Ordering Effects in Incremental Learning

Douglas Fisher*
Computer Science Dept.
Vanderbilt University
Nashville, TN 37235
dfisher@vuse.vanderbilt.edu

Abstract

Incremental systems often suffer from ordering effects: the knowledge structures that they form may vary with the presentation of objects. We discuss this phenomenon primarily in the context of an incremental, unsupervised system known as COBWEB, but speculate of related phenomena found in supervised systems such as ID4.

1 INTRODUCTION

Learning organizes knowledge in a manner that facilitates efficient and accurate interaction with the environment. The important defining aspects of a learning strategy include assumptions about the form of data accepted by the system, the form of the resulting knowledge structures, a search control strategy that explores the space of knowledge structures, and a measure of knowledge structure quality or fitness that is used to guide the learner's search for good knowledge structures.

Our primary concern when dealing with any learning system is validating whether the knowledge that is learned supports efficient and accurate inference. This can be decomposed into two subordinate concerns: (1) how well does knowledge structure quality, as reflected in the selected fitness measure, reflect on the external validation of a system in terms of the accuracy and efficiency of inference, and (2) how well does the control strategy that searches the space of plausible knowledge structures converge on structures that approximate the optimal according to the selected fitness measure. Ideally, we want a fitness measure that is highly correlated with external performance, and a control strategy that closely approximates optimal knowledge structures in all cases. However, in many situations, the environment imposes additional constraints on the learner that can detract from these ideals.

*Work was supported by NASA Ames grant NCC 2-645.

In particular, incremental learners assume that the data (observations, objects, etc.) given to the learner are presented in a *stream*. Observations are processed one at a time, each triggering some step in the larger search for a knowledge structure that covers this and prior observations. By itself, this is not an important constraint, since a learner can simply save each observation as it is observed, and process the observations made thus far *en masse*. However, an implicit assumption that underlies many incremental learning systems is that a learner should be prepared to react to the environment after each observation, and thus knowledge base revision after each observation should be efficient. These more important, albeit often implicit assumptions, may constrain the learner to knowledge structures that can be efficiently updated via selected search control methods. It is these constraints on a learner's memory of past experience and the limited ways in which this memory can be processed that lead to ordering effects.

We will discuss ordering effects and various issues that contribute to them in the context of Fisher's (1987b) COBWEB system. However, much of this discussion is relevant to other systems such as ID4 (Schlimmer & Fisher, 1986).

2 A REVIEW OF COBWEB

COBWEB is one of a larger class of unsupervised or clustering methods (Michalski & Stepp, 1983) that discover 'informative' categories in unclassified data. COBWEB incrementally forms a classification tree from a stream of observations that are represented as attribute-value pairs. Each node in the tree is a *probabilistic concept*, which gives the attribute-value distributions of the observations that are classified under that node. For example, Figure 1 illustrates a tree that has been constructed over four objects described by `size (small, medium, large)`, `color (blue, green, red)`, and `shape (square, sphere, pyramid)`. Probabilities conditioned on category (node) membership qualify the importance of

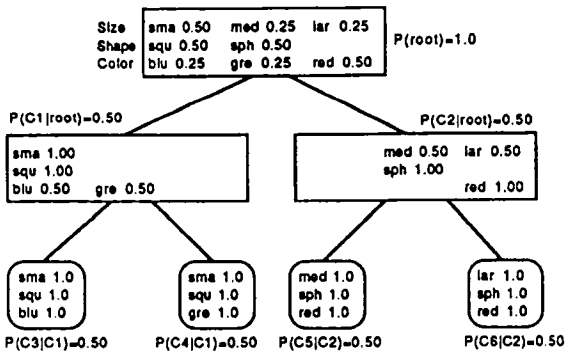

Figure 1: A COBWEB classification tree.

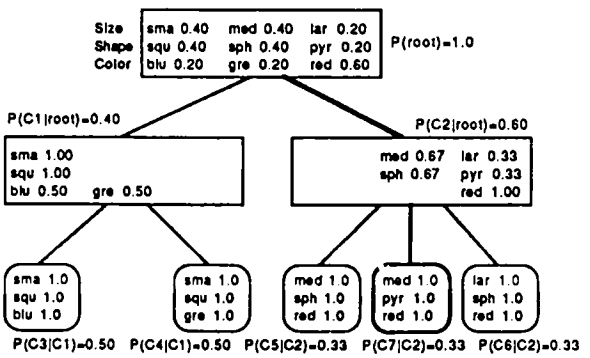

Figure 2: An updated classification tree.

attribute values in category definitions.[1] Leaves are singleton categories that correspond to previously seen objects; the probability of each observed attribute value at leaves is 1.0.

To classify a new object, COBWEB employs an evaluation function or fitness measure known as *category utility* (Gluck & Corter, 1985): $CU(C_k) = P(C_k)\sum_i\sum_j[P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2]$. Intuitively, this measures the degree that a category, C_k, promotes inference. $P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2$ measures how much C_k increases the predictability or certainty of attributes' values over the unconditioned case. This trades off against the size of the category, $P(C_k)$; larger categories are favored since predictions that stem from them apply to a larger portion of the data. The objective is to partition the data in a way that maximizes the average category utility, $\frac{\sum_k^N CU(C_k)}{N}$, of discovered categories. Thus, the top level of the tree ideally is the best partition of observed data. Lower levels recursively partition subsets of the data.

[1] COBWEB maintains actual probabilities over the observed data. However, these proportions merely approximate probabilities in the larger, unobserved domain.

A new object is first classified at the root of the existing classification tree. The probability distributions of the node's attributes are updated. The object is then compared to each child. Each child's attribute value distributions are tentatively updated to reflect the new object. The best category maximizes the difference in CU score of the original node with the CU score of the node after the new object has been added: $\text{MAX}[CU(C_{k'}) - CU(C_k)]$. Adding the object to this category maximizes the average CU of the partition relative to classification in other categories.

The tree of Figure 2 illustrates the addition of a new object. In this case it is classified under the rightmost, top-level sibling (C_2). In addition to classifying objects relative to existing nodes, there must be a way of creating new categories. A new singleton category is created if it yields a better average CU for the partition than placement in the best existing node. Figure 2 illustrates that a new node is created for the object after recursively classifying it under the top level. This indicates that the object is sufficiently different from the other children of C_2 to warrant its own category.

COBWEB has been used in a variety of data analysis, performance, and cognitive model settings (Fisher & Langley, 1990; Fisher, Xu et al, in press). Generally, the system is evaluated in terms of how well it improves prediction accuracy in different domains. Such an evaluation is concerned primarily with illustrating the merits of the evaluation function used to define categories. This paper, however, is largely concerned with how well different search control strategies converge on optimal partitions according to the evaluation function, and with presenting more robust control strategies than the simple classification strategy outlined previously in this section.

3 ORDERING EFFECTS

Incremental clustering has some benefits from a data analysis standpoint. In particular, it is clear to a human analyst/observer how each data point interacts and modifies a growing database (Fisher, Xu et al, 1991). However, it has long been recognized theoretically and experimentally (Fisher, 1987; Gennari, Langley, & Fisher, 1989; Biswas, et al, 1991) that COBWEB and other incremental systems suffer from *ordering* effects: discovered categories may differ depending on the presentation order of objects. In particular, if objects are ordered so that highly similar objects are consecutively presented then a very skewed (unbalanced) classification tree may evolve. The reason lies in category utility's bias to place observations into larger classes. This bias is present in evaluation functions used by AUTOCLASS (Cheeseman et al, 1988), Anderson and Matessa's (1991) system, and the 'simplicity' aspect of CLUSTER (Michalski & Stepp, 1983). As we noted above, there are good reasons for this

bias, notably so that it trades against other aspects of evaluation, thus producing an ideal level of abstraction. This bias takes the form of $P(C_k)$ in category utility. If initial objects are sufficiently similar then several large categories will develop; a new object, regardless of how dissimilar it is from existing categories, will be forced into one of them. Consider the following example from Fisher (1987a). Suppose that we have two categories that contain m_1 and m_2 observations, where $m_1 + m_2 = m$. The quality of this partition is partially given by:

$$\frac{\frac{m_1}{m}\sum_i\sum_j P(V_{ij}|C_1)^2}{2} + \frac{\frac{m_2}{m}\sum_i\sum_j P(V_{ij}|C_2)^2}{2}.$$

A new object will be evaluated with respect to C_1 and C_2 as well as checking whether a new singleton category should be created. A partition with a newly created singleton category will have quality partially given as:

$$\frac{\frac{m_1}{m+1}\sum_i\sum_j P(V_{ij}|C_1)^2}{3} + \frac{\frac{m_2}{m+1}\sum_i\sum_j P(V_{ij}|C_2)^2}{3} + \frac{\frac{1}{m+1}\sum_i\sum_j P(V_{ij}|C_{new})^2}{3}.$$

That is, an attempt to form a new singleton category after m previous observations will yield a category with a partial score of $\frac{1}{m+1}\sum_i P(A_i = V_{ij}|C_{new})^2$, which for a singleton category, C_{new} will equal $\frac{1}{m+1}\sum_i 1.0$. Despite the perfect predictability of attributes in a singleton, for sufficiently large m, the singleton will not be formed, because of a $P(C_{new}) = 1/(m+1)$ that is effectively 0. In addition, for a fixed m, the fewer the number of existing clusters over which the m observations are distributed, the more difficult it will be to form a new singleton class. Intuitively, this makes some sense – if relatively little variance has been observed over the data (i.e., as reflected in fewer clusters), then a reasonable bias is that less variance will be observed in the future, and the more difficult it should be to form new clusters. However, this is not a reasonable assumption in cases where the data description space is observed in a skewed manner. In contrast, if the objects are ordered so that they uniformly sample the object description space, then categories that accurately reflect the variation present in the data will evolve.

In particular, a 'similarity' ordering randomly selects a seed object, and iteratively selects the next object to be maximally similar to the previous object. Similarity is measured by a simple city-block metric, which counts the number of shared attribute values between objects. In contrast, a 'dissimilarity' ordering randomly selects an object, and iteratively selects the next object to be on average maximally-dissimilar from the previous objects. This method tends to uniformly sample the object space.

Table 1: CU scores under different orderings.

	Similarity	Random	Dissimilarity
Soybean	1.02 (0.13)	1.46 (0.02)	1.46 (0.02)
Iris	0.34 (0.04)	0.46 (0.01)	0.47 (0.00)
Mineral	0.45 (0.04)	0.54 (0.03)	0.57 (0.02)
Thermal	0.18 (0.11)	0.39 (0.01)	0.37 (0.02)

Table 2: Mean accuracy over all attributes (20 trials).

	Worst	Random	Best
Soybean	0.895	0.902	0.902
Iris	0.946	0.967	0.976
Mineral	0.613	0.617	0.619
Thermal	0.978	0.979	0.980

Table 1 summarizes the results of 20 trials in each of 4 domains with similarity, dissimilarity, and purely random orderings.[2] In particular, since ideally a COBWEB tree optimally partitions the data at the top level, Table 1 gives the mean category utility scores and standard deviations of top-most partitions. In general, the good news is that random orderings yield significantly better results than similarity orderings with $t(19, \alpha = 0.05)$, and random orderings approximate the quality of dissimilarity orderings, though there are significant differences in the means for the MINERAL and THERMAL domains. Perhaps surprisingly, ordering has almost no effect on prediction accuracy, which is the usual dimension along which COBWEB is evaluated (Fisher, 1987b). This is illustrated by Table 2. Two thirds of each dataset were used for training and one third for testing. In general, COBWEB does as well as many supervised systems for specified 'attributes', obtaining 100% and 98% accuracies on the class 'attributes' in the SOYBEAN and IRIS domains, respectively.

In general, others have explored the effects of ordering (Gennari, Langley, & Fisher, 1989; McKusick & Langley, 1991; Biswas, Weinberg, Yang, & Koller, 1991), though they have used classified data, and iteratively selected observations from different or the same classes. These strategies are roughly analogous to dissimilarity and similarity orderings, respectively, but the former methods rely on class information. In an unsupervised setting, it is typically the class structure which we want to discover. Thus, the dissimilarity strategy reported here generalizes these earlier tech-

[2] The IRIS and small SOYBEAN sets are well known. The MINERAL data set contains 25 objects of 10 attributes and is taken from data at various geological sites. The data was provided by Gautam Biswas and originates with the AMOCO company. The THERMAL domain was provided by Ray Carnes of Boeing and is composed of 60 objects of 10 attributes, and describes fault situations in a simulated thermal plant.

niques; we can actually use this strategy to prescriptively order data in an unsupervised setting when there is a sizable amount of data known *a priori*. Of course, to do so results in a nonincremental algorithm.

Dissimilarity ordering is one way to mitigate (or exploit) ordering effects. This procedure is external to the actual clustering algorithm. However, the more common approach is to build order-mitigating operators directly into the control structure of the algorithm. We turn to search control strategies that are more robust in the following section.

4 CONTROL STRATEGIES

COBWEB basically follows a top-down clustering strategy: observations are classified at the root, then placed in one cluster at the top level, from which clustering continues recursively downwards. The first stage in this process – the iterative allocation of objects among categories at a single level of abstraction (i.e., a single partition) – is common to many systems. A system by Anderson and Matessa (1991) does only this, but most other systems augment this process in order to mitigate ordering effects. Recall that COBWEB does not form a simple partition, but forms a tree with partitions at multiple levels of abstraction. The tree is a well-structured 'memory' of alternative partitions that can be exploited. In fact, COBWEB includes two operators that act upon this memory. *Splitting* allows a node in a partition to be replaced by a partitioning of the node's members. The subnodes that constitute the partitioning of the original node then become nodes in the larger partition. Splitting is illustrated in Figure 3. Rather than search through the possible ways to subpartition the node in question, the tree retains an approximation of the best partitioning: the children of the node. Thus, splitting removes a node and promotes its children to the next highest level, whenever doing so improves the average category utility score of this higher partition. The *merging* operator is roughly the inverse of splitting. Merging compresses two nodes into a larger node (with the original two nodes as children), if to do so improves the quality of the partition.

Taken together, merging and splitting can theoretically transform any classification tree into any other – trivially, by splitting a tree completely apart and then repeatedly merging to form a new structure. However, efficiency concerns require that they be used prudently. In particular, they are only employed when to do so yields immediate benefit in terms of partition quality. The difference between similarity and random-case results in Table 1, which were obtained with the aid of merging and splitting, attest to the limited heuristic foresight in employing these two operators. Thus, several authors have further augmented the basic COBWEB strategy. Fisher (1987a) and McKusick and Langley (1991) allow nodes to be individually promoted and

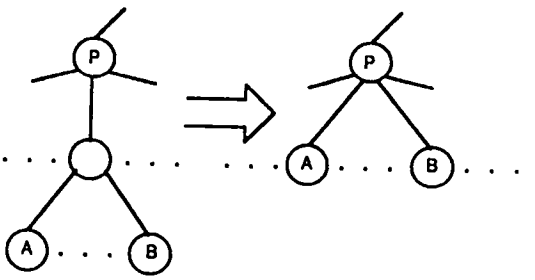

Figure 3: Splitting in COBWEB.

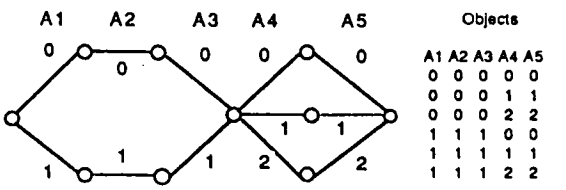

Figure 4: An artificial domain.

recombined relative to higher-level partitions if they appear to be misplaced. Lebowitz' (1988) UNIMEM and Reich and Fenves's (1991) BRIDGER simply delete 'misplaced' nodes and reclassifies the effected objects in the hierarchy. UNIMEM also buffers objects that can be (presumably randomized and) classified when a significant number have accumulated. Biswas et al (1991) extract a 'basic' level from the tree (corresponding closely to the tree's top level) after examining the data once, and then allow objects to be placed in other clusters if this appears reasonable; this continues until no more swaps are called for. Chen (1991) does something very similar, but retains tree structure throughout the 'redistribution' process. Experiments by Fisher (1987) simply reclassify data repeatedly with respect to an existing tree until all data items are classified as the previous iteration, thus stabilizing on a partition.

Each of these reclassification methods,[3] given sufficient data and/or time to reclassify previous data, will stabilize on a single partition. However, none of these methods can completely overcome the effects of initial ordering in search for an 'optimal' partition. Consider the finite state machine of Figure 4. This represents the dependency within two independent collections of attributes. Attributes A_1, A_2, and A_3 are perfectly

[3] *Iterative redistribution* was coined by Biswas, et al in reference to their ITERATE algorithm. We adopt *reclassification* to include this and other strategies surveyed above.

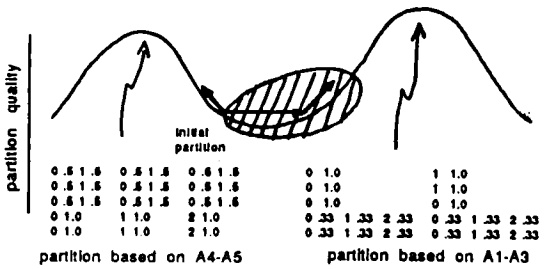

Figure 5: The limitations of reclassification.

correlated: a value on one of them specifies the values of the other two. This is true of the next set of two attributes as well. A partition that segregates objects according to values along either A_1–A_3 or A_4–A_5 represents a local optimal, but because of the fewer values per attribute and more attributes in the case of A_1–A_3, segregation along these attributes yields the globally optimal partition. Unfortunately, many orderings, may be such that partitions will form around the other set of interdependent attributes, particularly when other, irrelevant attributes are present. None of the reclassification strategies will overcome this initial bias if the globally-optimal partition is initially missed. Figure 5 illustrates the limitations with a specific example, and diagrams the general case: local optima can be reached via reclassification, but not escaped.

5 INCREMENTAL ISSUES IN COBWEB

In sum, we have discussed the ways in which the clustering quality measure (e.g., category utility) contributes to ordering effects in conjunction with the search control strategies that have been employed for incremental clustering. If we are concerned with maintaining the efficiency of knowledge base update during strict incremental learning, then we are limited in the ways that we can improve the robustness of the control strategy – we are stuck with some memory limitations and limited heuristic foresight, though we have seen some extensions to the basic COBWEB control strategy in an attempt to improve heuristic foresight. Of course, if we are not primarily concerned with maintaining efficiency, then we can easily and should adapt nonincremental methods to the clustering task (Fisher, Xu, & Zard, 1992). However, for the moment we assume that the efficiency (and limitations) of the search control strategy are held constant. Rather, we turn to the evaluation measure and ask whether alternatives might be better suited to incremental learning.

Table 3: Mean accuracy, partition quality, and partition size for two measures (20 trials).

	Original	Alternative
Accuracy	0.791 (0.028)	0.789 (0.025)
Quality	1.54 (0.088)	1.74 (0.055)
Size	3.35 (0.812)	3.95 (0.605)

5.1 The Evaluation Function

We have stated that incremental, unsupervised learning systems are biased to placing objects within larger clusters. This stems from a more fundamental bias of clustering systems, nonincremental or incremental, that some aspect of the quality function must reward larger clusters, else we may have little, if any, data compression. We cannot throw this bias away in any principled way, though we can modify its precise realization. In addition, earlier versions of COBWEB used the average CU over a set of clusters to assess partition quality. Averaging was important since simply summing over the CU scores of clusters in a partition would indicate that the set of singleton categories was the optimal partition over a set of data – again, no data compression. However, as we noted, averaging makes the creation of new clusters more difficult in cases where fewer clusters exist. This makes some sense under the assumption of a random or dissimilarity ordering, but we are not stuck with this bias.

Recently, we have introduced a measure that does not normalize CU across clusters by taking the average. Rather, we introduced an alternative form of CU that replaces terms such as $P(A_i = V_{ij}|C_k)^2$ with an alternative nonlinear measure, $P(A_i = V_{ij}|C_k) \log P(A_i = V_{ij}|C_k)$. This variation of CU, also due to Gluck and Corter is given by: $P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k) \log P(A_i = V_{ij}|C_k) - P(A_i = V_{ij}) \log P(A_i = V_{ij})$. While we could also average this over clusters, we adapt a measure for decision tree induction by De-Mantaras (1991) that normalizes this in an alternative fashion (Fisher & Hapenyengwi, in press). The motivation for this measure is not transparent, and so for illustrative purposes we will illustrate a simple variation of our initial CU measure that introduces nonlinearity into the $P(C_k)$ term and thus avoids the need to average over categories in order to obtain a partition score:

$$\sum_k P(C_k)^2 \sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2$$

Our expectation is that this measure is less sensitive to ordering effects.

Table 3 illustrates the results of initial experiments in the Soybean domain.[4] These experiments compared the relative merits of our original partition (averaged)

[4]The conventions for this study differ somewhat from

quality measure, and the one given above over 20 randomized orderings. For each measure, the Table shows the average accuracy obtained with each measure, the average partition score, and the average size of each partition. In addition, we show standard deviations along each of these dimensions. For the case of partition score we show a normalized standard deviation, which is the ratio of the standard deviation and mean – because each measure is scaled differently, this better reflects dispersion around the means in comparable terms. In general, the results indicate comparable accuracy between the two measures, larger partitions for the alternative, and smaller standard deviations across all dimensions, thus providing some tentative evidence for less order sensitivity on the part of the alternative measure.

In general, we are moving to alterative measures such as the one illustrated and the measure reported in Fisher and Hapenyengwi, in part due to issues of ordering sensitivity. In addition, we are also moving to measures that introduce nonlinearities into the evaluation measure using log. This is largely motivated for reasons of accuracy, but there may be some merit to these relative to ordering effects. In particular, $\sum_j P(A_i = V_{ij}|C_k)^2$ and $\sum_j P(A_i = V_{ij}|C_k) \log P(A_i = V_{ij}|C_k)$ are like-shaped functions: they are maximized when one value, V_{ij}, occurs with certainty, and other values have probability, 0.0; each function is minimized when values are evenly distributed. However, the difference between the maximum and minimum is more pronounced with the log 'squashing' function.

$$\sum_i \sum_j P(A_i = V_{ij}|C_k) \log P(A_i = V_{ij}|C_k)$$

will tend to contribute more weight relative to the $P(C_k)$ term, thus making it easier to create new singleton categories.

5.2 Assessing Tree Quality

Thus far, we have been exclusively concerned with assessing the quality of partitions. Category utility and related measures guide the system's search for high quality partitions. McKusick and Langley (1991) were to some extent concerned with assessing the quality of larger knowledge structures – trees. Gennari et al (1989) and Biswas et al (1991) noted that poor orderings could lead to highly skewed trees, which were almost linear rather than well balanced.

To guide incremental tree updates, we would like to quantify this notion of tree quality. As a starting point, we must again ask ourselves what external task the knowledge structure is intended to facilitate. Our experiments indicate that accuracy alone is inadequate as a performance dimension for discriminating 'good' (balanced) from poor (skewed) trees. Apparently, regardless of the particular tree structure, at a deep enough level where predictions are actually generated, skewed and balanced trees decompose the data in a similar manner. The primary difference however lies in the efficiency with which categorization proceeds with differing tree structures. In general, much of our discussion on ordering effects is akin to general issues of incremental update in such things as binary search trees (Knuth, 1973), and as in these structures measures that reflect good tree structure should take into account the *expected cost* of categorization as well as the accuracy that stems from this categorization. It is certainly the case that quantitative measures that reflect meaningful differences in tree quality can be quantified in terms of cost and accuracy.

We will not pursue the issue further here, but simply leave the reader with the idea that principled operations of knowledge structures at any level of complexity (e.g., partitions or trees) are best motivated by returning to the task that these operations are intended to optimize; we need not be satisfied with motivating our methods by qualitative intuitions alone.

6 ORDERING EFFECTS IN DECISION TREE INDUCTION

Schlimmer and Fisher (1986) and Utgoff (1989) have each defined incremental methods for supervised induction of decision trees. Schlimmer and Fisher's ID4 incrementally accumulates observations, and maintains the frequencies with which attribute values are observed. When statistics indicate that one 'most informative' attribute is significantly informative then a decision node is expanded with that attribute, and its values indicate subpopulations of observations. Attribute value statistics are then maintained at the terminals of each decision branch until a node can be expanded for a subpopulation. After expansion, statistics continue to be accumulated at a node, as well as its descendents. Should statistics accumulated over subsequent observations at a previously expanded node indicate that an alternative attribute is deemed more informative than the one used originally for expansion, then the node and its descendents are deleted, and the node is reexpanded with the alternative attribute; the subtree is regrown with subsequent observations. Utgoff's ID5 operates in much the same way, but rather than throwing away a node's descendents when a new attribute is deemed more informative than the original, it uses these descendents to good effect in a local reorganization of the subtree.

Many of the issues that we have discussed relative to unsupervised learning apply here. Notably, it is likely

that of past studies reported for the Soybean domain. Thus there are some differences in the average accuracies and partition qualities between those reported in Table 3 and previous tables.

that the differences that we saw between similarity and dissimilarity orderings effect these decision tree induction systems in similar ways. A similarity ordering which gives a very skewed sampling of the data description space is likely to lead to many subtree deletions in ID4 or reorganizations in ID5. A dissimilarity ordering, which uniformly samples the data description space, is likely to lead to relative stability in tree growth.

It is interesting to note that ID5's reorganization strategy and COBWEB's use of split and merge are similarly intended – both form alternative partitions by exploiting the memory of alternatives that is implicit in subtrees. However, ID5 does this in a manner that guarantees optimality in the sense that the selected divisive attribute at each node of the reorganized tree is the best in that context. As a result, ID5 will form the same tree as its nonincremental counterpart, ID3 (Quinlan, 1986). It may pay a price to insure this though; in most, but not all cases we expect that reorganization is cheap, but in some cases it may rival the cost of a nonincremental assessment. In contrast, work with COBWEB has generally ruled out any possibility of expensive reorganization, and thus its use of merging and splitting operators with limited heuristic foresight. Nonetheless, it would be informative to define a general procedure that performed more extensive reorganization if needed. Work on promotion by Fisher, and McKusick and Langley (see Section 4) moves in this direction, but again under the assumption that expensive reorganizations not simply be improbable, but impossible.

7 CONCLUDING REMARKS

We have described ordering effects in unsupervised, incremental systems and various control strategies and measures that are intended to mitigate these effects. The general issues are shared with incremental supervised systems, though we have only touched on these relationships. In addition, there are many issues that we have not delved into. For example, incremental learning is significantly complicated when relational data is assumed (Thompson & Langley, 1991). Relational data introduces a second source of nondeterminism; we are not only concerned with finding a best category for an observation, but with finding the best way to match the observation with summary relational representations of each category. Our guess is that these two sources of nondeterminism increase ordering effects considerably. Second, we have seen how dissimilarity orderings seem best in unsupervised and supervised systems, since they uniformly sample the observation space. While the learner may not be able to manipulate the data ordering directly (and still be called an incremental learner), it is worthwhile to investigate strategies that a (human or machine) teacher uses to order data (problems) to students. This analysis may inform the development of machine learners, but work on these issues can undoubtedly inform educational research on the best strategies for human training as well (Fisher & Yoo, in press).

Acknowledgements

Sections 2 through 4 are largely reprinted from Fisher, Xu, and Zard (1992).

References

Anderson, J. R., & Matessa, M. (1991). An interative Bayesian algorithm for categorization. In D. Fisher, M. Pazzani, & Langley, P. (Eds.), *Concept formation: Knowledge and experience in unsupervised learning.* San Mateo, CA: Morgan Kaufmann.

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

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AutoClass: A Bayesian classification system. *Proceedings of the Fifth International Machine Learning Conference* (pp. 54–64). Ann Arbor, MI: Morgan Kaufmann.

Chen, J. J. (1990). Conceptual Clustering Approach to Multichannel Pattern Analysis: Applications to Synergy Pattern Analysis of EMG Linear Envelopes in Gait Study (Doctoral Dissertation). Department of Biomedical Engineering, Vanderbilt University, Nashville, TN.

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

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

Fisher, D., & Hapenyengyi, G. (in press). Database management and analysis tools of machine induction. *Journal of Intelligent Information Systems.*

Fisher, D. H., & Langley, P. (1990). The structure and formation of natural categories. In G. H. Bower (Ed.), *The Psychology of Learning and Motivation.* San Diego, CA: Academic Press.

Fisher, D., & Pazzani, M. (1991). Computational models of concept learning. In D. Fisher, M. Pazzani, & Langley, P. (Eds.), *Concept formation: Knowledge and experience in unsupervised learning.* San Mateo, CA: Morgan Kaufmann.

Fisher, D., Xu, L., et al (in press). Selected applications of an AI clustering method to engineering applications. *IEEE Expert.*

Fisher, D., Xu, L., & Zard, N. (1992). Ordering effects in Clustering. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 163–168). Aberdeen, Scotland: Morgan Kaufmann.

Fisher, D., & Yoo, J. (in press). Categorization, concept learning, and problem solving: A unifying view. G. Nakamura, R. Taraban, & D. Medin (Eds.) *Categorization in Humans and Machines, The Psychology of Learning and Motivation, Vol. 29.*

Gennari, J., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence, 40,* 11–62.

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.

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

Knuth, D. E. (1973). *The Art of Computer Programming; Vol. 3: Searching and Sorting.* Reading, MA: Addison-Wesley.

Lebowitz, M. (1988). Deferred commitment in UNIMEM: Waiting to learn. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 80–86). Ann Arbor, MI: Morgan Kaufmann.

Martin, J. D. & Billman, D. (1991). Representational Specificity and Concept Learning. In D. Fisher & M. Pazzani (Eds.), *Concept formation: Knowledge and experience in unsupervised learning.* San Mateo, CA: Morgan Kaufmann.

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

Michalski, R. S., & Stepp, R. (1983). Automated construction of classifications: conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 5,* 219–243.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1,* 81–106.

Reich, Y., & Fenves, S. (1991). The formation and use of abstract concepts in design. In D. Fisher & M. Pazzani (Eds.), *Concept formation: Knowledge and experience in unsupervised learning.* San Mateo, CA: Morgan Kaufmann.

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

Stepp, R. (1987). Concepts in conceptual clustering. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 211–213). Milan, Italy: Morgan Kaufmann.

Thompson, K., & Langley, P. (1991). Concept formation in structured domains. In D. Fisher & M. Pazzani (Eds.), *Concept formation: Knowledge and experience in unsupervised learning.* san Mateo, CA: Morgan Kaufmann.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning, 4,* 161–186.