Improving Inference Through Conceptual Clustering

Douglas Fisher
Department of Information and Computer Science
University of California
Irvine, California 92717

Abstract

Conceptual clustering is an important way to summarize data in an understandable manner. However, the recency of the conceptual clustering paradigm has allowed little exploration of conceptual clustering as a means of improving performance. This paper presents COBWEB, a conceptual clustering system that organizes data to maximize inference abilities. It does this by capturing attribute inter-correlations at classification tree nodes and generating inferences as a by-product of classification. Results from the domains of soybean and thyroid disease diagnosis support the success of this approach.

I. Introduction

Machine learning is concerned with improving performance through automated knowledge acquisition and refinement [Dietterich, 1982]. Learning filters and incorporates environmental observations into a knowledge base that is used to facilitate performance at some task. Assumptions about the **environment**, **knowledge base**, and **performance task** all have important ramifications on the design of a learning algorithm. This paper is concerned with *conceptual clustering*, a task of machine learning that has not been traditionally discussed in the larger context of intelligent processing.

Conceptual clustering systems [Michalski and Stepp, 1983; Fisher, 1985; Cheng and Fu, 1985] accept a number of object descriptions (events, observations, facts), and produce a classification scheme over the observed objects. Importantly, conceptual clustering methods do not require the guidance of a teacher to direct the formation of the classification (as with *learning from examples*), but use an evaluation function to discover classes with good conceptual description. These evaluation functions generally favor classes exhibiting many differences between objects of different classes, and few differences between objects of the same class. As with other forms of learning, the context surrounding the conceptual clustering task can have important implications on the design of these systems.

Perhaps the most important contextual factor surrounding clustering is the performance task that benefits from conceptual clustering capabilities. While most systems do not explicitly address this task, exceptions do exist. In particular, Cheng and Fu [1985] and Fu and Buchanan [1985] have used clustering techniques to organize expert system knowledge. Generalizing on their use of conceptual clustering, classifications produced by conceptual clustering systems 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, 1984].

This paper describes the COBWEB system for conceptual clustering. COBWEB's design was motivated by both environmental and performance concerns. However, this paper is primarily concerned with performance issues – in particular, with the utility of COBWEB classification trees to facilitate inference during classification.[1] The following section motivates and develops an evaluation function used by COBWEB to guide class and concept formation. This measure, called *category utility* [Gluck and Corter, 1985], favors classes that maximize the amount of information that can be inferred from knowledge of class membership. Section 3 describes the COBWEB algorithm. The remainder of the paper focuses on the utility of COBWEB generated classification trees for inference, concentrating particularly on soybean disease diagnosis.

II. Concepts and Concept Quality

COBWEB uses a measure of concept quality called *category utility* [Gluck and Corter, 1985] to guide formation of object classes and concepts. While our primary interest in category utility is that it favors classes that maximize inference ability, Gluck and Corter originally derived category utility as a means of predicting certain effects observed during human classification. These effects stem from a psychological construct called the *basic level* that occurs in hierarchical classification schemes and seems to be where inference abilities are maximized.

[1] COBWEB is also distinguished from other systems in that it is incremental. Issues surrounding COBWEB's performance as an incremental system are given in [Fisher, 1987].

Category utility rewards information rich categories and is thus of generic value, but it can also be viewed (and more easily developed) as a tradeoff of intra-class object similarity and inter-class dissimilarity. Objects are described in terms of (nominal) attribute - value pairs (e.g., Color = red and Size = large). For an attribute value pair, $A_i = V_{ij}$, and class, C_k, intra-class similarity is measured in terms of a conditional probability $P(A_i = V_{ij}|C_k)$; the larger this probability, the greater the proportion of class members sharing the same value (V_{ij}), and thus the more *predictable* [Lebowitz, 1982] the value is of class members. Inter-class similarity is measured in terms of $P(C_k|A_i = V_{ij})$; the larger this probability, the fewer objects in contrasting classes that share this value, and thus the more *predictive* the value is of the class.

Attribute value predictability and predictiveness are combined into a measure of partition quality. Specifically,

$$\sum_{k=1}^{n} \sum_i \sum_j P(A_i = V_{ij}) P(C_k|A_i = V_{ij}) P(A_i = V_{ij}|C_k),$$

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

This function can also be regarded as rewarding the inference potential of object class partitions. More precisely, note that for any i, j, and k, $P(A_i = V_{ij})P(C_k|A_i = V_{ij}) = P(C_k)P(A_i = V_{ij}|C_k)$ (Bayes rule), so by substitution the above function equals

$$\sum_{k=1}^{n} P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k)^2.$$

$\sum_i \sum_j P(A_i = V_{ij}|C_k)^2$ is the *expected* number of attribute values that can be correctly guessed for an arbitrary member of class C_k. [2]

Finally, Gluck and Corter define category utility to be 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 knowledge of a partition $\{C_1, ..., C_n\}$, over the expected number of correct guesses with no such knowledge ($\sum_i \sum_j P(A_i = V_{ij})^2$). Formally, $CU(\{C_1, C_2, ..., C_n\})$ =

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

The denominator, n, is the number of categories in a partition, and averaging over categories allows comparison of different size partitions.

[2] This assumes a *probability matching* guessing strategy, meaning that an attribute value is guessed with a probability equal to its probability of occurring, as opposed to a probability maximizing strategy which assumes that the most frequently occurring value is always guessed (see [Fisher, 1987]).

Category utility can be computed given $P(C_k)$ is known for each category of a partition, as is $P(A_i = V_{ij}|C_k)$ for all attribute values. Such a category representation is termed a *probabilistic concept* [Smith and Medin, 1981]. Information on attribute value distributions distinguish probabilistic concepts from the logical (generally conjunctive) representations typically used in AI systems. Probabilistic representations subsume these types of logical representations, as there exists a simple mapping from probabilistic to logical representations. This increased generality comes at the cost of storing the probabilities, each of which can be computed from two integer counts, thus only increasing the proportionality constant of storage requirements.

III. COBWEB

COBWEB incrementally incorporates objects into a classification hierarchy. Given an initially *empty* hierarchy, over an incrementally presented series of objects, a hierarchical classification is formed, where each node is a probabilistic concept representing an object class (e.g., Birds ≡ BodyCover = feathers (1.0) and Transport = fly (0.88) and ...). The incorporation of an object is basically a process of classifying the object by descending the tree along an appropriate path, updating distributional information along the way, and performing one of several possible operators at each level.

A. Placing an Object in an Existing Class

Perhaps the most natural way of updating a partition of objects is to simply place a new object in an existing class. That is, after updating the distribution of attribute values at the root, the object may be incorporated into one of the root's children. To determine which child 'best' hosts a new object, the object is tentatively placed in each child. The partition that results from adding the object to a given node is evaluated using category utility. The node to which adding the object results in the best partition is the best existing host for the new object.

B. 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. Class creation is performed if it yields a better partition (by category utility). This operator allows COBWEB to adjust the number of classes at a partition to fit the regularities of the environment; the number of classes is not bounded by a system parameter (e.g., as in CLUSTER/2).

C. Merging and Splitting

While operators 1 and 2 are effective in many cases, by themselves they are very sensitive to initial input order-

Table 1: COBWEB control structure

FUNCTION COBWEB (Object, Root ⟨ of tree ⟩)
1) Update counts of the Root
2) IF Root is a leaf
   THEN Return expanded leaf to accommodate the new object
   ELSE Find that child of Root that best hosts Object and perform one of the following
      2a) Create a new class if appropriate
      2b) Merge nodes if appropriate and call COBWEB (Object, Merged node)
      2c) Split a node if appropriate and call COBWEB (Object, Root)
      2d) IF none of the above (2a,b, or c) then call COBWEB (Object, Best child of Root)

ing. To guard against the effects of initially skewed data, COBWEB also includes two operators for node *merging* and *splitting*. The function of merging is to take two nodes of a level (of n nodes) and 'combine' them in hopes that the resultant partition (of n-1 nodes) is of better quality. The merging of two nodes simply involves creating a new node and combining attribute distributions of the nodes being merged. The two original nodes are made children of the newly created node. 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 merging the *two* best hosts (as indicated by category utility) is evaluated.

Besides node merging, node splitting may also serve to 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. Splitting is considered only for the children of the best host among the existing categories.

COBWEB's control structure is summarized in Table 1. As an object is incorporated, at most one operator is applied at each tree level. Compositions of these primitive operators can be viewed as transforming a single classification tree. Fisher [1987] adopts the view that COBWEB is hill-climbing (without backtracking) through the space of possible classification trees. In order to maintain robustness, operators are not restricted to building the tree in a strictly top-down or bottom-up fashion, but the inverse operators of merging and splitting allow COBWEB to move bidirectionally in this space, thus allowing an *approximation* of backtracking through operator application. This strategy keeps update cost small ($B^2 log_B n$ where B is the average branching factor of the tree and n is the number of previously classified objects), while maintaining learning robustness. Fisher [1987] addresses the strengths and weaknesses of this approach in more detail.

IV. Classification and Inference

COBWEB forms classifications that tend to maximize the amount of information that can be inferred from category membership. This is a domain independent heuristic whose efficacy depends on the assumption that important properties are dependent on regularities or 'hidden causes' [Pearl, 1985; Cheng and Fu, 1985] in the environment, and that these regularities can be extracted and organized by a conceptual clustering system. The utility of classification trees for inference was tested in the several domains, including a set of 47 soybean disease cases [Stepp, 1984]. Each case (object) was described along 35 attributes. Four soybean diseases were represented in the data - Diaporthe stem rot, Charcoal Rot, Rhizoctonia Root Rot, Phytophthora rot. These disease designations were also included in each object description, making a total of 36 attributes (e.g., Precipitation = low, Root-condition = rotted, ..., Diagnostic-condition = Charcoal Rot).[3]

An experiment was conducted in which soybean disease cases were incrementally presented to COBWEB in order to see whether the resultant classification could be used for effective disease diagnosis. After incorporating every 5th instance, the remaining unseen cases were classified (but not incorporated) with respect to the classification tree constructed up until that point. 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. This leaf represented that previously observed object that best matched the test object. The diagnostic condition of the test object was guessed to be the corresponding condition of the leaf. The experiment was terminated after one half of the domain (of 47 cases) had been incorporated.

The graph of Figure 1 gives the results of the experiment. The graph shows that after 5 randomly selected instances, the classification could be used to correctly diagnose disease (over the remaining 42 unseen cases) 88% of the time. After 10 instances, 100% correct diagnosis was achieved and maintained. While these results seem impressive, they follow from the regularity of this domain. In fact, when COBWEB was run on the data with no information of Diagnostic condition at all, the four classes were 'rediscovered' as nodes in the resultant tree. This indicates that Diagnostic condition participates in a network of attribute correlations. In organizing classes around the correlated network of attributes, classes corresponding to the various Diagnostic conditions are generated (Figure 2).

[3]While Diagnostic condition was included in each object description, it was simply treated as another attribute. Diagnostic condition was not treated as a teacher imposed class designation as in learning from examples.

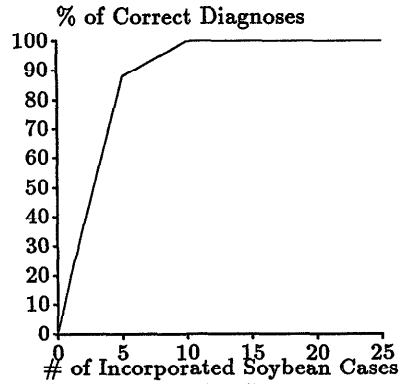

Figure 1: Success at inferring 'Diagnostic condition'

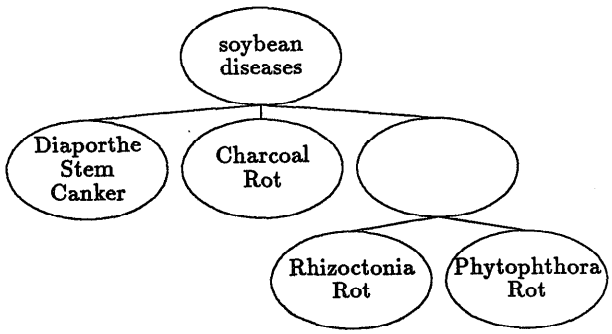

Figure 2: A partial tree over soybean cases

The success at inferring Diagnostic condition implies 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 conducted for Diagnostic condition was repeated for each of the remaining 35 attributes. The results (including Diagnostic condition) were averaged over all attributes and are presented in Figure 3. On average, correct induction of attribute values for unseen objects levels off at 88% using the COBWEB generated classification tree.

To put these results into perspective, Figure 3 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 value of the unknown attribute. Averaged results using this strategy level off at 72% correct prediction, placing it at 16% under the more complicated classification strategy. While averaged results are informative, the primary interest is determining a relationship between attribute interdependencies and the ability to correctly predict an attribute's value.

Dependence of an attribute, A_M, on other attributes, A_i, is a given as a function of

$$\Sigma_{j_M}[P(A_M = V_{Mj_M}|A_i = V_{ij_i})^2 - P(A_M = V_{Mj_M})^2],$$

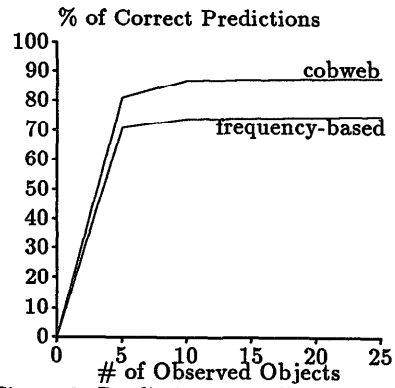

Figure 3: Prediction over all attributes

that is averaged over all attributes, A_i, not equal to A_M. This measures the average increase in the ability to guess a value of A_M given one knows the value of a second attribute. If A_M is independent of all other attributes, A_i, then dependence is 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 $P(A_M = V_{Mj_M}|A_i = V_{ij_i})^2 - P(A_M = V_{Mj_M})^2 = 0$.

In Figure 4 the advantage afforded by the COBWEB classification tree over the frequency-based method is shown as a function of attribute dependence. Each point represents one of the 36 attributes used to describe soybean cases. The graph indicates a significant positive correlation between an attribute's dependence on other attributes and the degree that COBWEB trees facilitate correct inference. For example, Diagnostic condition participates in dependencies with many other attributes and is also the most predictable attribute.

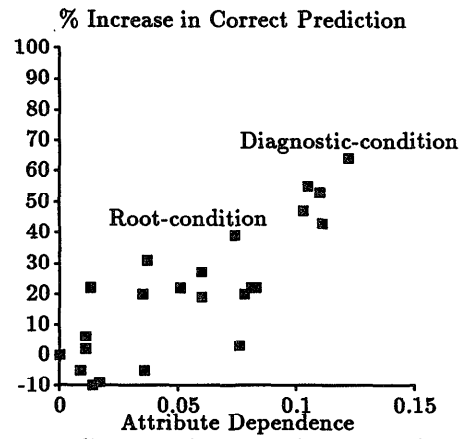

Figure 4: Prediction as function of attribute dependence

V. Concluding Remarks

To summarize, the soybean data strongly suggests that COBWEB captures the important inter-correlations between attributes, and summarizes these correlations at classification tree nodes. In doing so, COBWEB promotes inference of attributes in proportion to their participation in attribute inter-correlations. Similar results have been obtained using thyroid disease data [Fisher, 1987].

Experimentation above assumed classification proceeded all the way to leaves before predicting a missing attribute value. Further studies have indicated however, that for the *inductive* task of predicting properties of previously *unseen* objects, classification need only proceed to about one eighth the depth of the tree to obtain comparable inference results. This behavior emerges as a result of using intermediate node *default* values to determine when attribute value prediction is cost effective (cheap, but reasonably accurate). In COBWEB, default values occur at a level where the attribute approximates conditional independence from other attributes – in this case, knowing the value of other attributes will not aid in further classification, and prediction might as well occur at this level. In this light, COBWEB can be viewed as an incremental and *satisficing* version of a system by Pearl [1985].

Finally, this work casts conceptual clustering as a useful tool for problem-solving, by assigning it the generic, but well-defined performance task of inferring unknown attribute values. The future should find that as conceptual clustering methods utilize more complex representation languages [Stepp, 1984], so too can their behavior be interpreted as improving more sophisticated problem-solving tasks.

Acknowledgements

Discussions with Dennis Kibler suggested a performance task for conceptual clustering. Thanks also go to Jeff Schlimmer, Pat Langley, Rogers Hall, and anonymous AAAI reviewers for numerous ideas and helpful comments. This work was supported by Hughes Aircraft Company.

References

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

[Cheng and Fu, 1985] Y. Cheng & King-sun Fu. Conceptual clustering in knowledge organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 7*, 592-598.

[Clancey, 1984] W. J. Clancey. Classification problem solving. *Proceedings of the National Conference on Artificial Intelligence* (pp. 49-55). Austin, TX: William Kaufmann, Inc.

[Dietterich, 1982] T. Dietterich. Chapter 14: Learning and inductive inference. P. Cohen & E. Feigenbaum (Eds.), *The handbook of artificial intelligence.* Los Altos, CA: William Kaufmann, Inc.

[Fisher, in press] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning.*

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

[Fu and Buchanan, 1985] L. Fu & B. Buchanan. 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.

[Gluck and Corter, 1985] M. Gluck & J. Corter. 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.

[Lebowitz, 1982] M. Lebowitz. Correcting erroneous generalizations. *Cognition and Brain Theory, 5*, 367-381.

[Michalski and Stepp, 1983] R. Michalski & R. Stepp. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 5*, 396-409.

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

[Smith and Medin, 1981] E. Smith & D. Medin. *Categories and concepts.* Cambridge, MA: Harvard University Press.

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