partially because the class of endgames is severely restricted and partially due to the inclusion of the high-level attribute **black king can move adjacent to knight**. Identifying and defining this abstract attribute was itself a concept formation task. One step in pushing this process back onto the learning method itself would be to only *identify* high-level features without giving their *definitions*. Using this approach, instead of including the attribute **black king can move adjacent to knight** and its value in the description of each instance, the learning program would be given the task of attaining that concept. We did this with STAGGER by first training it to recognize situations in which the black king could move next to the knight using only the 16 attributes enumerated above.[7] The constructed characterization shown below correctly classifies more than 99% of the 3,259 boards.

king moves from black king to knight $\not> 2$ *and* (knight square type is open *or* king moves from white king to knight is > 2)

Having acquired a characterization for this concept, STAGGER is then able to use this characterization as a true or false attribute in subsequent concept attainment tasks. Given the task of determining whether a board is safe for the black knight, STAGGER scans all of the concepts it has previously acquired and adds an attribute for each of them to the instance description language. These new attributes are assigned a true or false value by projecting the appropriate concept's characterizations over all 'older' attribute-values. In this case, thought only the 16 attributes were used to describe the instances, the characterization acquired for a safe board is the same as in the more completely described case.

This approach to utilizing previous concept learning is not unique to STAGGER. Iba (1984) and Sammut and Banerji (1986) describe concept attainment systems which save each learned concept description and use them to rewrite the description of an instance. After training on one task, their systems are able to recognize subsequent instances as examples of previous concepts. This eases learning about subsequent, similar tasks.

In terms of the framework of component processes presented in Section 1.1, STAGGER is *aggregating* instance descriptions by adding higher-level attributes. Aggregation is the process of determining the appropriate *part-of* relationships, or deciding which elements of instance descriptions are objects (Langley & Carbonell, in press). This task may also be viewed as a search for the appropriate granularity of features used to describe objects, or the process of adjusting the level of the

[7] Positive instances constituted 75% of the randomly generated boards.

representation language. In concept attainment, this amounts to forming more complicated, higher-level features which can effectively describe instances or *aggregating* smaller, primitive features into more effectual units. STAGGER attempts to aggregate simple descriptions into more useful ones by incorporating an attribute describing each learned concept into the description of new instances. Thus learning about one concept becomes a stepping stone for the search through the space of characterizations for another. This is a source of positive transfer during concept learning.

Dietterich and Michalski (1981) describe the similar process of *constructive induction*, or producing new descriptors which were not present in the input events. As an example, they point to the use of fixed augmentation rules in Induce 1.2 (Dietterich & Michalski, 1981). A sample rule adds an attribute by counting the number of objects in an instance which possess a given feature. STAGGER is performing constructive induction, for it is producing new descriptors which were not present in the input instance. But it is also doing more, since previous learning is utilized in the rewriting process.

Return for a moment to the previous discussion regarding the sensitivity of various learning methods to the presentation order of instances (Section 3.3). If training instances of the concept **black king can move adjacent to knight** are mixed with instances of **safe**, then there would be no benefit or leverage for the attainment of either concept. However, if **black king can move adjacent to knight** is presented first, STAGGER can use that concept to speed learning about the **safe** concept. This is a second occasion in which sensitivity to the order of the instances is a desirable quality in a learning algorithm. Learning methods which utilize previous learning to enhance subsequent concept formation will be more widely applicable. As Sammut and Banerji (1986) note, it is precisely this ability in students that encourages teachers to structure class material according to ascending difficulty.

4. Related work

The nature of the learning methods employed in STAGGER and Anderson's ACT* model (Anderson, 1983) are quite similar; they both utilize a weighted, distributed concept description, and they both perform a bidirectional search from simple concepts to more general and more specific ones. In addition to being a model of learning, ACT* is also a model of cognitive performance. It is based on a bipartite division of memory: a *declarative* memory stores factual information while a *procedural* memory stores knowledge about how to do things. The learning methods applied to the procedural memory are the most similar to those employed in STAGGER. A concept, or learned procedure, is expressed in terms of production rules which specify an action and a situation under which that action is appropriate. ACT* iterates by first determining which rules are appropriate and then selecting one to fire. Though the concept description is distributed among a number of separate productions, in a given cycle, only one characterization is projected onto the current instance.

The productions are subject to two types of modification by the learning component of ACT*: compilation and tuning. The compilation processes compose multiple productions together into longer action sequences and proceduralize productions by replacing variables with specific values.

The tuning processes generalize, specialize, and strengthen existing productions. Generalization (or specialization) introduces a new more general (or more specific) copy of a production into procedural memory. These new productions vie with others on the basis of their strength and specificity. The strength of a production is a number roughly reflecting its accuracy and effectiveness. When a production is chosen for projection, its strength is incremented. If the production turns out to be incorrect, it is weakened by one-fourth of its value. Because they compete with each other, a production is effectively weakened when it is not applied because other productions are strengthened; it is also strengthened when another production applies and fails. The net change of a production's strength expressed as a probabilistic value is:

$$\Delta S = p(F) - \frac{p(F \wedge \neg O)}{4} S - p(\neg F) + \frac{p(\neg F \wedge \neg O)}{4} S$$

This calculation is effective at dampening the potentially spurious effects of generalization and specialization. Langley (in press) has further demonstrated the ability of a similar formula to increase tolerance to noise in a learning production system. However, adjusting the strength of a production in this manner does not accommodate the findings of contingency. To clarify this point, consider this strength change expressed in terms of the matching situations in Table 1:

$$\Delta S = C_P + I_N - \frac{I_N}{4} S - I_P - C_N + \frac{C_N}{4} S$$

In the case where the novel stimulus and the unpleasant stimulus always occur together, the production would become strong; it would not be weakened because there is no positive or negative infirming evidence. However, the other three cases each contain infirming evidence. Partial reinforcement (novel stimulus occurs alone) contains negative infirming evidence, yet this would incorrectly result in a relatively weak production. Partial warning (unpleasant stimulus occurs alone) contains positive infirming evidence, and the production would barely be weakened. This predicts learning as is appropriate in this case. Random reinforcement (both stimuli occur alone) would resemble the partial reinforcement case. The presence of negative infirming evidence would greatly weaken the production, and the positive infirming evidence would have little effect. This function's failure to tolerate differentially either type of infirming evidence, but not both, disallows accounting for both partial reinforcement and partial warning effects.

Langley's discrimination learning method (in press) is similar to Anderson's ACT* in many respects. The learned concepts are expressed as a set of production rules, one of which is projected at any given time. Refinement of rules occurs only via specialization and these rules are strengthened using an evaluation measure similar to Anderson's. In addition to demonstrating this method's ability to tolerate random noise, Langley has also shown that it can track simple changes in a concept definition over time. As the definition of a concept drifts, recently learned productions are weakened while the specialization process proposes new ones. The new productions are strengthened and eventually overwhelm any previous learning. Because this program is based on a strengthening evaluation function, however, it considers any infirming evidence as indicative of change. It does appear that this method should correctly model the effects of extensive training to a defunct concept definition; extensive pretraining would give proportionately large strengths to effective characterizations. These strong productions would have an inhibitory effect on subsequent relearning of new characterizations for the same concept.

Quinlan (1983, 1985) describes an algorithm named ID3 which constructs its characterization in a manner similar to STAGGER; the characterization is initially simple, and it is refined until sufficiently effective. For ID3, a characterization is represented as a decision tree. Its root is an attribute of the instances (e.g., size), and arcs descending to subdecision trees are labeled with a value for this attribute (e.g., small). The leaves of the tree are marked either as examples or nonexamples of the concept. To project the decision tree over a particular instance, the root attribute is tested and the appropriate arc is followed until a leaf is reached. Some hint of prototypicality may be obtained by considering the depth at which a leaf was reached. Since the most important tests are made first, a prototypical positive instance matches at a shallow point in the tree.

In the process of constructing the decision tree, ID3 chooses an attribute for the root test which maximizes the amount of information gained by testing it. The information theory measure used to evaluate each potential root also mirrors performance in the four contingency situations. The thread common to the evaluation measures used in STAGGER and ID3 is the notion of statistical independence. The latter states that two events are statistically independent when the probability of their joint occurrence is equal to the product of their individual probabilities, or $p(E_1 \wedge E2) = p(E_1)p(E2)$ (Fine, 1973). In short, LS and LN measure statistical independence, since $LS = LN = 1$ if and only if the two events are statistically independent. Similarly, the information theoretic measure indicates that a feature is irrelevant to the outcome precisely when the two events are statistically independent (Schlimmer, 1986). There are a number of measures currently in use which also measure statistical dependence in one way or another, including *category utility* (Gluck & Corter, 1985) and our previous work (Granger & Schlimmer, 1985a, 1985b). Seemingly appropriate measures, however, may not make the proper measurement. For example, neither $p(O|F)$ nor $p(F|O)$

correspond to statistical independence as can be shown algebraically (Schlimmer, 1986).

Quinlan (1983) has extensively demonstrated the ability of ID3 to tolerate random noise in both the class information of instances and the values of individual features. Furthermore, ID3 is also able to construct sophisticated characterizations. For example, in one chess endgame task, ID3 constructs a decision tree of over 393 tests (Quinlan, 1979). As in STAGGER, the major source of bias in ID3 is parsimony, for ID3 attempts to build the simplest decision trees that will accurately characterize the instances. This is the motivation behind the use of the information theoretic measure.

However, refinement in ID3 is nonincremental. Quinlan assumes that all of the instances are available at the outset for examination by the program. As a result, ID3 examines a large number of the instances at one time and does not have mechanisms for modifying an existing tree to incorporate new instances. ID3 could be modified to track concept changes, but the resulting algorithm would be relatively inefficient in terms of storage space and processing time. One way to do this would be to store a queue of instances. As each new instance was added to the queue, all of them could be tested. If too few of them were correctly classified by the current decision tree, a new decision tree could be constructed from scratch.

The incremental nature of a learning algorithm does not guarantee that it will be able to deal with concept drift over time. Mitchell (1982), for example, reports on the version space learning method in which an appropriate description of observed instances is formed via a bidirectional search through a space of possibilities. Whereas STAGGER searches the space of possible characterizations from the middle toward both sides, the version space method searches from most specific and most general toward the middle. The concept description is distributed across a set of maximally general characterizations and a complementary set of maximally specific characterizations. This entire concept description is used in projection to determine if a new instance is likely to be positive or negative. Additionally, this method has the desirable property that the degree to which a static concept is learned may be measured by noting the distance between the two characterization sets.

Though relational information is utilized, the version space method assumes the strong bias that a conjunctive characterization can accurately capture the concept to be learned. In later work (Mitchell, Utgoff, & Banerji, 1983), a modification was proposed which would form disjunction descriptions or tolerate limited noise in instances (but not both interestingly). Though this method is incremental and can refine its concept description as new observations are made, it may not be able to track concept changes. Specifically, it is unclear whether the algorithm could be modified to allow learned characterizations to change and recross the search boundaries if the definition of a concept drifted over time.

The techniques used in STAGGER also have a fair amount in common with connectionist approaches. Though these approaches differ from STAGGER because they avoid using symbolic representations, their representation of the concept description is distributed, is modified incrementally, and is tolerant of noisy instances. For instance, Hampson and Kibler (1983) present a learning method based on a multilayer perceptron network. The learned concept is represented as a set of weighted connections between neuron-like units. They show that this network can acquire any Boolean function of its inputs. This concept representation is distributed, and each unit participates in projection.

Hampson and Kibler further demonstrate that the evaluation process which adjusts the strengths between nodes yields the classes of learning exhibited in contingency experiments. Moreover, they show how their method can track concept drift. Flexibility in their approach naturally arises out of the concept representation. When the definition of a concept changes, the network begins to falter and appropriate changes in the connection weights are made. If the weight-altering threshold is set so that the network freezes its weights immediately following an accurate classification of the instances, extensive pretraining has no effect on subsequent relearning. However, if the weights continue to be modified well beyond initially satisfactory performance in order to increase the confidence of the projection process, the effects of overtraining will be retained by the network and significantly more training on subsequent definitions will be required to overcome the imprinted weights.

Holland's genetic algorithm (Holland, 1975) is an incremental method which utilizes a distributed concept description and is able to tolerate noise as well as track changes in concepts over time. In his framework, characterizations are represented as bit patterns which are subject to natural selection style laws of propagation. Initially a *population* of these bit patterns are randomly generated. All of these are then matched against a new instance and a cumulative projection is made. An external function provides an evaluation of the *fitness* of each of these characterizations, and weak characterizations are discarded. To fill in the vacancies, copies of the most effective characterizations are made by mixing their patterns together in a manner similar to genetic mating. The processes of projection, evaluation, and refinement then iterate with this new set of bit patterns.

The genetic algorithm is relatively tolerant of noisy instances. However, depending on the specific evaluation function used, it may or may not be making a contingency-like distinction between random and systematic noise. Secondly, this learning method is able to track changing concepts. This is undoubtedly due to the strong influence of the laws of genetic propagation on the design of the method's refinement operators. Furthermore, this method may also exhibit the effects of over-training. Though it does not appear to keep any record of pattern effectiveness, successful patterns may come to dominate the breeding space. These patterns may then exhibit extreme reluctance to be bred out, given a large amount of previous training. Mauldin (1984) reports on a similar phenomena in which the genetic algorithm converges prematurely on a suboptimal solution due to a nearly complete dominance of the breeding space.

5. Conclusions

The framework reviewed in this paper presents seven processes for the task of learning from experience: clustering, initialization, projection, evaluation, refinement, aggregation, and storage. STAGGER ignores the clustering task since it assumes that all instances are prelabeled with appropriate class information; all work in learning from examples makes the same assumption. Initialization in STAGGER is based on a distributed concept description composed of a set of weighted characterizations. Each characterization is a symbolic expression of conjunctive, disjunctive, and negated attribute-values. Projection of learned concept descriptions onto new instances utilizes the cumulative effect of the weighted elements and obeys the principle of prototypicality. Furthermore, projection matches instances against previously learned concepts and, in doing so, affords a savings in subsequent learning.

The learning processes in STAGGER evaluate characterizations and search for more effective characterizations. The process of evaluating characterizations mirrors results in psychology and statistics, providing a clear differentiation between random and systematic variations. Another possible evaluation function, strengthening, was shown to be inadequate for this purpose. Refinement introduces new characterizations and is a heuristically guided beam-search through the space of possibilities. Search operators propose more specific, more general, and inverted versions of existing characterizations. Evaluation driven backtracking allows tracking a change in a concept's definition over time, differentiating properly between random variation and genuine change, as well as conforming to the effects of over-training. Simple aggregation in STAGGER augments the descriptions of instances in terms of previous learning. A nodes-and-links memory model implementation of STAGGER, which addresses the issues of memory storage and retrieval, is not described in this paper but is discussed by Granger and Schlimmer (1985b).

As concept change over time within a domain, it is valuable to have a learning method for tracking these drifting concepts. STAGGER can be viewed as providing fundamental, robust mechanisms for dealing with noise and drift in learning. These mechanisms are by no means intended as a complete, closed solution. Rather, we hope that this case study has provided insights that will encourage other researchers to adapt pivotal portions of this method as they address the difficult problems currently challenging the field of machine learning.

Acknowledgments

Thanks to the machine learning group at UCI, especially to Dennis Kibler and Pat Langley. Countless discussions with these fellow researchers have enlightened us on many issues and prodded many ideas into code. Michal Young deserves our gratitude for his early formulation of some of the algorithms used here. We also thank Ross Quinlan for his suggestion of a natural refinement to the projection process. Special thanks to Doug Fisher who read a number of the drafts of this paper and willingly helped us hone a number of these ideas. Lastly, we sincerely appreciate the comments provided by the reviewers.

This research was supported in part by the Office of Naval Research under grants N00014-84-K-0391 and N00014-85-K-0854, the National Science Foundation under grants IST-81-20685 and IST-85-12419, the Army Research Institute under grant MDA903-85-C-0324, and by the Naval Ocean Systems Center under contract N66001-83-C-0255.

Appendix A: Derivation of *LS* and *LN*

Section 2.3 states that the *LS* and *LN* measures can be computed via a simple formula composed of the situation types in Table 1. This appendix shows the derivation by first substituting in joint probabilities for each of the conditional probabilities and then use the counts of situation types to estimate the joint probabilities. Here are the conditional probability forms of *LS* and *LN* again:

$$LS = \frac{p(F|O)}{p(F|\neg O)} \qquad LN = \frac{p(\neg F|O)}{p(\neg F|\neg O)}$$

The definition of conditional probability states that $p(A|B) = p(A \wedge B)/p(B)$. The above equation may therefore be rewritten as:

$$LS = \frac{p(F \wedge O)p(\neg O)}{p(F \wedge \neg O)p(O)} \qquad LN = \frac{p(\neg F \wedge O)p(\neg O)}{p(\neg F \wedge \neg O)p(O)}$$

Noting that the outcome is either a positive or negative instance and the feature is either a matched or an unmatched characterization, the above probabilities may be approximated with counts of the situation types in Table 1 as follows:

$$\begin{aligned}
p(F \wedge O) &\approx \mathrm{C_P}/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})\\
p(F \wedge \neg O) &\approx \mathrm{I_N}/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})\\
p(\neg F \wedge O) &\approx \mathrm{I_P}/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})\\
p(\neg F \wedge \neg O) &\approx \mathrm{C_N}/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})
\end{aligned}$$

Marginal probabilities of the form $p(A)$ may be similarly approximated by noting that $p(A) = p(A \wedge B) + p(A \wedge \neg B)$.

$$\begin{aligned}
p(F) &\approx (\mathrm{C_P} + \mathrm{I_N})/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})\\
p(\neg F) &\approx (\mathrm{I_P} + \mathrm{C_N})/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})\\
p(O) &\approx (\mathrm{C_P} + \mathrm{I_P})/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})\\
p(\neg O) &\approx (\mathrm{I_N} + \mathrm{C_N})/(\mathrm{C_P} + \mathrm{I_P} + \mathrm{I_N} + \mathrm{C_N})
\end{aligned}$$

Substituting these approximations yields:

$$LS = \frac{\dfrac{C_P}{C_P+I_P+I_N+C_N} \times \dfrac{I_N+C_N}{C_P+I_P+I_N+C_N}}{\dfrac{I_N}{C_P+I_P+I_N+C_N} \times \dfrac{C_P+I_P}{C_P+I_P+I_N+C_N}}$$

$$LN = \frac{\dfrac{I_P}{C_P+I_P+I_N+C_N} \times \dfrac{I_N+C_N}{C_P+I_P+I_N+C_N}}{\dfrac{C_N}{C_P+I_P+I_N+C_N} \times \dfrac{C_P+I_P}{C_P+I_P+I_N+C_N}}$$

Simplifying by canceling $C_P+I_P+I_N+C_N$ results in equation 4 presented in Section 2.3.

$$LS = \frac{C_P(I_N + C_N)}{I_N(C_P + I_P)} \qquad LN = \frac{I_P(I_N + C_N)}{C_N(C_P + I_P)}$$

References

Anderson, J.R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Bruner, J.S., Goodnow, J.J., & Austin, G.A. (1956). *A study of thinking*. New York: John Wiley & Sons.

Dietterich, T.G., & Michalski, R.S. (1981). Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods. *Artificial Intelligence, 16*, 257–294.

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.

Easterlin, J.D. (1985). *A survery of concept formation in machine learning*. Unpublished manuscript, University of California, Department of Information and Computer Science, Irvine, CA.

Fine, T.L. (1973). *Theories of probability*. New York: Academic Press.

Fitzgerald, R.D. (1963). Effects of partial reinforcement with acid on the classically conditioned salivary response in dogs. *Journal of Comparative and Physiological Psychology, 56*, 1056–1060.

Gamzu, E., & Williams, D.R. (1971). Classical conditioning of a complex skeletal response. *Science, 171*, 923-925.

Gluck, M.A., & Corter, J.E. (1985). Information, uncertainty, and the utility of categories, *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum Associates.

Granger, R.H., Jr., & Schlimmer, J.C. (1985a). Combining numeric and symbolic learning techniques, *Proceedings of the Third International Machine Learning Workshop* (pp. 44–49). Skytop, PA.

Granger, R.H., Jr., & Schlimmer, J.C. (1985b). Learning salience among features through contingency in the CEL framework. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 65–79). Irvine, CA: Lawrence Erlbaum Associates.

Granger, R.H., Jr., & Schlimmer, J.C. (in press). The computation of contingency in classical conditioning. In G.H. Bower (Ed.), *The psychology of learning and motivation: Advances in research and theory*. New York: Academic Press.

Hampson, S., & Kibler, D. (1983). A Boolean complete neural model of adaptive behavior. *Biological Cybernetics, 49*, 9–19.

Holland, J.H. (1975). Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press.

Iba, W.F. (1984). *Machine learning in terms of concepts in terms of previously learned concepts.* Unpublished manuscript, University of California, Board of Computer and Information Sciences, Santa Cruz, CA.

Langley, P. (in press). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development.* Cambridge, MA: MIT Press.

Langley, P., & Carbonell, J.G. (in press). Language acquisition and machine learning. In B. MacWhinney (Ed.), *Mechanisms of language acquisition.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Lowerre, B., & Reddy, R. (1980). The HARPY speech understanding system. In W. Lea (Ed.), *Trends in speech recognition.* Englewood Cliffs, NJ: Prentice-Hall.

Mauldin, M.L. (1984). Maintaining diversity in genetic search. *Proceedings of the National Conference on Artificial Intelligence* (pp. 247–250). Austin, TX: Morgan Kaufmann.

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

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

Mitchell, T.M., Utgoff, P.E., & Banerji, R. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga.

Quinlan, J.R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), *Expert systems in the micro electronic age.* Edinburgh: Edinburgh University Press.

Quinlan, J.R. (1983). Learning from noisy data. *Proceedings of the Second International Machine Learning Workshop* (pp. 58–64). Urbana–Champaign, IL.

Quinlan, J.R. (1985). *Induction of decision trees* (Technical Report 85.6). New South Wales, Australia: New South Wales Institute of Technology, School of Computing Sciences.

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

Sammut, C., & Banerji, R.B. (1986). Learning concepts by asking questions. In R. S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Schlimmer, J.C. (1986). *A comparison of feature salience measures* (Technical Report 86–13). Irvine, CA: University of California, Department of Information and Computer Science.

Siegel, S., & Domjan, M. (1971). Backward conditioning as an inhibitory procedure. *Learning and Motivation, 2,* 1–11.

Smith, E.E., & Medin, D.L., (1981). *Categories and concepts.* Cambridge, MA: Harvard University Press.

Wasserman, E.A., Chatlosh, D.L., & Neunaber, D.J. (1983). Perception of causal relations in humans: Factors affecting judgments of response-outcome contingencies under free-operant procedures. *Learning and Motivation, 14,* 406–432.