Information and Computer Science

Technical report

UNIVERSITY OF CALIFORNIA
IRVINE

Contingency and Salience Assignment: Incremental Learning in the CEL Framework

Richard H. Granger
Michal Young
Jeffrey C. Schlimmer

Irvine Computational Intelligence Project
Computer Science Dept.
Cognitive Sciences Program
and
Center for the Neurobiology of Learning and Memory
University of California
Irvine, California 92717

Telephone: (714) 856-6360
CSNET Address: Granger@UCI-750a

ABSTRACT

The problem of deciding which of many possible features in a training sequence are the salient or predictive ones is a well-known problem in machine learning. This problem of *salience assignment* is difficult when attempting to learn in an unpredictable and reactive environment. Components of Experiential Learning (CEL) is a framework for the development of computational theories of learning in this type of environment [Granger 1983, Granger and McNulty 1984]. In this paper, we review the CEL processes and explain a specific computer model LURN (Learning by Unconscious ReasoNing) which illustrates the use of an incremental method for performing salience assignment. An additional constraint on salience assignment arises from experimental results in animal psychology which indicate that living learning engines (i.e., humans and animals) make a sharp distinction between simple pairing (or strengthening) of associated events versus *contingent* salience assignment [Rescorla 1966]. LURN calculates the contingent predictiveness of features in a noisy environment, in accordance with these experimental results.

Topic area: Cognitive Modelling
Keywords: Learning, Contingency, Salience Assignment
Paper length: 4700 words

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

1 Introduction

1.1 The Salience Assignment Problem

A rat in a laboratory cage hears a tone. It also hears the air conditioning system start, and sees a lab assistant taking notes. Shortly afterwards, it feels an unpleasant electric shock. What does the rat learn?

Since the late 1960s, psychological experiments have made it clear that what the rat learns from above episode depends on the relationship, over several trials, among the several plausible cues to the unpleasant event. In Machine Learning research in Artificial Intelligence, this corresponds to a *salience assignment* problem: which of the many possible cues are the predictive, or salient ones, i.e., the ones to be learned? Rats solve the salience assignment problem under constraints more severe than those faced by most AI systems. For instance, learning must be *incremental*, for the rat in a natural setting must make good use of the experiential data already gathered even while gathering more. Moreover, the environment may not provide perfect predictors; the animal must make predictions as best it can when cues indicate only a change in the probability of an event.

The CEL model of learning and memory [Granger 1983, Granger and McNulty 1984] is a framework for explaining a variety of learning phenomena. LURN (Learning by Unconscious ReasoNing) is a computer model embodying some specific hypotheses within the CEL framework. LURN accounts for some data from animal learning experiments. In particular, LURN shows how an animal can learn which features of the environment are predictive of future stimuli, as demonstrated by Rescorla's [1966, 1968] experiments with dogs and rats.

1.2 A Method for Salience Assignment

In this paper we present a method of determining which features of an event are predictive of other events and distinguishing useful cues from context and background noise. The method determines the relevance of individual predefined features, and also forms new feature descriptions by conjoining or negating existing features. The effectiveness of the method is due to taking into account, in addition to successful predictions and errors of commission, also errors of omission and events in which the absence of a cue correctly prevented prediction of a second event. This extension of the idea of strengthening and weakening corresponds to the distinction in psychology between learning based on number of pairings and learning based on contingency. Using this method, the LURN program exhibits contingency-based learning behavior, modeling the learning behavior of animals and humans in classical conditioning tasks. The program is able to function correctly even with a large number of erroneous training instances.

2 Background: What is learned

2.1 Association and Contingency

How does a rat in a classical conditioning experiment distinguish predictive cues from other, non-predictive features of the environment? We might initially guess that the animal forms an 'association' between a feature (call it F1) like a tone and a following feature (F2) like a shock. In the classical conditioning paradigm, the second stimulus (F2) elicits an uncontrollable reaction with no prior training. This 'association' between the F1 and F2 might then become 'strengthened' [Anderson 1983] each time these stimuli are paired together.

Experimental psychologists have studied this type of question in various learning

paradigms. Animal learning studies differ from most artificial intelligence work in cognitive modeling in that they attempt to elucidate the representations and mechanisms involved at a lower level. However, researchers in animal learning are careful to look for effects that will 'scale up' to human learning. There is extensive evidence in the psychological literature that mammals (if not birds and lower animals) are excellent models of many human learning and memory abilities, and that most key results on animals do prove to also be true of humans, once tested [see, e.g., Bower and Hilgard 1981]. Hence, models of learning in lower mammals will contribute to the study of learning in higher mammals (e.g., us), even though such models cannot teach us everything about human learning.

Animal learning experiments show that a more sophisticated salience assignment method is at work than simple strengthening on the basis of number of pairings, described above. What actually happens is that over a number of trials, a rat incrementally learns to attend to some features of the environment over others, differentially noting which of the features (tone, lights, other noises, other visual cues) turn out to be more useful as predictors of the shock's onset. Rescorla [1966, 1967, 1968] has shown that animals do not simply learn an association based on the number of times they are paired together. Rather, they learn a combination of information about how often the relevant F1 (tone) precedes the occurrence of F2 (shock) measured against how often F2 occurs by itself, without the presence of F1. For instance, if much less than half of all soundings of a tone are followed by a delivery of food, but all deliveries of food are preceded by tones, then the animal will associate food with the tone even though this association leads to a high likelihood of disappointment. Rescorla shows that animals learn all and only relationships in which the probability of the relevant feature cue F1 preceding F2 is greater than the probability of F2 occurring without that cue. Formally stated, this means that a positive association between two stimuli (F1 and F2) will occur if and only

if $p(F2|F1) > p(F2|\overline{F1})$.

This measurement of relative probabilities is referred to in the psychological literature as *contingency*; animals (and humans, in similar circumstances) exhibit *contingency-driven learning* in the sense that they somehow maintain incrementally updated knowledge of the relative predictiveness of features. Through experience the animal must pick out the relevant features from the background of uncorrelated features and use only the relevant features to predict future events.

2.2 Behavior vs. Behaviorism

The study of animal learning has sometimes been erroneously associated with 'behaviorism', which would have it that nothing is learned by an animal except the association between stimulus and response itself. This extreme view of psychology fails to provide a satisfactory account for a wide range of behavioral data, such as latent learning, in which animals clearly exhibit learning in spite of the absence of any simple goal-based appetitive or avoidance-based reinforcer. In today's animal and human psychology, the strict behaviorist viewpoint is all but dead. As Dickinson [1983] points out, a theory that equates learning with behavioral change will have a great deal of trouble explaining the many phenomena of "behaviorally silent" learning, including, for instance, sensory preconditioning. The CEL framework describes learning in terms of mental processes and representations, and so is completely incompatible with the behaviorist perspective. Behavioral changes are of interest to us as *evidence* of other changes. In this respect our work is in the mainstream of modern animal and human learning theory.

2.3 Related work

Much work in machine learning has involved separating relevant features of a state from irrelevant features. As early as 1959, Samuel's checkers program constructed a linear

evaluation function by adjusting the weight of features depending on performance of the function, and by experimentally adding features from a predetermined pool. Samuel's program achieved masters level play, but the approach of constructing a linear evaluation function did not prove to be powerful enough for adoption in other domains.

More recent work has concentrated on symbolic descriptions. Winston's ARCH program [Winston 1975] learned structural descriptions of objects from a set of positive and negative instances. Instances and concepts both are represented by semantic nets. Objects (nodes in the net) are removed when they are part of the difference between two positive instances. The most relevant features (objects and relations) are those which are parts of differences between positive instances and negative instances. In order to isolate these relevant features, ARCH depends on negative instances which differ from a positive instance in only a single feature. Also, ARCH is not able to function correctly when erroneous training instances are presented.

An alternative to starting with a single positive instance and generalizing, as in ARCH, is starting with an over-general description and specializing it. Specialization may be more useful than generalization for problem-solving systems because the performance component of the system is initially too rash, and generates negative instances which can be used to refine the concept. This is the approach taken by Langley [1983, 1984] in the SAGE.2 system, which learns appropriate conditions for applying operators in a number of problem-solving domains. An "instance," for SAGE.2, consists of the whole working memory when a production was fired. Candidate differences between positive instances (selection context) and negative instances (rejection context) are found by a path-finding process which follows chains of features appearing in more than one working memory element. New rules proposed by the path-finding process are initially weak, but are strengthened whenever they are re-invented or when they are fired and lead toward the goal state. Rules are weakened when the credit assignment algorithm

determines that they are not leading toward the goal state.

SAGE.2 succeeds in identifying features which are relevant for identifying the proper situations in which to fire an operator. It learns incrementally without help from a teacher. The performance component of SAGE.2 is initially quite rash, but the learning component is conservative: New rules are introduced at low strengths, and become effective only after being reinvented several times. This is in marked contrast to Winston's ARCH program, which specializes a concept immediately upon presentation of a single negative instance. The conservative learning strategy of SAGE.2 allows it deal with negative instances which differ from positive instances in many ways, and which therefore lead SAGE.2 to postulate many ways of discriminating between positive and negative instances. It may also give SAGE.2 some immunity to erroneous training instances.

Strengthening and weakening is also used in the ACT* family of programs developed by John Anderson [Anderson 1983]. The ACT* framework, like many others including SAGE.2, is based on production rules. ACT* creates new rules through processes of composition, proceduralization, generalization, and discrimination. Of these, generalization and discrimination address the problem of discovering which features are relevant for determining when an operator should be applied. ACT* creates a generalized rule by omitting a condition from the antecedent part of another rule. Discrimination adds a new clause either to the antecedent or to the consequent part of a rule. As in SAGE.2, newly introduced rules initially have very low weights. Weights are increased when they are reinvented or are activated through the spread of activation in memory. They are weakened by negative feedback.

The ACT* framework has been used to account for a wide variety of data from the psychology of human learning. The scheme for strengthening and weakening rules, however, does not appear to be consistent with the basic psychological data concerning contingency.

3 The CEL Framework and the LURN Model

The CEL architecture [Granger 1982, 1983; Granger and McNulty 1984] provides a set of mechanisms for establishing, storing, indexing and retrieving memory traces based on experience. Within this architecture, the LURN model implements a number of specific hypotheses about the nature of traces, their indexing, and the detailed operation of the encoding and retrieval processes in memory to account for experimental data in human and animal psychology.

3.1 Overview of CEL

The CEL framework consists of twelve processes, or operators, and a set of representations based on sequential traces of events. The operators are separable, by function, into five classes: reception, recording, retrieval, reconstruction, and refinement. There are two reception operators: DETECT and SELECT. DETECT performs sensory input and special preprocessing. For example, specialized processing performed by the visual system would be encapsulated within DETECT. SELECT acts as an active filter on the sensory information processed by DETECT, allowing the model to attend to specific elements of a sensory stream of information.

The recording operators are NOTICE, COLLECT, DETOUR, and INDEX. The NOTICE operator maintains a list of desirable and undesirable states. NOTICE continuously matches these states against the output from SELECT; NOTICE tags events with a hedonistic value, and then appends them onto a data structure called short term memory (STM). NOTICE also sets a global value reflecting the pleasure of the organism as a whole. If this global value falls to one extreme or another, then either of two other operators are triggered: COLLECT or DETOUR. These operators 'package' a number of event specifications stored in short term memory (STM) into a schema (memory trace).

A schema is represented in CEL as a sequence of events, and each event is represented as a set of (unstructured) features. COLLECT packages a desirable schema; DETOUR, an undesirable one. Both of these operators hand their newly created schema to the fourth operator: INDEX. INDEX stores each schema in long term memory (LTM) using appropriate indexing schemes so that they may be retrieved at a later time.

The retrieval operators REMIND and ACTIVATE perform retrieval and selection of stored schemata. REMIND matches the last event specification in STM against the indexing structure built by INDEX in long term memory (LTM). Schemata matching within a certain threshold are placed in intermediate term memory (ITM). ACTIVATE then examines each of the schemata in ITM and chooses one for reenactment on the basis of a number of metrics. This chosen schema is called the current predictive schema (CPS). The competition between these REMINDed schemata is similar to conflict resolution in a spreading activation framework [Anderson, 1983].

SYNTHESIZE and ENACT are the two reconstruction operators. The first, SYNTHESIZE, matches the CPS with to the events appearing in STM. If event descriptions in the CPS and STM match, then the ENACT operator attempts to perform any actions in the next event of the CPS. If SYNTHESIZE is not able to match an event description in the CPS with the most recent event in STM, it triggers the refinement operator BRANCH. Lastly, if SYNTHESIZE has been able to successfully match each event description in the CPS with sequential event specifications in STM, then it will trigger the refinement operator REINFORCE.

The fifth class of operators are the refinement operators REINFORCE and BRANCH. REINFORCE incrementally strengthens a schema. BRANCH packages event specifications into a schema for indexing, effectively reducing the strength of the current predictive schema (CPS) by adding a competing schema. BRANCH also changes organism state values associated with surprise or failed expectation. One of these state values

triggers SELECT to relax its filtering process. Another has the effect of increasing the number of event specifications that are packaged as a schema for indexing.

The CEL operators and their paths of interactions are sketched in figure 1.

4 Contingency in LURN

Contingency is defined by Rescorla [1966] as $p(F2|F1)/p(F2|\overline{F1})$. This is similar to the *logical sufficiency* value used in some expert systems employing Bayesian statistics, notably the Prospector system [Duda 1979]. Our model shows how such a value can be incrementally learned and used in a reactive environment.

To determine the predictive value of a feature (or group of features), LURN notes the frequency of four combinations: F1 occurs and then F2 occurs *(prediction)*, F1 occurs and then F2 does not occur (error of *commission*), F1 does not occur and then F2 does occur (error of *omission*), or F1 does not occur and neither does F2 (*non-prediction*). The first and last of these combinations strengthen the association between F1 and F2, while errors of commission and of omission weaken the association.

	F2 present	F2 absent
F1 present	++ Prediction	+− Error of Commission
F1 absent	−+ Error of Omission	−− Non-prediction

Table 1: Possible combinations of F1 and F2

The four categories described above are needed to compute logical necessity and logical sufficiency by application of Bayes rule. However, it would be psychologically implausible to note all the times that F1 does not occur and F2 also does not occur.

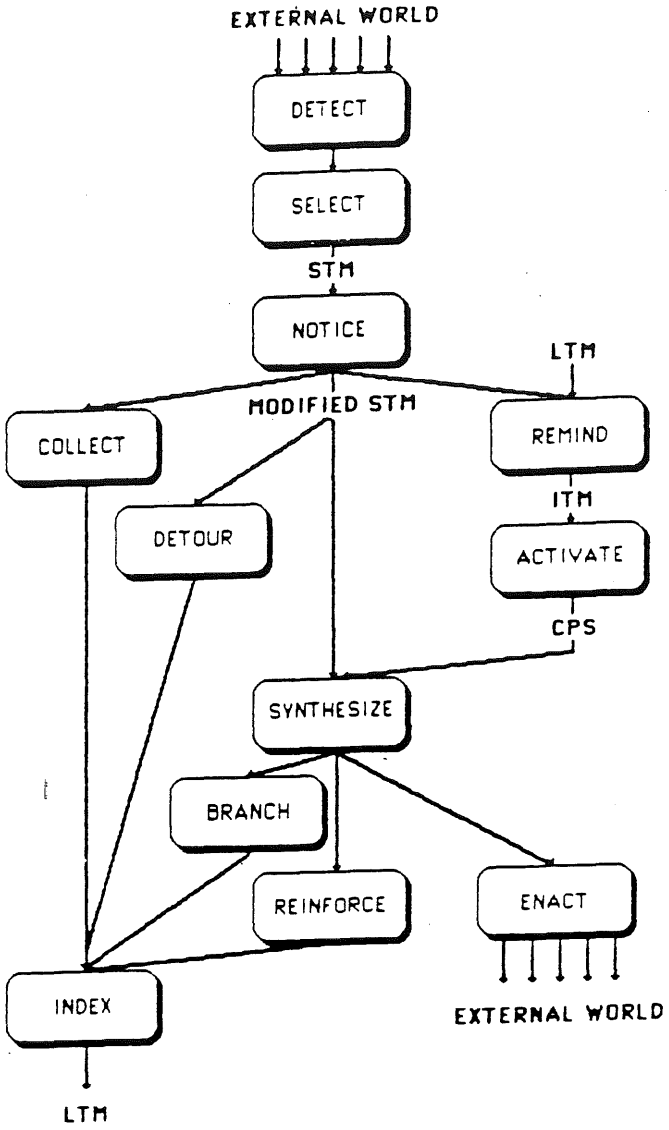

Figure 1: Outline of CEL operators

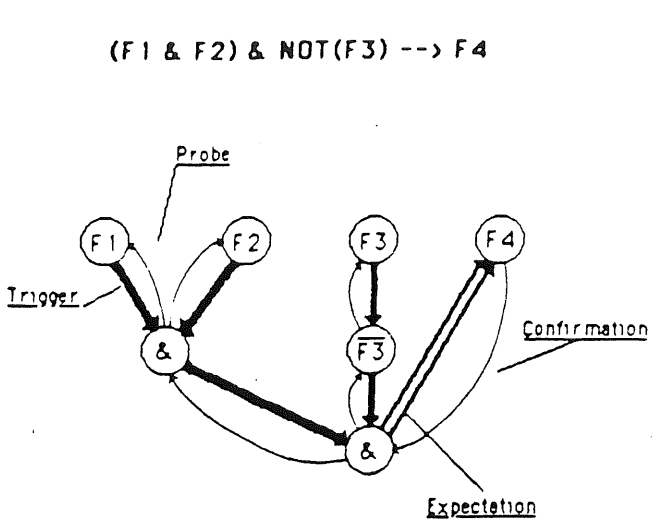

Figure 2: A hypothetical memory network.

Therefore the learning mechanism of CEL notes relations between features only when one or the other has become active. We envision this as a spreading activation process, in which a node may become active either by being present in STM or by being expected (triggered by another node). Thus the paired non-occurrence of F1 and F2 is noted only if some other combination of features causes F2 to be expected.

The restriction that links between nodes are modified only if at least one of the nodes is activated leads to a systematic undercounting of non-predictions. Thus the expectations simulated by the model are not the same as would be achieved by application of Bayes rule.

Memory in our model is organized as nodes and links, with inhibitory and excitatory links strengthened and weakened by the differences in activation between nodes. We hypothesize a single node type (which computes a boolean combination of its inputs) and four kinds of links: probe, trigger, expectation, and confirmation. Trigger and probe links are paired, as are expectation and confirmation links. The expectation–confirmation pairs allow successes, errors of commission, and errors of omission to be recognized. The trigger–probe pairs allow us to have nodes which compute the logical negation of their inputs, but which are not constantly excited in the absence of input

stimuli. If the network in figure 2 were activated through nodes F1 and F2, but not F3 or F4, activation would spread from the first AND node to the second, and a probe would propagate backward from the second AND to the NOT node, which would then satisfy the second AND node. This would then trigger an expectation of node F4, and an error of commission would be recognized.

Our current implementation uses a simplification of this scheme: For each feature (F2), we keep a table of counts with entries for each feature or combination hypothesized as a relevant cue (F1). Instead of adjusting the strengths of links between nodes, we compute weights from these counts. The simplified scheme is advantageous for experimental change while our theoretical view of memory develops.

4.1 Establishing a Memory Trace

In response to a pleasant or unpleasant stimulus (F2), CEL forms a memory trace, or schema, containing several events that lead up to it (see [Granger and McNulty 1984] for details of trace-establishment processes in CEL).

Each of the features in these preceding events is potentially a cue which can be used to predict F2. The task is to decide which of these potential cues are relevant or *predictive*, which are background or *context* cues, and which are uncorrelated.

4.2 Gathering Evidence

All counts in LURN's memory are initially 1. These counts will be updated only when the index node is triggered by matching cues in the environment (REMIND) and one of the schemas beneath it is chosen (ACTIVATE) to become the current predictive schema (CPS). SYNTHESIZE then follows the CPS and attempts to match it to events as they occur, eventually triggering REINFORCE or BRANCH, depending on whether the match between the current episode and the CPS is successful or unsuccessful.

REINFORCE is responsible for noting when a prediction has succeeded. Cues in the index node which match features in the environment during the current experience are called successes, and their success scores are incremented. Cues in the index node which did not match features in the environment are called omissions, and their omission scores are incremented.

BRANCH is responsible for relevance assignment when predictions fail. BRANCH scores a commission for each cue feature that matched the environment and a non-prediction for each cue feature that was absent from the environment. Features present in the environment but not present in the schema are added to the cue table with an initial score of 1 commission, no successes, no omissions, and no non-predictions.

4.3 A Detailed Example

Assume LURN is simulating a situation where tones, lights, noises, and shocks are occurring. LURN's job is to construct a memory record that allows it to learn which of the many features of the environment are the ones that predict the occurrence of shock, so that it can avoid it. In this section we will illustrate what LURN's memory would look like in three circumstances: (1) where shock is randomly paired with a number of environmental features (*random contingency*); (2) shock is reliably preceded by a conjunction of predictive features (e.g., tone and light) (*positive contingency*); and (3) shock only occurs in the absence of a particular feature cue, so that the cue (e.g., tone) becomes a 'safety signal' - i.e., the animal (or LURN) can predict that no shock will occur after the cue (*negative contingency*).

After training in the random condition, i.e., with tone, light, shock, etc., *independently* occurring at regular intervals, LURN will have a score function similar to table 2 (note that successes are indicated by '++', commissions by '+−', omissions by '−+', and non-predictions by '−−'. The figures in tables 2, 3, and 4 are taken from runs of

	++	+−	−+	−−	LS	LN
Cage	9	9	1	1	1.0	1.0
Tone	4	4	6	6	1.0	1.0
Light	4	4	6	6	1.0	1.0
Buzz	4	5	6	5	0.81	1.22
Whrr	4	4	6	6	1.0	1.0

Table 2: Random Contingency

our computer model.)

LURN in a positive continency condition, on the other hand, would have a score function like table 3. The conjunction of light and tone is proposed by the LURN program, as discussed in section 4.5. This chart illustrates important differences between contingency learning and more intuitive notions of strengthening based on number of pairings. Cage and tone receive the same number of pairings with shock, but tone is a much better predictor of shock. Moreover, tone was involved in a greater number of mistaken predictions (errors of commission) than was buzz, but tone is still recognized as the better predictor.

Finally, in a negative contingency situation, LURN's memory would have gathered statistics like table 4. In training situations with aversive stimuli, negatively contingent cues have been shown to increase pursuit behavior and decrease avoidance behavior (even when the avoidance or pursuit behavior is not related to the stimulus whose absence is predicted). Negatively contingent cues may also serve to suggest avoidance or escape reactions.

	++	+−	−+	−−	LS	LN
Cage	17	6	1	1	1.48	0.52
Tone	17	4	1	3	3.24	0.25
Light	17	4	1	3	3.24	0.25
Buzz	6	3	12	4	0.89	1.33
Whrr	14	6	4	1	0.88	1.50
And[Tone,Light]	15	2	1	2	2.65	0.18

Table 3: Positive Contingency

	++	+−	−+	−−	LS	LN
Cage	22	6	1	1	1.57	0.43
Tone	1	3	22	4	0.30	4.88
Light	12	3	11	4	1.09	0.75
Buzz	11	5	12	2	0.80	2.19
Whrr	11	4	12	3	0.92	1.33
Not[Tone]	22	4	1	2	2.54	0.23

Table 4: Negative Contingency

4.4 Characterizing Cues

With counts of omissions, successes, commissions, and non-predictions, we can say what we mean by *useful* cues (both positive and negative), *context,* and *uncorrelated* cues. A positive cue has a greater than average ratio of successes to errors of commission (high LS), and a negative cue has a smaller than average ratio of successes to commissions (LS less than 1). Both context cues and uncorrelated cues have ls values in the vicinity of 1, but context cues have a small number of errors of omission relative to errors of commission.

Positive cue	$ls >> 1$
Negative cue	$ls << 1$
Context	$ls \approx 1, omissions < commissions$
Uncorrelated	$ls \approx 1, omissions \geq commissions$

An intuitive way of stating the difference between uncorrelated and context cues is that one can be relatively sure that uncorrelated features are not necessary for predicting an event (F2), since F2 has repeatedly occurred in absence of the uncorrelated cue. One cannot be sure, though, about the importance of context. It is impossible to know whether or not it is a necessary precondition until F2 has been predicted a few times in the absence of the context feature. Then the context cue will either become a positive cue (if the prediction was successful) or an uncorrelated cue (if the prediction failed).

4.5 Combining features

It is not sufficient to note relations between individual features. It is also necessary to note useful combinations of features. We will use the term 'clause' to refer either to a memory node representing a single feature or a node representing a boolean combination of features. The LURN model uses current associations between clauses to suggest new

combinations. Clauses containing boolean AND or NOT are introduced to discriminate between positive and negative instances.[1] Generation of these clauses is guided by current relations between clauses and the features to be predicted. When a clause is satisfied in a negative instance but not in a positive instance, and that clause has an LS value less than 1, it is a candidate for negation. When a clause with a LN value below 1 is present in a negative instance it is a candidate for conjunction with a clause that is unsatisfied in the negative instance and also has a LN value less than 1. We can think of the value of clauses as guiding a plausible move generator for searching the space of discriminated conditions.

Propose	When (error of commission)
Not[A]	LS(A) << 1, A satisfied
And[A,B]	LN(A) << 1, A satisfied LN(B) << 1, B unsatisfied

5 Experience with the LURN System

5.1 Robustness

Real-world environments invariably entail some degree of noise, so a learning engine must be able to tolerate erroneous training instances. We have been pleasantly surprised by the performance of the LURN program in these circumstances. Figure 3 depicts the performance of LURN when trained with various rates of erroneous instances.

The line plotted with circles in figure 3 shows performance under conditions of uni-

[1] It is desirable to allow both discrimination (through conjunctions of clauses) and generalization (through disjunction). At this time, however, the LURN model proposes only conjunctions and negations. A weaker form of generalization is achieved by dropping clauses with low predictive value.

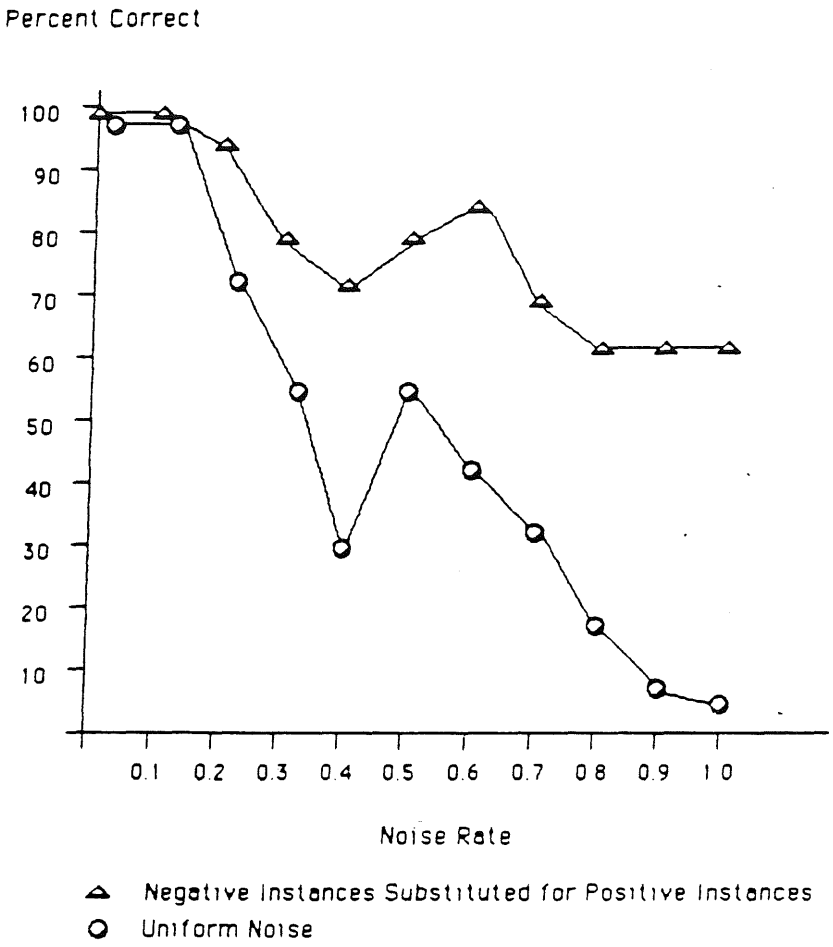

Figure 3: Performance of LURN as a function of noise.

form noise, that is, equal chance of substituting a positive instance for a negative instance and vice versa. As the error rate approaches 0.3, the LURN's performance falls toward a chance level (50%). As one would expect, error rates in excess of .5 cause LURN to acquire the opposite of the training concept and to perform at less than chance level.

LURN's tolerance of erroneous training instance is partly due to the smooth weighting function. In addition, though, we found that robustness depends critically on the introduction of combined features (section 4.5). With no noise in the data, LURN can achieve perfect performance for simple conjunctive classifications even when the combination proposer is disabled, since the extreme values of LS and LN are sufficient to express logical necessity and logical sufficiency. But when even a small number of erroneous instances are introduced, performance falls off precipitously unless combinations are proposed. The predictive value of a combination of features is higher than that of any of its component features, and the influence of erroneous instances on that value is correspondingly less.

The triangles in figure 3 plot the performance of LURN when negative instances are substituted for positive instances, but there are no spurious positive instances. This is similar to partial reinforcement in conditioning. LURN's performance remains well above chance in this case even for levels of noise in excess of 50% because the tone and light are in positive contingency relation with shock even when the absolute probability of shock following the cues is low. (LURN's level of performance is similar when the only noise is spurious positive instances.)

5.2 Annotated Run-Time Output

We have implemented LURN in Franz Lisp on a VAX 11/750 running under Unix. In the following transcript of a LURN run, the conjunction of tone and light is a positively contingent cue for the onset of shock. Annotations are separated from actual program

output by semicolons.

...

```
Detecting: cage, light, tone, whrr      ; these are external cues.
    doubtful of shock occurrence (odds = 0.3 < 1).
Detecting: shock                        ; LURN doesn't predict the shock, but
Updating Expectations                   ;  gets one.
    marking successes                   ; satisfied cues get a success.
    marking omissions                   ; unsatisfied cues get an omission.

Detecting: cage, light, tone, whrr, buzz
    strongly expecting shock (odds = 3.25 >> 1).
Detecting: shock                        ; LURN successfully predicts the shock.
Updating Expectations
    marking successes                   ; satisfied cues get a success.
    marking omissions                   ; unsatisfied cues get an omission.

Detecting: cage, light
    doubtful of shock occurrence (odds = 0.64 < 1).
Detecting: nothing                      ; LURN doesn't predict the shock and
                                        ;  doesn't get one.  this is a
                                        ;  successful prediction, but the counts
                                        ;  are not updated when a node isn't
                                        ;  activated.

Pattern                                 ; LTM looks like this now:
    ++  +-  -+  --     ls   ||   ln   |
    ------------------------------------
not[whrr]                   ||        |
    2   1   3   1     0.89  ||  1.33  |; a mistakenly introduced clause.
    ------------------------------------
tone                        ||        |
    4   2   1   2     2.0   ||   0.5  |; tone has a low LN.
    ------------------------------------
light                       ||        |
    4   2   1   2     2.0   ||   0.5  |; light also has a low LN.
    ------------------------------------
buzz                        ||        |
    3   1   2   3     1.88  ||  0.42  |
    ------------------------------------
whrr                        ||        |
    3   3   2   1     0.75  ||   1.5  |
    ------------------------------------
cage                        ||        |
    4   3   1   1     1.14  ||  0.86  |
    ------------------------------------
```

```
Detecting: cage, light, whrr, buzz
    strongly expecting shock (odds = 2.14 >> 1).
Detecting: nothing                      ; LURN makes an error of commission.
Updating Expectations
    marking commissions                 ; satisfied features get a commission.
    marking non-predictions             ; unsatisfied features: a non-prediction.
Suggesting Possible New Feature
    and[light,tone]                     ; LURN suggests a new feature clause.

        ; LURN's contingency information suggests this new feature
        ;  through a contingency driven discrimination process.
        ;  The light cue is satisfied (present) in this instance
        ;  and has a LN << 1.  The tone cue is not satisfied
        ;  (missing) and also has a LN << 1.  Their conjunction
        ;  is suggested as a new clause.

...
```

5.3 Explanation of program behavior

LURN calculates its confidence in predicting the shock by using the LS and LN values associated with each of the features. Logical sufficiency (LS) and logical necessity (LN) are calculated by

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

where s is the count of successful predictions, c is errors of commission, o is errors of omission, and n is non-predictions. LURN updates counts for satisfied and unsatisfied features when F2 is expected or when F2 is detected. LURN doesn't update counts when F2 is neither expected or detected.

Confidence in LURN is calculated by multiplying together the LS values of each satisfied feature and the LN values of each unsatisfied feature. This confidence measure is then interpreted in terms of odds: much less than 1 indicates that F2 is not expected; about 1 indicates uncertainty; much greater than 1 indicates that F2 is expected.

LURN introduces new clauses only on errors of commission. By restricting the proposal of new clauses to immediately after errors, LURN avoids endlessly making proposals when it reaches proficiency. Possible new clauses are proposed from the satisfied and unsatisfied features with LN values below a bound of 1.

6 Conclusions: Achievements and Limitations

6.1 Contingency versus simple strengthening of associations

LURN expands the concept of "strength" or "weight" of a memory trace to include sufficient information to determine inter-event contingency. The salience of individual features and of combinations of features is determined incrementally, and performance remains quite good even with a considerable number of erroneous training instances.

6.2 Future Work

The field of animal learning is rich in important data, little of which has been examined from the viewpoint of artificial intelligence. We have begun work on blocking [Kamin 1968, 1969] and on differences in response latencies between animals in instrumental and classical conditioning procedures. Weaknesses remain in some aspects of our model. In particular, neither the encoding of features (unstructured attributes) nor the mechanisms for learning combinations of features (which currently propose only conjunctions and negations) are as strong as they need to be. We expect the behavioral data to continue to guide us toward a more complete and powerful model.

References

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

2. Bower, Gordon H. and Hilgard, Ernest R. *Theories of Learning.* Englewood: Prentice-Hall, 1981.

3. Duda, R. O., Gaschnig, J. G., Hart, P. 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, 1979.

4. Granger, Richard H. Identification of components of episodic learning: The CEL process model of early learning and memory. *Cognition and Brain Theory 6*(1), 1982, 5–32.

5. Granger, Richard H. An artificial intelligence model of learning and memory that provides a theoretical framework for the interpretation of experimental data in psychology and neurobiology. Department of Computer Science Technical Report 220, University of California, Irvine, March, 1983.

6. Granger, Richard H. and McNulty, Dale M. Learning and memory in machines and animals: An AI model that accounts for some neurobiological data. *Proceedings of the 6th Annual Conference of the Cognitive Science Society,* Boulder, Colorado, 1984.

7. Hintzman, Douglas L. *The Psychology of Learning and Memory.* San Francisco: W. H. Freeman and Company, 1978.

8. Kamin, Leon J. Predictability, surprise, attention, and conditioning. In Jones, Marshall R., ed., *Miami Symposium on the Prediction of Behavior, Aversive Stimulation.* University of Miami Press, Coral Gables, FA, 1968.

9. Kamin, Leon J. "Attention-like" processes in classical conditioning. In Campbell, Byron A. and Church, Russell M., eds., *Punishment and Aversive Behavior.* Conference on Punishment, Princeton, NJ, 1967. New York: Appleton-Century-Crofts, 1969.

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

11. Langley, Pat. Learning to search: From weak methods to domain-specific heuristics. The Robotics Institute, Carnegie-Mellon University Technical Report CMU-RI-TR-84-21, Carnegie-Mellon University, Pittsburgh, 1984.

12. Rescorla, Robert, Predictability and number of pairings in Pavlovian fear conditioning. *Psychonomic Science 4*(11), 1966, 383-384.

13. Rescorla, Robert. Pavlovian conditioning and its proper control procedures. *Psychological Review 74*(1), 1967, 71–80.

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

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

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