Constraints on Tree Structure in Concept Formation

Kathleen B. McKusick* and Pat Langley
AI Research Branch, Mail Stop 244-17
NASA Ames Research Center
Moffett Field, CA 94035 USA

Abstract

We describe ARACHNE, a concept formation system that, uses explicit constraints on tree structure and local restructuring operators to produce well-formed probabilistic concept trees. We also present a quantitative measure of tree quality and compare the system's performance in artificial and natural domains to that of COBWEB, a well-known concept formation algorithm. The results suggest that ARACHNE frequently constructs higher-quality trees than COBWEB, while still retaining the ability to make accurate predictions.

1 Background and Motivation

The task of concept formation involves the incremental acquisition of concepts from unlabeled training instances (Fisher &; Pazzani, in press). Much of the recent research on this topic builds on Fisher's (1987) COBWEB. Fisher's system assumes that each instance is described as a conjunction of attribute-value pairs, and employs a probabilistic representation for concepts. In particular, COBWEB represents each concept *Ck* as a set of attributes *Ai* and a subset of their possible values V_{ij}. Associated with each value is the conditional probability of that value given membership in the concept, $P(A_i = V_{ij} \backslash C_k)$. In addition, each concept has an associated probability of occurrence, $P(C_k)$- COBWEB organizes its conceptual knowledge into a hierarchy, with nodes partially ordered according to their generality; thus, the root node summarizes all instances that have been observed, terminal nodes correspond to single instances, and intermediate nodes summarize clusters of observations.

COBWEB integrates the processes of classifying instances and incorporating them into memory. The system sorts each new instance *I* down the hierarchy, starting at the root, locating nodes that summarize classes into which the instance fits well. At a given node TV, COBWEB retrieves all children and considers placing the instance in each child node *C* in turn; it also considers creating a new child based on the instance. The algorithm uses an evaluation function, *category utility* (Gluck & Corter, 1985), to determine the "best" resulting partition, then incorporates the instance into memory accordingly. The system then recurses, sorting the instance through memory until it produces a disjunct or reaches a terminal node. Other research on concept formation (Anderson & Matessa, in press; Hadzikadic & Yun, 1989; Lebowitz, 1987) has assumed a similar control structure.

Our designs for ICARUS (Langley, Thompson, Iba, Gennari &: Allen, in press) - an integrated cognitive architecture - use COBWEB as the underlying engine for classification and concept formation. ICARUS invokes Fisher's algorithm to acquire primitive concepts, which in turn serve as background knowledge for the rest of the system. In theory, COBWEB produces structures containing concepts that correspond to concepts in the data, at several levels of generality. However, our experience with. COBWEB suggests that its ability to form identifiable concepts is limited. Because the system's evaluation function is oriented toward maximizing predictive accuracy, the hierarchies it constructs may not reflect the underlying class structure of the domain. This behavior is especially apparent with noisy data and with certain orders of training instances. This has implications for systems that use these concepts as building blocks for other knowledge structures. For example, if a unified *cat* concept has not been formed, it cannot be used as part of a larger knowledge structure, such as a living room.

In this paper we describe ARACHNE, a concept formation system that seeks to construct well-formed concept hierarchies while maintaining high predictive accuracy. ARACHNE'S focus on the structural quality of the hierarchies it constructs is new to unsupervised learning. However, Van de Velde's (1990) supervised IDL algorithm seeks to induce decision trees with high accuracy and desirable structural properties. Although ARACHNE'S structural goals are different from those of IDL, both systems use structural principles to guide tree formation.

The ARACHNE algorithm bears many similarities to COBWEB, but employs different criteria for tree formation and uses alternative restructuring operators, which we describe in the next section. We then present experimental studies that compare the behavior of the two systems on both accuracy and tree quality. We close with general observations about the two systems and directions for future work.

2 The ARACHNE System

Like COBWEB, ARACHNE represents knowledge as a hierarchy of probabilistic concepts, and it classifies new instances by sorting them down this hierarchy. The system differs from COBWEB in its concern for the structure

*Also affiliated with Sterling Federal Systems.

of the concept tree it constructs, in the learning algorithm it employs, and in the way it classifies instances. Below we discuss each of these differences in turn.

2.1 Constraints on Memory Organization

ARACHNE'S main goal is to create well-structured concept trees, but this requires some specification of "desirable" structures. We have chosen to state these as formalized constraints, but we have been careful to focus on local constraints that can be tested efficiently. Our hope has been that global properties would tend to emerge from these local concerns, even though we could not guarantee this would occur. The system's approach assumes that one has some *similarity metric S* that lets one determine the similarity of two instances, an instance and a concept, or two concepts. The constraint framework does not depend on any particular metric.

Recall that, in our framework, each concept is a probabilistic abstraction of the nodes below it in the hierarchy, and each child is a specialization of its parent. This structure suggests two local constraints on the structure of concept trees, which we believe reflect useful notions of well-formed hierarchies. The first deals with the relation between a child, its parent, and its siblings:

DEFINITION. Concept *N* is *horizontally well placed in* a concept tree w.r.t. similarity metric *S* if *P* is the parent of TV and for all siblings A' of TV, *S(N, P) > S(N, K).*

Thus, a concept is horizontally well placed if it is of equal or greater similarity to its parent than to any sibling. If a node TV violates this constraint, it suggests that one should consider merging TV with one of its siblings.

A second constraint concerns the relative similarity between a child, its parent, and its grandparent:

DEFINITION. Concept TV is *vertically well placed* in a concept tree w.r.t. similarity metric *S* if *P* is the parent of TV, *G* is the parent of P, and S(TV, P) > 5(TV, *G).*

Thus, a concept is vertically well placed if it is more similar to its parent than to its grandparent. If a node TV violates this constraint, it suggests that one should consider promoting TV to become a child of its grandparent (and thus a sibling of its parent).

Taken together and applied over all the nodes in a concept tree, these constraints can be used to define a characteristic of the entire tree:

DEFINITION. A concept tree is *well organized* w.r.t. similarity metric *S* if all concepts in the tree are horizontally and vertically well placed with respect to 5.

ARACHNE seeks to generate concept hierarchies that are well organized for a given similarity metric. We hypothesize that such hierarchies will reflect concepts that are inherent in the domain.

2.2 ARACHNE'S Control Structure

The ARACHNE learning algorithm has many similarities to COBWEB but also important differences. The system accepts an instance / and a concept node TV as arguments, and incorporates / into the hierarchy below TV. If TV is a terminal node, ARACHNE (like COBWEB) extends the hierarchy downward, creating a new concept *P* that summarizes TV and /, making TV and / children of the new concept. If TV is not a terminal node, the system averages I into the existing probabilistic description and stores the instance as a new child. At this point, ARACHNE considers two operators for restructuring the hierarchy, and this is where it most diverges from its predecessor.

The system first checks each child *C* of TV in turn (including the new child /) to make sure it obeys the constraint that it be vertically well placed. If *C* violates this condition, ARACHNE promotes C, removing it as a child of TV and making it a child of TV's parent. (Actually, the system must recheck each constraint after applying the promote operator, since this changes the description of TV). This ensures that no children of TV are more similar to their grandparent than to their parent. Thus, storing a new instance as a child of a concept can cause a sibling instance or concept to "bubble up" to a higher location in memory.

ARACHNE'S next step involves checking each child of *N* to make sure it obeys the constraint that it be horizontally well placed. If two or more children are more similar to each other than either is to TV, the system merges the most similar pair. This involves replacing these siblings with a new node that is their probabilistic average, taking the union of their children as its children. ARACHNE then recursively considers merging this new node's children. In some cases, this leads to recreation of the original siblings at a lower level in the hierarchy; in other cases, it produces further reorganizations in the subhierarchy. In particular, if the original instance is merged with an existing concept, recursive calls of the merge operator can effectively sort it down through memory.

Once it has merged two nodes at a given level, ARACHNE checks the remaining nodes for satisfaction of horizontal well placement. If it finds two or more nodes that violate this constraint, it again merges the most similar, then repeats this process until all nodes at this level satisfy the constraint. In this way, a single new instance can cause the system to merge successively many of the nodes previously stored at a given level, including pairs of nodes dissimilar from it. For instance, suppose ARACHNE had stored four instances of cats under a common parent, and a dog instance is added (through merging from above). Here the system would first merge the two most similar cats, then merge a third into the resulting node, and finally the fourth. The result would be two concepts, one representing the abstraction of four cat instances and the other based on a single dog. This iterative merging process differs from that used in COBWEB, which merges nodes only when they are similar to a new instance. Thus, we expect ARACHNE will create well-structured trees regardless of the order in which instances are presented.

2.3 Similarity and Prediction in ARACHNE

Recall that ARACHNE'S constraints and control structure rely on the ability to measure the similarity between nodes and/or instances. At each level, the system uses its similarity metric to decide in which class an instance belongs by determining which class descrip-

tion is most similar to that of the instance. ARACHNE also uses its similarity measure to determine the depth to which it should sort an instance, halting whenever the best similarity score at the next level is no better than that at the current level. The metric also plays a role in deciding when to invoke the merge and promote operators. Hadzikadic and Yun (1989) have also used a similarity metric to guide the concept formation process; both their INC system and ARACHNE differ in this way from COBWEB, which uses an evaluation function over an entire partition of nodes. Although ARACHNE can use different similarity functions, our tests with the system have used a simple measure of "probabilistic overlap" between the attributes of nodes and instances.

ARACHNE uses the same similarity function and essentially the same control structure for prediction that it uses in learning. The system sorts an instance down the hierarchy in accordance with its constraints, except that no promotion is allowed and only merges that involve the instance are executed.[1] Thus an instance sorts to the class at which it would ordinarily become a disjunct, and a prediction is made from the last node to which it sorted. ARACHNE includes a simple recognition criterion to foster prediction from internal nodes and thus avoid overfitting. As it sorts an instance through memory, the system makes a prediction from an internal node if its modal values perfectly match all the values of the instance.

3 Comparative Studies

Now that we have described ARACHNE, we must still demonstrate that hierarchies constructed according to its constraints have desirable structural properties, and that ARACHNE is competitive with COBWEB in terms of predictive ability. In designing the algorithm, we suspected that good structure would, if anything, enhance the latter ability, and sought to show this experimentally. To this end, we designed a set of comparative studies, which we report after summarizing the dependent variables we used to measure the systems' behaviors.

3.1 Accuracy of Class Prediction

Typically, researchers have evaluated inductive learning systems by training them on a set of examples and then measuring their ability to make predictions about new examples. For supervised learning methods, the prediction task involves identifying the *class name* of a novel instance, given training instances that include this information as part of their description. In contrast, the training data for unsupervised systems like COBWEB and ARACHNE does not include class information. Thus, Fisher (1987) introduced the task of flexible prediction, which requires the system to predict the values of one or more arbitrary attributes that have been excised from the test instances. Martin (1989) and Gennari (1990) have used similar performance measures.

[1] Recall that during learning, ARACHNE can merge any two nodes at the current level. It is not limited to considering only merges that involve the instance being sorted.

However, the class name can be used for prediction straightforwardly, even with unsupervised learning methods. A typical unsupervised system does not include the class name in the description of training instances, but there is nothing to prevent one from including such class information, provided the system does not use it to determine concepts. To take advantage of this idea in evaluating systems like ARACHNE and COBWEB, we associate the class name with each instance description as an extra attribute, hiding the label so that it does not affect clustering. However, we do let the system retain probabilities at each concept for the class labels of instances summarized by the node. To predict the class name of a new instance, the system simply classifies the instance to a node in the hierarchy and predicts the most frequently occurring label at that node.

We chose to predict class names in our comparative studies because they provide a good baseline for predictive ability. Class names are never provided by the environment; domain experts assign them based on regularities they have observed over time. Thus, they are designed to be predictable from observed features. In contrast, Fisher's notion of flexible prediction fails to distinguish between attributes that can be predicted trivially, ones that can be predicted with appropriate knowledge, and ones that cannot be predicted at all.

3.2 Quality of Tree Structure

Informal inspections of the trees constructed by ARACHNE and COBWEB suggested that the former system was frequently building "better" trees, in that fewer instances were situated in classes where they did not seem to fit well. ARACHNE also seemed to construct fewer "junk" nodes - spurious clusters of instances that have little in common. The challenge was to quantify these observations and to construct a measure of tree quality that could be applied to the hierarchies of both systems. This measure should not favor the guiding organizational constraints of either system. For example, we might have evaluated whether the category utility of the top-level partitioning was optimized, as in Fisher's (1987) study of tree quality in COBWEB, or how well the hierarchies adhered to global variants of the constraints set forth for ARACHNE. But these measures would be biased in favor of one of the systems.

Instead, we devised two dependent measures which we could apply to trees in artificial domains for which we knew the "correct" concept hierarchy. A good hierarchy should contain nodes that correspond to concepts embodied in the data. If one knows that a domain contains well-defined, distinct concepts, then a natural measure of tree quality should reflect the degree to which the learned tree respects the known structure of the data used to build it. In keeping with this idea, we counted the percentage *of formed* concepts, or those nonterminal nodes whose modal values exactly matched the modal values of a concept known to exist in the data.

Furthermore, we defined a measure of *well-placed instances,* singleton nodes that are descendents of a target concept and that match 50% or more of the modal attribute values of the target concept. Concepts containing

well-placed instances tend to adhere closely to their expected concept description, and show minimal presence of attribute values in frequencies that vary from the expected. A high percentage of well-placed instances in a tree implies a large number of accurate concepts.

3.3 Experimental Procedure

In our experimental studies, we carried out groups of ten runs,[2] presenting ARACHNE and COBWEB with identical sets of randomly selected training instances for each run. We used the same test set for every run in a group. In experiments not concerned with order effects, the training instances were randomly ordered. In all cases, test instances were taken from the same distribution as the training instances. Thus, if the training data had a noise level of 90%, on average the test data would have that noise level as well.

Both systems sorted training instances one at a time through their hierarchies. Learning took place as each instance was incorporated, as probabilities in the concept descriptions changed and the hierarchies were restructured. After each training instance, the entire test set was presented to each system, which used the hierarchy it had formed thus far to predict the class name of each test instance. We compared this to the "actual" class name associated with the test instance, giving the system a score of one if the prediction was correct and zero otherwise. No learning was done on the test instances, so it was possible to construct learning curves which plot average accuracy on the test set as a function of the number of training instances seen.

For our studies we used two versions of COBWEB that built identical hierarchies but differed in their prediction mechanisms. Recall that ARACHNE has a recognition criterion to foster prediction from internal nodes; it stops sorting an instance if its values perfectly match the modal values of a node it has reached in memory. To control for the effect on predictive accuracy, we created a version of COBWEB, denoted COBWEB', which included this mechanism. Since this noticeably affected performance only in the noisy artificial domain, we report COBWEB' results only in that section.

After each system had processed the complete training set on each run, we ran the final hierarchies through our assessor of tree quality, which reported the number of concepts found and the percentage of well-placed nodes. Since we devised these measures especially for our artificial data sets, which had known, clearly-defined concepts, we did not attempt to apply them in the natural domains we tested.

3.4 Behavior on Natural Domains

We now consider some hypotheses about the relative behavior of COBWEB and ARACHNE, and the experiments we carried out to test them using the dependent measures and procedure described above. Our first step in evaluating ARACHNE was to examine its behavior on some standard problems from the machine learning literature. For this purpose, we selected the domain of Congressional voting records, which Fisher (1987) has used in tests of COBWEB, and the domain of soybean diseases, which Michalski and Chilausky (1980) used in their experiments on supervised learning. The prediction in this case is straightforward:

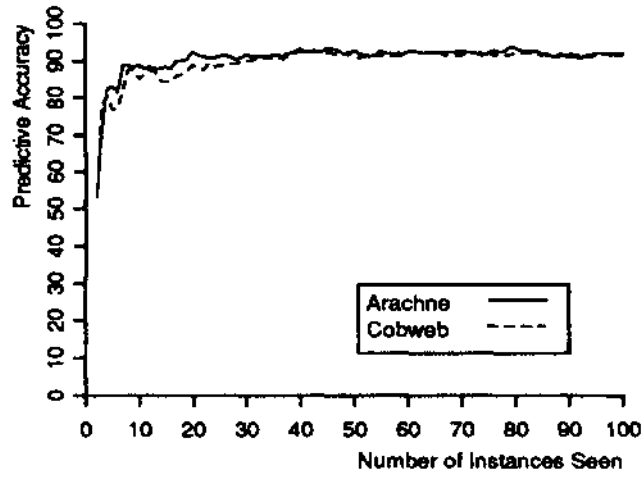

Figure 1. Learning curves for ARACHNE and COBWEB on congressional voting records, using predictive accuracy as a performance measure.

Hypothesis: ARACHNE *will show better predictive accuracy than* COBWEB *in natural domains.*

The congressional voting domain contains 435 instances of sixteen Boolean attributes each (corresponding to yea or nay votes), with each falling into one of two classes (Democrat and Republican). In contrast, the soybean data set contains 683 instances from 19 classes, each described in terms of 35 symbolic attributes.[3] Thus, the two data sets differ in the number of attributes and diverge even more in the number of prespecified classes.

To evaluate our hypothesis, we presented both COBWEB and ARACHNE with random samples of 100 training instances and a test set of 25 instances from the congressional domain and 150 training instances and a test set of 95 instances (five from each class) from the soybean domain. The results, averaged over ten runs for the first domain and five for the latter, reveal similar accuracies in class prediction throughout the course of learning. On the congressional records, ARACHNE reaches its asymptote slightly earlier than COBWEB, at about 20 instances rather than 40 instances. On the soybean data, COBWEB reaches asymptote earlier, at about 120 instances rather than 140 instances. But in both cases the two asymptotes are basically equivalent and, in general, the differences we had anticipated did not emerge, forcing us to reject our hypothesis. However, we should not conclude too swiftly that ARACHNE and COBWEB always perform at comparable levels; these two domains simply may not have characteristics that bring out their differences. This suggests another approach, to which we now turn.

[2] The soybean results are one exception to this rule; in this case our averages are based on five runs.

[3] This data set is much more complex than the four-class version used by Stepp (1984) and Fisher (1987).

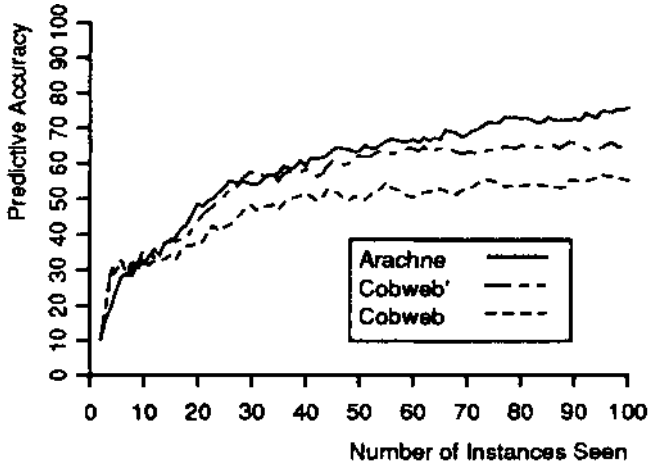

Figure 2. Learning curves for ARACHNE and COBWEB on an artificial domain with high attribute noise, using predictive accuracy as a performance measure.

3.5 Effects of Attribute Noise

Although studies with natural domains show relevance to real-world problems, artificial domains are more useful for understanding the reasons for an algorithm's behavior. A common use of such domains involves varying the noise level. Noise in a training set confounds a learning system; it blurs boundaries between classes, making misclassification of instances more likely and the underlying concepts more difficult to discern. A system trained on noisy data is susceptible to *overfitting,* predicting attributes at a more specific concept than it should.

We designed ARACHNE'S performance and learning algorithms to be robust in noisy domains, and this suggests a simple prediction about its behavior:

Hypothesis: ARACHNE *will be less affected by noise than* COBWEB *w.r.t. both accuracy and tree structure.*

Intuitively, ARACHNE should build better trees because of its more powerful reorganization operators and concern with constraints, and it should be less subject to overfitting because of its ability to predict from well-formed internal nodes.

To test this hypothesis, we designed artificial data sets at two levels of noise. Each contained instances with four attributes, each of which could take on ten distinct values. The attributes had a prototypical value but could take on other "noise" values at some specified probability. In the low-noise data set (noise level I), the modal value for each attribute occurred with probability 0.7, while three "noise" values occurred with probability 0.1. In the noisier data set (noise level II), the modal value for each attribute occurred with probability 0.5, while five noise values occurred with probability 0.1. Because a noise value appearing in one class was the modal value of some other class, class descriptions overlapped to some extent. About 24% of the noise level I instances and only about 6% of the noise level II instances should conform perfectly to the modal class description, with the remainder being noisy variants.

We carried out ten runs at each of these noise levels, presenting COBWEB and ARACHNE with 100 training examples in each case. Figure 2 shows the learning curves for noise level II; similar results were achieved at noise level I. The graph plots the average predictive accuracy against the number of instances seen. At noise level II, ARACHNE asymptotes at 76% accuracy, while COBWEB asymptotes at 55% and COBWEB', which predicts from internal nodes, at 65%. In this domain, tree quality was lower for COBWEB (5.5 target concepts and 37% well-placed nodes) than for ARACHNE (6.6 target concepts and 52% well-placed nodes). Accuracy differences were significant at the .001 level for ARACHNE and COBWEB and at the .01 level for ARACHNE and COBWEB'. Differences in tree quality were significant at the .025 level.

These results only partly agree with our hypothesis. ARACHNE'S asymptotic accuracy is higher than COBWEB'S at both noise levels, with the difference increasing with noise. By predicting from well-formed internal nodes, ARACHNE is less susceptible to overfitting. However, the picture is more ambiguous with respect to tree quality. Both systems lose tree quality as noise increases, but ARACHNE suffers less than COBWEB, presumably because of its restructuring operators. This experiment lends evidence to the view that predictive ability and tree quality are not perfectly correlated. Good tree structure does not guarantee good prediction, and good predictive performance does not mean the underlying hierarchy is organized to contain concepts inherent in the data, but both are important factors.

3.6 Effects of Priming in Noisy Domains

Our explanation of the previous results supposed that noise in early training instances could mislead both systems, but that COBWEB was more susceptible to this effect than ARACHNE. If so, we should be able to eliminate this difference by *priming* both systems with well-ordered, noise-free training instances from each class. This is equivalent to giving them background knowledge about idealized categories. This suggests a third prediction about their behavior:

Hypothesis: ARACHNE *and* COBWEB *will behave similarly on both accuracy and tree structure when primed with well-ordered, noise-free training data.*

The intuition here is that, with less need to reorganize memory to recover from misleading observations, ARACHNE'S restructuring operators become less important and differences between the systems should be reduced. To test this prediction, we provided each system with 40 noise-free instances at the start of each run, four identical prototypes from each class, producing an idealized hierarchy. We followed these data with the same noisy training sets used for the second experiment.

Figure 3 shows the results of this experiment at noise-level II, which are somewhat surprising; results at noise-level I are analogous though less pronounced. Priming improves ARACHNE'S predictive accuracy significantly, raising it to 90% from 76% without priming. COBWEB shows an accuracy of only 63%, as did COBWEB'. This was an improvement over COBWEB'S former level of 55%,

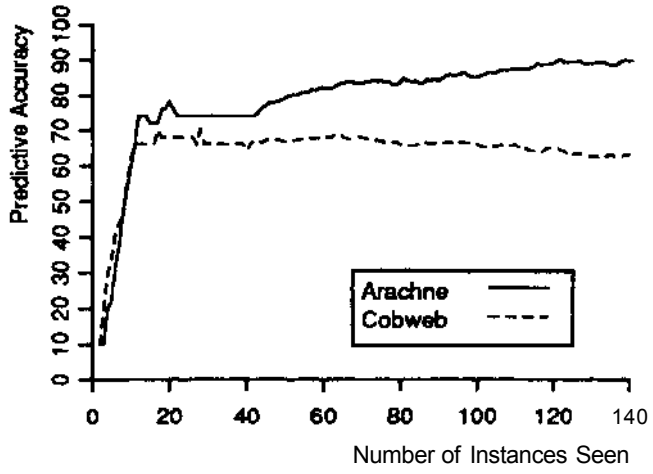

Figure 3. Learning curves for ARACHNE and COBWEB on noisy artificial data with primed concept trees.

but still below ARACHNE'S performance. For both systems, priming improves tree quality. ARACHNE'S quality, initially higher than COBWEB'S, improves from 52% well-placed nodes without priming to 89%; the 1alters tree quality rises from 37% to 79% with priming. Between-system differences in predictive accuracy were statistically significant at the .001 level; tree quality differences were significant at the .01 level.

This experiment disconfirms our hypothesis. Though priming generally improves the predictive accuracy of the systems, ARACHNE still performs significantly better than COBWEB. Furthermore, with priming ARACHNE builds better trees at both noise levels. This suggests that ARACHNE can make better use of the background knowledge encoded in a primed tree. This may be partly due to COBWEB'S greater tendency to misplace instances (incorporate them into an inappropriate concept), which affects both the accuracy of the concept description (it obtains superfluous noise) and presumably the system's ability to index and retrieve the object. Notably, although priming led both systems to higher accuracy initially, ARACHNE reached the same asymptote as without priming, whereas COBWEB'S accuracy actually dropped as it saw noisy training instances. This provides further evidence that ARACHNE benefits from its ability to predict from well-formed internal nodes, thus avoiding the overfitting to which COBWEB is susceptible.

3.7 Effects of Instance Order

Another one of our concerns in designing ARACHNE was stability with respect to different orders of training instances. Order effects are apparent in COBWEB when different presentations of the same data produce hierarchies with different structure. ARACHNE'S operators for merging and promotion should enable recovery from nonrepresentative training orders, and this leads to another prediction:

Hypothesis: COBWEB *will suffer more from order effects than* ARACHNE *w.r.t. tree quality but not accuracy.*

Gennari (1990) has reported that training order affects the structure of COBWEB trees but does not alter their accuracy, so we did not expect ARACHNE to outperform its predecessor on the latter measure. However, we did expect it to construct well-structured concept hierarchies regardless of the training order. Our experience with COBWEB suggested it has difficulty when every member of a class is presented at once, followed by every member of a new class, and so forth. Thus, we tested the above hypothesis by presenting both systems with ten random orderings and ten "bad" orderings of 200 training instances from the low-noise data set described earlier. The bad orderings were strictly ordered by class, so the systems saw 20 examples of each class in turn.

Table *1.* Tree quality in artificial domains, measured as percentage of well-placed nodes, for (1) unprimed and (2) primed runs, (3) poor and (4) random training orders, at low noise; for (5) unprimed and (6) primed runs, at high noise.

	Noise Level I				Noise Level II	
	(1)	(2)	(3)	(4)	(5)	(6)
ARACHNE	74%	95%	88%	86%	52%	89%
COBWEB	82%	89%	77%	84%	37%	79%

In this experiment our hypothesis was confirmed. Instance order did not affect predictive accuracy, although naturally the learning rate for the bad ordering was slower, since the systems did not see a representative of the final class until the 181st instance. Random and bad orderings produce hierarchies capable of analogous predictive accuracy, approximately 90% for ARACHNE and 80% for COBWEB. However, tree quality differs significantly in the two situations. Both systems locate most or all of the concepts at some level, but COBWEB is vulnerable to misplaced instances with the pathological ordering. Whereas ARACHNE arrives at 88% well-placed nodes with the bad ordering, and a similar 86% with random ordering, COBWEB averages only 74% well-placed nodes when learning from the bad ordering, compared to 84% for the random ordering. Differences in predictive accuracy were significant for both conditions at the .001 level. Differences in tree quality between the two systems were not significant for random orderings, but were significant at the .001 level for bad orderings.

This experiment provides additional evidence that predictive accuracy is not an adequate measure of tree quality. Apparently order effects that lead to a decrease in tree quality do not affect COBWEB'S ability to index instances and predict accurately. This is consistent with Gennari's results and with our hypothesis.

3.8 Cost of Classification

Analysis of COBWEB reveals an average-case assimilation cost that is logarithmic in the number of objects in the tree and quadratic in the branching factor (Fisher, 1987). Analysis of ARACHNE is confounded by the fact that in theory its restructuring of the hierarchy is not guaranteed to halt. Such cases are pathological and ARACHNE'S behavior on all our data sets was tractable.

To quantify each system's efficiency in practice, we measured the number of attribute values inspected as a function of n, the number of objects incorporated, over five runs of the soybean data set, the more challenging of the natural domains we tested. Both systems appeared linear in n; COBWEB with a correlation of 0.999 and ARACHNE with one of 0.976. However, ARACHNE actually inspected more attributes than COBWEB, having a linear coefficient thirty times that of its predecessor. We believe that a heuristic cutoff mechanism that, limits reorganization of the hierarchy would reduce cost without loss of predictive accuracy or tree quality.

4 Discussion

In this paper we identified some problems with Fisher's (1987) COBWEB, a concept formation system that achieves high predictive accuracy but does not always create well-structured concept trees. In response, we developed ARACHNE, an algorithm with explicitly-stated constraints for well-formed probabilistic concept hierarchies, which incrementally constructs such trees from unsupervised training data. We also reported four experiments that compared ARACHNE with COBWEB on both predictive accuracy and tree quality. We found the systems achieved comparable accuracy on two natural domains, but we found significant differences in both accuracy and tree quality using artificial data. In particular, COBWEB tends to overfit more than ARACHNE when trained and tested on noisy instances, and it benefits less from priming with noise-free data with respect to tree quality. Also, the quality of COBWEB trees suffers from misleading orders of training instances, while ARACHNE's tree structure is relatively unaffected.

Despite these encouraging results, we need more comparative studies before drawing firm conclusions about one system's superiority over the other. Also, ARACHNE has clear limitations that should be removed in future work. Preliminary studies suggest that the current similarity metric has difficulty distinguishing irrelevant attributes from relevant ones. However, since relevance can be estimated from stored conditional probabilities, we are confident that a different similarity function will add this capability. Also, the existing system can handle numeric attributes, but only by using Euclidean distance between means as its distance metric; future versions of ARACHNE should employ a probabilistic measure that lets it combine symbolic and numeric data in a unified manner. The system still tends to form "junk" concepts; avoiding them may require additional constraints or more powerful operators for restructuring the concept tree. We should also compare ARACHNE'S behavior to other noise-tolerant learning algorithms, including Fisher's (1989) variant on COBWEB and supervised methods for pruning decision trees (Quinlan, 1986).

Nevertheless, we believe the present work has shown the importance of examining the structural quality of concept hierarchies in addition to their predictive accuracy. It has also shown that explicit constraints on tree structure, combined with restructuring operators for correcting violated constraints, can produce well-formed trees with high predictive accuracy despite noise and misleading orders of training instances. We expect future work in this paradigm will lead to even more robust systems for incremental, unsupervised concept learning.

Acknowledgements

We thank W. Iba, J. Allen, K. Thompson, D. Kulkarni, and W. Buntine for discussions that led to many of the ideas in this paper. The above also provided comments on an earlier draft, as did M. Drummond and L. Leedom. J. Alien formatted the figures.

References

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

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.

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

Fisher, D. H. (1989). Noise-tolerant conceptual clustering. *Proceedings of the Eleventh International Joint ConferenceArtificial Intelligence* (pp. 825-830). Detroit: Morgan Kaufmann.

Fisher, D. H., & Pazzani, M. (Eds.) (in press). *Concept formation: Knowledge and experience in unsupervised learning.* San Mateo, CA: Morgan Kaufmann.

Gennari, J. H. (1990). *An experimental study of concept formation.* Doctoral dissertation, Department of Information &: Computer Science, University of California, Irvine.

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: Morgan Kaufmann.

Langley, P., Thompson, K., Iba, W., Gennari, J. H., &: Allen, J. A. (in press). An integrated cognitive architecture for autonomous agents. In W. Van De Velde (Ed.), *Representation and learning in autonomous agents.* Amsterdam: North Holland.

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning, 1*, 103-138.

Martin, J. D. (1989). Reducing redundant learning. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 396-399). Ithaca: Morgan Kaufmann.

Michalski, R. S., & Chilausky, R. L. (1980). Learning by being told and learning from examples. *International Journal of Policy Analysis and Information Systems, 4,* 125-160.

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

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

Van de Velde, W. (1990). Incremental induction of topological^ minimal trees. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 66-74). Austin: Morgan Kaufmann.