Machine Learning, 18, 5–22 (1995)

A Branch and Bound Incremental Conceptual Clusterer

ARTHUR J. NEVINS
Department of Computer Information Systems, Georgia State University, Atlanta, GA 30302

Editor: Tom Dietterich

Abstract. A computer program is described that is capable of learning multiple concepts and their structural descriptions from observations of examples. It decomposes this conceptual clustering problem into two modules. The first module is concerned with forming a generalization from a pair of examples by extracting their common structure and calculating an information measure for each structural description. The second module, which is the subject of this paper, incrementally incorporates these generalizations into a hierarchy of concepts. This second module operates without reference to any underlying representation language and utilizes only the information measure provided by the first module, while employing a branch and bound procedure to search the hierarchy for concepts from which to form new clusters. This ability to search the hierarchy is used as the basis of a hill climbing strategy which has as its goal the avoidance of local peaks so as to reduce the sensitivity of the program to the order in which the observations are encountered.

Keywords: Conceptual clustering, characterization module, information measure, branch and bound, incremental learning, hill climbing

1. Introduction

This paper addresses the problem in machine learning known as conceptual clustering. A sequence of observations is presented which the machine must group into clusters. For each cluster, the machine will produce a conceptual description that represents a generalization of the observations appearing in the cluster. Clusters themselves may serve as raw material out of which still larger clusters may be formed. This can produce a hierarchical organization of clusters with clusters nearer the root of the hierarchy represented by more general descriptions.

Conceptual clustering involves two problems: (1) The *clustering* problem (i.e., the grouping of observations into different clusters and incorporation of the clusters into a hierarchy) and (2) the *characterization* problem (i.e., the determination of a descriptive concept for each such cluster). As Fisher (1987) has pointed out, these two problems cannot be independent in conceptual clustering since the results of characterization must be used to determine the quality of the clusters.

Previous approaches to the clustering problem have depended upon some underlying representation language such as feature vectors (Kolodner, 1983; Lebowitz, 1987; Fisher, 1987; Hanson & Bauer, 1989; Hadzikadic & Yun, 1989), ordered pairs of feature vectors (Cheng & Fu, 1985), unordered groups of feature vectors (Gennari, Langley, & Fisher, 1989), or a modified first-order predicate calculus (Michalski, 1980; Michalski & Stepp, 1983). In each case, the underlying representation language was exploited in both the characterization problem and the clustering problem even though a representation language is essential

only for the characterization problem. The fact that "clustering" and "characterization" are not independent in conceptual clustering does not mean that the solution to the clustering problem must depend upon details of how the characterization problem was solved. It need only depend upon the *results* of characterization.

The computer program HIERARCH described in this paper consists of a characterization module and a clustering module. The approach taken in the design of HIERARCH was to disentangle the characterization and clustering problems from each other by freeing the clustering module from the constraints of *any* representation language. The only requirement is that the solution to the characterization problem (i.e., output of the characterization module) provide an appropriate information measure for any conceptual description it generates. This enables research on characterization and clustering to proceed apart from each other with advances in either of the two problems improving the overall effectiveness of the conceptual clustering algorithm. It is a step toward answering Simon's (1983) call for devising "learning procedures that don't involve knowing the details of the internal languages and programs." The clustering module of HIERARCH is the subject of this paper.

Like some previous conceptual clustering programs such as COBWEB (Fisher, 1987) and UNIMEM (Lebowitz, 1987), HIERARCH learns incrementally (i.e., a hierarchy is continually fashioned as new observations are encountered). HIERARCH is also similar to COBWEB and UNIMEM in that it allows partial matching. Thus, it does not insist that class membership depend upon satisfying an exact set of logical conditions (e.g., a penguin is a bird even if it is not a typical bird). However, unlike COBWEB which achieves partial matching through the use of probabilistic descriptions or UNIMEM which does it by means of parameters that govern the extent to which a mismatch will be tolerated, HIERARCH achieves partial matching by exploiting the information measure produced by the characterization module.

An important aspect of HIERARCH is the way it controls its search for solutions. Like COBWEB and UNIMEM, HIERARCH is a hill-climber that builds its hierarchies in neither an entirely top-down nor bottom-up fashion and has operators for undoing past mistakes by merging and/or deleting nodes. However, like all hill-climbers, it runs the risk of encountering local peaks and consequently some global considerations were introduced into the operators used by HIERARCH.

The ITERATE algorithm of Biswas et al. (1991) attacks this hill-climbing problem in a non-incremental manner by first finding an initial partition and then globally redistributing individual objects so as to improve partition quality. Fisher, Xu & Zard (1992) present evidence that even local redistribution can be superior to global redistribution if the local redistribution involves entire categories of objects and the global redistribution is confined to individual objects. The incremental ARACHNE system of McKusick & Langley (1991) introduces more powerful operators for redistributing categories. These operators will promote a child that resembles its grandparent more than its parent and merge two children that resemble themselves more than their parent.

HIERARCH senses that a child has been misplaced in the hierarchy by analyzing the result of an information measure applied to the child and its parent. However, rather than promote the misplaced child, as in ARACHNE, it makes a more global attempt at redistributing the child by removing it from the hierarchy and then reinserting it as if it were just another

object to be learned. This global attempt at redistribution is done in a manner that does not compromise the incremental nature of the program and is not confined to individual objects, as was the case with ITERATE, but is applied to categories of objects.

Since the willingness to reinsert categories into the hierarchy can necessitate repeated searches of the hierarchy, particular concern was given to the procedure that would be used for conducting the search. Initially, an effort was made to employ a procedure that confined the search to the "Best K" nodes at each level of the search space. The author had employed this procedure successfully for the balancing of assembly lines (Nevins, 1972). The procedure was later rediscovered and given the name "beam search" (Lowerre, 1976). The decision to employ beam search was influenced in part by its success at other tasks including speech recognition (Lowerre, 1976), image analysis (Rubin & Reddy, 1977), and the learning of structural descriptions from examples (Dietterich & Michalski, 1981). However, in the context of the current effort at conceptual clustering it did not prove to be as successful as anticipated and eventually it was abandoned. Instead, a branch and bound procedure was incorporated into HIERARCH which turned out to be more natural and adept at exploiting the information measures.

The method by which HIERARCH constructs its hierarchies is described in Section 2 followed by the application of branch and bound in Section 3. An evaluation of HIERARCH is the focus of Section 4 and this is followed by some concluding remarks in Section 5.

2. Hierarchy building in HIERARCH

HIERARCH uses the information values returned by the characterization module to guide its search through the hierarchy. At the root of the hierarchy is the "universal concept", whose description is empty. Any object is capable of being stored as a child of the universal concept, but that would only be done if there were no other acceptable location in the hierarchy at which the object could be stored.

HIERARCH builds its hierarchy in such a way that as one descends from the root to lower levels of the hierarchy one will encounter nodes where the information measure yielded higher values. The hierarchy building module requires that the information measure return a higher value for a more specific concept (i.e., since a more specific concept has a lower probability of accepting an observation, its very act of accepting an observation is more informative). However, a concept with a higher information value need not be more specific. In particular, we say that C is an *instance* of P if and only if the information measure applied to the generalization of C and P yields a value that is equal to the value produced when the measure is applied to P (i.e., the generalization does not involve a loss of information).

One could have insisted that each node in the hierarchy be more specific than any of its ancestors. However, the assumption of increasing specificity would have made it difficult if not impossible to perform partial matching, since each node then would have needed to meet every condition required of its parent without exception. By insisting instead that a node have a higher information value than its parent, one is saying in effect that the child is "kind of" more specific even if it does not exactly qualify as a special case of its parent. The philosophical position expressed here in HIERARCH is that monotonicity of "information" is more fundamental than monotonicity of "specificity" as one descends the hierarchy.

Although UNIMEM would lose an observation if the observation were a child of a concept being dissolved, it was not its intention to lose track of *all* occurrences of an observation. Both UNIMEM and COBWEB sought to retain all observations and HIERARCH does likewise. Genarri et al. (1989) limited the number of retained observations with the help of a program parameter, but such considerations are beyond the scope of the present paper.

2.1. *Constraints on the placement of an object in the hierarchy*

HIERARCH does not allow a node E to be placed as a child of a node B if the amount of information in the parent B that is not accounted for by the child E exceeds either (1) the amount of information from the parent B that is accounted for by child E or else (2) exceeds the amount of information that the child E possesses in excess of that possessed by its parent B. Thus, rather than allow E to be placed under B on the basis of parameters such as "enough" features in common or feature values sufficiently "close" to each other as in UNIMEM, the placement is controlled by information theoretic considerations.

It should be noted that the acquisition of these information values in no way depends upon the details of the characterization module. In particular, if $I(X)$ represents the amount of information in concept X and $G(Y, Z)$ represents the generalization of Y and Z, then both $I(X)$ and $G(Y, Z)$ can be obtained as outputs from the characterization module. The amount of information from parent B that is accounted for by child E is $I(G(B, E))$, while the amount of information from parent B that is not accounted for by child E is $I(B) - I(G(B, E))$.

For example, suppose each concept or observation is described by a feature vector composed of five binary attributes, which can assume values of 1, 0, or $*$ (unspecified). As an information measure, count the number of specified attributes. Thus, $B = (1, *, 0, 1, 0)$ would have an information value of 4 bits and $E = (1, 0, 1, 1, 0)$ would have an information value of 5 bits. Consequently, of the 4 bits of information specified in B only 3 are accounted for by E, since E conflicts with B in the third feature. This is reflected in the fact that their generalization $G(B, E)$ is $(1, *, *, 1, 0)$, which contains 3 bits of information. However, since E possesses 1 bit of information in excess of B (i.e., 5 versus 4 bits) and E also accounts for 3 bits of the information in B and each of these quantities (i.e., the 1 bit excess and 3 bits accounted for by E) are as great as the 1 bit of information in B that is not accounted for by E, the placement of E as a child of B would be allowed.

By contrast, for $E' = (0, 0, 1, 1, 0)$ the generalization $G(B, E') = (*, *, *, 1, 0)$ represents a loss of 2 bits from B and consequently E' would never be allowed as a child of B, since the 2 bits not accounted for by E' (i.e., $I(B) - I(G(B, E'))$) is greater than the 1 bit of information that E' possesses in excess over B. For $B' = (1, *, *, 0, 0)$, the generalization $G(B', E') = (*, *, *, *, 0)$ also represents a 2 bit information loss. Although E' has the requisite 2 bits of information in excess over B', the placement of E' as a child of B' still would never be allowed, since E' only accounts for 1 of the 3 bits in B' and this is less than the 2 bits in B' that are not accounted for by E'.

2.2. *Search for a best node*

Both concepts, which are internally created by HIERARCH, and observations can serve as nodes in the hierarchy. When HIERARCH incorporates an example E (i.e., observation or concept) into its hierarchy, it will search for a "best node" B to which E is "most strongly" related. Either B will serve as a "host" for E or else B will be "merged" with E. B becomes a host for E if E is added to the hierarchy in such a manner that it is positioned as a child of B (i.e., situated immediately above E in the hierarchy is its parent B). B is merged with E if, after removing B from the hierarchy, the generalization $G(B, E)$ is inserted into the hierarchy either at the position formerly occupied by B or at some other more advantageous position. Regardless of where $G(B, E)$ gets stored, it would serve as the parent of both B and E, which is why this is referred to as a merger.

When searching for the best node B for E, one needs some means of measuring the strength of the relationship between B and E, since the node B of greatest strength will be selected for association with E, either as the parent of E or, in the event that E and B are merged, as a sibling of E. The measure employed by HIERARCH depends upon whether E is being considered as a child of B or as a sibling of B.

If E is being considered for insertion as a child of B, then this measure is the information in B accounted for by E minus the information in B not accounted for by E. Representing the second term as $L(B, E) = I(B) - I(G(B, E))$ this measure becomes $I(G(B, E)) - L(B, E) = I(B) - (I(B) - I(G(B, E)) - L(B, E) = I(B) - 2 * L(B, E)$.

Suppose on the other hand that E is being considered for merger with B rather than insertion under B. HIERARCH requires of the characterization module that any generalization it produces must be such that each of the objects being generalized must be able to account for all of the information in the resulting generalization. Consequently, one possibility for this measure of strength could be simply the information $I(G(B, E))$ contained in the generalization. This is suggested by the fact that the strength associated with the insertion of E under an existing concept B would also be the information of this new parent of E if the child (i.e., E) could account for all the information (i.e., $I(B)$) in its parent.

However, the situation is not quite symmetrical between an insertion under an existing concept and the creation of a new concept by means of the generalization $G(B, E)$, since the latter must itself be inserted at some position in the hierarchy. What had been a good location for B might not be a good location for $G(B, E)$. On the other hand, since B is being carried along with its new parent $G(B, E)$, the final destination of $G(B, E)$ could be a region of the hierarchy that is more conducive to $G(B, E)$ than it is to B. Whereas B might be happy with its new parent $G(B, E)$ it might not be so happy with its new grandparent or with its new "uncles" (i.e., the siblings of $G(B, E)$). For this reason, one does not want to make it too easy for some object like E to come along and through its generalization $G(B, E)$ carry B off to another part of the hierarchy.

The approach taken here is to exact a price from the generalization $G(B, E)$ by deducting from $I(G(B, E))$ an amount based upon information in the children (i.e., B and E) that $G(B, E)$ fails to take into account. One possibility is to deduct from $I(G(B, E))$ the amount $\mathrm{Max}(I(B), I(E)) - I(G(B, E))$ so that the deduction would be based on the child that

suffered more from the generalization. Another possibility is to deduct from $I(G(B,E))$ the amount

$$\begin{aligned}&\mathrm{Max}\big[\mathrm{Min}(I(B),I(E)) - I(G(B,E)),\\ &\quad \mathrm{Max}(I(B),I(E)) - \mathrm{Min}(I(B),I(E))\big],\end{aligned}$$

which attempts to strike a compromise between the child that suffered less and the child that suffered more. Both of these possibilities will be explored in Section 4.

2.3. *Overall control structure*

HIERARCH would search elsewhere to store a new generalization $G(B,E)$ than under the former parent P of B if insertion of $G(B,E)$ under P would violate a constraint of Section 2.1. If it has been decided to search elsewhere for the placement of an object, then that object is put on a special list called the *globalization* list where it awaits subsequent processing. Meanwhile, $G(B,E)$ replaces B as a child of P until its final status is resolved. However, once it has been decided to transfer $G(B,E)$ to some other location in the hierarchy, it is removed as a child of P. If this transfer leaves P with fewer than two children then P is dissolved unless P still possesses a child that is on the globalization list (i.e., no action is taken regarding the dissolution of P if its sole remaining child is still waiting to be processed).

However, eventually any concept possessing fewer than two children gets dissolved, since the original reason for its creation no longer applies (i.e., it no longer would represent a merger). The child of a dissolved concept is not automatically promoted, since its new parent (i.e., its former grandparent) then would contain less information than the parent that was just dissolved. Instead, such a child is placed on the globalization list for subsequent processing.

The overall control structure of the program is summarized in Figure 1. Although not explicitly stated in Figure 1, during the branch and bound search for the best node B for E, a merger of B with E is not allowed if E is currently a child of B and B does not have at least two children other than E. For if such a merger of $G(B,E)$ were allowed, the eventual removal of E as a child of B would leave B with fewer than two children thereby causing B to become dissolved, which in turn destroys the basis for the generalization $G(B,E)$. This still would not preclude B from serving as host for E, since its selection then would confirm that E currently is being stored at the appropriate location.

The reason for bypassing anything on the globalization list GL when searching for a best node B for E is that if E merged with some B on GL, the subsequent processing of B could undo the merger by transferring B away from $G(B,E)$ to some other part of the hierarchy. Furthermore, if E had been allowed to merge even with a descendant of B, the transfer of this merger to another part of the hierarchy could undermine the viability of B by removing this descendant from the scope of B; by not allowing such a merger, the algorithm assures that when B eventually is selected for processing from GL its status is that of a node that has not already been earmarked for dissolution.

The algorithm selects from GL the member with the least amount of information in order to assure that if two objects P and D are on GL and D is a descendant of P, then P will

```
Let GL represent the globalization list, which initially
  consists of just the new observation;
CycleCount := 0;
REPEAT CycleCount := CycleCount + 1;
  E := member of GL with least amount of information;
  IF CycleCount > 1 THEN Q := parent of E;
  Use branch and bound search procedure to find node B whose
     association with E is regarded as having the greatest
     strength, where the search is confined to those nodes
     that are neither on GL nor possess ancestors on GL;
  IF CycleCount > 1 THEN remove E as a child of Q;
  IF B is designated as a host for E
    THEN BEGIN insert E into the hierarchy as a child of B;
         IF CycleCount = 1 THEN add all siblings of E and all
         ancestors of E to GL that are not already on it;
         END
    ELSE
      BEGIN remove B from its parent P;
      insert the generalization G(B,E) as a child of P;
      IF CycleCount = 1
        THEN add all children of P and all ancestors of G(B,E) to
             GL that are not already on it
        ELSE IF insertion of G(B,E) as a child of P is not
                allowed by the constraints of Section 2.1
             THEN add G(B,E) to GL;
      END;
Remove E from GL;
IF CycleCount > 1 THEN
  WHILE Q is not the universal concept and
        has no child on GL and has fewer than two children
    DO BEGIN P := parent of Q;
       Remove Q as a child of P;
       IF Q has exactly one child C
         THEN BEGIN add C as a child of P;
              add C to GL;
              END;
       Q := P;
       END;
UNTIL GL is empty;
```

Figure 1. Overall control structure of HIERARCH.

be processed before D. For if D has been processed before P, and D were transferred to another part of the hierarchy, then its removal from the scope of P could undermine the viability of P.

The key to bringing the iterative process of Figure 1 to a halt is the requirement that, for any node B that was a participant in the merger that created its parent P in the hierarchy, no merger $G(B, E)$ be allowed that does not have a higher measure of strength than that associated with the formation of P. For without such a merger $G(B, E)$ that might carry away the child B of P to another part of the hierarchy, the concept P would have enough children so that it never would get dissolved. However, since the measure of strength associated with a merger cannot continue to increase indefinitely as new mergers are formed, it means that eventually a point must be reached when the population of concepts in the

hierarchy is such that it is no longer possible to dissolve an existing concept P. But since one cannot keep forming mergers indefinitely from a fixed pool of objects without dissolving at least one of the mergers somewhere along the way, the production of new mergers must likewise eventually cease thereby assuring that the iterative process will terminate.

3. The branch and bound search

Since the number of nodes tends to grow exponentially at each successive level of the hierarchy, it should be evident that the practicality of the above procedure depends upon an effective means by which one can search the hierarchy for a best node. The general approach employed here is known as "branch and bound." Branch and bound procedures have been widely used in operations research (e.g., Ignall & Schrage, 1965; Kolesar, 1967), and one such procedure is the basis for the well known Alpha-Beta method in game playing (Knuth & Moore, 1975).

The basic idea behind "branch and bound" is that if an upper bound can be placed on the value a node is capable of producing, then the node can be ignored if this upper bound cannot meet a lower bound required of the best node. Obviously, the trick in employing branch and bound is to devise effective bounds, since tighter bounds enable one to steer clear of more paths in the search space without overlooking the best node. Fortunately, the use of an information measure that increases as one descends the hierarchy turns out to be a natural instrument for exploiting the potentiality inherent in branch and bound.

The search for the best node B for E necessitates generalizing E with various alternative nodes in the hierarchy. However, since forming a generalization is by far the single most time-consuming step, it is critical that the number of these generalizations be kept to a minimum. Consequently, a major function of the branch and bound procedure will be to instruct HIERARCH as to when it should attempt a generalization.

3.1. Finding an upper bound on the candidate generalization

Attached to each node B in the hierarchy are two numbers: the information $I(B)$ associated with node B and an information loss $L(P, B) = I(P) - I(G(P, B))$ = information in the parent P of B that is not accounted for by B.

As described in Section 2.2, the search for a best node B for E is based upon the maximization of the strength of the relationship between B and E, and this strength cannot exceed $I(G(B, E)) = I(B) - L(B, E)$. Obviously, the upper bound $I(G(B, E))$ is of only theoretical interest, since it cannot be obtained without performing the generalization $G(B, E)$, which is precisely what the use of the bound is attempting to avoid. However, it is possible to obtain an upper bound that approximates $I(G(B, E))$ based upon a generalization that took place earlier between E and some ancestor of B during the search through the hierarchy. This approximation, which will be selected as the upper bound, is just $I(B) - ML(B, E)$ where $ML(B, E)$ is a lower bound to $L(B, E)$. This lower bound can be regarded as a minimum information loss derived from the experience of generalizing E with some ancestor of B.

In particular, suppose E already had been generalized with the parent P of B. If B were an instance of P then $ML(B, E)$ would equal $L(P, E)$, since all aspects of P would be present in B and hence any inability of E to capture aspects of P also would mean that E could not capture the same aspects of B. But if B were not an instance of P, then $ML(B, E)$ would need to be tempered by the extent to which some of the inability of E to capture aspects of P might have involved aspects of P that B also could not capture. Consequently, $ML(B, E) = \text{Max}(0, L(P, E) - L(P, B))$.

Even if E had not been generalized with the parent P of B, $ML(B, E)$ would be obtained as $\text{Max}(0, ML(P, E) - L(P, B))$. In other words, when descending the hierarchy, the minimum loss is updated by deducting the failure of a node to capture information in its parent, with $ML(\text{RootNode}, E) = 0$ by definition.

Upon backtracking, HIERARCH also attempts to revise its estimate of $ML(P, E)$ based upon information it uncovered during the search. In particular, if it discovered an information loss $L(B, E)$ associated with a child B of P such that $L(B, E) > ML(P, E)$ then it would like to reset $ML(P, E)$ to $L(B, E)$. Note, however, that some of the loss $L(B, E)$ could have been based on aspects of B that were unique to B (i.e., aspects not present in P). However, the portion that is unique to B could not possibly exceed $I(B)$ minus the information shared by P and B, with this shared amount being $I(G(P, B)) = I(P) - L(P, B)$. Consequently it would reset $ML(P, E)$ equal to $L(B, E)$ minus the amount $I(B) - (I(P) - L(P, B))$ if doing so would raise the value of $ML(P, E)$.

3.2. *Finding lower bounds to narrow the search*

It remains to find at least one lower bound required of a node which, by exceeding the node's potential (i.e., its upper bound UB), can signal that the generalization of the node should be avoided. One obvious choice for such a lower bound is the highest strength of any relationship encountered so far during the search. Denoting this lower bound by the name BEST, there would be no point in generalizing a node unless its generalization stands a chance of producing an information value as high as BEST. If its generalization produces information lower than BEST, it only would be rejected in favor of the node that produced BEST.

There is a serious problem associated with exclusive reliance on BEST as the lower bound. BEST works fine if a good value for it can be found early in the search. However, often one will not be so fortunate and will find oneself with a fairly low value for BEST that easily is exceeded by the UB of many of the nodes encountered during the search. This would be particularly true if the search began in a region that had nothing in common with the object E being incorporated into the hierarchy.

One way around this difficulty is suggested by the fact that such a region of the hierarchy would of necessity be accompanied by substantial information losses when generalizations are attempted. These information losses can be used heuristically as the basis for an additional lower bound by focusing on the most recent ancestor RA for which a generalization had been performed. Prior to such a generalization, the information $I(RA)$ attached to RA could serve as an optimistic information value for the generalization of E with RA. However, once the information loss $L(RA, E)$ is uncovered, it exposes the undue optimism

of $I(RA)$. As seen in Section 3.1, this exerts a depressing effect on the upper bound associated with any descendant B of RA, since it raises the value of $ML(B,E)$ in the formula $I(B) - ML(B,E)$ for the upper bound. As the descent continues down the hierarchy below RA, nodes are encountered whose increasing information values help to raise UB and restore confidence shaken by the information loss associated with the generalization of E and RA.

This suggests that $I(RA)$ be used as an additional lower bound along with BEST. This will postpone generalizing any descendant of RA until a sufficiently large UB for that node makes it likely that the information loss from the generalization of RA can be recovered. Thus, instead of requiring $UB \geq$ BEST, HIERARCH requires that $UB \geq$ MAX(BEST, $I(RA)$).

Furthermore, if generalization of a node B were skipped but backtracking revealed that one of its descendants got generalized, then node B also would be generalized provided node B had more than one other child that had not yet been examined. The rationale for this generalization of node B is that, since one of its descendants already needed to be generalized, a higher information loss (i.e., obtained by generalizing node B) might be needed to avoid generalizing additional descendants of B.

4. Evaluation of HIERARCH

When evaluating hierarchies created by the program, it is desirable to employ some objective measuring instrument rather than rely just on evidence that appears "intuitively satisfying." The measuring instrument employed in this section assumes that each observation in the task domain consists of a single feature vector. Such task domains have been well studied in the literature on conceptual clustering (e.g., Fisher, 1987). The characterization module used for the experiments described in this section produces a generalization of two feature vectors by dropping the values of a feature whenever they disagree. This makes the feature "unspecified" as discussed in Section 2.1.

The motivation for the measuring instrument employed in evaluating HIERARCH is discussed in Section 4.1. The measuring instrument is based on a "recognition index" for measuring the quality by which a given concept is capable of recognizing a given observation. This recognition index is described in Section 4.2 followed by a description of the measuring instrument in Section 4.3. Five experiments involving the use of this measuring instrument are described in Section 4.4 followed by their application to the classification by HIERARCH of soybean diseases, U.S. Senate voting records, and thyroid diseases in Sections 4.5, 4.6 and 4.7 respectively.

4.1. Motivation for the measuring instrument

One possibility would be to proceed as in Fisher (1987) where, after each observation is learned, an attempt is made to predict feature values associated with unseen observations. When the percent of correct predictions was averaged over all features for the same soybean task as in Section 4.5 (including a feature that represented "diagonistic condition"), the

successful prediction rate reported for COBWEB was 87 percent after incorporating the twenty-fifth observation, whereas it was only 74 percent when the prediction was based upon the most frequently observed value for the feature.

The problem with this evaluation, if it were to be used to measure the quality of a hierarchy, is that one can also do very well by basing the predictions on the old observation that comes closest to matching the new observation (i.e., excluding the feature for which the prediction is being made, find the observation that has the greatest number of features with specified values that are in agreement with the new observation and then make the prediction based on the value of the excluded feature for the observation so selected). For example, the soybean experiments described in Section 4.5 involved ten different orders in which the observations were presented. Two of these observation sequences were decidedly non-random in that the observations were presented by disease category (i.e., first all the observations in one category were presented, then all the observations in another category, etc.). Excluding these two non-random sequences, this predictor (i.e., based on "closest match") had a success rate, after examining the twenty-fifth observation, of 88 percent on two occasions, 87 percent on three occasions, 86 percent on two occasions and 85 percent on one occasion.

The difficulty with employing such a measure to evaluate the quality of a conceptual hierarchy is that it would not significantly downgrade a hierarchy that consisted exclusively of observations (i.e., a hierarchy in which there were no concepts and all observations were stored as children of the root node). Instead, the approach taken here will be to evaluate the hierarchy based on how well it recognizes new observations using a 'recognition index' that employs conditional probabilities in the same spirit as Fisher (1987).

4.2. The recognition index

Let $P(C \mid F \cdot j)$ denote the probability that an observation in the hierarchy will be stored under concept C given the fact that the value of its j_th feature is $F \cdot j$. Let $P(F \cdot j \mid C)$ denote the probability that an observation stored in the hierarchy under concept C will have a value $F \cdot j$ for its j_th feature. Let $P(C)$ and $P(F \cdot j)$ be the corresponding unconditional probabilities.

When determining the quality by which a concept C is capable of recognizing a given observation, we will focus on the feature values $\{F \cdot j\}$ associated with the given observation. For each feature value $F \cdot j$ associated with the observation, one would like to take into account both $P(C \mid F \cdot j) - P(C)$ (i.e., how much the feature value adds to the likelihood of the concept) as well as $P(F \cdot j \mid C) - P(F \cdot j)$ (i.e., how much the concept adds to the likelihood of the feature value). From Bayes Theorem, both of these two expressions must possess the same sign, and consequently, if one makes a positive contribution then so will the other. This would suggest taking the product of $P(C \mid F \cdot j) - P(C)$ times $P(F \cdot j \mid C) - P(F \cdot j)$, except that when both make a negative contribution the result would be positive. To prevent this occurrence, the product is multiplied by -1 when each expression makes a negative contribution. This product deals directly with differences between conditional and unconditional probabilities and treats categories C and feature/values $F \cdot j$ symmetrically.

Consequently, the recognition index for evaluating the quality by which concept C recognizes an observation whose feature values are given by $\{F \cdot j\}$ is just the sum of the products $[P(C \mid F \cdot j) - P(C)] \times [P(F \cdot j \mid C) - P(F \cdot j)]$ over all the feature vales $F \cdot j$, with each product being multiplied by -1 if $P(C \mid F \cdot j) - P(C) < 0$, and where, for purposes of normalization, the sum is multiplied by 100 divided by the number of features.

4.3. The measuring instrument

When presented with an observation to be recognized, the first step is to conduct a branch and bound search for the node that relates best to the observation. The next step is to examine the best node and all its ancestors and from them select the node that gives the highest value for the recognition index. While any of these nodes could be regarded as a "recognizer" for the observation, the one with the highest recognition index will be regarded as the "best recognizer."

When evaluating how HIERARCH builds a hierarchy from a given set of observations, each time a new observation is incorporated into the hierarchy, the best recognizer for each of the remaining unincorporated observations is found and the recognition index associated with the best recognizer is computed. The average of these "best" recognition indices is then calculated, where the average is taken over the remaining unincorporated observations. Since this calculation of the average is repeated each time a new observation is incorporated, a grand average is taken of all these averages. This grand average will be referred to here as the measuring instrument M1.

The measure M1 is a kind of predictor in that it is based on unseen (i.e., as yet unincorporated) observations. However, it does not directly measure the quality of the hierarchy that finally emerges after all the observations have been processed. In fact, it can be fairly sensitive to the order in which the observations are presented in that it will give a relatively low value if the early observations are unrepresentative.

For this reason, a second measuring instrument, M2, is also employed that measures the quality of the final hierarchy for the set of observations from which it was created. The measure M2 is just the average recognition index associated with the best recognizer, where recognition is based on the final hierarchy and the average is taken over all the observations from the original set of observations.

4.4. The experiments

There were ten different orders by which the observations were presented for the soybean task, six different orders for the U.S. Senate voting records, and another six orders for the thyroid task. The measures M1 and M2 were computed for each of these twenty-two orders of presentation. Since this was repeated for five different experiments E1 through E5, to be described below, the measures M1 and M2 were each computed 110 times.

E1 employed an information measure that counted the number of features whose values were specified in the description of the concept or observation. It was the measure discussed in Section 2.1.

E2 was the same as E1 except that the globalization aspect of HIERARCH was turned off. Thus, when a new observation was incorporated into the hierarchy, it was done under the additional constraint that no existing concept or observation would be transferred to another part of the hierarchy.

E3 was the same as E1 except that its information measure computed a weighted (rather than an unweighted) sum of the specified features, with each weight dependent on the overall probability of the value for the given feature. The idea is that a feature value with a very high probability of occurrence is not interesting, since the value is expected and therefore conveys little information. On the other hand, a feature value with a very low probability of occurrence is also not very interesting, since it is so idiosyncratic that it is likely to be associated with a very small cluster. In order to strike a balance between these two considerations, if P were the probability of some value associated with a given feature, then its weight would be $\mathrm{Min}(P, 1-P)$ divided by $\mathrm{Max}(P, 1-P)$. Thus, the weight would reach a maximum of one when $P = .5$ and would gradually fall to zero as P approached 0 or 1.

E4 was the same as E3 except that E4 employed the first measure of strength described in Section 2.2 whereas E3 (as did also E1 and E2) employed the second measure.

E5 computed M1 and M2 for an algorithm that placed each new observation as a child of the root node as discussed in Section 4.1. For E5, the best recognizer was the observation that came closest to matching the new observation.

4.5. Soybean diseases

One of the more interesting applications of previous conceptual clustering programs is the clustering of diseased soybean plants, as it is a real-world example of concern to plant pathologists. The CLUSTER/2 program (Michalski & Stepp, 1983; Stepp, 1984) had classified 47 diseased soybean plants, each described by a vector of 35 feature/value pairs. CLUSTER/2 was able to obtain a four cluster solution (i.e., each plant fell into one of four disease categories) which agreed with the conclusions of expert plant pathologists.

The soybean data was presented to HIERARCH and the five experiments described in the previous section were carried out, once for each of ten different orders of presentation of the observations. The results are summarized in Table 1. The rows S1 through S10 in Table 1 correspond to the ten different sequences of observations, whereas the columns E1 through E5 correspond to the five different experiments. The entries in the table are in the form M1/M2 where M1 and M2 are the two measuring instruments described in Section 4.3.

It should perhaps not be surprising that the measure M1 shows a much greater variability across observation sequences than does M2, since M1 is a predictor of unseen observations and is therefore sensitive to whether or not the early observations in the sequence are representative of what will come later. However, it is interesting to note that the benchmark experiment E5 showed by far the smallest variation across the different sequences. In fact, for the two non-random sequences S4 and S9, where the observations were presented by disease category, the gap between E5 and the other experiments narrowed considerably (and vanished altogether for S9). This suggests that a major benefit of a hierarchy, as

Table 1. The soybean experiments.

	E1	E2	E3	E4	E5
S1	5.95/6.46	5.28/5.83	5.75/6.46	5.18/6.00	4.04/1.61
S2	6.37/6.46	5.76/5.71	6.14/6.46	6.11/6.16	4.06/1.61
S3	6.39/6.46	6.32/6.46	6.33/6.46	6.38/6.46	4.44/1.61
S4	4.42/6.46	4.10/6.01	4.36/6.46	4.36/6.46	3.34/1.61
S5	6.15/6.46	5.83/6.04	6.06/6.46	5.90/5.99	4.23/1.61
S6	6.44/6.46	5.58/5.38	6.39/6.46	6.25/6.46	4.09/1.61
S7	6.68/6.46	5.17/5.10	6.66/6.46	6.58/6.46	4.58/1.61
S8	6.55/6.46	5.28/5.39	6.49/6.46	6.47/6.46	4.51/1.61
S9	3.92/6.46	2.57/3.30	3.85/6.46	3.95/6.48	3.92/1.61
S10	6.56/6.46	6.13/6.05	6.54/6.46	6.48/6.46	4.50/1.61

opposed to a flat linear list of observations as in E5, may lie in its ability to capitalize on representative observations.

For any given observation sequence, two experiments with different values of M1 reflect different learning rates for that sequence (i.e., all other things being equal, a higher value for M1 would likely mean that, as new concepts are learned, higher values were more often obtained for the average best recognition index during the computation of the grand average). However, it is the measure M2 that reflects the final state of the hierarchy after all the observations have been learned.

Every entry of M2 that is as high as 6.46 in Table 1 reflects a final hierarchy that contains a cluster for each of the four disease categories (i.e., each of the 47 observations is contained in one of these four clusters, but none of the four clusters contains an observation from more than one disease category). On the other hand, none of the cases where M2 was less than 6.46 was able to create all four of these "correct" clusters (i.e., clusters that were able to cover a disease category without being corrupted by the presence of an observation from some other disease category). Consequently, the value of $M2 = 6.46$ represents a threshold that needed to be attained if clusters were to be found for all four categories.

As can be seen from the values of M2 in Table 1, Experiments E1 and E3 were able to find all four disease categories for each of the ten observation sequences and E4 was successful for seven of the ten sequences. However, Experiment E2 (i.e., the experiment that turned off the globalization) was able to achieve this in only one of ten attempts.

When the 47 observations were presented to the final hierarchy for purposes of recognition, it turned out that, for those hierarchies where $M2 = 6.46$, the best recognizer of an observation (i.e., the cluster with the highest value of the recognition index) was the cluster that represented the disease category. A cluster that represented a disease category did not always appear as a child of the root node, as it was often the case that it was part of a larger cluster. However, regardless of where the disease category appeared in the hierarchy, the cluster that represented the correct disease category was always identified as the best recognizer so long as M2 was 6.46. Since there were 47 observations and 27 occasions when M2 equaled 6.46, this meant that for each of the $47 * 27 = 1269$ attempts at recognition when M2 equaled 6.46, the cluster that represented the correct disease category was also the best recognizer.

Although Experiment E4 failed on three of its attempts to reach the threshold of 6.46, it was the only experiment ever to exceed 6.46. It managed this for Sequence S9 when it achieved a value of 6.48 for M2. For the hierarchy that produced M2 = 6.48, the best recognizer did not always correspond to the cluster that represented the entire disease category, since on occasion it corresponded to a sub-cluster within the disease category.

4.6. U.S. Senate voting records

Another task asked of HIERARCH was to analyze a listing of fourteen "key votes" (Fisher, 1987) of U.S. Senators during the 1985 session of Congress as published in *Congressional Quarterly*.

All votes were described as either yes, no, or uncertain, with "uncertain" reserved for those situations where the Senator either did not vote or gave no indication as to how he would have voted. The data consisted of 100 feature vectors (one for each Senator) with each feature vector containing 14 feature/value pairs. The results are summarized in Table 2.

The measure M2 proved to be useful in identifying hierarchies where there were problems. For example, the low value of M2 = 5.14 for Experiment E1 on Sequence S6 revealed a hierarchy that did not contain a large, predominantly democratic cluster. It found one cluster consisting of sixteen democrats and two republicans and another cluster consisting of fifteen democrats and one republican but was unable to bring these two clusters together as part of a common cluster. It fared somewhat better on the republican side, as it found a cluster consisting of thirty-two republicans and two democrats, but even this was not as good as that obtained by many of the other hierarchies.

If one averages the results of the predominantly democratic and republican clusters obtained by Experiment E1 for the other five sequences together with those obtained for Experiments E3 and E4 for all six sequences, one gets a predominantly democratic cluster consisting of thirty nine democrats and three republicans and a predominantly republican cluster consisting of 44 republicans and two democrats, out of a U.S. Senate in 1985 consisting of fifty-three republicans and forty-seven democrats.

As expected, Experiment E2 did not do as well, and this was reflected in its generally lower values for M2. On Sequences S1, S2, and S3, it was unable to form a large, predominantly democratic cluster. On Sequence S4, it obtained clusters of 27 democrats and

Table 2. The Senate voting records experiments.

	E1	E2	E3	E4	E5
S1	6.89/6.76	5.77/5.54	6.71/6.91	6.84/6.82	1.96/1.31
S2	7.18/6.51	5.97/5.67	7.15/5.51	7.04/5.82	2.84/1.31
S3	7.26/6.72	6.34/5.76	7.27/5.93	7.40/6.66	3.25/1.31
S4	5.96/6.27	5.06/5.12	5.91/6.18	5.98/6.02	3.15/1.31
S5	7.91/6.78	6.95/5.97	7.84/6.39	7.41/5.87	3.75/1.31
S6	6.59/5.14	7.12/6.89	6.89/6.53	6.80/6.44	2.60/1.31

25 republicans, but both were still well below the previously mentioned averages of 39 democrats and 44 republicans.

4.7. *Thyroid diseases*

One of the more challenging tasks presented to COBWEB (Fisher, 1987) involved the classification of thyroid diseases. The data consisted of one hundred and fifty observations, fifty of which were for Hypothyroid, fifty for Sick-euthyroid, with the remaining fifty regarded as Negative, since they did not represent any disease. HIERARCH was presented this data, and the results are summarized in Table 3.

Sequence S4 was non-random in that it consisted of first the Negative observations, followed by the Hypothyroid, and then the Sick-euthyroid. As had been the case for the soybean diseases, the measure M1 showed the smallest gap between Experiment E5 and the other experiments when the observations were presented by disease category (i.e., non-randomly).

It was hoped that Experiment E3 would continue its success demonstrated in the previous two tasks. This turned out not to be the case, at least for two of the sequences. The low values that E3 recorded of M2 = 2.77 for Sequence S2 and M2 = 2.56 for Sequence S6 reflected an inability to coalesce Hypothyroid observations into a single huge cluster. However, Experiment E3 did very well for the other four sequences. If one averages the results of the sequences S1, S3, S4 and S5 for Experiment E3, one finds that 90.5 percent of the Hypothyroid observations are contained in a single cluster consisting exclusively of Hypothyroid, while 86.5 percent of the Sick-euthyroid observations are in a single cluster of which 91.1 percent are Sick-euthyroid.

It should be recalled that Experiment E4 used the same information measure as E3 and differed only in that it employed a different measure of strength. One strategy, explored further in the concluding section, is to combine the results of E3 and E4 by selecting the hierarchy that gave the larger value for M2. This combination strategy would have performed very well for all six sequences of observations. If one averages the results of all six sequences using this strategy, one finds that 90 percent of the Hypothyroid observations are contained in a single cluster that is 99.6 percent Hypothyroid, while 84 percent of the Sick-euthyroid observations are in a single cluster that is 92 percent Sick-euthyroid.

Table 3. The thyroid experiments.

	E1	E2	E3	E4	E5
S1	3.16/3.33	2.82/2.92	3.15/3.41	3.03/2.67	1.59/0.93
S2	3.12/3.55	2.29/3.29	2.94/2.77	3.13/3.29	1.73/0.93
S3	3.28/3.00	3.23/3.09	3.17/3.32	3.07/3.48	1.31/0.93
S4	2.80/3.48	2.27/2.96	2.30/3.55	2.57/3.53	1.89/0.93
S5	2.91/3.36	2.47/2.86	2.75/3.41	2.95/3.39	1.52/0.93
S6	3.02/3.34	2.84/2.84	3.00/2.56	3.09/3.15	1.57/0.93

5. Concluding remarks

The present approach to conceptual clustering is one of 'divide and conquer' through a clean separation of the characterization and clustering modules. The clustering module of HIERARCH does not even look at any of the conceptual descriptions produced by the characterization module. It only needs to know the amount of information associated with a concept and requires only that the information measure be a local property of the concept and never increase when two concepts are generalized. Thus, the clustering module of HIERARCH can be regarded as a 'knowledge weak' problem solver that derives its ability by virtue of a search procedure that can exploit the information measure that is applied to concepts in the hierarchy. Its knowledge ultimately is obtained from the characterization module, and its enrichment depends indirectly upon the knowledge representation employed by the characterization module as transmitted through the medium of the information measure.

As suggested at the end of the previous section, one possible extension of the algorithm would be to employ a number of different measures of strength, each based on the same information measure. As new observations arrive, a separate hierarchy could be formed for each measure of strength. After a certain number of observations have been processed, one hierarchy could be selected based on the best value obtained for some global measuring instrument. The process then could be repeated as additional observations arrive, only this time each measure of strength would begin with the hierarchy that was selected by the global measuring instrument at the end of the previous cycle.

More research also is needed that is directed at developing and improving such global instruments for measuring the quality of a hierarchy, especially for domains involving structured objects. Although the experiments of the previous section were confined to unstructured objects (i.e., individual feature vectors), there is nothing in the clustering module of HIERARCH that suggests any dependence on feature vectors. Indeed, experience with HIERARCH has indicated an ability at classifying structured objects when the characterization module was able consistently to produce good generalizations. The development of more effective and efficient characterization modules for such domains is an important and challenging task.

Acknowledgments

This research was supported in part by the National Science Foundation under Grant DCR-8408389.

References

Biswas, G., Weinberg, J.B., Yang, Q., & Kollar, G.R. (1991). Conceptual clustering and exploratory data analysis. *Proceedings of the Eighth International Machine Learning Workshop* (pp. 591–595). Evanston, Il: Morgan Kaufmann.

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

Dietterich, T., & Michalski, R. (1981). Inductive learning of structural descriptions. *Artificial Intelligence,* 16, 257–294.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning,* 2, 139–172.

Fisher, D., Xu, L., & Zard, N. (1992). Ordering effects in clustering. *Proceedings of the Ninth International Machine Learning Workshop* (pp. 163–168). San Mateo, CA: Morgan Kaufmann.

Genarri, J., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence,* 40, 11–61.

Hadzikadic, M., & Yun, D. (1989). Concept formation by incremental conceptual clustering. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 831–836). Detroit, MI: Morgan Kaufmann.

Hanson, S., & Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning,* 3, 343–372.

Ignall, E., & Schrage, L. (1965). Application of the branch and bound technique to some flow-shop scheduling problems. *Operations Research,* 13, 400–412.

Kolesar, P. (1967). A branch and bound algorithm for the knapsack problem. *Management Science,* 13, 723–735.

Kolodner, J. (1983). Maintaining organization in a dynamic long-term memory. *Cognitive Science,* 7, 243–280.

Knuth, D., & Moore, R. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence,* 6, 293–326.

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning,* 2, 103–136.

Lowerre, B. (1976). The HARPY Speech Recognition System. Ph.D. thesis, Carnegie-Mellon University, Department of Computer Science.

McKusick, K.B., & Langley, P. (1991). Constraints on tree structure in concept formation. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence* (pp. 810–816), Sydney: Morgan Kaufmann.

Michalski, R. (1980). Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-2, 349–361.

Michalski, R., Stepp, R., & Diday, E. (1981). A recent advance in data analysis: Clustering objects into classes characterized by conjunctive concepts. In L. Kanal & E. Rosenfield (Eds.), *Progress in Pattern Recognition,* 1, 33–55.

Michalski, R., & Stepp, R. (1983). Automated construction of classifications: conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-5, 396–410.

Nevins, A. (1972). Assembly line balancing using best bud search. *Management Science,* 18, 529–539.

Rubin, S., & Reddy, R. (1977). The locus model of search and its use in image interpretation. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 590–595). Cambridge, MA.

Simon, H. (1983). Why machines should learn? In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach.* Los Altos, CA: Morgan Kaufmann.

Stepp., R. (1984). *Conjunctive conceptual clustering: A methodology and experimentation.* Ph.D. thesis, University of Illinois at Urbana-Champaign, Department of Computer Science.

Received May 7, 1993
Accepted August 3, 1993
Final Manuscript September 27, 1993