From *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining* (2001). ACM Press.

Generalized Clustering, Supervised Learning, and Data Assignment

Annaka Kalton
Pat Langley
ISLE / 2164 Staunton Court
Palo Alto, CA 94306
[akalton,langley]@isle.org

Kiri Wagstaff
Dept. of Computer Science
Cornell University
Ithaca, NY 14850
wkiri@cs.cornell.edu

Jungsoon Yoo
Computer Science Dept.
Middle Tenn. State University
Murfreesboro, TN 37132
csyoojp@mtsu.edu

ABSTRACT

Clustering algorithms have become increasingly important in handling and analyzing data. Considerable work has been done in devising effective but increasingly specific clustering algorithms. In contrast, we have developed a generalized framework that accommodates diverse clustering algorithms in a systematic way. This framework views clustering as a general process of iterative optimization that includes modules for supervised learning and instance assignment. The framework has also suggested several novel clustering methods. In this paper, we investigate experimentally the efficacy of these algorithms and test some hypotheses about the relation between such unsupervised techniques and the supervised methods embedded in them.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence–*Learning*

General Terms

Algorithms, design, experimentation

Keywords

Clustering, supervised learning, iterative optimization

1. INTRODUCTION AND MOTIVATION

Although most research on machine learning focuses on induction from supervised training data, there are many situations in which class labels are not available and which thus require unsupervised methods. One widespread approach to unsupervised induction involves *clustering* the training cases into groups that reflect distinct regions of the decision space. There exists a large literature on clustering methods (e.g., Everitt [3]), a long history of their development, and increasing interest in their application, yet there is still little understanding of the relation between supervised and unsupervised approaches to induction.

In this paper, we begin to remedy that oversight by examining situations in which a supervised induction method occurs as a subroutine in a clustering algorithm. This suggests two important ideas. First, one should be able to generate new clustering methods from existing techniques by replacing the initial supervised technique with a different supervised technique. Second, one would expect the resulting clustering methods to behave well (e.g., form desirable clusters) in the same domains for which their supervised components behave well, provided the latter have labeled training data available.

In the pages that follow, we explore both ideas in the context of iterative optimization, a common scheme for clustering that includes K-means and expectation maximization as special cases. After reviewing this framework in Section 2, we describe an approach to embedding any supervised algorithm and its learned classifier in an iterative optimizer, and in Section 3 we examine four supervised methods for which we have taken this step. In Section 4, we report on experimental studies designed to test our hypotheses about the relations between behavior of the resulting clustering methods and that of their supervised components. In closing, we review related work on generative frameworks for machine learning and consider some directions for future research.

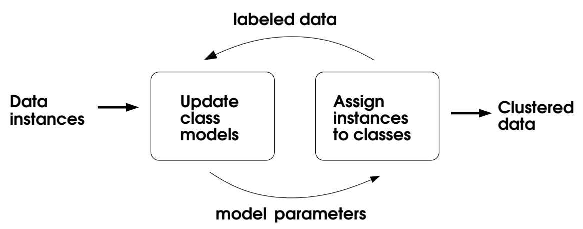

Figure 1: The iterative optimization procedure.

2. GENERALIZED CLUSTERING

Many clustering systems rely on the notion of *iterative optimization*. As Figure 1 depicts, such a system iterates between two steps – class model creation and data reassignment – until reaching a predetermined iteration limit or until no further changes occur in reassignments. There are many variations within this general framework, but the basic idea is best illustrated with some well-known example methods.

2.1 K-means and EM as Iterative Optimizers

Two clustering algorithms that are popular for their simplicity and flexibility are K-means [2] and expectation maximization (EM) [1]. Both methods have been studied experimentally on many problems and have been used widely in applied settings. Here we review the algorithms briefly, note their key similarities, and show how their differences suggest a more general clustering framework.

The K-means algorithm represents each class by a centroid, which it computes by taking the mean for each attribute over all the instances belonging to that class. In geometric terms, this corresponds to finding the center of mass for the cases associated with that class. Data reassignment involves assigning each instance to the class of the closest centroid.

In contrast, EM models each class by a probability distribution that it extracts from the training data in the class model creation step. If the data are continuous, each class is generally modeled by an n-dimensional Gaussian distribution that consists of a mean and variance for each attribute. In the discrete case, $P(a_j = v_{jl}|c_k)$ is extracted for each possible combination of class c_k, attribute a_j, and attribute value v_{jl}. In both cases, when finding these parameters, the contribution of each instance x_i is weighted by $P(c_k|x_i)$. Data reassignment is done by recalculating $P(c_k|x_i)$ for each instance x_i and class c_k using the new class models.

2.2 A General Framework

Although both of the above clustering algorithms incorporate iterative optimization, they employ different methods for developing class models. Thus, we can view them as invoking a different supervised learning technique to distinguish among the classes. The two algorithms also differ in how they assign instances to classes: K-means assigns each instance to a single class, whereas EM uses partial assignment, in that each instance is distributed among the classes. We will refer to the absolute method as the "strict" paradigm and to the partial method as "weighted".

These observations lead to a general framework for clustering that involves selecting a supervised learning algorithm and selecting one of these assignment paradigms. In the context of K-means and EM, this framework immediately suggests some variants. By using the weighted paradigm with the K-means classifier, we obtain a weighted K-means algorithm. Similarly, combining EM's probabilistic classifier with the strict paradigm produces a variant in which each instance is assigned entirely to its most probable class. This variant has been explored under the name of "strict-assignment EM", although the partial assignment method is more commonly used.

Although the classifiers utilized in K-means and EM can be easily modified to operate with either assignment method, other supervised algorithms can require more sophisticated adaptations, as we will see shortly.

3. SUPERVISED LEARNING METHODS

As we have argued, it should be possible to embed any supervised learning method within our generalized clustering framework. However, our evaluation has focused on four simple induction algorithms that have limited representational power [6], because the clustering process itself aims to generate the disjoint decision regions that more powerful supervised methods are designed to produce. Below we describe these algorithms in some detail, including the adaptations we made for use in the weighted paradigm. These adaptations involve altering model production to take into account the weights of instances and revising instance reassignment to generate class weights for every instance, which are then used to produce the next generation of class models.

3.1 Prototype Modeler

Our first supervised algorithm, which plays a role in K-means, creates a prototype [13] or centroid for each class by extracting the mean of each attribute from training cases for that class. Such a prototype modeler classifies an instance by selecting the class with the centroid closest to it in n-dimensional space. Because the distance metric is sensitive to variations in scale, our version normalizes all data to values between zero and one before creating the prototypes.

In the weighted paradigm, the mean for each attribute becomes a weighted average of the training cases. The relative proximity of each instance to a given centroid determines the associated weight for that centroid's class. Formally, we can express this by

$$w(x_i|c_k) = 1 - \frac{distance(x_i, prototype(c_k))}{\sum_{m=1}^{|C|} distance(x_i, prototype(c_m))} \ ,$$

where $|C|$ is the number of classes. The new centroid is then composed of the weighted mean for each attribute, with the mean of attribute a_j for cluster c_k being calculated by

$$\frac{\sum_{i=1}^{|X|} w(x_i|c_k) \cdot x_{ij}}{\sum_{i=1}^{|X|} w(x_i|c_k)} \ ,$$

where x_{ij} is the value of the jth attribute of instance x_i and where $|X|$ is the total number of instances.

3.2 Naive Bayesian Modeler

We selected naive Bayes [2] as our second induction algorithm. As described in the context of EM, this technique models each class as a probability distribution described by $P(c_k)$ and $P(a_j = v_{jl}|c_k)$ for each class c_k, attribute a_j, and attribute value v_{jl}. For nominal attributes, naive Bayes represents $P(a_j = v_{jl}|c_k)$ as a discrete conditional probability distribution, which it estimates from counts in the training data, and it estimates the class probability $P(c_k)$ in a similar manner. For continuous attributes, it typically uses a conditional Gaussian distribution that it estimates by computing the mean and variance for each attribute from training data for each class. To calculate the relative probability that a new instance belongs to a given class c_k, naive Bayes employs the expression

$$P(c_k|x_i) \propto P(c_k) \prod_j P(a_j = v_{jl}|c_k) \ ,$$

which assumes that the distribution of values for each attribute are independent given the class.

When operating normally as a strict classifier, naive Bayes returns the class with the highest probability for each instance. In the weighted case, the conditional distributions are calculated using a weighted sum rather than a strict sum, while the expression

$$w(x_i|c_k) = \frac{P(c_k|x_i)}{\sum_{m=1}^{|C|} P(c_m|x_i)}$$

determines the weight used in the data reassignment process.

3.3 Perceptron List Modeler

Another simple induction method, the perceptron algorithm [12], also combines evidence from attributes during classification, but uses the expression

$$output = \begin{cases} 1 & \text{if } \sum_{m=1}^{|A|} w_m \cdot x_{im} > threshold \\ 0 & \text{otherwise} \end{cases}$$

to assign a test case to the positive (1) or negative (0) class. Each weight w_m specifies the relative importance of an attribute m; taken together, these weights determine a hyperplane that attempts to separate the two classes. The learning algorithm invokes an error-driven scheme to adjust the weights associated with each attribute.[1] Because a perceptron can only differentiate between two classes, we employed an ordered list of perceptrons that operates much like a decision list. The algorithm first learns to discriminate between the majority class and others, generating the first perceptron. Instances in the majority class are removed, and the system trains to distinguish the new majority class from the rest, producing another perceptron. This process continues until one class remains, which is treated as a default.

Although the perceptron traditionally assumes all-or-none assignment, it seems natural to interpret the scaled difference between the sum and the threshold as a likelihood. The weighted variant multiplies the update for each attribute weight by the weight for each instance, so that an instance with a smaller weight has a smaller effect on learning. To prevent small weights from causing endless oscillations, it triggers an updating cycle through the data only if an incorrectly classified instance has a weight of greater than 0.5, although all instances are used for the actual update.

In reassignment, the weighted method calculates the difference between the instance value and the threshold, scaled by the sigmoid

$$w(x_i|c_k) = \frac{1}{1+e^{5\cdot(threshold-sum)}} ,$$

which produces bounds on the weight size. If an instance were evaluated as being perfectly at the threshold, the function would return 0.5. The factor 5 in the exponent of e distributes the resulting weights over a larger range, so the algorithm will not give a weight close to 0.5 for all instances. Otherwise the sigmoid is not tight enough to be useful for a generally small range of values.

3.4 Decision Stump Modeler

For our final supervised learning algorithm, we selected decision-stump induction (e.g., Holte [4]), which differs from the others in selecting a single attribute to classify instances. To this end, it uses the information-theoretic measure

$$info(S) = -\sum_{k=1}^{|C|} \frac{freq(c_k,S)}{|S|} \cdot log_2\left(\frac{freq(c_k,S)}{|S|}\right) ,$$

where $freq(c_k, S)$ is the frequency of class c_k in a training set S with $|C|$ classes. If the attribute is continuous, the algorithm orders its observed values and considers splitting between each successive pair, selecting the split with the highest score. The method applies this process recursively to the values in each subset, continuing until further divisions gain no more information, as measured by

$$gain = info(T) - \sum_{m=1}^{|P|} \frac{|T_m|}{|T|} \cdot info(T_m) ,$$

where T is the training set, T_m is a given subset of T, and $|P|$ is the number of branches. If the attribute is nominal, the algorithm creates a separate branch for each attribute value. Each branch of the stump is then associated with the majority class of those training cases that are sorted to that branch.

To accommodate weighted assignment, we adjust the equations above to sum over the weights of instances, rather than over strict frequencies, and keep simple statistical information for each branch. The reassignment weight given to each instance for class c_k is calculated by

$$w(x_i|c_k) = \frac{\sum_{m=1}^{|B|} w(x_m|c_k)}{\sum_{n=1}^{|C|}\sum_{m=1}^{|B|} w(x_m|c_n)} ,$$

where $|B|$ is the number of instances associated with the branch to which that instance is sorted.

4. EXPERIMENTAL STUDIES

We had two intuitions about our clustering framework that suggested corresponding formal hypotheses.[2] First, we expected that each algorithm would exhibit a "preference" for one of the data assignment paradigms by demonstrating better performance in that paradigm across different data sets. Second, we anticipated that, across data sets, high (low) predictive accuracy by a supervised method would be associated with relatively high (low) accuracy for the corresponding clustering algorithm. In this section, we describe our designs for the experiments to test these hypotheses and the results we obtained.

4.1 Experiments with Natural Data

To test these hypotheses, we ran the generalized clustering system with each algorithm-paradigm combination on a battery of natural data sets. We also evaluated each supervised algorithm independently by training it and measuring its predictive accuracy on a separate test set. The independent variables were the assignment paradigm (for the clustering tests), the supervised learning algorithm, the data set, and the number of instances used in training. The dependent variables were the classification accuracies on unseen data.

We used a standard accuracy metric to evaluate both the supervised classifiers and the clustering algorithms:

$$accuracy(T) = \frac{\sum_{x_i \in T} \delta(x_i)}{|T|},$$

where T is the test set, and where $\delta(x_i) = 1$ if x_i is classified correctly and 0 otherwise.

When evaluating accuracy, we trained each classifier on the labeled data set with the test set removed. Because the clustering algorithms create their own classes, we added a step in which each completed cluster is assigned the actual

[1]For the purposes of this study, we used a learning rate of 0.05 and 50 iterations through the training data, which did well on all our classification tasks.

[2]Naturally, we also expected that no single algorithm combination would outperform all others on all data sets, but this is consistent with general findings in machine learning, and so hardly deserves the status of an hypothesis.

Table 1: Supervised accuracies on four data sets.

	Prototype	Bayes	Perceptron	Stump
Promoters	**86.0**	**87.0**	76.0	70.0
Iris	49.3	**94.7**	46.0	**93.3**
Hayes-Roth	32.3	61.5	**79.2**	43.1
Glass	84.8	79.0	39.0	**97.6**

class of its majority population. For example, if a given cluster consists of 30 instances that are actually class A and 10 that are actually class B, all instances in the cluster will be declared members of class A, with an accuracy of 75% for that cluster. This approach loses detail, but it let us evaluate each clustering algorithm against the "correct" clusters.

We selected four data sets from the UCI repository – Promoters, Iris, Hayes-Roth, and Glass – that involved different numbers of classes (two to seven), different numbers of attributes (five to 57), and different attribute types (nominal, continuous, or mixed). Another factor in their selection was that each led to high classification accuracy for one of the supervised methods but (typically) to lower accuracy for the others, as shown with bold font in Table 1. This differentiation on supervised training data seemed a prerequisite for testing the predicted correlation between accuracies for supervised learning and clustering.

Moreover, remember that our four supervised methods each has restricted representational power that is generally limited to one decision region per class. As a result, the fact that one such method obtains high accuracy in each of these domains suggests that each of their classes maps onto to a single cluster. This lets us assume that the number of classes in each data set corresponds to the number of clusters, further increasing the chances of meaningful results.

For each data set, we collected a learning curve using ten-fold cross-validation, recording results for each increment of 25 data points. Typically, clustering accuracy ceased to improve early in the curve, although the supervised accuracy often continued to increase. The results we report here all involve accuracy as measured at the last point on each curve.

Table 2: Unsupervised accuracies for two alternative data assignment paradigms (strict/weighted).

	Prototype	Bayes	Perceptron	Stump
Promoters	62.0/**77.0**	**52.0**/41.0	49.0/**57.0**	19.0/**26.0**
Iris	27.3/**51.3**	83.3/**88.0**	26.7/**32.0**	**55.3**/53.3
Hayes-Roth	37.7/**39.2**	30.0/**40.0**	38.5/38.5	34.6/**36.2**
Glass	**84.8**/51.0	44.8/**61.9**	26.2/**34.3**	**77.1**/74.3

Recall that our first hypothesis predicted each supervised method would construct more accurate clusters when combined with its preferred data assignment paradigm. The results in Table 2, which shows the classification accuracies for each method-paradigm combination on the four domains, disconfirms this hypothesis. In general, each supervised algorithm sometimes did better with one assignment scheme and sometimes with the other, depending on the domain. Both naive Bayes and the prototype learner showed

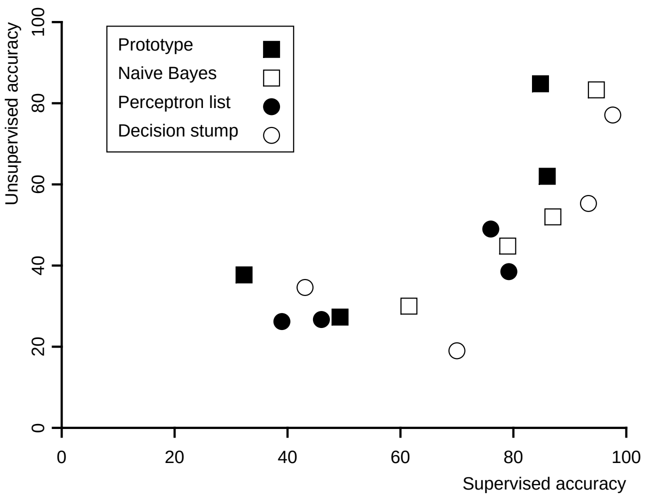

Figure 2: Supervised and unsupervised accuracies, using strict data assignment, for four algorithms on four natural data sets ($r = 0.745$).

large shifts of this sort, though swings for the decision-stump learner were less drastic. Only the perceptron list method showed any support for our prediction, favoring weighted assignment on three data sets and a tied result on the fourth.

After addressing our first hypothesis, we proceeded to test our second claim, that relatively higher (lower) accuracy in supervised mode is associated with relatively higher (lower) accuracy on unsupervised data, i.e., that they are correlated positively. Our original plan was to measure the unsupervised accuracy of each learning algorithm when combined with its preferred data assignment paradigm. Having rejected the notion of such preference, we resorted instead to measuring the relation between supervised accuracy and that achieved by clustering with strict assignment, followed by a separate measure between the accuracy of supervised learning and weighted assignment.

To this end, we computed the correlation between the supervised accuracies using the 16 algorithm-domain combinations in Table 1 and the analogous strict accuracies from Table 2. The resulting correlation coefficient, $r = 0.745$, was significant at the 0.01 level and explained 55 percent of the variance. Figure 2 shows that supervised accuracy is a reasonable predictor of unsupervised accuracy, thus generally supporting our hypothesis. We also calculated the correlation between supervised accuracies and the weighted accuracies from Table 2. Here the correlation was $r = 0.653$, which was also significant at the 0.01 level and explained 43 percent of the variance.

4.2 Experiments with Synthetic Data

Our encouraging results with natural data sets show that our framework has relevance to real-world clustering problems, but they can give only limited understanding for the reasons underlying the phenomena. For this reason, we decided to carry out another study that employed synthetic data designed to reveal the detailed causes of these effects.

One standard explanation for some induction methods outperforming others relies on the notion of *inductive bias*, which reflects the fact that some formalisms can represent certain decision regions more easily than others. Since our four supervised learning methods have quite different inductive biases, we designed four separate learning tasks, each intended to be easily learned by one of these methods but not by others.

Each learning task incorporated two continuous variables and three classes, with a single contiguous decision region for each class. Thus, the domain designed with decision stumps in mind involved splits along one relevant attribute, the prototype-friendly domain involved three distinct prototypes, and so forth. The naive Bayesian classifier is difficult to foil, but for every other supervised method, we had at least one domain on which it should do relatively poorly. For each domain, we devised a generator that produced 125 random instances from either a uniform or, for the Bayes-friendly domain, a Gaussian distribution for every class, creating the same number of instances for each one.

The geometric metaphor clarifies one reason that a given method should outperform others in both supervised and unsupervised mode, but it also suggests a reason why the correlation between behavior on these two tasks is imperfect. Conventional wisdom states that clustering is easy when clusters are well separated but difficult when they are not. Thus, our data generator also included a parameter S that let us vary systematically the separation between the boundaries of each class. The predictive variables for each domain ranged from 0 to 1, so we varied the separation distance from $S = 0$ to $S = 0.24$.

Although we expected our synthetic domains to reproduce the positive correlation we observed with natural data, we also predicted that cluster separation should influence this effect. In particular, we thought the correlation would be lower when the gap was small, since iterative optimization would have difficulty assigning instances to the "right" unlabeled classes, whereas supervised learning would have no such difficulty. However, the correlation should increase monotonically with cluster distance, since the process of finding well-separated clusters should then be dominated by the inductive bias of the supervised learning modules.

Our experimental runs with synthetic data did not support these predictions.[3] Despite our attempts to design data sets that would distinguish among the supervised learning methods, correlations between supervised and unsupervised accuracies when cluster separation $S = 0$ were considerably lower ($r = 0.488$ for strict and $r = 0.377$ for weighted) than for our studies with natural domains, though still marginally significant at the 0.1 level. Moreover, our experiments showed no evidence that correlation increases with cluster separation, giving $r = 0.413$ for strict and $r = 0.337$ for weighted when $S = 0.12$, and giving $r = 0.462$ for strict and $r = 0.219$ for weighted when $S = 0.24$.

Figure 3, which plots the accuracies for strict unsupervised learning against supervised accuracy when cluster separation $S = 0$, suggests one reason for this negative result. Apparently, the correlations are being reduced by a "ceiling effect" in which the supervised accuracies (generally much higher than for our results on natural domains) show little variation, whereas the unsupervised accuracies still range widely. The supervised methods typically learn very accurate classifiers across all four synthetic domains, even though

[3]This study also revealed no evidence for a preferred data assignment scheme, with the best combinations shifting across both domain and separation level.

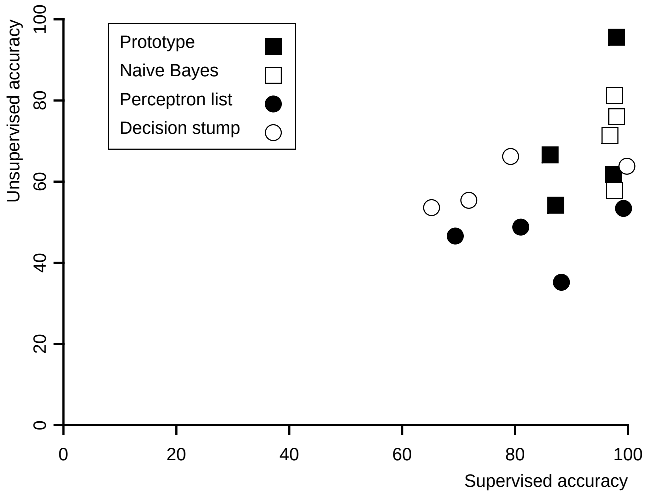

Figure 3: Supervised and unsupervised accuracies, using strict data assignment, for four algorithms with $S = 0$ on four synthetic data sets ($r = 0.488$).

we did our best to design them otherwise. Analogous plots for higher values of the separation parameter S show even stronger versions of this effect, indicating that supervised induction benefits more from cluster separation than does unsupervised clustering, which explains why the correlation does not increase as predicted.

Our expectations rested on the intuition that inductive bias and cluster separation are the dominant factors in determining the behavior of an iterative optimizer. From these negative results, and from the high correlations on natural domains, we can infer that other factors we did not vary in this experiment play an equal or more important role. Likely candidates include the number of relevant attributes, the number of irrelevant attributes, the amount of attribute noise, and the number of classes, all of which are known to affect the predictive accuracy of learned classifiers. These domain characteristics should be varied systematically in future studies that draw on synthetic data to explore the relation between clustering and supervised learning.

5. RELATED WORK

As we noted earlier, there exists a large literature on clustering that others (e.g., Everitt [3]) have reviewed at length. Much of this work relies on iterative optimization to group training cases, and there exist many variants beyond the K-means and expectation-maximization algorithms familiar to most readers. For instance, Michalski and Stepp's [11] Cluster/2 used logical rule induction to characterize its clusters and assign cases to them. More recently, Zhang, Hsu, and Dayal [15] have described the K-harmonic means method, which operates like K-means but invokes a different distance metric that usually speeds convergence. However, despite this diversity, researchers have not proposed either theoretical frameworks for characterizing the space of iterative optimization methods or software frameworks to support their rapid construction and evaluation.

In the broader arena, there have been some efforts to link methods for supervised and unsupervised learning. For example, Langley and Sage [8] adapted a method for inducing univariate decision trees to operate on unsupervised data and thus generate taxonomy, and, more recently, Langley [6] and Liu et al. [9] have described similar but more sophisticated approaches. The relationship between supervised and unsupervised algorithms for rule learning is more transparent; Martin [10] has reported one approach that adapts supervised techniques to construct association rules from unlabeled data. But again, such research has focused on specific algorithms rather than on general or generative frameworks.

However, other areas of machine learning have seen a few frameworks of this sort. Langley and Neches [7] developed PRISM, a flexible language for production-system architectures that supported many combinations of performance and learning algorithms, and later versions of PRODIGY [14] included a variety of mechanisms for learning search-control knowledge. For classification problems, Kohavi et al.'s [5] MLC++ supported a broad set of supervised induction algorithms that one could invoke with considerable flexibility. The generative abilities of MLC++ are apparent from its use for feature selection and its support for novel combinations of existing algorithms. This effort comes closest to our own in spirit, both in its goals and its attempt to provide a flexible software infrastructure for machine learning.

6. CONCLUDING REMARKS

In this paper, we presented a framework for iterative optimization approaches to clustering that lets one embed any supervised learning algorithm as a model-construction component. This approach produces some familiar clustering techniques, like K-means and EM, but it also generates some novel methods that have not appeared in the literature. The framework also let us evaluate some hypotheses about the relation between the resulting clustering methods and their supervised modules, which we tested using both natural and synthetic data.

Our first hypothesis, that each supervised method had a preferred data assignment scheme with which it produced more accurate clusters, was not borne out the experiments. Clustering practitioners can continue to combine prototype learning with strict assignment (giving K-means) and naive Bayes with weighted assignment (giving EM), but we found no evidence that these combinations are superior to the alternatives. However, our experiments did support our second hypothesis by revealing strong correlations between the accuracy of supervised algorithms on natural data sets and the accuracy of iterative optimizers in which they were embedded. We augmented these results with experiments on synthetic data, which gave us control over decision regions and separation of clusters. These studies also produced positive correlations between supervised and unsupervised accuracy, but failed to reveal an effect of cluster separation.

Clearly, there remains considerable room for additional research. The framework supports a variety of new clustering algorithms, each interesting in its own right but also important for testing further our hypotheses about relations between supervised and unsupervised learning. We should also carry out experiments with synthetic data that vary systematically other factors that can affect predictive accuracy, such as irrelevant features and attribute noise. Finally, we should explore further the role of cluster separation and the reason it had no apparent influence in our studies.

Although our specific results are intriguing, we attach more importance to the framework itself, which supports a new direction for studies of clustering mechanisms. We encourage other researchers to view existing techniques as examples of some generative framework and to utilize that framework both to explore the space of clustering methods and to reveal underlying relations between supervised and unsupervised approaches to induction. Ultimately, this strategy should produce a deeper understanding of the clustering process and its role in the broader science of machine learning.

7. REFERENCES

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, Series B, 1977.

[2] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[3] B. Everitt. *Cluster Analysis*. Halsted Press, New York, 2nd edition, 1980.

[4] R. C. Holte. Very simple classification rules perform well on most commonly used data sets. *Machine Learning*, 3:63–91, 1993.

[5] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 740–743, IEEE Computer Society Press, 1994.

[6] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1996.

[7] P. Langley and R. T. Neches. PRISM user's manual. Technical report, Carnegie-Mellon University, Department of Computer Science, Pittsburgh, PA, 1981.

[8] P. Langley and S. Sage. Conceptual clustering as discrimination learning. In *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 95–98, London, Ontario, Canada, 1984.

[9] B. Liu, Y. Xia, and P. Yu. Clustering through decision tree construction. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, Washington, DC, 2000.

[10] J. D. Martin. Focusing attention for observational learning: The importance of context. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989. Morgan Kaufmann.

[11] R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, San Francisco, CA, 1983.

[12] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, Washington, DC, 1962.

[13] E. E. Smith and D. L. Medin. *Categories and Concepts*. Harvard University Press, Cambridge, MA, 1981.

[14] M. M. Veloso and J. G. Carbonell. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, 10:249–278, 1993.

[15] B. Zhang, M. Hsu, and U. Dayal. K-harmonic means – A spatial clustering algorithm with boosting. In *Proceedings of the PKDD-2000 Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining*, 2000.