Machine Learning 2: 139-172, 1987


# Knowledge Acquisition Via Incremental Conceptual Clustering

DOUGLAS H. FISHER (DFISHER@ICS.UCI.EDU)

*Irvine Computational Intelligence Project, Department of Information and Computer Science, University of California, Irvine, California 92717, U.S.A.*

(Received: October 6, 1986)
(Revised: July 4, 1987)

**Keywords:** Conceptual clustering, concept formation, incremental learning, inference, hill climbing

**Abstract.** Conceptual clustering is an important way of summarizing and explaining data. However, the recent formulation of this paradigm has allowed little exploration of conceptual clustering as a means of improving performance. Furthermore, previous work in conceptual clustering has not explicitly dealt with constraints imposed by real world environments. This article presents COBWEB, a conceptual clustering system that organizes data so as to maximize inference ability. Additionally, COBWEB is incremental and computationally economical, and thus can be flexibly applied in a variety of domains.

## 1. Introduction

Machine learning is concerned with improving performance by automating knowledge acquisition and refinement. This view is reflected in the simple model of intelligent processing (Dietterich, 1982) shown in Figure 1. In this model, a learning system accepts environmental observations and incorporates them into a knowledge base, thereby facilitating some performance task. Assumptions about the environment, knowledge base, and performance element all impact the design of a learning system (Rendell, 1986). This article is concerned with *conceptual clustering*, a learning task that has not traditionally been discussed in the larger context of intelligent processing. As with other forms of learning, such a treatment can have important implications on the design of clustering systems.

Conceptual clustering is a machine learning task defined by Michalski (1980). A conceptual clustering system accepts a set of object descriptions (events, observations, facts) and produces a classification scheme over the observations. These systems do not require a 'teacher' to preclassify objects, but use an evaluation function to discover classes with 'good' con-

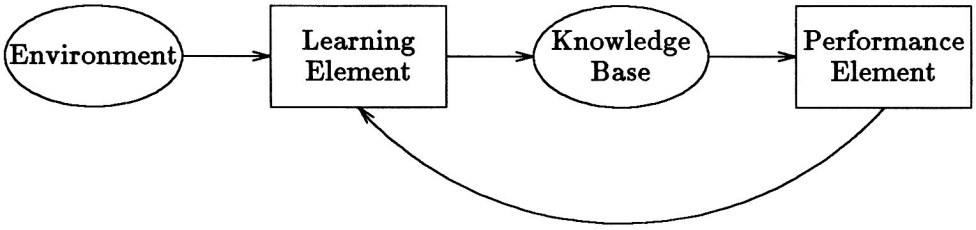


*Figure 1.* A model of learning and performance.

ceptual descriptions. Thus, conceptual clustering is a type of *learning by observation* (as opposed to *learning from examples*) and is an important way of summarizing data in an understandable manner. However, because this paradigm has been formalized only recently, previous work has not focused on explicating these methods in the context of an environment and performance task. Traditional treatments have concentrated almost exclusively on the clustering (i.e., learning) mechanism and the form of the resultant classification (i.e., knowledge base). This paper presents COBWEB, a conceptual clustering system that is inspired by environmental and performance concerns.

The most important contextual factor surrounding clustering is the general performance task that benefits from this capability. While most systems do not explicitly address this task (and thus the often asked question: "How do you know the resulting classifications are any good?"), some exceptions exist. CLUSTER/S (Stepp & Michalski, 1986) augments object descriptions with attributes that may be useful for inferring conditions under which domain-dependent goals (e.g., 'survival') can be attained. Cheng and Fu (1985) and Fu and Buchanan (1985) use clustering to organize expert system knowledge. Abstracting these uses of conceptual clustering, classification schemes can be a basis for effective inference of unseen object properties. The generality of classification as a means of guiding inference is manifest in recent discussions of problem solving as classification (Clancey, 1985). For example, a tiger may be recognized by its observed features and thus be regarded as life-endangering, or a set of symptoms may suggest a particular disease from which a cure can be inferred. COBWEB favors classes that maximize the information that can be predicted from knowledge of class membership.

A second contextual factor surrounding learning is the environment. Conceptual clustering systems typically assume environmental inputs are indefinitely available for examination and thus the environment is amenable to nonincremental processing of observations. However, real world environ-

ments (Carbonell & Hood, 1986; Langley, Kibler, & Granger, 1986; Sammut & Hume, 1986) motivate incremental object assimilation. Learning methods that incrementally process observations are gaining prominence (Lebowitz, 1982; Kolodner, 1983; Reinke & Michalski, 1985; Rendell, 1986; Schlimmer & Fisher, 1986). In response to real world considerations, COBWEB has been constructed as an incremental method of conceptual clustering.

In summary, COBWEB's design was motivated by concerns for environment and performance: learning is incremental and seeks to maximize inference abilities. The following section develops conceptual clustering as a process of search. Section 3 follows with a detailed description of COBWEB in terms of this framework. Section 4 demonstrates that COBWEB performs effective incremental learning of categories that are useful for predicting unknown object properties. Section 5 concludes with a summary of results, a discussion of shortcomings, and suggestions on how future work might rectify some of these problems.

## 2. Background: Learning as search

This section develops a framework for understanding COBWEB in terms of a pervasive paradigm of AI – *search.* Concept learning as search was first proposed to describe learning from examples and the main ideas are summarized here. The search model is then extended to conceptual clustering. Finally, a form of incremental learning is described along two dimensions: *search control* and *search direction.*

### 2.1 Concept learning from examples

The predominant form of concept learning studied in AI has been *learning from examples.* This task assumes that objects are classified by a 'teacher' with respect to a number of object classes. The goal is to derive concepts that appropriately describe each class. Mitchell (1982) has characterized learning from examples as a process of *search.* For each object class, the learner navigates through a space of concept descriptions until it finds an appropriate concept.

Using the search framework, systems for learning from examples have been characterized along a number of dimensions (Mitchell, 1982; Dietterich & Michalski, 1983; Langley & Carbonell, 1984). One dimension is the *search control strategy*; this can vary from exhaustive strategies, such as depth-first or breadth-first search, to heuristic methods like hill climbing, beam search, or best-first search. A second dimension, the *direction of search*, follows from the observation that concept descriptions are ordered

*Table 1.* Animal descriptions.

| *Name* | *BodyCover* | *HeartChamber* | *BodyTemp* | *Fertilization* |
|---|---|---|---|---|
| 'mammal' | hair | four | regulated | internal |
| 'bird' | feathers | four | regulated | internal |
| 'reptile' | cornified-skin | imperfect-four | unregulated | internal |
| 'amphibian' | moist-skin | three | unregulated | external |
| 'fish' | scales | two | unregulated | external |

by generality. Specifically, a learner may use *generalization* operators to search from specific concepts to more general ones, or it may use *specialization* operators to search from general to specific.

While learning from examples is a simple context for introducing the idea of learning as search, the real interest is in extending this framework to conceptual clustering. This task differs from learning from examples in that no teacher preclassifies objects; the task of the learner is to discover appropriate classes, *as well as* concepts for each class.

## 2.2 Conceptual clustering

Clustering forms a classification tree over objects. For example, given the animal descriptions in Table 1, clustering might result in the classification tree shown in Figure 2. Methods of conceptual clustering (Michalski, 1980; Michalski & Stepp, 1983) differ from earlier methods of *numerical taxonomy* (Everitt, 1980) in that clustering quality is not solely a function of individual objects, but is dependent on concepts that describe object classes (e.g., the *simplicity* of concepts) and/or the map between concepts and the classes they cover (e.g., the *fit* or generality of derived concepts). Despite differences in representation (Rendell, 1986) and quality judgements (e.g., understandability versus inference ability), all conceptual clustering systems evaluate class quality by looking to a summary or concept description of the class.

There are two problems that must be addressed by a conceptual clustering system:

- The *clustering* problem involves determining useful subsets of an object set. This consists of identifying a set of object classes, each defined as an extensional set of objects.
- The *characterization* problem involves determining useful concepts for each (extensionally defined) object class. This is simply the problem of learning from examples.

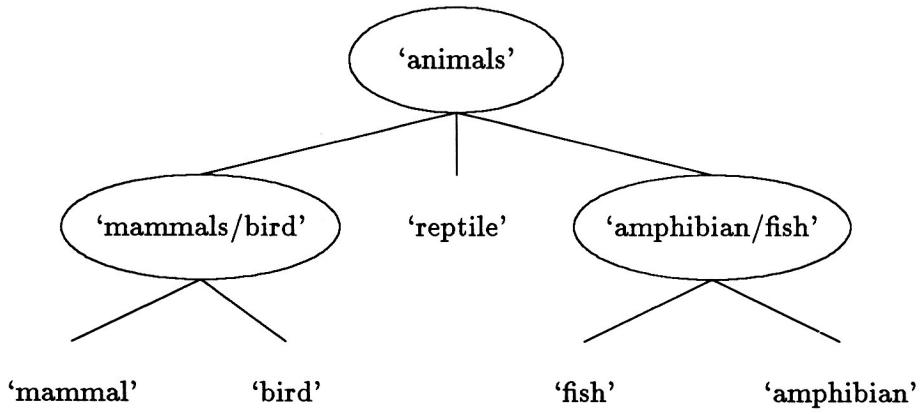


*Figure 2.* A hierarchical clustering over animal descriptions.

Fisher and Langley (1985, 1986) adapt the view of learning as search to fit conceptual clustering. Clustering and characterization dictate a two-tiered search, a search through a space of object clusters and a subordinate search through a space of concepts.[1] In the case of hierarchical techniques this becomes a three-tiered search, with a top-level search through a space of hierarchies.

As with learning from examples, the dimensions of *search control* and *search direction* can also be used to distinguish conceptual clustering systems. For instance, most systems transform a single classification tree throughout processing and thus hill climb through the space of hierarchies (Michalski & Stepp, 1983; Fisher, 1985). On the other hand, Langley and Sage's (1984) DISCON system makes a nearly exhaustive search of hierarchy space. Second, when searching through a space of hierarchies, search direction may dictate building a tree top down (*divisive* techniques) by continually dividing nodes (Langley & Sage, 1984; Michalski & Stepp, 1983; Fisher, 1985) or building a tree bottom-up (*agglomerative* methods) by continually fusing nodes (Hanson & Bauer, 1986; Cheng & Fu, 1985).

Search control and direction are important dimensions of concept learning. The next section motivates choices along these dimensions for purposes of incremental learning.

[1]Methods of numerical taxonomy do not result in concept descriptions for discovered classes. By definition, clustering and characterization cannot be independent in conceptual clustering; the results of characterization (i.e., a set of concepts) must be used to determine the quality of object classes (i.e., the result of clustering).

## 2.3 Incremental concept induction

Many concept learning systems, whether they carry out learning from examples or conceptual clustering, are *nonincremental* – all objects must be present at the outset of system execution. In contrast, *incremental* methods accept a stream of objects that are assimilated one at a time. A primary motivation for using incremental systems is that knowledge may be rapidly updated with each new observation, thus sustaining a continual basis for reacting to new stimuli. This is an important property of systems that are used under real-world constraints (Carbonell & Hood, 1986; Langley, Kibler, & Granger, 1986; Sammut & Hume, 1986). Search-intensive methods may be appropriate in a nonincremental system, but may be too costly for incremental processing, since they require updating a frontier of concept hypotheses and/or examining a list of previously seen objects. Schlimmer and Fisher (1986) imply that incremental processes are profitably viewed as strategies operating under diminished search control. Specifically, they use a hill-climbing strategy (with no backtracking) to implement and test incremental variants of Quinlan's (1983) ID3 program.

Schlimmer and Fisher demonstrate that the cost of object incorporation can be significantly reduced, while preserving the ability of the learning system to converge on concept descriptions of high quality. The ability to achieve high quality concept descriptions, despite the limitations of hill climbing, is maintained by extending the set of available operators. Rather than restricting search to be unidirectional, both generalization and specialization operators are supplied. Bidirectional mobility allows an incremental system to recover from a bad learning path.

In learning from examples, Winston's (1975) 'ARCH' program fits this view of incremental processing; it employs a hill-climbing strategy with operators for both generalization and specialization. This view can also be extended to conceptual clustering. For instance, Fisher and Langley (1985, 1986) view Lebowitz' (1982, 1986a) UNIMEM as an incremental conceptual clustering system. Given a new object and an existing hierarchy that was built from previous observations, the program incorporates the object into the hierarchy. This results in a classification hierarchy that covers the new object as well as previously seen objects. Since UNIMEM maintains only one hierarchy following each observation, it can be viewed as hill climbing through a space of classification hierarchies. Second, UNIMEM does not build its hierarchies in an entirely top-down or bottom-up fashion. Instead, it has operators for merging nodes in an agglomerative manner and deleting nodes and associated subtrees. Node deletion selectively undoes the effects of past learning and thus approximates backtracking.

While existing descriptions of UNIMEM and similar systems like CYRUS (Kolodner, 1983) are not framed as search, desirable search properties can be abstracted from them. These systems use diminished search control and greater operator flexibility to navigate through hierarchy space, and thus employ a practical strategy for incremental learning. The advantage of viewing these systems in terms of search is that it requires explicit consideration of the 'goal' of learning and of the system's ability to achieve or approximate this goal. The search framework forces analysis to move beyond anecdotal characterizations of system behavior.

## 3. COBWEB: Incremental conceptual clustering

UNIMEM and CYRUS, along with the conceptual clustering work of Michalski and Stepp, have inspired the COBWEB system. COBWEB is an incremental system for hierarchical conceptual clustering. The system carries out a hill-climbing search through a space of hierarchical classification schemes using operators that enable bidirectional travel through this space. This section describes COBWEB, filling in the details of the general incremental strategy. Specifically, the section gives

- the *heuristic evaluation measure* used to guide search,
- the *state representation*, including the structure of hierarchies and the representation of concepts,
- the *operators* used to build classification schemes, and
- the *control strategy*, including a high level description of the system.

### 3.1 Category utility: A heuristic evaluation measure

COBWEB uses a heuristic measure called *category utility* to guide search. Gluck and Corter (1985) originally developed this metric as a means of predicting the *basic level* in human classification hierarchies. Briefly, basic level categories (e.g., *bird*) are retrieved more quickly than either more general (e.g., *animal*) or more specific (e.g., *robin*) classes during object recognition. More generally, basic level categories are hypothesized to be where a number of inference-related abilities are maximized in humans (Mervis & Rosch, 1981).

Identifying preferred concepts in humans is important from a cognitive modeling standpoint, but it also provides a basis for developing principled criteria for evaluating concept quality in AI systems. Category utility can be viewed as a function that rewards traditional virtues held in clustering generally - similarity of objects within the same class and dissimilarity of objects in different classes. In particular, category utility is a tradeoff

between intra-class similarity and inter-class dissimilarity of objects, where objects are described in terms of (nominal) attribute-value pairs like those in Table 1. Intra-class similarity is reflected by conditional probabilities of the form $P(A_i = V_{ij}|C_k)$, where $A_i = V_{ij}$ is an attribute-value pair and $C_k$ is a class. The larger this probability, the greater the proportion of class members sharing the value and the more *predictable* the value is of class members. Inter-class similarity is a function of $P(C_k|A_i = V_{ij})$. The larger this probability, the fewer the objects in contrasting classes that share this value and the more *predictive* the value is of the class.

These probabilities are dispositions of individual values, but they can be combined to give an overall measure of partition quality, where a partition is a set of mutually-exclusive object classes, $\{C_1, C_2, ..., C_n\}$. Specifically,

$$\sum_{k=1}^{n} \sum_{i} \sum_{j} P(A_i = V_{ij}) P(C_k|A_i = V_{ij}) P(A_i = V_{ij}|C_k), \qquad 3-1$$

is a tradeoff between intra-class similarity (through $P(A_i = V_{ij}|C_k)$) and inter-class dissimilarity (through $P(C_k|A_i = V_{ij})$) that is summed across all classes (k), attributes (i), and values (j). The probability $P(A_i = V_{ij})$ weights the importance of individual values, in essence saying it is more important to increase the class-conditioned predictability and predictiveness of frequently occurring values than for infrequently occurring ones.

Function 3-1 balances traditional concerns of intra- and inter-class similarity (i.e., predictability and predictiveness). However, it also rewards the inference potential of object class partitions. More precisely, for any $i$, $j$, and $k$, $P(A_i = V_{ij})P(C_k|A_i = V_{ij}) = P(C_k)P(A_i = V_{ij}|C_k)$ by Bayes rule, so by substitution function 3-1 equals

$$\sum_{k=1}^{n} P(C_k) \sum_{i} \sum_{j} P(A_i = V_{ij}|C_k)^2. \qquad 3-2$$

In words, $\sum_i \sum_j P(A_i = V_{ij}|C_k)^2$ is the *expected* number of attribute values that can be correctly guessed for an arbitrary member of class $C_k$. This expectation assumes a guessing strategy that is *probability matching*, meaning that an attribute value is guessed with a probability equal to its probability of occurring. Thus, it assumes that a value is guessed with probability $P(A_i = V_{ij}|C_k)$ and that this guess is correct with the same probability.[2]

[2]Probability matching can be contrasted with *probability maximizing*. The latter strategy assumes the most frequently occurring value is always guessed. While this strategy may seem superior at a cursory level, it is not sensitive to the *distribution* of all attribute values and is not as desirable for heuristically ordering object partitions.

*Table 2.* Probabilistic representation of {fish, amphibian, mammal}.

| *Attributes* | *Values and probabilities* |
|---|---|
| BodyCover | scales [0.33], moist-skin [0.33], hair [0.33] |
| HeartChamber | two [0.33], three [0.33], four [0.33] |
| BodyTemp | unregulated [0.67], regulated [0.33] |
| Fertilization | external [0.67], internal [0.33] |

Finally, Gluck and Corter define category utility as the *increase* in the expected number of attribute values that can be correctly guessed ($P(C_k)$ $\sum_i \sum_j P(A_i = V_{ij}|C_k)^2$) given a partition $\{C_1, ..., C_n\}$ over the expected number of correct guesses with no such knowledge ($\sum_i \sum_j P(A_i = V_{ij})^2$). More formally, $CU(\{C_1, C_2, ..., C_n\})$ equals

$$\frac{\sum_{k=1}^{n} P(C_k)[\sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2]}{n} \qquad 3-3$$

The denominator, $n$, is the number of categories in a partition. Averaging over categories allows comparison of different size partitions. If an attribute value, $A_i = V_{ij}$, is independent of class membership, then $P(A_i = V_{ij}|C_k) = P(A_i = V_{ij})$ and $P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2 = 0$. If this is true for all the attribute's values then the attribute is effectively *irrelevant* for any expression of category makeup.

### 3.2 Representation of concepts

At the basis of any classification scheme is a representation of individual concepts. Given its sensitivity to the distribution of attribute values, the choice of category utility as a heuristic measure dictates a concept representation other than the logical, typically conjunctive, representations used in AI. Category utility can be computed from $P(C_k)$ of each category in a partition and $P(A_i = V_{ij}|C_k)$ for each attribute value. A summary representation that lists attribute values and associated probabilities is a *probabilistic* concept (Smith & Medin, 1981), an example of which is shown in Table 2.

Attribute value probabilities are computed from two integer counts. For example, a concept for the class of birds has an entry, $P$(flies|bird), that is computed by $\frac{\#times\text{-}a\text{-}bird\text{-}was\text{-}observed\text{-}to\text{-}fly}{\#times\text{-}a\text{-}bird\text{-}was\text{-}observed}$. Both counts of the quotient are stored at the node corresponding to 'birds'. As convenient, concepts will be alternatively discussed in terms of attribute-value probabilities and

the integer counts that underlie these probabilities.

In COBWEB, a probabilistic concept labels each node in the classification tree and summarizes the objects classified under the node. Probabilistic concept trees are unlike strict discrimination networks or decision trees (Feigenbaum & Simon, 1984) in that probabilistic (and not logical) descriptors label nodes (and not arcs) of the tree. Classification using a probabilistic concept tree is done using a partial matching function to descend the tree along a path of 'best' matching nodes.[3] The following section shows how COBWEB adapts this general procedure for tree update.

## 3.3 Operators and control

COBWEB incrementally incorporates objects into a classification tree, where each node is a probabilistic concept that represents an object class. The incorporation of an object is a process of classifying the object by descending the tree along an appropriate path, updating counts along the way, and performing *one* of several operators at each level. These operators include

- classifying the object with respect to an existing class,
- creating a new class,
- combining two classes into a single class, and
- dividing a class into several classes.

While these operators are applied to a single object set partition (i.e., set of siblings in the tree), compositions of these primitive operators transform a single classification tree. The emergent search strategy is one of hill-climbing through a space of classification trees.

### *3.3.1 Operator 1: Placing an object in an existing class*

Perhaps the most natural way of updating a set of classes is to simply place a new object in an existing class. In order to determine which category 'best' hosts a new object, COBWEB tentatively places the object in each category. The partition that results from adding the object to a given node is evaluated using category utility (3–3). The node that results in the best partition is identified as the best existing host for the new object.

[3]This procedure is *polythetic*; the choice of what path to follow at each node is dependent on an object's values along many attributes. In contrast, most decision trees are *monothetic* in that descent at each node is based on the value of a single attribute. Fisher (1987) discusses the relative advantages of the probabilistic tree approach.

### *3.3.2 Operator 2: Creating a new class*

In addition to placing objects in existing classes, there is a way to create new classes. Specifically, the quality of the *partition* resulting from placing the object in the best existing host is compared to the partition resulting from creating a new singleton class containing the object. Depending on which partition is best with respect to category utility, the object is placed in the best existing class or a new class is created. This operator allows COBWEB to automatically adjust the number of classes in a partition. The number of classes is not bounded by a system parameter (e.g., as in CLUSTER/2), but emerges as a function of environmental regularity.

### *3.3.3 A simple example*

Figure 3 demonstrates the effect of operators 1 and 2 in three snapshots. Snapshot (a) shows a classification tree that has been previously built over the 'fish' and 'amphibian' objects of Table 1. Listed with each node (class) are the probability of the class and the probabilities of attribute values conditioned on class membership. For example, the probability of having scales is 0.5 for objects classified at the root of snapshot (a), while scales are assured with probability 1.0 for objects classified at $C_1$ (a singleton class containing only 'fish'). Space prohibits showing more than one attribute value for each node, but all values exhibited over objects of a node are stored with their respective conditional probabilities. For example, node $C_0$ of tree (b) is completely specified by the probabilistic concept of Table 2. Probabilities reflect attribute value distributions over observed objects. As with any inductive program, there is an implicit assumption that the observations collectively approximate the environment as a whole. However, distributions are not permanent, but change in response to further observation (Cheeseman, 1985).

Snapshot (b) shows a new class being created. The transition from (a) to (b) is caused by incorporating the 'mammal' object of Table 1. The probability, $P(\text{scales}|C_0)$, reflects this addition at the root. Creating a new singleton class ($C_3$) corresponding to 'mammal' yields a better partition than adding the object to either of the existing classes.

Snapshot (c) demonstrates an object being added to an existing class. Adding 'bird' to the tree of snapshot (b) causes appropriate alterations at the root; e.g., scales now occur in only one quarter of the observed animals. Adding 'bird' to the existing class corresponding to 'mammal' yields the best possible partition. Since this node is a leaf in snapshot (b), incorporation of 'bird' involves expanding the leaf to accommodate the new object, as well as the previously classified one.

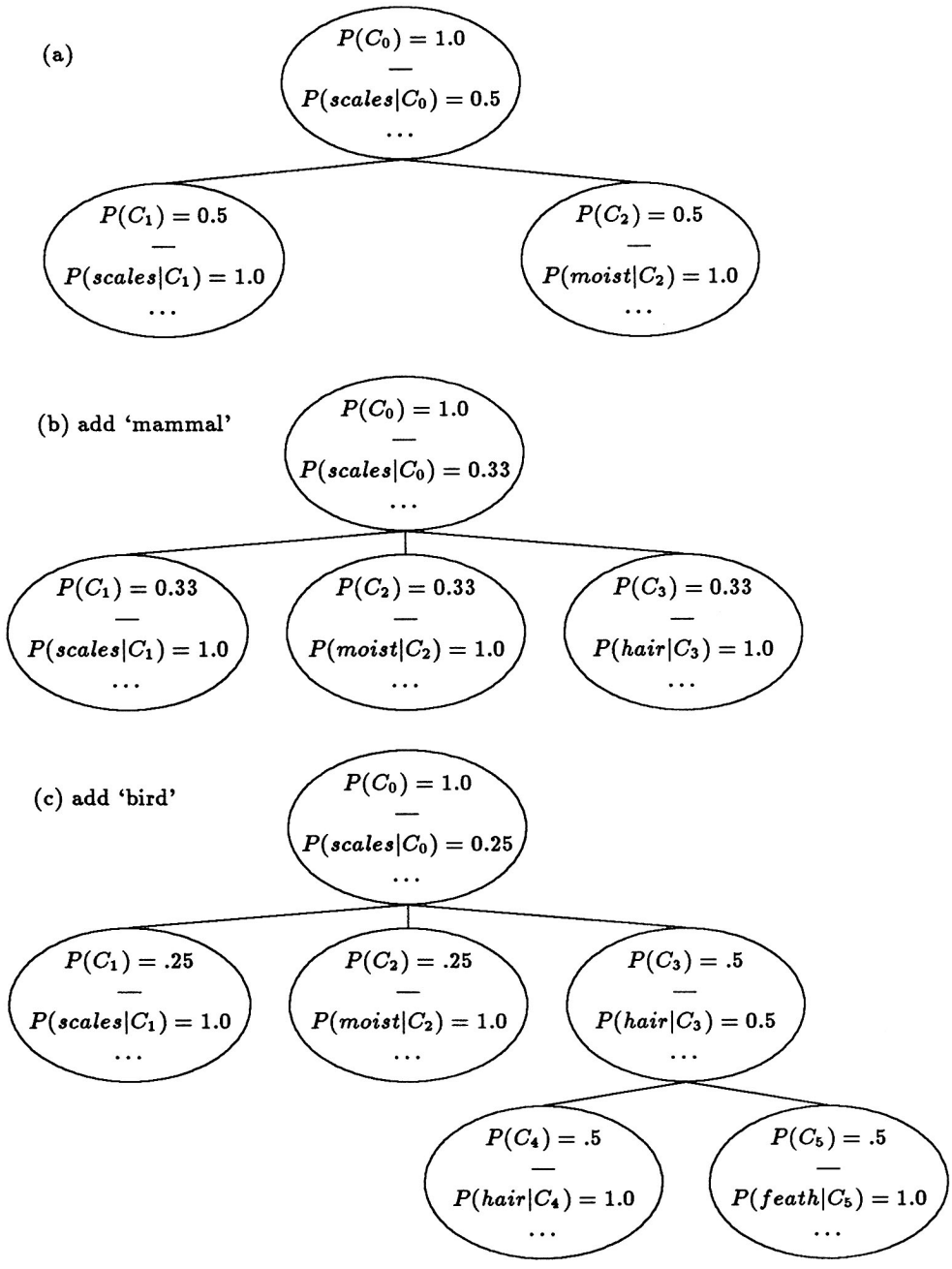


*Figure 3.* Adding 'mammal' and 'bird' to an existing classification tree. Each node represents an object class, $C_i$, that is summarized by a set of probabilities, $P(\text{value}|C_i)$.

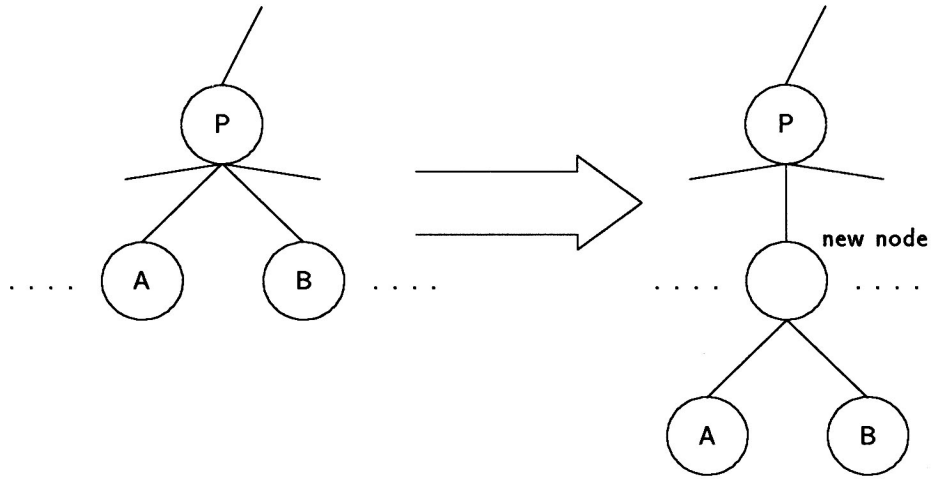


*Figure 4.* The effect of node merging.

Figure 3 demonstrates how a concept hierarchy is constructed over sequential observations and how distributions change to reflect increasing information. While the figure shows probabilities at each node, recall that they are actually computed from two integer counts. Stored at each node is a count of the number of objects classified under the node. Additionally, each attribute-value entry includes an integer count of the number of objects classified under the node possessing that value. Probabilities are computed on demand for evaluation purposes, but it is the underlying counts that are updated.

### *3.3.4 Operators 3 and 4: Merging and splitting*

While operators 1 and 2 are effective in many cases, by themselves they are very sensitive to the ordering of initial input. To guard against the effects of initially skewed data, COBWEB includes operators for node *merging* and *splitting*. Merging takes two nodes of a level (of $n$ nodes) and 'combines' them in hopes that the resultant partition (of $n - 1$ nodes) is of better quality. Merging two nodes involves creating a new node and summing the attribute-value counts of the nodes being merged. The two original nodes are made children of the newly created node as shown in Figure 4. Although merging could be attempted on all possible node pairs every time an object is observed, such a strategy would be unnecessarily redundant and costly. Instead, when an object is incorporated, only the *two* best hosts (as indicated by category utility) are considered for merging.

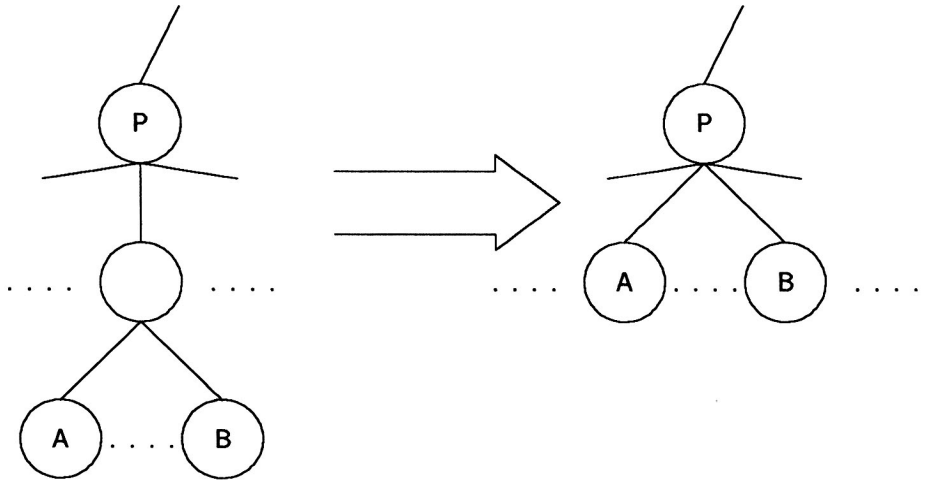


*Figure 5.* The effect of node splitting.

An example of merging occurs when adding a second instance of 'fish' from Table 1 to the last tree in Figure 3; nodes $C_1$ and $C_2$ are identified as the best and second best hosts, respectively. Merging these nodes results in a partition superior to that obtained by incorporating the object in the best host.

As with node merging, splitting may increase partition quality. A node of a partition (of $n$ nodes) may be deleted and its children promoted, resulting in a partition of $n + m - 1$ nodes, where the deleted node had $m$ children as shown in Figure 5. Splitting is considered only for the children of the best host among the existing categories.

Node merging and splitting are roughly inverse operators and allow COBWEB to move bidirectionally through a space of possible hierarchies. Splitting can be invoked to undo the effects of a prior merging should conditions change and vice versa. In general, merging is invoked when initial observations suggest that the environment is a space of highly similar objects, relative to the actual structure of the environment suggested by subsequent observations. Splitting is invoked when the environment is more 'compressed' than suggested by initial input. Merging and splitting decrease the sensitivity of COBWEB to input ordering due to their inverse relation.[4]

[4]Space has not permitted discussion of a fifth operator, node *promotion*, that allows selectively promoting a single node without splitting its parent (Fisher, 1987).

*Table 3.* The control structure of COBWEB.

```
FUNCTION COBWEB (Object, Root ( of a classification tree ))
1) Update counts of the Root
2) IF Root is a leaf
   THEN Return the expanded leaf to accommodate the new object
   ELSE Find that child of Root that best hosts Object and
        perform one of the following
        a) Consider creating a new class and do so if appropriate
        b) Consider node merging and do so if appropriate and
           call COBWEB (Object, Merged node)
        c) Consider node splitting and do so if appropriate and
           call COBWEB (Object, Root)
        d) IF none of the above (a, b, or c) were performed
           THEN call COBWEB (Object, Best child of Root)
```

#### *3.3.5 COBWEB's control structure*

Table 3 summarizes the control strategy that COBWEB uses to organize its learning operators. Using this strategy, the system tends to converge on classification trees in which the first level (the root is the 'zeroth' level) is the optimal partition (with respect to category utility) over the entire object set. For example, over the objects of Table 1, COBWEB consistently converges on the tree of Figure 2. However, as the example for node merging indicates, some orderings may require more than one instance of the same object to converge. Furthermore, while merging and splitting desensitize the system to the effects of initial input ordering, *all* hill-climbing approaches are susceptible to problems related to initial input ordering. This limitation and other matters are discussed in the following section.

## 4. Evaluation of COBWEB

This section evaluates COBWEB with respect to Dietterich's learning model. The model posits three elements that surround learning: the knowledge base, the performance element, and the environment. First, the general form of COBWEB classification trees (knowledge base) is reviewed by examining an anecdotal domain. Next, the utility of this acquired knowledge for inference (performance task) is examined in the domain of soybean diseases. The section concludes with an investigation of COBWEB's effectiveness as an incremental learner (environment).

### 4.1 More on classification trees and concepts

Given a sequence of objects, COBWEB forms a classification tree that summarizes and organizes those objects. For example, given the animal

*Table 4.* Descriptions of two categories from congressional voting tree.

| | $N_1$ ('conservative') $[P(value\|N_1), P(N_1\|value)]$ | $N_2$ ('liberal') $[P(value\|N_2), P(N_2\|value)]$ |
|---|---|---|
| *Normative values* | Toxic Waste – yes [0.81, 0.90]<br>Budget Cuts – yes [0.81, 0.81]<br>SDI reduction – no [0.93, 0.88]<br>Contra Aid – yes [0.88, 0.88]<br>Line-Item Veto – yes [0.91, 0.90]<br>MX Production – yes [0.90, 0.95] | Toxic Waste – no [0.88, 0.78]<br>Budget Cuts – no [0.90, 0.78]<br>SDI reduction – yes [0.83, 0.90]<br>Contra Aid – no [0.83, 0.83]<br>Line-Item Veto – no [0.86, 0.88]<br>MX Production – no [0.93, 0.87] |

descriptions of Table 1, the system formed the tree of Figure 2 along with a probabilistic concept for each node. A domain that illustrates COBWEB's use of probabilistic concepts is congressional voting records. Members of the U.S. Senate were represented by 14 key votes taken in 1985.[5] These votes ranged from domestic issues, such as emergency farm credits, gun control, and school prayer, to foreign affairs issues like Nicaraguan 'Contra' aid. Party affiliations (democrat and republican) were not included in the representation. Each attribute corresponded to one of the 14 votes, with each attribute having two possible values, 'yes' or 'no'.

Based on these data, COBWEB formed a classification tree that partitioned senators at the top level into groups corresponding roughly to 'liberals' and 'conservatives'. Democrats predominantly inhabited the 'liberal' cluster and republicans dominated the 'conservative' cluster. Table 4 shows these nodes with a number of predictable and predictive values. Lower level nodes further distinguish these top-level groups. For instance, one node under the 'conservative' cluster contains eight of the ten democrats classified in the 'conservative' cluster. This smaller group corresponds roughly to the concept of 'southern democrat' or alternatively 'Jackson democrat',[6] which differed from other members of the 'conservative' group by opposing budget cuts (with probability 0.92).

An important feature of the congressional classification tree is that none of the 'conservative', 'liberal', and 'southern democrat' clusters contained any perfectly common values. However, several values occurred with high probability and collectively represent strong tendencies of their respective classes. In general, a number of authors (Hanson & Bauer, 1986; Rendell, 1986; Smith & Medin, 1981) point out that many object classes are more

[5]Votes were designated as key votes by the *Congressional Quarterly*, 1985, pp. 6–15, and not selected by the author. This domain was inspired by similar data used by Lebowitz (1986b).

[6]Seven of eight senators were from southern states. A 'Jackson' democrat is one who votes conservatively on military/foreign policy issues and liberally on social issues.

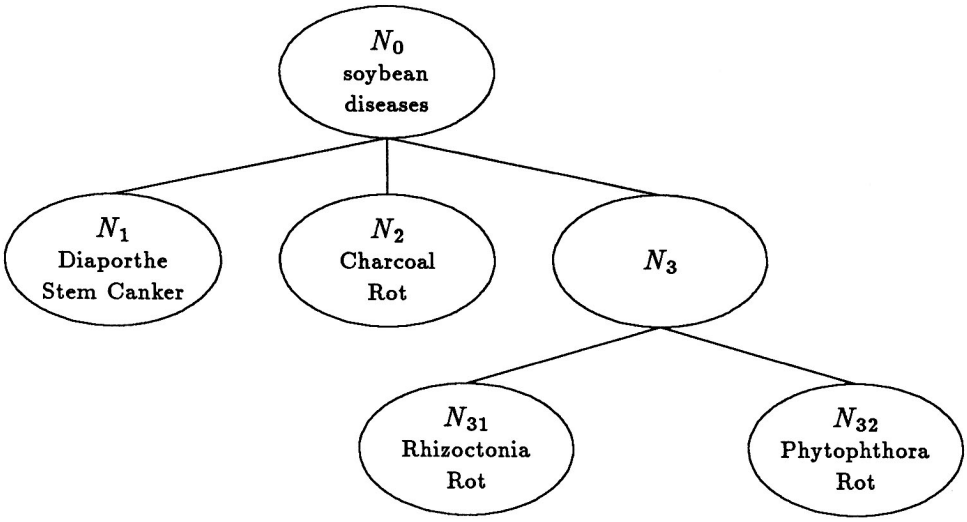


*Figure 6.* Classification tree of soybean case histories.

amenable to a probabilistic representation than to a logical one; important conditions may only be true with high probability.

Concept properties that can be inferred with high confidence go by a number of names, including *normative* (Kolodner, 1983) and *predictable* (Lebowitz, 1982). In Kolodner's (1983) CYRUS, normative values are those that are present with a probability greater than a constant threshold (e.g., 0.67). COBWEB takes a different approach to identifying 'norms' (Fisher, 1987) by looking at a tradeoff between value predictability and predictiveness. Briefly, a value may only be normative at nodes that render it (approximately) conditionally independent of other attribute values. This approach generalizes constant threshold strategies, since an attribute with a highly probable value (e.g., 0.67) will tend to approximate independence from other attributes; this is trivially true when the probability is 1.0. The values in Table 4 are examples of normative values that were identified by COBWEB. These values can be viewed as *default* values (Brachman, 1985) with probabilistic qualifiers.

Normative values allow for a compact summary of category information and provide a link between probabilistic and symbolic representations. In this regard, Hanson and Bauer (1986) suggest the use of *polymorphic concepts* to delineate category membership. A polymorphic set of normative conditions is true if some number of the total set are true. For example, the polymorphous concept {Contra Aid = yes, LineItem Veto = yes, Gramm-

*Table 5.* Descriptions of the 'Charcoal Rot' cluster from the soybean tree. The predictability and predictiveness for each normative value are given in brackets, i.e., [$P$(value|Charcoal Rot), $P$(Charcoal Rot|value)].

| | $N_2$ ('Charcoal Rot')<br>[$P(value\|N_2)$, $P(N_2\|value)$] |
|---|---|
| *Normative values* | Precipitation = below-normal [1.0, 1.0]<br>Temperature = above-normal [0.60, 1.0]<br>Stem-cankers = absent [1.0, 1.0]<br>Fruit-pod-condition = normal [1.0, 0.50]<br>Canker-lesion-color = tan [1.0, 1.0]<br>Outer-stem-decay = absent [1.0, 0.48]<br>Internal-stem-discoloration = black [1.0, 1.0]<br>Sclerotia-internal-external = present [1.0, 1.0] |

Rudman = yes} can be used to classify senators as 'conservatives' if two of the three conditions are true. In Michalski and Stepp's (1983) terms, polymorphic rules can supply *simple* and tightly *fitting* representations of data.

COBWEB forms classes that may be best represented probabilistically. However, such representations do not preclude compact and understandable concept descriptions. In fact, while classes of the congressional domain exhibited no necessary and sufficient conditions, such classes arise in many domains. This is demonstrated in a domain of soybean case histories. COBWEB was tested in a domain of 47 soybean disease cases taken from Stepp (1984). Each case (object) was described along 35 attributes. An example case history is {Precipitation = low, Temperature = normal, ..., Root-condition = rotted}. Four categories of soybean disease were present in the data – Diaporthe Stem Canker, Charcoal Rot, Rhizoctonia Root Rot, and Phytophthora Rot – but for this experiment they were not included in object descriptions. When COBWEB was run on the instances the four classes were 'rediscovered' as nodes of the resultant classification tree shown in Figure 6.[7]

Normative values for the 'Charcoal rot' class are given in Table 5. An important characteristic of this domain, not observed in the congressional domain, is that many classes arise that can be described by necessary and

[7]Stepp's (1984) CLUSTER system also rediscovered the disease classes. However, Stepp's system is search-intensive and nonincremental; its results are independent of input order. On the other hand, COBWEB is order dependent and required three iterations through the data before converging on the reported tree. Issues related to convergence are described in 4.3.

sufficient values. For example, Charcoal rot has a total of seven necessary values (i.e., $P$(value|Charcoal rot) = 1.0), six sufficient values (i.e., $P$(Charcoal rot|value) = 1.0), and five necessary *and* sufficient values. Classes with necessary and sufficient conditions naturally emerge from the more general process of looking for classes with predictable and predictive values.

### 4.2 The utility of classifications for inference

A central claim of this paper is that classification structures produced by conceptual clustering systems, particularly COBWEB, are useful for making inferences. As described earlier, the current system employs an evaluation function that prefers categories from which more can be predicted. The efficacy of this domain-independent heuristic requires that important properties be dependent on regularities or 'hidden causes' (Cheng & Fu, 1985; Pearl, 1985) in the environment. This section describes experimentation aimed at verifying COBWEB's ability to uncover such regularities, thereby improving prediction.

COBWEB's utility for inference was demonstrated by running the system on instances from the soybean domain a second time. However, in these experiments diagnostic condition was included in each object description, though it was simply treated as a 36th attribute. In building a classification tree, diagnostic condition was not used to classify objects as it would in learning from examples.

After incorporating every fifth instance, the remaining **unseen** cases were classified (but not incorporated) with respect to the classification tree constructed up until that point. Thus, in the tradition of Quinlan (1983), the input was implicitly divided into *training* and *test* sets. The goal was to determine if induction over the training set improved inference over the test set through a process of classification. Classification was performed in a manner similar to incorporation, except that statistics were not updated, nor were merging, splitting, or class creation performed. Test instances being classified contained no information regarding diagnostic condition, but the value of this attribute was inferred as a byproduct of classification. Specifically, classification terminated when the test object was matched against a leaf of the classification tree. The leaf represented the previously observed object that best matched the test object. The diagnostic condition of the test object was predicted to be the corresponding condition of the leaf.

Figure 7 gives the results of the experiment. The graph shows that after five instances the classification tree could be used to correctly infer diagnostic condition (over the remaining 42 unseen cases) 88% of the time. After

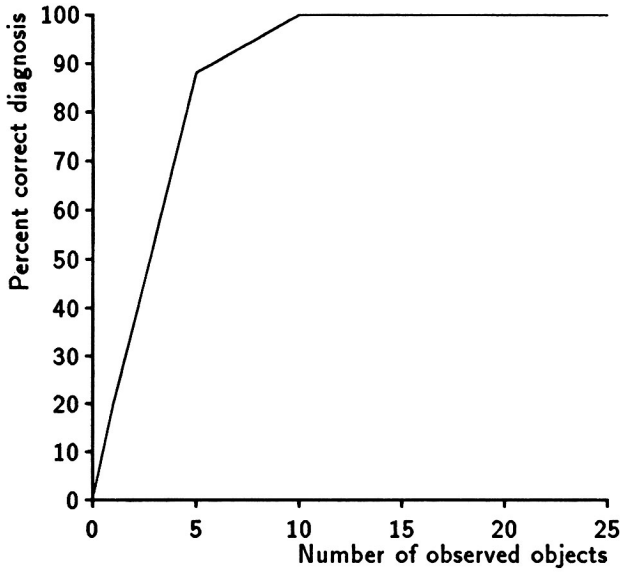


*Figure 7.* Diagnostic success with soybean cases.

ten instances, 100% correct diagnosis was achieved and maintained for the rest of the run. While impressive, these results follow from the regularity of this domain. COBWEB's 'rediscovery' of expert-defined diagnostic conditions in section 4.1 indicates that diagnostic condition participates in a network of attribute correlations. Forming classes around these correlations is rewarded by category utility, resulting in classes corresponding to the human-defined diseases.

The success at inferring the diagnostic condition suggests a relationship between an attribute's dependence on other attributes and the utility of COBWEB classification trees for induction over that attribute. To further characterize this relationship, the induction test was repeated for each of the remaining 35 attributes. The results of these tests (including diagnostic condition) were averaged over all attributes and are presented in Figure 8. On the average, correct induction of attribute values for unseen objects reaches 87% after COBWEB has observed one half of the soybean case histories.[8]

To put these results into perspective, Figure 8 also graphs the averaged results of a simpler, but reasonable inferencing strategy. This 'frequency-based' method dictates that one always guess the most frequently occurring

[8]Recall that this is an *induction* task. As would be expected, when classifying previously observed objects, correct prediction of missing attribute values is nearly 100%.

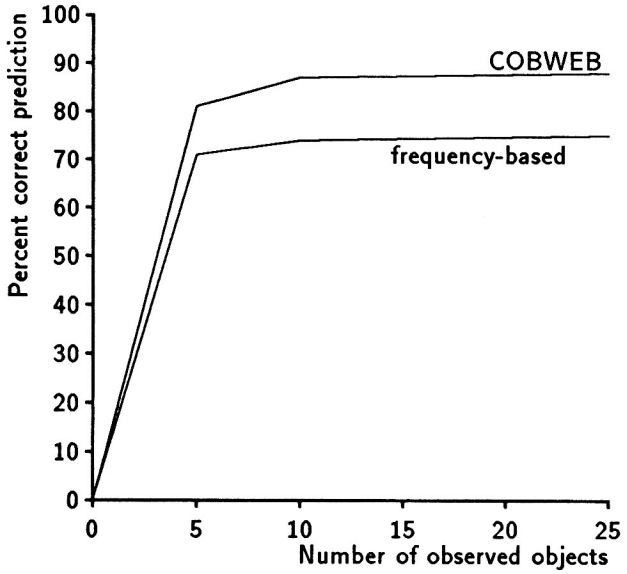


*Figure 8.* Success at inference averaged over all soybean attributes.

value of the unknown attribute. Averaged results using this strategy level off at 74% correct prediction, placing it at 13% under the more complicated classification strategy. However, these averaged results do not tell the whole story – the primary interest is in determining a relationship between attribute correlations and the ability to correctly infer an attribute's value using COBWEB's classification trees.

To characterize the relationship between attribute dependence and inferencing ability, it is necessary to introduce a measure of attribute dependence. The dependence of an attribute $A_M$ on other attributes $A_i$ can be defined as:

$$\frac{\sum_i \sum_{j_i} P(A_i = V_{ij_i}) \sum_{jM} [P(A_M = V_{Mj_M} | A_i = V_{ij_i})^2 - P(A_M = V_{Mj_M})^2]}{|\{i | A_i \neq A_M\}|} \qquad 4-1$$

This function is derived in much the same way as category utility, but it measures the average increase in the ability to guess a value of $A_M$ given the value of a second attribute, $A_i \neq A_M$. $V_{ij_i}$ signifies the $j_i$th value of attribute $A_i$. If $A_M$ is independent of all other attributes, $A_i$, then equation 4–1 equals 0 since $P(A_M = V_{Mj_M} | A_i = V_{ij_i}) = P(A_M = V_{Mj_M})$ for all $A_i$, and thus $P(A_M = V_{Mj_M} | A_i = V_{ij_i})^2 - P(A_M = V_{Mj_M})^2 = 0$.

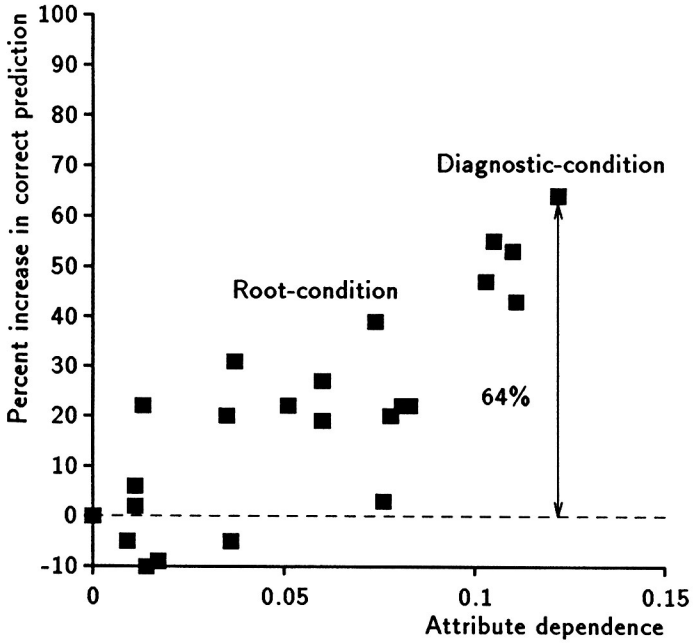


*Figure 9.* Increase in correct soybean inference as a function of dependence.

Figure 9 shows the increase in correct prediction afforded by COBWEB's classification tree (after 25 instances) over the frequency-based method as a function of attribute dependence. Each point on the scatter graph represents one of the 36 attributes used to describe soybean cases. There is a significant positive correlation between an attribute's dependence on other attributes and the degree that COBWEB trees improve inference.[9] For example, diagnostic condition is most dependent on other attributes and it is also the attribute for which prediction benefits most from classification. For this attribute, the frequency-based approach results in 36% correct prediction compared to 100% correct prediction using the tree generated by COBWEB (i.e., 64% difference). Prediction of attributes that approximate independence from other attributes does not benefit from classification and in the case of four attributes it is less effective than the frequency-based approach. The important methodological point to be gleaned from this analysis is that prediction tasks vary in difficulty and COBWEB has been characterized with respect to domains that span a spectrum from difficult to easy. In general, little attention has been paid to Simon's (1969) point

[9]The Pearson product-moment coefficient is 0.88, indicating a highly significant correlation.

that domains must be characterized before the advantage of a learning system can be evaluated.

In addition to the soybean domain, tests were conducted using a domain of 150 thyroid patient case histories (Fisher, 1987) and a similar pattern of correct prediction was exhibited.[10] In general, experiments indicate that COBWEB is a viable means of organizing observations to support inference.[11] The utility of classifications is assured in domains where there are data dependencies involving important attributes (e.g., diagnostic condition). In this light, the performance task associated with COBWEB can be contrasted with that of learning from examples. While a learning from examples system seeks to maximize inference with respect to a single, teacher-selected 'attribute' (e.g., diagnostic condition), COBWEB seeks to maximize a probabilistic average across all attributes. The predictability of an attribute appears to rise roughly in proportion to the degree it participates in intercorrelations. Attributes that participate in few interdependencies come to be treated as irrelevant. The effectiveness of this approach stems from observations that real-world domains tend to exhibit significant degrees of data dependence (Mervis & Rosch, 1981).

Before moving on, it is important to point out that effective prediction is not restricted to classification schemes that employ probabilistic concept representations or that were formulated using category utility. Undoubtedly, many functions represent a tradeoff of attribute-value predictiveness and predictability that operationalizes the performance criterion of maximizing inference ability. These may even be functions applied to logical concept representations. As Medin, Wattenmaker, and Michalski (1986) point out, predictiveness and predictability are continuously-valued analogs of logical sufficiency and necessity, respectively. Category utility can be viewed as a continuously-valued analog of quality measures used with logical representations, most notably CLUSTER/2's measures of *simplicity* and *fit* (Michalski & Stepp, 1983). As discussed earlier, using category utility tends to result in concept descriptions that are compact and understandable. Conversely, it is likely that a tradeoff between simplicity and fit will reward classification schemes with greater utility for inference.

While many domains exhibit significant regularity, the ability to uncover this regularity may be hindered by a number of factors. One difference between COBWEB and other conceptual clustering systems is that it is incremental. The inability to examine all instances simultaneously can

[10] These data were kindly supplied by J. R. Quinlan.

[11] Experiments (Fisher, 1987) indicate that classification need not proceed all the way to leaves as with the experiments reported here. Instead, using normative or default values (briefly described earlier), classification need only proceed to about one tenth the depth of the tree on average to obtain comparable inference results.

be a significant barrier to learning. The following discussion explores the impact of incremental processing.

### 4.3 COBWEB as an incremental system

COBWEB is an incremental conceptual clustering system that can be viewed as hill climbing through a space of classification trees. The program does not adopt a purely agglomerative or divisive approach, but uses divisive (splitting) as well as agglomerative (merging) operators in tree construction. Schlimmer and Fisher (1986) propose three criteria for evaluating incremental systems that were inspired by Simon (1969). These criteria (adapted for conceptual clustering) are:

- the *cost of incorporating* a single instance into a classification tree,
- the *quality* of learned classification trees, and
- the *number of objects* required to converge on a stable classification tree.

They argue that for incremental systems in general, incorporation cost should be low, thus allowing real-time update. However, this may come at the cost of learning lower quality classifications and/or requiring a larger sample of objects to find a good classification than a similarly biased non-incremental, search-intensive method. This section evaluates COBWEB using these criteria and verifies it to be an economical and robust learner.

#### *4.3.1 Cost of assimilating a single object*

To compute the cost of incorporating an object into an existing classification tree, assume that $B$ is the average branching factor of the tree and that $n$ is the number of previously classified objects, so that $log_B n$ approximates the average depth of a leaf. Also, let $A$ be the number of defining attributes and let $V$ be the average number of values per attribute. In comparing an object to a node during classification, the appropriate counts of the node are incremented and the *entire set* of children to which the node belongs is evaluated by category utility. This costs $O(BAV)$ and must be made for each of the $B$ children (on the average). Therefore, comparing an object to a set of siblings requires $O(B^2AV)$ time. In addition to testing an object with respect to existing nodes, class creation is evaluated (one comparison), as is merging (one comparison), and splitting ($B$ comparisons). These costs are additive so that $O(B^2AV)$ remains a legitimate approximation of comparison cost.

In general, classification proceeds to a leaf, the approximate depth of which is $log_B n$. Thus, the total number of comparisons necessary to incorporate an object is roughly

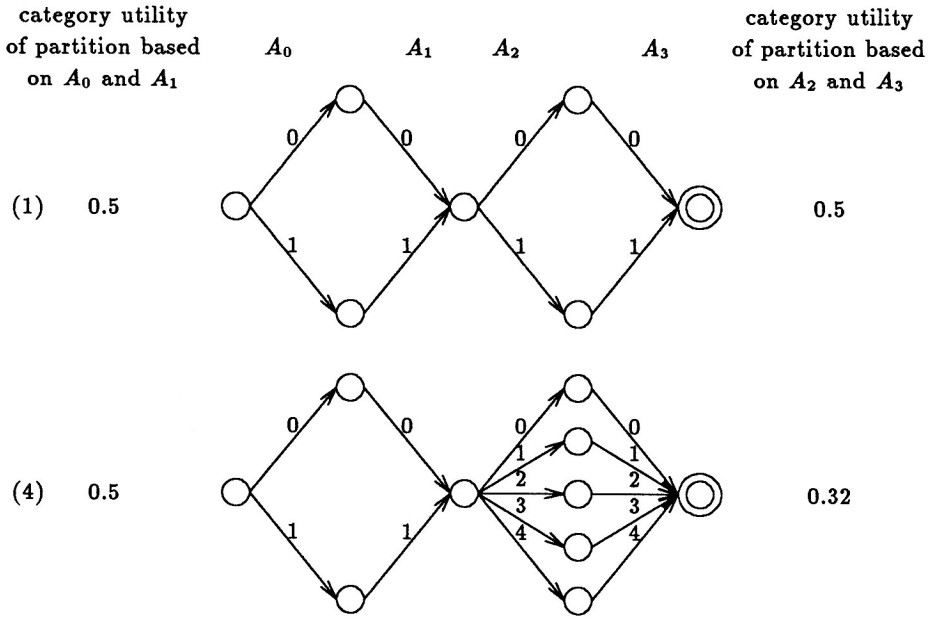


*Figure 10.* Artificial domains with global and local optima.

$$\text{cost} = O(B^2 \log_B n \times AV).$$

In COBWEB, the branching factor is dependent on regularity inherent in the environment; it is not bounded above by a constant as in CLUSTER/2 (Michalski & Stepp, 1983), or by the average number of values per attribute as in RUMMAGE (Fisher, 1985) and DISCON (Langley & Sage, 1984). This makes an exact upper bound on update cost difficult to come by analytically. However, tests in a variety of domains indicate that the branching factor of trees produced by COBWEB range from two to five. This agrees with Michalski and Stepp's (1983) intuition that most 'good' classification trees have small branching factors and lends support to bounding the branching factor in their system. In any case, the cost of adding a single object in COBWEB is significantly less expensive than rebuilding a classification tree for a new object using a search-intensive method such as CLUSTER/2. Polynomial or exponential cost will be associated with any search-intensive clustering system.

### *4.3.2 The quality of classifications*

COBWEB differs from other incremental systems like UNIMEM and CYRUS in that it explicitly seeks classification trees in which the first

level is optimal with respect to a measure of clustering quality. Although node splitting and merging reduce the sensitivity of COBWEB to initial sample skew and aid convergence on optimal partitions, *all* hill-climbing approaches are susceptible to becoming trapped in local optima. While natural domains were appropriate for earlier discussion, the pliability of artificial domains make them better suited for demonstrating the range of a system's behavior. Thus artificial domains were used to test COBWEB's ability to converge on optimal partitions with respect to category utility.

COBWEB was tested in four artificial domains in which globally and locally optimal partitions existed. The difference between the quality of the global and local optimum (in terms of category utility) was systematically varied from domain to domain. Figure 10 gives state machines representing two of these domains: the domain with the least (domain 1) and greatest (4) disparity between the global and local optima. Each state machine represents a domain whose objects it recognizes. Objects in each domain are represented by attributes $A_0$ through $A_3$. For example, one member of domain 4 is $\{A_0 = 0, A_1 = 0, A_2 = 4, A_3 = 4\}$. A state machine representation was chosen because it provides a compact and pictorial view of the correlation between attributes. In each domain, the optimal partition is a segregation based on the values of attributes, $A_0$ and $A_1$, including domain 1 in which there is a tie. Partitioning based on attributes $A_2$ and $A_3$ form a partition of lesser quality: a local optimum.

COBWEB was run 20 times on random samples of 50 objects for each domain. Figure 11 shows the results. The vertical axis is the percentage of runs in which the optimal partition was discovered, while the horizontal axis represents the distance between the category utility of the optimal and local optima normalized to lie in $[0, 1]$.[12] The graph indicates that as the distance between global and local partitions grows, the possibility of becoming trapped in local optima rapidly diminishes. The system's inability to converge on optimal partitions in extreme cases is a direct result of its hill-climbing strategy; a search-intensive method would typically discover the optimal partition in all situations. However, since category utility measures the degree to which a partition promotes correct prediction of object properties (i.e., the expected number of correctly guessed attribute values), COBWEB finds the global optimum when it is most important to do so (i.e., when there is most at stake in terms of correct inference). The program will stumble into local optima only when there is little lost in terms of inference ability.

[12]Distance was normalized by taking the optimal score, subtracting by the local score, and dividing by the optimal score. A normalized score of zero indicates the 'global' and 'local' optimum are tied, while a score of one indicates there is only one optimum or peak in the domain.

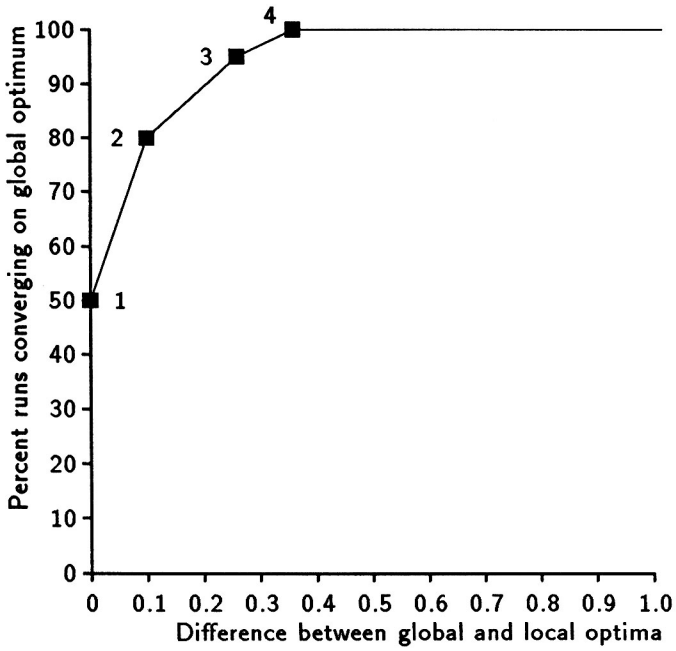


*Figure 11.* Convergence on optimal partitions.

### *4.3.3 Number of objects required for convergence*

Experiments indicate that COBWEB tends to converge on classification trees in which the first level is optimal. In this section, COBWEB is discussed in terms of a third criterion for evaluating incremental methods – the number of objects required to converge on a 'stable' partition. Again, four artificial domains were used to examine the program's behavior. These domains systematically differed in terms of the quality of the optimal partition. The two domains of least and greatest quality are pictured in Figure 12. The domains were selected so that the optimal partition of each domain is unambiguous and easily visualized, but the ease with which it can be discerned with incremental object presentation varies across domains.

COBWEB was run on five random orderings of objects from each domain. During learning, 100 random objects were intermittently selected from the domain and were classified using the classification tree formed thus far by COBWEB. If the top-most partition segregated the sample in the same manner as the optimal partition of the environment as a whole, the two partitions were regarded as equivalent. Figure 13 shows the results of this experiment. As the quality of the optimal partition grows, fewer

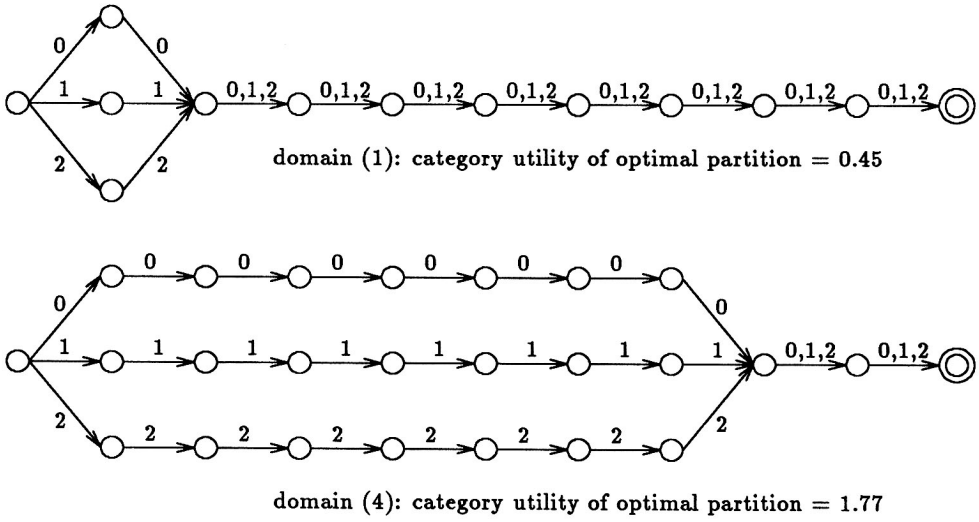


*Figure 12.* Artificial domains with various optimal partition values.

objects are required to find the optimal partition. Inversely, as quality decreases, the number of objects required for convergence appears to exponentially accelerate upwards. Although a search-intensive method would probably also increase, the rate of acceleration would likely be shallower.

While COBWEB may require many objects to stabilize on a partition, it appears to converge rapidly in domains exhibiting significant regularity, as indicated by higher category utility scores for the optimal partition. To put some of the previously examined domains in context, the partition (i.e., first level of the classification tree) formed for the congressional domain measured 1.20, the soybean domain measured 1.50, and the thyroid domain measured 0.50. While these scores literally represent a function of the expected number of attributes that can be correctly predicted, informally they indicate the degree of attribute interdependencies. Given the graph in Figure 13, the category utility values inherent in these natural domains provide further evidence for their learnability by COBWEB.

In summary, experiments indicate that COBWEB is a cost effective means of learning classification trees; update cost is low and convergence time and hierarchy quality gracefully degrade with decreasing environmental regularity. However, in some ways the system may be too extreme in allowing only one object to be incorporated at a time. For example, Hanson and Bauer (1986) describe an incremental method that allows a specified number of objects to be simultaneously considered. While raising update cost (presumably by a constant factor), this generalized approach

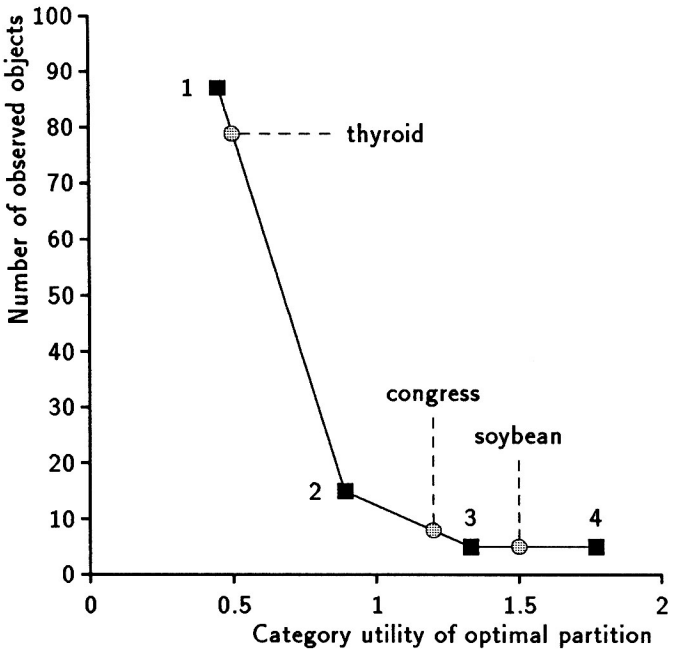


*Figure 13.* Number of objects required to converge.

may improve convergence time and hierarchy quality in cases of little environmental regularity.

## 5. Concluding remarks

COBWEB is a conceptual clustering system that is incremental, economical, and robust. Furthermore, classifications produced by the system are effective tools for inference. While the system uses an evaluation function consistent with preferences in human categorization (Gluck & Corter, 1985), it should not be regarded as a cognitive model, but as a general-purpose clustering method. COBWEB explicitly seeks to optimize its classification tree with respect to category utility, and it does not show how such preferences emerge as the result of more primitive measures. However, Fisher (1987) describes an offspring of this system that can be viewed as a cognitive model and that accounts for certain psychological phenomena, including basic-level and typicality effects.

From a machine learning standpoint, this research has been greatly influenced by work in conceptual clustering, especially Michalski and Stepp

(1983), Fisher and Langley (1985, 1986), and other concept formation systems, particularly UNIMEM (Lebowitz, 1982). COBWEB seeks classifications that maximize a heuristic measure (as in conceptual clustering systems) and uses a search strategy abstracted from incremental systems such as UNIMEM and CYRUS. Describing COBWEB in terms of search has motivated an evaluation of its behavior with respect to the cost and quality of learning. Importantly, this analysis is probably extendable to UNIMEM and CYRUS and it seems likely that they share many of COBWEB's strengths and limitations.

Hopefully, this discussion will lead to comparative studies, something that has been lacking in previous reports. This paper has carefully avoided claims of superiority with respect to related systems. In fact, Fisher (1987) discusses reasons why several systems might perform better along some dimensions (e.g., prediction). Rather, this work embraces a methodology for empirically characterizing learning systems. COBWEB's characterization downplays anecdotal evidence, stressing instead the appropriate use of natural and artificial domains and abstract measures for characterizing domains as well as algorithms. This study enumerates dimensions along which comparisons between concept formation systems can be made in the first place.

Future work will focus on rectifying a number of COBWEB's limitations. One limiting aspect is the object description language: nominal attribute-value pairs. One way to relax this constraint is to allow numeric, continuously-valued attributes. Gennari, Langley, and Fisher (in press) have described CLASSIT, a variation on COBWEB that rewards partitions formed around 'dense' value areas of numeric attributes. An alternative approach is employed by Michalski and Stepp (1983). Their system discretizes continuous attribute domains into ranges based on how well they contribute to higher-order conceptual descriptions. A range of values can then be treated like a nominal value.

A second way of relaxing object description constraints is to allow *structured* objects and concepts. Manipulating structured representations is an important prerequisite for applying conceptual clustering methods in sophisticated problem-solving domains. As CLUSTER/2 and UNIMEM served as precursors to COBWEB, CLUSTER/S (Stepp, 1984; Stepp & Michalski, 1986) and RESEARCHER (Lebowitz, 1986a) are likely starting points for work on incremental clustering of structured objects. Work by Vere (1978) on 'clustering' relational productions shows how conceptual clustering methods might be applied to operator descriptions.

Finally, future work will focus on improving COBWEB's hill-climbing search strategy. While the limitations of hill climbing are generally not problematic, there are a number of real-world problems in which these lim-

itations would significantly manifest themselves. One problem is that of tracking changes in the environment (e.g., the change of seasons), which Schlimmer and Granger (1986) have studied in the context of learning from examples. In real-world domains, a concept formation system must be cognizant of changes in the regularity of the environment. Tracking concept drift is equivalent to the problem of dealing with extremely skewed data. Under extreme skew, a hill-climbing strategy results in a classification tree whose utility may irreversibly and progressively degrade. A principled solution to this and similar problems would involve monitoring the effectiveness of classification trees for performance and using this information to trigger global changes in the classification scheme, rather than the local alterations typically made.

Thus, while the impact of conceptual clustering on performance has been discussed, solutions to a number of important problems will involve looking to ways that performance can affect clustering. In terms of Dietterich's learning model (Figure 1), the insight that performance impacts learning remains an unrealized impetus for future work in conceptual clustering and concept formation.

## Acknowledgements

I thank members of the UCI machine learning group for their helpful comments on earlier drafts of the paper, including Pat Langley, Jeff Schlimmer, Dennis Kibler, Rogers Hall, and Rick Granger. Dennis Kibler and Pat Langley gave initial direction to this work and contributed to its early conceptual development. Discussions with Jeff Schlimmer continually suggest new ideas and hone old ones. Dennis Volper suggested the domains of Figure 10. Comments by Ryszard Michalski and other reviewers helped improve the style, understandability, and correctness of the paper.

This work was supported by Contract N00014-84-K-0345 from the Office of Naval Research and a gift from Hughes Aircraft Company.

## References

Brachman, R. J. (1985). I lied about the trees. *AI Magazine*, *6*, 80–93.

Carbonell, J. G., & Hood, G. (1986). The World Modelers Project: Objectives and simulator architecture. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: A guide to current research.* Boston, MA: Kluwer.

Cheeseman, P. (1985). In defense of probability. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 1002–1009). Los Angeles, CA: Morgan Kaufmann.

Cheng, Y., & Fu, K. (1985). Conceptual clustering in knowledge organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *7*, 592–598.

Clancey, W. J. (1984). Classification problem solving. *Proceedings of the National Conference on Artificial Intelligence* (pp. 49–55). Austin, TX: Morgan Kaufmann.

Dietterich, T. G. (1982). Learning and inductive inference. In P. R. Cohen & E. A. Feigenbaum (Eds.), *The handbook of artificial intelligence.* Los Altos, CA: Morgan Kaufmann.

Dietterich, T. G., & Michalski, R. S. (1983). A comparative review of selected methods of learning from examples. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Everitt, B. (1980). *Cluster analysis.* London: Heinemann Educational Books.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, *8*, 305–336.

Fisher, D. H. (1985). *A hierarchical conceptual clustering algorithm* (Technical Report 85-21). Irvine, CA: University of California, Department of Information and Computer Science.

Fisher, D. H. (1987). *Knowledge acquisition via incremental conceptual clustering.* Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.

Fisher, D. H., & Langley, P. (1985). Approaches to conceptual clustering. *Proceedings of the Ninth International Conference on Artificial Intelligence* (pp. 691–697). Los Angeles, CA: Morgan Kaufmann.

Fisher, D., & Langley, P. (1986). Methods of conceptual clustering and their relation to numerical taxonomy. In W. Gale (Ed.), *Artificial intelligence and statistics.* Reading, MA: Addison-Wesley.

Fu, L., & Buchanan, B. G. (1985). Learning intermediate concepts in constructing a hierarchical knowledge base. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 659–666). Los Angeles, CA: Morgan Kaufmann.

Gennari, J. H., Langley, P., & Fisher, D. H. (1987). *Models of incremental concept formation* (Technical Report). Irvine, CA: University of California, Department of Information and Computer Science.

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.

Hanson, S. J., & Bauer, M. (1986). Machine learning, clustering, and polymorphy. In L. N. Kanal & J. F. Lemmer (Eds.), *Uncertainty in artificial intelligence.* Amsterdam: North-Holland.

Kolodner, J. L. (1983). Reconstructive memory: A computer model. *Cognitive Science, 7*, 281–328.

Langley, P., & Carbonell, J. G. (1984). Approaches to machine learning. *Journal of the American Society for Information Science, 35*, 306–316.

Langley, P., Kibler, D., & Granger, R. (1986). Components of learning in a reactive environment. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: A guide to current research.* Boston, MA: Kluwer.

Langley, P., & Sage, S. (1984). Conceptual clustering as discrimination learning. *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence* (pp. 95–98). London, Ontario, Canada.

Lebowitz, M. (1982). Correcting erroneous generalizations. *Cognition and Brain Theory, 5*, 367–381.

Lebowitz, M. (1986a). Concept learning in a rich input domain: Generalization-based memory. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Lebowitz, M. (1986b). Integrated learning: Controlling explanation. *Cognitive Science, 10*, 219–240.

Medin, D. L., Wattenmaker, W. D., & Michalski, R. S. (1986). *Constraints and preferences in inductive learning: An experimental study comparing human and machine performance* (Technical Report ISG 86-1). Urbana, IL: University of Illinois, Department of Computer Science.

Mervis, C. B., & Rosch, E. (1981). Categorization of natural objects. *Annual Review of Psychology, 32*, 89–115.

Michalski, R. S. (1980). Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems, 4*, 219–243.

Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

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

Pearl, J. (1985). Learning hidden causes from empirical data. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 567–572). Los Angeles, CA: Morgan Kaufmann.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Reinke, R., & Michalski, R. S. (1986). Incremental learning of concept descriptions. *Machine intelligence* (Vol. 11). Oxford University Press.

Rendell, L. (1986). A general framework for induction and a study of selective induction. *Machine Learning*, *1*, 177–226.

Sammut, C., & Hume, D. (1986). Learning concepts in a complex robot world. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: A guide to current research.* Boston, MA: Kluwer.

Schlimmer, J. C., & Fisher, D. H. (1986). A case study of incremental concept induction. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). Philadelphia, PA: Morgan Kaufmann.

Schlimmer, J. C., & Granger, R. H. (1986). Beyond incremental processing: Tracking concept drift. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 502–507). Philadelphia, PA: Morgan Kaufmann.

Simon, H. A. (1969). *The sciences of the artificial.* Cambridge, MA: MIT Press.

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

Stepp, R. E. (1984). *Conjunctive conceptual clustering: A methodology and experimentation* (Technical Report UIUCDCS-R-84-1189). Doctoral dissertation, Department of Computer Science, University of Illinois, Urbana.

Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering: Inventing goal-directed classifications of structured objects. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Vere, S. A. (1978). Inductive learning of relational productions. In D. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* Orlando, FL: Academic Press.

Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision.* New York: McGraw-Hill.