REPORT NO. UIUCDCS-R-85-1206 UILU-ENG-85-1711

85-30

UTILITY PATTERNS AS CRITERIA FOR EFFICIENT GENERALIZATION LEARNING

by

Larry Rendell

April 1985

REPORT No. UIUCDCS-R-85-1206

UTILITY PATTERNS AS CRITERIA FOR EFFICIENT GENERALIZATION LEARNING

by

Larry Rendell

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield Avenue
Urbana, Illinois 61801

April 1985

This paper will appear in the 1985 Conference on Intelligent Systems and Machines.

UTILITY PATTERNS AS CRITERIA FOR EFFICIENT GENERALIZATION LEARNING

Larry Rendell

Department of Computer Science,
University of Illinois at Urbana-Champaign,
1304 West Springfield Avenue, Urbana, Illinois 61801

Efficient task performance and *ability to predict* are two consequences of *induction* (generalization learning). Because of its import and subtlety, induction is a fundamental problem of many fields of study. Recently, artificial intelligence has begun to focus more sharply on essential issues and principles underlying generalization learning. Of the many AI implementations of induction, the author's *probabilistic learning system PLS* is one of the best evidenced combinations of efficiency, effectiveness, power, scope, and extensibility. This paper considers PLS as a paradigm for these qualities and addresses some issues and principles of practical mechanized induction.[1]

1. INTRODUCTION

Efficiency and prediction follow generalization. *Generalization learning* or *induction* compresses large numbers of "similar" *objects, patterns,* or *events* into meaningful *classes, categories,* or *concepts.*[2] Since generalization means fewer categories to manage, induction promotes economy of space and time; since categorization merges observed events, attendant concepts descriptions anticipate similar events (see Figs. 1, 2).

In terms of both application and theory, induction is one of the most important problems of artificial intelligence [Dietterich 82, Michalski 83, Mitchell 85] and pattern recognition [Fu 80, Watanabe 69].[3] In an expert system, for example, computer generalization allows mechanized knowledge acquisition, and consequently reduced costs. Present expert systems are prone to unexpected error, whereas induction would increase reliability and obviate maintenance: even in its primitive state, automated induction has outperformed a knowledge engineering approach [Michalski 80].

1. This work was supported in part by an operating grant form the Natural Sciences and Engineering Research Council of Canada.

2. Induction may be examined more formally. Aspects of induction may be stated more formally. Given a set S of observed objects, a simple kind of induction is the inference of a larger set, *class*, or *hypothesis* T, such that $S \subset T$. Since T is a generalization it may not be true; our confidence in T is called its *credibility*. This is the most abstract view of induction, i.e. class formation with associated credibility criterion [Watanabe 69, 72]. (This view even covers creation of structure, which involves concept formation and reconsideration of what constitutes an object, e.g. first chess pieces, then "chunks" of knowledge such as attack formations. This paper just alludes to structure; more can be found in [Rendell 85a].) The classification view is useful for purposes such as discovering the difficulty of a generalization problem or measuring the power of an inductive system.

3. Generalization learning is also a fundamental problem in philosophy [Christensen 64] and psychology [Medin 84].

1.1. INHERENT DIFFICULTY

Induction produces regular, coherent classes or concepts. Their discovery and expression requires some language such as predicate logic [Watanabe 69], semantic networks [Winston 84], frames or schemata [DeJong 85], feature vectors [Duda 73], formal grammars [Fu 82], etc. The more expressive the language, the more powerful a learning system can be, but the more difficult the process. Although a great deal of research has been done, present induction systems are limited in scope, efficiency, or extensibility [Dietterich 83].

Induction is a difficult problem for a variety of reasons. First, the number of possible generalizations is large, compared with the number which can be explored through evaluation or construction of explicit hypotheses. For example, if a small 10×10 grid of bits encodes letters of the alphabet, the number of different classes (hypotheses) is $27^{[2^{100}]}$, and very few of these are sensible. The second major difficulty is that noisy and sparsely distributed data offer little help in distinguishing any hypotheses which are considered.

1.2. CREDIBILITY AND OTHER CONSTRAINTS

Certain constraints can reduce the combinatorial explosion of hypotheses. A straightforward tactic is simply to limit the description language without confining its power too much (e.g. permitting conjunctions but not disjunctions in logic expressions [Michalski 83]).

As Watanabe showed in his "theorem of the ugly duckling", no one classification (hypothesis) is intrinsically better than any other [Watanabe 69]. In order to select an appropriate concept, we must rely on some external criterion. This criterion is the meaningfulness, quality, or *credibility;* it expresses some ascribed elegance or purpose of a generalization [Watanabe 69].

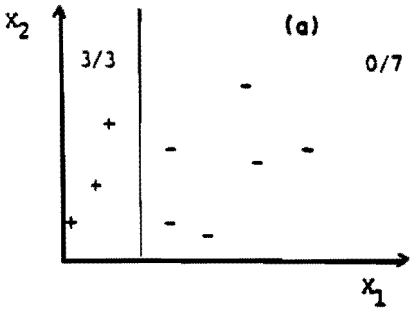
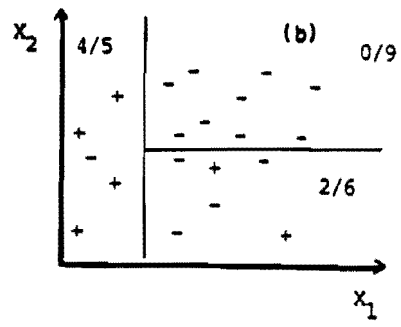


Figure 1. Deterministic versus probabilistic induction in two dimensional feature space. Ideally, classes can be neatly differentiated into positive and negative instances (a). But more commonly, exceptions occur (b). In mild cases, these are just anomalies whose effects can be recorded using proportions.



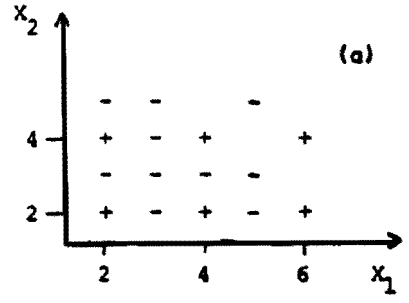
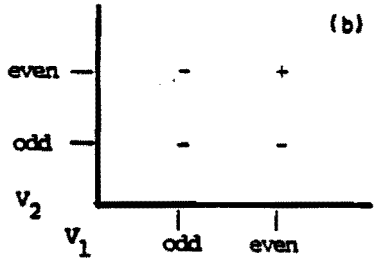


Figure 2. Constructive induction. If positive training examples cannot be delineated using a few rectangles, the original attributes (a) may be transformed into new descriptions (concepts); then simple induction is possible (b). Concept formation is the hard part.



Credibility may be used *to evaluate* a hypothetical class or concept; credibility may also be employed *to restrict* candidate hypotheses, producing only the more plausible ones [Michalski 83, Rendell 83a, 85a]. Credibility provides a means to impose or discover underlying order, regularity, or structure in desired concepts, and can promote efficient and effective induction.

Examples of credibility include "simplicity" of concept description, "sparseness of fit" to the data [Michalski 83], and "invariance" under transformation of task elements [Ernst 82]. A relaxed form of invariance has long been used in statistics and pattern recognition, viz. "similarity" in cluster analysis [Anderberg 73].

1.3. PLS, EFFICACY, AND EFFICIENCY

The author's scheme involves a special kind of similarity constraint which utilizes the whole data environment. The precise form of this similarity has to do with success or "utility" in the performance of some task [Rendell 83a, 83d, 85b]. This approach, used in the *probabilistic learning system PLS*, has produced unique results such as convergence to optimal heuristics [Rendell 83a, 83b, 85c]. PLS can handle noisy environments and incremental learning. The system is economical with regard to both space and time requirements [Rendell 83a, 85c].

This paper examines some of the underlying reasons for the efficiency and effectiveness of PLS. The next section explores the representation language used for objects, concepts, and related information. Section 3 considers some ways of imposing order using information structures and examines implications. Section 4 utilizes imposed regularity for efficiency in induction algorithms. The final section summarizes the main points.

2. EFFECTIVE REPRESENTATION

Much of the AI work on generalization learning has been rather simplified, implementing straightforward kinds of induction, disregarding uncertainty and noise, and omitting complex structuring needed for real-world knowledge acquisition (see the systems described in [Dietterich 82]). In this section we consider some of these aspects of knowledge representation for induction.

Generalization learning systems start with objects or events and produce classes or concepts. Effective systems also tend to represent *intermediate knowledge*. Intermediate information is used to create more knowledge.[4] Consider for example recording the degree of assurance that an object belongs to a class: as evidence mounts, a hypothesis is first tentative, then more credible, and finally well established. This important information mediates objects and concepts, improving efficiency and effectiveness [Rendell 85a]. Some forms of intermediate information will be discussed after the basic requirements for objects and classes.

2.1. LANGUAGE FORMS

As mentioned earlier, different approaches to generalization learning have employed various means to represent objects and concepts (e.g. logic, semantic nets, etc.). Though a particular language is usually more natural than other languages for a given domain, they are alternative, rather than unrelated modes of expression. This is clear in studies of equivalences [Banerji 71, Brachman 83, Kanal 72, Levesque 84, Schubert 7[illegible] A

4. One interesting model is discussed in [Lenat 84]. Undisputed reasons for the power of such systems await further analysis [Ritchie 84].

description of an object or concept can usually be mapped into a corresponding representation in another language, although there are certain exceptions (for example feature vectors cannot express structure without some augmentation). Perhaps less well appreciated is that naturalness of final expression does not preclude intermediate languages. In fact multiple representations may not only be sensible, but very appropriate [Fu 82, Rendell 85a]. With this flexibility in mind, we note that a practical approach to induction may begin with a *feature space* description of objects.[5]

2.2. EXPRESSION OF OBJECTS

An event or object is a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where n is the number of *attributes* or *features* x_i describing the data. In vision for example, the set of attributes might be the light intensity (gray level) for a total of n squares (pixels) of a grid representing an image. A grid is expressed as an n-dimensional vector, a point in feature space. If the problem is to discover those images which contain some symbol, each vector observed becomes a positive or negative instance of the class. Fig. 1 shows a simpler example, where n is only 2.

Any inductive problem may begin with feature space descriptions; however, to construct "higher level" features or concepts, transformation of attributes may be necessary (see Fig.2).[6] Attribute transformation requires formation of logic descriptions or grammars in order to represent relationships among objects or among their parts. We shall return to this topic later in the paper (see also [Fu 82, Rendell 85a]); for the present, we disregard cases needing structure alteration, and describe classes which are quite uniform in the original feature space (as in Fig. 1).

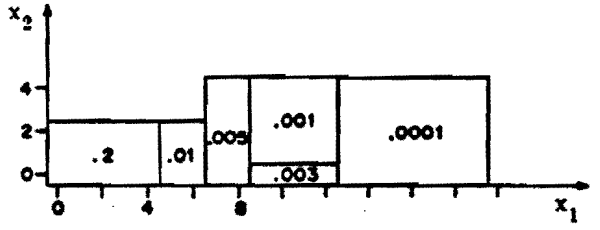


Figure 3. Rectangular utility classes. Feature space can be partitioned into rectangles representing similar probability of occurrence of some event, such as appearance in a solution to a problem or a win in a game. Effective clustering creates rectangles whose characteristics match the problem domain.

5. The term "feature space" is usually associated with variables having linearly ordered ranges. We do not consider the case of nominal scales, for which the term "event space" has been used [Michalski 83]. There have been very many applications of feature space representations, from sensory processing to problem solving [Rendell 83a, 85c, Tou 74].

6. Difficult induction requiring transformation of attributes is sometimes called "constructive" induction. We shall term the easier kind "simple".

2.3. REPRESENTATION OF CLASSES

When domain and features permit, a class has a concise description, as in Fig. 3 where the leftmost *hyperrectangle* r is $(0 \leq x_1 \leq 4) \cap (0 \leq x_2 \leq 2)$. PLS efficiently manipulates these concise expressions. Storing domain knowledge in *modifiable* feature space cells began with the original probabilistic learning system, *PLS1*. [Rendell 76, 83a] and continued with substantial extensions in [Coles 84, Rendell 83b, 83c, 84, 85a, 85b, 85c].[7] Section 3 of this paper will explain how PLS1 clusters objects into rectangular classes having uniform utility (e.g. probability of success in a task).

The meaning is simplest if only two classes are possible, and an event x is a positive or negative instance of a class r (this situation is illustrated in Fig. 1a). In most natural representations, however, there are more than two categories. One kind of multiple classification depends on *utility* u, the usefulness for some task; for example u could represent the likelihood that $\mathbf{x} \in \mathbf{r}$ leads to a desired result (see Fig. 3). PLS models this situation. In one use of PLS, an object x is a state in a problem or a game, and the utility classes r are probability classes, i.e. the probability that x will appear in a good solution or win.

2.4. CODING UNCERTAINTY

Real-world situations always involve some degree of noise or uncertainty: observations are often unreliable and attributes incomplete. In the PLS model of these situations, positive versus negative instances of a concept are counted, and repeated samplings produce a *probability* $u = k/t$, where k is the number of successes and t is the sample size (see Figs. 1 and 3).[8] If we classify according to probability, we have as many categories as probability values (6 in Fig. 3). Associated with each u is some set r of feature values; u can be considered the name of class r. In Fig. 3, r is the leftmost rectangle, representing probability $u(r) = 0.2$ of occurrence of some event E, i.e. u(r) is the conditional probability $Pr(E|r)$ of E, given that object $\mathbf{x} \in \mathbf{r}$. In PLS, the event E is some overall goal in a task domain, so u(r) is measurable. Notice that from one point of view, the potential number of probability classes is the number of u values, while from another point of view, we have returned to the situation in which there are

7. PLS feature space partitioning is similar to attribute-valued logic VL_1 [Michalski 73, 83]. PLS implements only ordinal or interval scales, but unlike VL_1, even early versions of PLS managed noise and learned incrementally in difficult domains [Rendell 76, 83a]; extensions can handle nonlinearity and structuring [Rendell 83b, 83c, 85b, 85c]. All PLS systems are efficient.

8. AI sometimes uses untraditional formalisms for probability. Stortliffe's "certainty factor" in MYCIN is nonstandard. Mitchell's "degree of match" in LEX is also probabilistic [Mitchell 78]. Any method which incoporates differential experience is essentially probabilistic.

only two classes (success or not). In the latter view, the feature space representation is a discrete function u(r) indicating probability of inclusion in the "success" class.[9]

2.5. AUGMENTING FEATURE SPACES

Probability is one kind of goal-oriented *utility* u which can be associated with a feature vector **x** or feature space neighborhood r. In the deterministic case, utility becomes binary, indicating membership in just a single class. The precise definition of utility is less important than its abstract use as a measure of quality in the performance of some task; this could be defined variously.[10]

Associating utility with features is equivalent to extending or *augmenting* feature space; coordinates are added to give (**x**, u) or (r, u). As well as utility u, various other kinds of information can be identified with **x** or r. One example is the error in u [Rendell 83a]. This association of appropriate information with r is one step in overcoming the structure limitations of pure feature space approaches. We can use augmented feature space representations to create expressions in other representation languages which express structure but cannot easily be used to derive it [Rendell 85a].

Augmented feature spaces can be used to represent intermediate knowledge structures (information derived by a learning system for its own use), as well as the final products of the inductive process (classes or concepts). Augmented feature spaces can support a mechanism to derive higher-level, structurally-oriented representations.

Unfortunately, choosing effective representations is only part of the solution to automated induction -- efficient manipulation is also required. Intuitively, it is apparent that effective representation and efficient manipulation are closely related to the degree of "pre-existing" or "natural" regularity. The following section examines the relationship between this pair of critical issues.

3. SOURCES OF ORDER

Learning systems always generalize partly on the basis of pre-existing knowledge (they always are somewhat "model-driven", they use domain knowledge). This knowledge may be expressed in a variety of forms: as information structures, as algorithms, or as control structures for algorithms. These various factors determine the cost, power, and difficulty of induction. For example, if attributes are ordinal features, and if these features affect utilities smoothly, generalization is easier (see Fig. 1). In the following, we consider some kinds of regularity which may be imposed on data, classes, and domains.

A rational basis for guiding induction and creating order is the utility. Since utility is the essence of task performance, it should regulate class formation and underlie information structures. Structures use attribute vectors, so attributes influence order or regularity.

3.1. FEATURE QUALITY

The overriding determiner of regularity is the attributes describing objects. There are related two ways in which attributes may aid induction: one involves the amount of detail they describe, the *grain size* (data compression), and the other has to do with the regularity they impose, the *utility coherence*.

Grain Size (Original Data Compression). As suggested by Fig. 2, attributes have greater or lesser degrees of abstraction. In a game such as checkers, *low-level, primitive* attributes give contents of individual squares, while *high-level, abstract* features summarize properties relevant to winning (e.g. "piece advantage", "mobility", "center control", etc.). Depending on attributes, one point in feature space may represent more than one case. If the attributes are primitive, a feature space representation will compress the data very little or not at all, and the space may be very large (e.g. about 10^{20} for checkers). On the other hand, if attributes are high-level, the feature space will contain considerably fewer points than there are primitive objects, and consequently induction will be easier, since order is then inherent in compressed feature vector descriptions. A feature vector compacts data having common or similar utility.

Utility Coherence (Uniformity or Smoothness). Another means of providing order for integrating events into classes depends on the relationships between utility and features. To simplify in an example, suppose utility is binary, and indicates class membership. Fig. 2 shows two different patterns of just one "concept" (two classes, positive and negative).[11] These two patterns have different degrees of regularity. For a simple concept description, either situation in Fig. 1 needs only a straightforward partition of feature

9. An application of this appears in [Rendell 83a, 85c]. There, an object x represents a state in a problem or a game, and "success" means leading to discovery of a good solution or of a win. If x falls within a rectangle r, then the function u(r) = H(x) may be used as an *evaluation function* or *heuristic* for state-space search to assess state x.

10. An alternative definition for utility in problem solving is the estimated solution path length. In simple concept learning, utility is the sign of the training instance.

11. When events are specified as positive and negative examples (training instances) of a concept, the learning is *supervised*. This corresponds to the definition of simple induction stated in Footnote 2: a set S of events is given, of which $S_+ \subset S$ are positive examples, and $S_- \subset S$ are negative. From these, a supervised learning system must form a class T_+ such that $S_+ \subset T_+$, and $T_+ \cap S_- = \emptyset$. T_+ is the general class or concept. In *unsupervised* learning the number of classes is not known, nor is their meaning. Lack of supervision makes induction more difficult.

space, while Fig. 2 requires formation of the intermediate concept of "evenness". The situations of Fig. 1 allow elementary methods of class formation: these include discovery of parameters in linear discriminant functions, or insertion of partitions in feature space clustering [Tou 74]. However, mechanized structuring is very difficult, and no general and tractable methods exist. The *simple* (easier) kind of induction has also been called *selective*, and the other, whereby abstract concepts or features are created, is *constructive* [Michalski 83], the *problem of new terms* [Dietterich 82], or *conceptual knowledge acquisition* [Rendell 85c].

The significant differences between selective and constructive induction become apparent in a more concrete example such as checkers, where high-level features ("piece advantage", "center control", etc.) take the roles of the coordinates in Fig. 1. Here the basis for classification (utility) might be multivalued (perhaps indicating likelihood of a win). Utility varies only gradually with high-level features. In contrast, if the attributes are not abstract, but primitive (e.g. the contents of individual checkerboard squares), then utility is very irregular (much worse than Fig. 2 suggests). Here, concepts such as piece advantage would have to be created before straightforward partitioning algorithms could be effective.

3.2. REGULARITY & INDUCTIVE DIFFICULTY

If constructive induction is required, it too can be of varying degrees of difficulty. Symmetries like the one of Fig. 2 can be more or less complex, and as the complexity of a concept increases, so does the complexity of the language needed to describe it (e.g. full predicate logic). When disorder (entropy) is so bad that utility patterns are completely random, no generalization can occur which is reliable for any other case.

In practice, however, any interesting real-world situation exhibits some degree of commonality. This regularity appears not only across instances of a problem (e.g. all checker games), but also across problems of a class (e.g. all board games). In other words, induction requires classification, not only at the event level, but also at the level of the domain. As an example of the latter, utility is often invariant or similar under translation, rotation, etc. (these apply in divergent domains, from board games to vision).

In summary, there are many kinds and degrees of regularity. They can be captured by data structuring, concept structuring, and domain structuring. In the following section we shall examine how structured information may be converted into efficiency in algorithms; we shall also consider some ideas for avoiding loss of power.

4. DYNAMICS OF ORDER

This section of the paper examines algorithmic aspects of induction, using PLS as a paradigm. Details of the induction methods, and other aspects of PLS may be found in [Coles 84, Rendell 83b, 83c, 84, 85a, 85b, 85c].

4.1. EFFICIENCY IN SIMPLE INDUCTION

Let us consider some reasons for the efficiency of generalization algorithms used by the simpler systems PLS1 and PLS2 (which we shall collectively refer to as PLS2). PLS2 can be used in situations where some *utility* u can be defined. This covers a large class of problems. For example, u can simply indicate class membership (for single concept learning), or u can measure desirability for some performance task. Recall from Sections 2.3 and 2.4 that u is a property of a feature vector x or rectangle r (where r is aligned with feature space axes).

The form of the inductive hypotheses in PLS2 is "rectangle r has utility u". A perfect learning system would generate rectangles r of just the right sizes, so that r's would be of small extent in volumes of the space where utility varies rapidly, but r's would otherwise be large (causing various efficiencies). Each (r, u) pair would be perfectly accurate.

PLS2 attempts to approach this ideal by several forms of incremental feedback, credit assignment, and credit localization. While the system is complex, it is essentially a set of utility class rewriting rules for pairs (r, u). One rule alters u (effectively moving r to a different class. Other operators adjust rectangles; these rules include forming, refining (specializing), merging (generalizing), and recombining, so that the feature space partition is iteratively improved during task performance.

Here we shall consider just one rule: refinement (i.e. specialization, differentiation). Refinement employs a special kind of clustering which examines contextual information. A rectangle r is tentatively split in two, and the split becomes permanent if it is "better than" other tentative refinements, and if the split is "good enough". These decisions depend on the *utility divergence* in the two subrectangles. Utility divergence is measured as the variation of utility in feature space, accounting for observation error.[12] This is appropriate since utility is used to guide task performance. Utility divergence results in "simplicity" and "fit", as a *consequence* of goal directedness (compare with [Michalski 83]).

12. In more precise detail: The criterion for splitting involves a dissimilarity (distance) measure d. If u_1 and u_2 are the two utilities for a tentative dichotomy, and ϵ_1 and ϵ_2 their errors, then $d = |\log u_1 - \log u_2| - \log(\epsilon_1 \epsilon_2)$. This distance is computed for all boundary insertions parallel to any feature space axis. If the largest d is positive, the corresponding split is retained. The process repeated until additional refinement is unwarranted by the data (until $d \leq 0$). Notice that larger d means more assured dissimilarity.

This splitting scheme has several efficiencies. One is that a rectangle is economically stored (just two feature vectors specify the extreme corner points). Second, rectangles compress objects into fewer categories (utility classes). Because few rectangles are required, PLS2 makes efficient use of both storage space and processing time. Finally, the generation of rectangles is inexpensive because of the small number of dichotomies to be tested. The time complexity of the splitting algorithm is knm, where k is the number of objects observed, n is the dimensionality of the feature space, and m is the number of rectangles (utility classes) formed.

Like clustering (refinement), PLS2's other utility class rewriting rules for $\{(r,u)\}$ are also efficient. (These rules include u modification, r generalization, and r recombination.)

The overall scheme can be considered an improvement of signature tables [Samuel 67]. Both approaches use feature space rectangles to represent heuristic knowledge, but in PLS2 more knowledge is autonomously acquired. Instead of having the user compress feature ranges and dimensionality, PLS2 and the data determine sizes, shapes, and effective dimensionality of feature space rectangles.

PLS2 and most other systems for generalization learning perform simple induction, where variation of utility with attributes is gradual, and feature space can be meaningfully partitioned by insertion of a few boundaries (see Fig. 1 and Section 3.1).

4.2. EFFICIENT CONSTRUCTIVE INDUCTION

The great majority of realistic inductive problems require discovery of new and appropriate object descriptions. Constructive induction is required when the variation of utility with original attributes is drastic (e.g. consider Fig. 2 or the 32 attributes e_i, defined as the contents of individual squares of a checker board). No realistic constructive induction has been tractably mechanized, although some limited algorithms have been devised [Ernst 82, Michalski 83, Rendell 85a].

A new system *PLS0* addresses a hard problem of constructive induction: mechanized creation of high-level features from very detailed, elementary attributes or *primitives* (e.g. the e_i above). PLS0 breaks down the problem into several levels, each with a reduced time complexity [Rendell, 85a]. The system naturally unites representations (feature spaces, logic descriptions, and formal grammars) as a consequence of its multi-layer design. PLS0 requires little background knowledge, and the approach is quite general.

Instead of clustering entire feature spaces, PLS0 examines appropriate primitive subspaces. The system alligns *patterns* of *utility similarity* -- that is: *utility surfaces*, not utility values are clustered. PLS0 can incrementally construct tentative components for high level features. More structured elements are created from these components after they have been validated (attained some credibility). Gradually structures emerge to become abstract features.

Just as traditional clustering is an effective and efficient means for selective induction, this scheme for constructive induction introduces more general techniques for economical use of scarce information. These techniques include layered information compression resulting in a "divide and conquer" approach, and subtle overlaying of data for simultaneous strengthening of inference and discovery of structure. A complex form of regularity is induced.

5. SUMMARY

Generalization learning systems attempt the difficult problem of induction, i.e. the creation of classes or concepts from data. Powerful induction systems tend to possess certain characteristics, each of which contributes to efficiency and effectiveness:

- Intermediate information structures are used to generate layers of more abstract knowledge.
- This mediating knowledge includes probabilistic information (although it may be disguised), whose purpose is to overcome noise and make maximal use of sparse data.
- Scarce and unreliable data are nevertheless used incrementally to increase credibility of tentative hypotheses.
- Credibility measures and other constraints are flexibly employed, not only to test generated hypotheses, but also to guide their formation.
- Knowledge representations are dynamically and appropriately selected; these representations are restrictive enough to aid efficiency, yet expressive enough to learn structure in observed events.

ACKNOWLEDGEMENTS

I would like to thank Chris Matheus and Raj Seshu for their helpful comments on earlier drafts of this paper.

REFERENCES

Anderberg, M.R., *Cluster Analysis for Applications*, Academic Press, 1973.

Banerji, R.B., Some linguistic and statistical problems in pattern recognition, *Pattern Recognition 3*, 409-419, 1971.

Brachman, R.J., What IS-A is and isn't, *IEEE Computer, Special Issue on Knowledge Representation*, 30-36, October, 1983.

Christensen, R., *Foundations of Inductive Reasoning*, Entropy, Ltd., 1964.

Coles, D. and Rendell, L.A., Some issues in training learning systems and an autonomous design, *Proc. Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, 1984.

Dietterich, T.G., London, B., Clarkson, K., and Dromey, G., Learning and inductive inference, STAN-CS-82-913, Stanford University, also Chapter XIV of *The Handbook of Artificial Intelligence*, Cohen, P.R., and Feigenbaum, E.A. (Ed.), Kaufmann, 1982.

Dietterich, T.G. and Michalski, R.S., A comparative review of selected methods for learning from examples, in Michalski, R.S. et al (Ed.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 41-81, 1983.

Duda, R.O. and Hart, P.E., *Pattern Classification and Scene Analysis*, Wiley, 1973.

Ernst, G.W. and Goldstein, M.M., Mechanical discovery of classes of problem-solving strategies, *J. ACM 29*, 1-33, 1982.

Fu, K.S., *Syntactic Pattern Recognition and Applications*, Prentice-Hall, 1982.

Kanal, L. and Chandrasekaran B., On linguistic, statistical and mixed models for pattern recognition, in Watanabe, S. (Ed.), *Frontiers of Pattern Recognition*, Academic Press, 163-192, 1972.

Lenat, D.B. and Brown, J.S., Why AM and Eurisko appear to work, *Artificial Intelligence 23*, 269-294, 1984.

Levesque, H.J., A fundamental tradeoff in knowledge representation and reasoning, *Proc. Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, 141-152, 1984.

Medin, D.L. and Smith, E.E., Concepts and concept formation, *Annual Review of Psychology 35*, 113-138, 1984.

Michalski, R.S., Discovering classification rules using variable-valued logic system VL_1, *Proc. Third International Joint Conference on Artificial Intelligence*, 162-172, 1973.

Michalski, R.S., A theory and methodology of inductive learning, *Artificial Intelligence 20*, 2 (1983), 111-161; reprinted in Michalski, R.S. et al (Ed.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 83-134, 1983.

Michalski, R.S. and Chilausky, R.L., Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis, *Int. J. Policy Analysis and Information Systems 4*, 2, 125-161, 1980.

Mitchell, T.M., Version spaces: An approach to concept learning, Stanford University Ph.D. Dissertation, 1978.

Mitchell, T.M., Mahadevan, S., and Steinberg, L.I., LEAP: A learning apprentice for VLSI design, *Proc. Ninth International Joint Conference on Artificial Intelligence*, (to appear), 1985.

Rendell, L.A., A method for automatic generation of heuristics for state-space problems, Dept of Computer Science CS-76-10, University of Waterloo, 1976.

Rendell, L.A., A new basis for state-space learning systems and a successful implementation, *Artificial Intelligence 20*, 4, 369-392, 1983a.

Rendell, L.A., A learning system which accommodates feature interactions, *Proc. Eighth International Joint Conference on Artificial Intelligence*, 469-472, 1983b.

Rendell, L.A., A doubly layered, genetic penetrance learning system, *Proc. Third National Conference on Artificial Intelligence*, 343-347, 1983c.

Rendell, L.A., Toward a unified approach for conceptual knowledge acquisition, *AI Magazine 4*, 4, 19-27, Winter 1983d.

Rendell, L.A. and Burgess, J., A uniform learning system for problems and games, CIS Dept. Report CIS84-6, University of Guelph, 1984.

Rendell, L.A., Substantial constructive induction using layered information compression: Tractable feature formation in search, *Proc. Ninth International Joint Conference on Artificial Intelligence*, (to appear), 1985a.

Rendell, L.A., Genetic plans and the probabilistic learning system: Synthesis and results (submitted), 1985b.

Rendell, L.A., Conceptual knowledge acquisition in search, in Bole, L. (ed.), *Knowledge Based Learning Systems*, Springer-Verlag, (to appear), 1985c.

Ritchie, G.D. and Hanna, F.K., AM: a case study in AI methodology, *Artificial Intelligence 23*, 249-268, 1984.

Samuel, A.L., Some studies in machine learning using the game of checkers II--recent progress, *IBM J. Res. and Develop. 11*, 601-617, 1967.

Schubert, L.K., Extending the Expressive Power of Semantic Networks, *Artificial Intelligence 7*, 163-198, 1976.

Tou, T.T. and Gonzales, R.C., *Pattern Recognition Principles*, Addison-Wesley, 1974.

Watanabe, S., *Knowing and Guessing: A Formal and Quantitative Study*, Wiley, 1969.

Watanabe, S., Pattern recognition as information compression, in Watanabe, S. (Ed.), *Frontiers of Pattern Recognition*, Academic Press, 561-567, 1972.

Winston, P.H., *Artificial Intelligence* (second edition), Addison Wesley, 1984.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-85-1206	2.	3. Recipient's Accession No.

4. Title and Subtitle

UTILITY PATTERNS AS CRITERIA FOR EFFICIENT GENERALIZATION LEARNING

5. Report Date: April 1985

6.

7. Author(s): Larry Rendell

8. Performing Organization Rept. No. R-85-1206

9. Performing Organization Name and Address

Department of Computer Science UIUC
1304 W. Springfield Avenue #222
Urbana, IL 61801

10. Project/Task/Work Unit No. 11

11. Contract/Grant No. A2497

12. Sponsoring Organization Name and Address

Natural Sciences and Engineering Research Council of Canada
200 Kent Street
Ottawa, Canada K1A 1H5

13. Type of Report & Period Covered: technical

14.

15. Supplementary Notes

16. Abstracts

Efficient task performance and ability to predict are two consequences of induction (generalization learning). Because of its import and subtlety, induction is a fundamental problem of many fields of study. Recently, artificial intelligence has begun to focus more sharply on essential issues and principles underlying generalization learning. Of the many AI implementations of induction, the author's probabilistic learning system PLS is one of the best evidenced combinations of efficiency, effectiveness, power, scope, and extensibility. This paper considers PLS as a paradigm for these qualities and addresses some issues and principles of practical mechanized induction.

17. Key Words and Document Analysis. 17a. Descriptors

machine learning
inductive inference
conceptual clustering

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited	UNCLASSIFIED	10
	20. Security Class (This Page) UNCLASSIFIED	22. Price