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. This 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 component behaves well, provided the latter has 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.

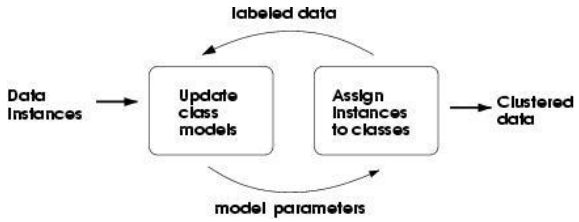

Figure 1: The iterative optimization procedure.

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 change in reassignments occur. There are many variations within this general framework; but the basic idea is best illustrated with some well-known example methods.

KDD 2001 San Francisco, CA USA

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 widely used in applied settings. Below 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 of the data points belonging to that class. In geometric terms, this corresponds to finding the center of mass for the instances 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 a n-dimensional Gaussian distribution that consists of a mean and variance for each attribute. In the discrete case, $P(c_k|a_j = v_{jl})$ is extracted for each possible combination of class c_k, attribute a_j, and attribute value l for a_j (v_{jl}). In both cases, when finding these parameters, the contribution of each instance x_i is weighted by $P(x_i \in c_k)$. Data reassignment is done by recalculating $P(x_i \in c_k)$ for each instance x_i and class c_k using the new class models.

2.2 A General Framework

Although both of these clustering algorithms are based on iterative optimization, they employ different methods for developing class models. Thus, we can view them as invoking a different supervised learning algorithm 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 the "weighted" paradigm.

These observations lead to a general framework for clustering algorithms 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 strictly 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 paradigm, 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 within our generalized clustering framework. We selected four simple algorithms for evaluation, which we describe below 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 give a weight for each class to every instance, which are then used to produce the next generation of the class models.

3.1 Prototype Modeler

As described in the context of K-means, the prototype modeler [13] creates a prototype or centroid for each class by extracting the mean of each attribute from data points belonging to that class. Classification of an instance is done by selecting the class with the centroid closest to it in n-dimensional space. Because the distance metric is highly 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, where the mean of attribute a_j for cluster c_k is calculated by

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

where x_{mj} is the value of the jth attribute of instance x_m.

3.2 Naive Bayesian Modeler

Naive Bayes [2] is a simple probabilistic learning algorithm. As described in the context of EM, each class is modeled by a probability distribution described by $P(c_k)$ and $P(c_k|a_j = v_{jl})$ for each class c_k, attribute a_j, and attribute value v_{jl}. For nominal attributes, naive Bayes represents $P(c_k|a_j = v_{jl})$ 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 probability that a new instance belongs to a given class c_k, naive Bayes employs the expression

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

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

When operating as a strict classifier, the naive Bayes algorithm returns the class with the highest probability for each instance. In the weighted case, the mean and variance are found using a weighted sum rather than a strict sum, while the expression

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

determines the weight used in the $w(x_i|c_k)$ in the reassignment process.

3.3 Decision Stump Modeler

A decision stump [4] relies on a single attribute to classify instances. The associated induction algorithm selects this attribute using the information-theoretic measure

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

If the attribute is continuous, the algorithm generates up to $|C|$ different partitions of that attribute. It decides how much to partition the data by the information gained in that partition:

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

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

To accommodate weighted assignment, we adjust the information gain equation 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 branch of the decision stump to which that instance is sorted.

3.4 Perceptron List Modeler

The perceptron modeler is not strictly a simple perceptron but is based on a list of simple perceptrons [12]. During classification, a perceptron uses the expressions:

$$output(P) = \begin{cases} 0 & \text{if } sum < threshold \\ 1 & \text{otherwise} \end{cases}$$

and

$$sum = \sum_{m=1}^{|A_i|} x_{im} \cdot w_m \ ,$$

where w_m is a weight that specifies the relative importance of attribute m. The learning algorithm operates by adjusting the weights associated with each attribute.[1] The perceptron is limited to differentiating between two classes, so we employed an ordered list of perceptrons that operates much like a decision list. The algorithm first learns to differentiate between the majority class and others, which produces the first perceptron. Instances belonging to the majority class are removed, and the system trains to discriminate 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 weight of each instance by the learning rate before updating the model, so that an instance with a small weight has only a small effect. 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 a sigmoid:

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

that produces bounds on the weight size. If an instance were evaluated as being perfectly at the threshold, the function would return 0.5. The extra exponential term in e causes a more extreme assignment. Otherwise the sigmoid was not tight enough to be useful for the generally small range of values. Because the perceptron is embedded in a list, the class weights are not guaranteed to sum to one.

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 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 paradigm/algorithm combination on a battery of natural data sets. We also evaluated each supervised algorithm independently by training it and calculating 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:

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

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

[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 a wide variety of 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.

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

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 was assigned the actual class of its majority population. For example, if a given cluster consists of 20 instances that are actually class A and 10 that are actually class B, all instances in the cluster would be declared members of class A. This approach loses evaluation detail, but it let us evaluate each clustering algorithm against the "correct" clusters.

We selected four data sets from the UCI repository – Promoters, Iris, Glass, and Hayes-Roth – that involved different numbers of classes, different numbers of attributes, and different attribute types (nominal, continuous, or mixed). Another factor in their secltion was that each led to high classfication accuracy for one of the supervised methods but (typically) to lower accuracy for the others, as shown 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, note 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 corresponds to a single cluster. This lets us assume that the number of classes in each data set corresponded 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. For the two largest data sets, we limited the number of training cases to 500, having observed that classification accuracy generally leveled off after two or three hundred instances. Typically, clustering accuracy ceased to improve early in the learning curve, although the supervised classification accuracy often continued to improve.

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

	Bayes	Proto	Percept	Stump
Iris	10.0/10.0	10.0/10.0	10.0/10.0	10.0/10.0
Promoters	10.0/10.0	10.0/10.0	10.0/10.0	10.0/10.0
Hayes-Roth	10.0/10.0	10.0/10.0	10.0/10.0	10.0/10.0
Glass	10.0/10.0	10.0/10.0	10.0/10.0	10.0/10.0

Recall that our first hypothesis predicted that 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, generally support this hypothesis. In particular, the prototype learner and perceptron lists typically fared better when combined with strict assignment, whereas naive Bayes and decision stumps did better with weighted assignment. All but decision stumps showed a substantial difference in their average accuracies across the four data sets.

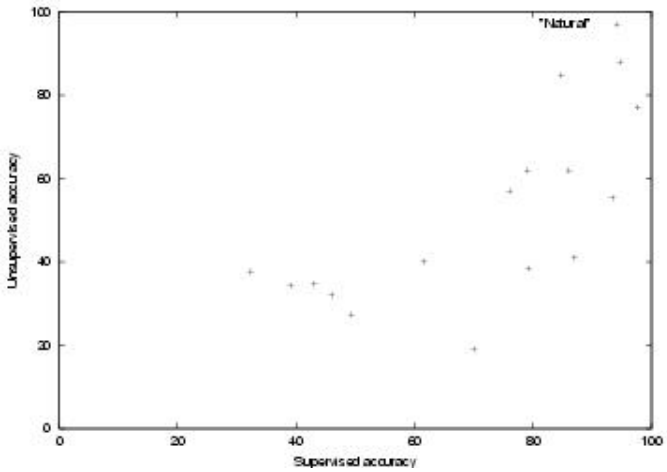

Figure 2: Supervised and unsupervised accuracies for four algorithms on four natural data sets.

Using the preferred assignment paradigm for each induction algorithm, we proceeded to test our second hypothesis, that higher (lower) accuracies in supervised mode would be associated with higher (lower) accuracies on unsupervised data. To this end, we computed the correlation between these two accuracies over the 16 algorithm-domain combinations. The resulting correlation coefficient, $r = 0.49$, was significant at the *** level and explains 70 percent of the variance. Figure 2 shows that supervised accuracy is a reasonable predictor of unsupervised accuracy, thus generally supporting our hypothesis.

4.2 Experiments with Synthetic Data

Our encouraging results with natural data sets shows that our framework has relevance to real-world clustering problems, but they can give only limited understanding for the causes 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 representational 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 involved two continuous variables and three classes, with a single contiguous decision region for each class. Figure 3 (a) shows the decision regions for the domain designed with decision stumps in mind, whereas Figure 3 (b) shows the regions for a domain that the prototype method should easily learn. Some algorithms, especially naive Bayes, are difficult to foil, but for every supervised

method, we had at least one domain on which it should do poorly. For each domain, we devised a generator that produced *** random instances from a uniform distribution for each class, creating the same number of instances for each class.

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 that let us vary systematically vary the separation between the instances for each class. The predictive variables for each domain ranged from *** to ***, so we varied the separation distance from *** to ***.

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 low when the gap was small, since iterative optimization would have difficulty assigning instances to the "right" unlabeled classes. However, the correlation should increase monotonically with cluster distance, since the process of finding well-separated cluster should be dominated by the inductive bias of the supervised learning modules.

Here we need some paragraphs that describe our results on the synthetic data, maybe including a table with accuracies but especially a graph that plots correlation as a function of cluster separation.

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 [10] Cluster/2 uses logical rule induction to characterize its clusters and assign cases to them. More recently, Zhang et al. [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] has described a similar but more sophisticated approach. The relationship between supervised and unsupervised algorithms for rule learning is more transparent; Martin [9] has reported one approach that adapts supervised techniques to let them 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 tasks, Kohavi et al.'s MLC++ [5] supported a broad set of supervised induction algorithms that one could invoke with considerable flexibility. The generative abilities of this framework are apparent from its use in feature selection and its support for novel combinations of existing algorithms. The MLC++ 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 the 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.

As we predicted, each supervised method had a preferred data assignment scheme with which it produced more accurate clusters. In particular, clustering practitioners will be pleased to know that strict assignment works best for the prototype learner and weighted assigment for naive Bayes, which corresponds to the popular K-means and EM algorithms. Moreover, we found a strong correlation 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 by running experiments on synthetic data, which gave us control over decision regions and separation of classes. These studies also produced a correlation 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 intereting in its own right but also important for testing further our hypotheses about relations between supervised and unsupervised learning. We should also carry out additional experiments with synthetic data that vary systematically other factors that can affect predictive accuracy, such as irrelevant features and attribute noise. Finally, we should also 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 a 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. *Tools with Artificial Intelligence*, chapter MLC++: A machine learning library in C++, pages 740–743. IEEE Computer Society Press, 1994.

[6] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 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] 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.

[10] 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.

[11] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[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, 1993.

[15] B. Zhang, M. Hsu, and U. Dayal. K-harmonic means - a spatial clustering algorithm with boosting. In *Temporal, Spatial, and Spatio-Temporal Data Mining Workshop at PKDD 2000*, 2000.