Adaptive Clustering: Better Representatives with Reinforcement Learning

Abraham Bagherjeiran, Christoph F. Eick, Ricardo Vilalta

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number UH-CS-05-06

18th March 2005

Keywords: supervised clustering, meta-learning, reinforcement learning

Abstract

Adaptive clustering uses reinforcement learning to learn the reward values of successive data clusterings. Adaptive clustering applies when external feedback exists for a clustering task. It supports the reuse of clusterings by memorizing what worked well in a previous context. It explores multiple paths in a reinforcement learning environment when the goal is to find better cluster representatives based on arbitrary environmental feedback. Our experiments apply adaptive clustering to instance-based learning relying on a distance function modification approach. The results show that adaptive clustering can find better representatives, if compared with traditional instance-based learning, such as k-nearest neighbor classifiers. Moreover, we introduce as a by-product a new instance-based learning technique that classifies examples by solely using cluster representatives; the technique shows high promise in our experimental evaluation.

Adaptive Clustering: Better Representatives with Reinforcement Learning

Abraham Bagherjeiran, Christoph F. Eick, Ricardo Vilalta

Abstract

Adaptive clustering uses reinforcement learning to learn the reward values of successive data clusterings. Adaptive clustering applies when external feedback exists for a clustering task. It supports the reuse of clusterings by memorizing what worked well in a previous context. It explores multiple paths in a reinforcement learning environment when the goal is to find better cluster representatives based on arbitrary environmental feedback. Our experiments apply adaptive clustering to instance-based learning relying on a distance function modification approach. The results show that adaptive clustering can find better representatives, if compared with traditional instance-based learning, such as k-nearest neighbor classifiers. Moreover, we introduce as a by-product a new instance-based learning technique that classifies examples by solely using cluster representatives; the technique shows high promise in our experimental evaluation.

Index Terms

supervised clustering, meta-learning, reinforcement learning

I. Introduction

A clustering algorithm finds groups of objects in a pre-defined attribute space. Since the objects have no known group membership, clustering is an unsupervised learning technique. A clustering algorithm optimizes some explicit or implicit criterion inherent to the data. The squared summed error, for example, is a criterion whose optimization is the primary concern of the k-medioids [12] clustering algorithm. A large amount of work (Section II) details the limitations of these algorithms: the main criticism is that these criteria are excessively simplistic and do not accurately capture the user's conception of the true data essence. The purpose of this paper is to introduce adaptive clustering as a solution to this problem.

Adaptive clustering applies when one seeks a clustering algorithm that can meet objectives beyond the simple least-squares criteria. Adaptive clustering makes the clustering process sensitive to external rewards though the use of reinforcement learning [11]. Moreover, it supports memorizing clusterings that worked well in a given context, supporting the reuse of "good" clusterings that were found in the past. This approach differs from that of both unsupervised and supervised learning. Although the objects have no class label, adaptive clustering seeks to find good clusters based on feedback with respect to previously found clusterings.

Adaptive clustering applies when external feedback exists for a clustering task. Applications include interactive clustering for summary generation, clustering for streaming data and multi-agent coordination and control. In interactive summary generation, such as spatial clustering, adaptive clustering improves the clustering in response to user-provided feedback regarding the quality of the obtained clusters. When mining data streams, adaptive clustering retains a model of the clustering that it can reuse as new objects arrive without re-clustering all the data. For multi-agent control, a clustering represents the properties of groups of agents, and the adaptive clustering algorithm uses external feedback to modify the clusters and thereby the agents toward some externally specified goal. A relatively simple but illustrative application is in classification where adaptive clustering is used to enhance an instance-based classifier.

II. Related Work

A. Clustering

1) Unsupervised Clustering: A traditional unsupervised clustering algorithm maximizes a known criterion function given the data. One such criterion is the squared summed error. Given a set of n objects O the algorithm

defines a set of k clusters $X = \{c_1, \ldots, c_k\}$ each with a representative object $\bar{c_i}$ such that the error:

$$E(X) \quad = \quad \sum_{i=1}^{k} \sum_{o \in O} (o - \bar{c_i})^2$$

is a minimum. The representative objects that minimize the above error are the mean vectors of clusters of the data. The k-means algorithm [14] uses centroids as cluster representatives, and the k-medioids algorithm takes its representatives from the original data.

The k-means and other partitioning algorithms typically use the Euclidean or Manhattan distance metrics. Many extensions to these and other partitioning algorithms attempt to employ more sophisticated distance metrics. Another group of approaches attempts to learn the distance metrics. The approach explored in this paper learns attribute weights with respect to the following object distance function d:

$$d(o_i, o_j) \quad = \quad \sum_{l} w_l \left| o_{i_l} - o_{j_l} \right|$$

where o_i and o_j are l-dimensional objects and $\mathbf{w}$ is a weight vector whose components are non-negative and $\sum_l w_l = 1$. The absolute value in the above formula indicates the original component-wise distance. The distance, then, is a weighted combination of the difference between two objects; when all the weights are equal the distance is exactly the Manhattan distance.

2) Supervised Clustering: Although supervised clustering has many alternative names, the algorithms all have access to the known class of the objects in the data set. As a classification algorithm, supervised clustering is simply a (spatially-oriented) classifier. Many of the algorithms adapt distance functions with the goal of obtaining a better clustering. The LVQ algorithm [13] modifies a distance function based on the performance of the classifier. A generalization to this approach adapts the distance metric to better separate examples within classes [10]. Another approach uses support vector machines [5] to learn to separate classes and then uses this separation to generate a better distance function [6]. Other methods directly integrate supervised learning with a traditionally unsupervised clustering algorithm using fitness functions that are based on the class purity within a cluster [7].

3) Semi-Supervised Clustering: In between unsupervised and supervised clustering, semi-supervised clustering algorithms adapt the clustering based on background knowledge. Most work in this area relies on the preferences of a user to inform the algorithm whether or not the clustering is useful. One example puts the clustering algorithm in a content-based image retrieval setting so that users can tell if two images should or should not belong to the same cluster. Based on a series of evaluation examples, the algorithm can adapt the clusters to meet the objectives [4]. Similar work learns a distance metric based on pairs of well-separated examples. Each pair of examples has a distance label that indicates that these examples should be far apart or close together. With this information [17], the distance function is adapted so that the shape of the clusters satisfies these constraints.

B. Reinforcement Learning

Reinforcement learning [11] (RL) lies between unsupervised and supervised learning because it expects feedback for its solutions rather than the correct answer. RL algorithms typically assume the existence of a state space, a set of actions, and feedback. The state space is the set of all possible situations. The environment presents the true state to the learner and it executes one of a the set of actions in the environment. The environment or a critic provides feedback which is typically in the form of a scalar feedback signal to indicate whether or not the actions were good. RL algorithms are often used for control of simple processes.

One of the most popular reinforcement learning algorithms is the Q-learning algorithm [11]. This algorithm maintains a table for each state-action pair that contains the learner's perception of the value of the pair. The value is based on how rewarding it is to execute the action in the state given that the agent subsequently proceeds in the best way it currently knows. This table presents a practical difficulty when the state space is large. The problem is further aggravated when the learner wants to learn a predictive model of the environment. With model-based reinforcement learning algorithms, the learner must keep a probability for each state with the state action pair. The amount of memory that the learner must maintain grows with the number of states s and actions a; the complexity is $O(sa)$ for Q-learning and $O(s^2a)$ for model-based algorithms. These problems greatly impede the use of this simple RL algorithm in large state spaces.

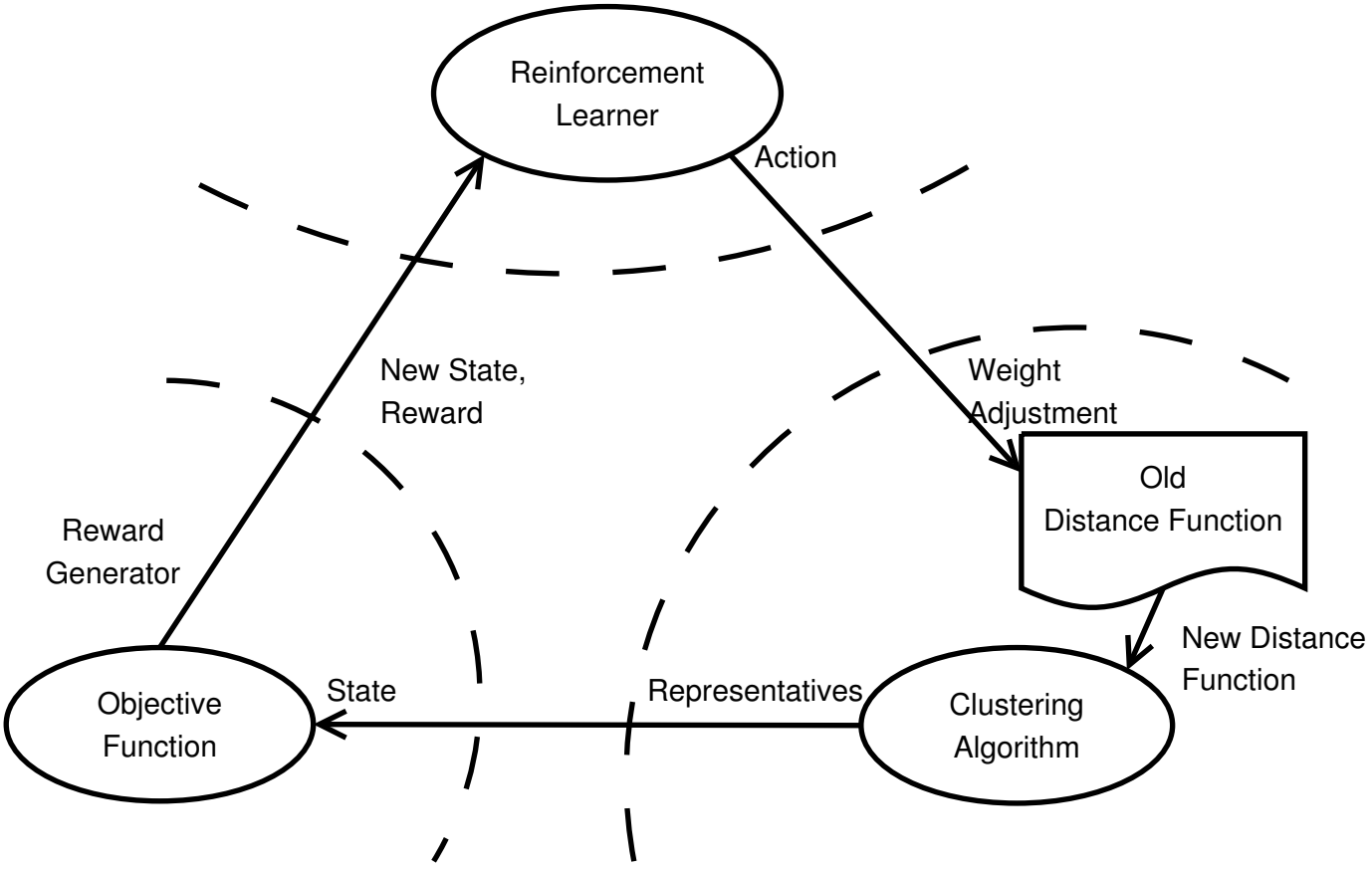

Fig. 1. The adaptive clustering environment.

Since RL algorithms like Q-learning are expensive to implement in large state spaces, a more complicated RL algorithm called prioritized sweeping [15] has been proposed. This algorithm presumes that in a large state space, maintaining a value estimate for individual states makes learning unnecessarily slow. Its solution is to selectively update several states during each interaction with the environment. The states are updated based on a priority related to their value and recent frequency of activation.

Recent research combines reinforcement learning and clustering to make more efficient RL algorithms for large state spaces. Since the state space is often excessively large and sometimes not very informative, the algorithm can cluster the states with an existing clustering method to significantly reduce the number of important states [3], [9]. Other work extends these approaches to reinforcement learning in a hierarchical environment. Youngblood uses hierarchical clustering to form a hierarchical probabilistic model of the environment and uses RL algorithms to control the learned model [18]. Bakker et. al. [1] clusters observations from the environment into aggregate observations. Lower-level RL components operate on the original observations and perform primitive actions while higher-level RL component pass execution to one of the lower-level components upon receipt of an aggregate observation. This allows for a self-reorganizing hierarchical reinforcement algorithm. In all of these approaches, clustering is used to enhance reinforcement learning, whereas adaptive clustering applies reinforcement learning to clustering.

III. Adaptive Clustering

In brief, adaptive clustering uses reinforcement learning to guide the search for good clusters based on external feedback. This section details our approach.

A. Clustering Environment Model

One can make clustering a reinforcement learning problem with states, actions, and rewards in a possibly non-deterministic environment. The new algorithm, adaptive clustering, employs a reinforcement learning controller for a clustering algorithm whose environment is as shown in Figure 1. The reinforcement learning algorithm adjusts the parameters of the clustering algorithm to produce better clusters. The result of each iteration of the clustering algorithm is a set of mutually exclusive and exhaustive clusters each of which has a representative object: e.g. the centroid. The clustering environment forms the state with these representatives. After observing the current state, the algorithm applies an action to adjust the weights of the distance metric. Given the state-action pair, the controller receives a reward. By learning weight-changing actions for the given set of representatives, the controller can retain its clustering knowledge for new instances of the same data distribution; this is particularly useful for online clustering.

The state space contains the current cluster representatives and distance function weights. Clustering algorithms usually operate in real-valued domains, but RL algorithms require distinct states. Since the use of discrete state information is more common in the literature, the cluster representatives first undergo a discretization step that divides the range of the variable into discrete intervals. Since adaptive clustering utilizes an existing partition-based clustering algorithm that relies on normalized data in the interval $[0, 1]$, the preprocessing step can effectively assume that the each of the coordinates lie between 0 and 1. Given k clusters each of which is a point in a d-dimensional space with m possible values for each attribute, the number of states is $m^{d(k+1)}$; the additional 1 is for the weight vector. The length of the state vector $\mathbf{s}$ is $d(k+1)$ and has the following form:

$$\begin{aligned} \mathbf{s} &= [\bar{c_1}, \ldots, \bar{c_k}; \mathbf{w}] \\ &= [\bar{c_{1_1}}, \ldots, \bar{c_{1_d}}; \ldots; \bar{c_{k_1}}, \ldots, \bar{c_{k_d}}; w_1, \ldots, w_d] \end{aligned}$$

where $x_{i,j}$ is the jth coordinate value of the ith centroid and the optional weights. This large state space presents practical difficulties that we address in Section IV.

Given a state, the learner can take one of several actions. The action either increases or decreases a single attribute weight. Given the old weight vector $\mathbf{w}$ of length d, the action uses the following formula to generate the new weight vector $\mathbf{w}'$:

$$\begin{aligned} w_i' &= \begin{cases} w_i \pm \Delta w_i & i = i^\star \\ w_i & i \neq i^\star \end{cases} \\ w_i' &= \frac{w_i'}{\sum_l w_l'} \end{aligned}$$

where $i^\star$ is the target attribute weight from the action, the constant $\Delta \in [.25, .5]$ is the randomly chosen percent change of the target weight, and the last equation is the renormalization of the weights. Given d attributes, the learner chooses $2d$ actions to increase or decrease an attribute indicated by either addition or subtraction of Δw_i.

Generically, the reward function must combine the influence of multiple objectives into a single scalar value. The current implementation returns the average of several component rewards, and leaves for future work the discovery of more sophisticated reward functions.

An important issue complicates the transformation of a clustering task to reinforcement learning. Reinforcement learning algorithms, in general, assume that the environment satisfies the Markov [11] assumption that the current state only depends on the immediately preceding state. Clustering, however, is an iterative process in which the representatives change almost deterministically with each iteration. At the level of the adaptive clustering algorithm, states would appear to change non-deterministically in response to different weight actions. From this level, the change in the state is simply assumed to be probabilistic–though possibly not Markovian.

B. Search Strategy

In a traditional reinforcement learning environment, agents are only allowed to perform actions in the current state; that is, it is not possible for an agent to jump from one state to another state that has been visited in the past. In adaptive clustering, however, it is possible to search multiple paths in parallel and to support more sophisticated

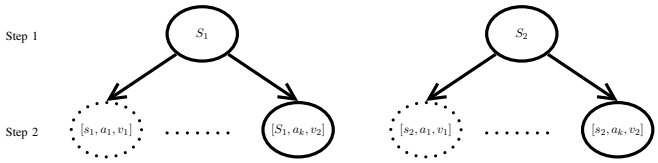

Fig. 2. The adaptive clustering search strategy.

forms of exploration that perform actions on previously visited states. Therefore, our adaptive learning architecture, employs a search algorithm that operates on the top of the reinforcement controller whose main goal is to select states for further expansion.

Reinforcement learning, in general, assumes that the environment is outside the control of the agent; thus, algorithms like Q-learning use reward information to more intelligently choose which actions to perform on the current state. Our approach, extends this framework to support intelligent state selection functions in addition to selecting actions intelligently. This allows us to utilize parallel reinforcement learning agents that share value information which enables us to evaluate the reward value of states more quickly.

In our current implementation, each of the parallel reinforcement learning agents applies the Q-learning algorithm to maintaining a single table of state-action values across all the agents and maintains an open list $L = \{S_1, \ldots, S_{|L|}\}$ of search states each of which consists of:

$$S_i \quad = \quad [\mathbf{s}, a, v]$$

where $\mathbf{s}$ is the state vector from the environment, a is the action to execute, and v is the Q-value of the state-action pair according to value iteration. The search algorithm reads the Q-value of each state-action pair in the search state and keeps the top $|L|$ most valuable states for execution. The existing prioritized sweeping Q-learning algorithm performs value iteration after execution of all actions in the open list. This search algorithm, then, is an example of a local beam search [16] in which the algorithm learns to determine which search states are best to expand.

Figure 2 further illustrates the search strategy. The figure assumes that the open list is of size 2. In search step 1, the algorithm creates a new open list that contains all single-action successors of the search states in its current open list. The bottom-row states contain the current state and the action to execute in the state. The algorithm looks up the Q-value of each of the four bottom-row states and then keeps the states having the largest value (shown in bold). The reinforcement learning agents then take each of the 2 actions in their respective states. The system returns a reward value for each transition and the prioritized sweeping algorithm performs value iteration on all states in the new open list at the end of the search step. The search continues until a fixed-number of steps have elapsed and retains the search state with the highest reward value as the current best solution.

The performance of the adaptive clustering algorithm largely depends on the performance of the underlying clustering algorithm. The reinforcement learning algorithm is fairly uncomplicated since states are stored in a lookup table and learning simply alters the states' values. Each action alters an array of weights. To generate the new state, however, the clustering algorithm must use the new weights to perform several iterations and generate new representatives. The reward function also uses some information about the data and can potentially be as expensive as the iteration of the clustering algorithm. In total, each step of the adaptive clustering algorithm performs a constant number of scans of the dataset. This makes the algorithm itself computationally expensive though still linear in the number of objects.

IV. Experimental Evaluation

The experiments demonstrate that adaptive clustering can find better clusterings with respect to an externally defined objective function. Finding a good objective function is difficult in two respects. First, the objective may only be a heuristic specific to particular datasets; thus, it would be difficult to generalize the results to new datasets. Second, it would be difficult to compare any custom objective function to a standard. To overcome these difficulties, we utilize an objective function that is very popular in machine learning algorithms for supervised learning– information gain. With this objective function, the clusterings and distance function weights that adaptive clustering

learns can enhance existing classification algorithms. The comparison of the enhanced classifiers will show how well adaptive clustering works.

To be clear, the purpose of the experiments is not to show that adaptive clustering is better at learning distance functions for classification but to demonstrate that the distance functions that adaptive clustering learns make better clusterings and are also useful for classification. The purpose of the experiments is to show that adaptive clustering is a better clustering algorithm under externally defined objectives.

A. Objective Function

The experiments use k-means as the underlying clustering algorithm which uses centroids as cluster representatives and creates clusters by assigning objects in a dataset to the nearest centroid. The objective function determines the worth of a particular clustering as the percent information gain [16] of the objects in the clusters with respect to the original unclustered data according to the following formula:

$$\begin{aligned} R(X) &= \frac{H(O) - \sum_{i=1}^{k} \frac{|c_i|}{|O|} H(c_i)}{H(O)} \\ O &= \bigcup_{i=1}^{k} c_i \\ \emptyset &= c_i \cap c_{j \neq i} \end{aligned}$$

where O is the original dataset, X is a clustering on O, H is the entropy function, and c_i is one of the k mutually exclusive and exhaustive sets that result from the clustering algorithm. Information gain uses the class label to determine the weighted average of the entropy of the data in each of the clusters. Although this is only applicable for supervised learning, adaptive clustering problem places no constraints on the information the rewarder can use to generate a reward.

B. Procedure

The experiments in this section test the adaptive clustering algorithm with several different parameter settings. The first parameter is the size of the open list which corresponds to the number of reinforcement learning agents. If the size is set to 1, the search model becomes traditional reinforcement learning, in which actions are applied to the current state. The second parameter is the classifier that uses the learned weights; two instance-based classifiers are used in the experiments that will be described later. The third parameter k, is the numbers of clusters generated by the clustering algorithm and is a multiple of the number of classes, C .

In each of the state encodings, the states should not be used in their continuous form. Many of the datasets for the experiment include numeric attributes whose cluster centroids may contain many significant digits. Table-based RL algorithms of which prioritized sweeping is one assume that the states are discrete and distinct from each other. To accommodate these algorithms, each attribute value and each weight first pass through a discretization filter that uses the following formula to filter an attribute value v into v':

$$v' = \lfloor 100v \rfloor$$

The constant 100 is not completely arbitrary since the data has already been normalized so that each attribute and weight lies in the interval $[0, 1]$; thus, v' should capture the first two or three significant digits of each attribute. The number of possible states is exponential in the dimensionality and number of clusters. Although this seems large, the actual implementation makes generous use of hash tables that grow only with the addition of new states. This significantly reduces the actual table size in practice. Any reference in the algorithm to states transitions that have yet to be visited generates the transitions and necessary information on the fly.

After the process of building the model on the training data, testing it is relatively simple. The experiment learns the attribute weights and then applies uses them with one of two instance-based classifiers: 1-NN [16] and NNCC. The 1-NN classifier is the one nearest-neighbor classifier. The Nearest-Neighbor Centroid Classifier (NNCC) clusters the training data, retains the cluster centroids, and assigns to the centroid the most common class in the cluster. At each incoming point, the NNCC finds the nearest centroid and assigns its class to the point. This algorithm

seems simplistic but it has been shown to be useful for certain datasets [7] and for the purpose of nearest neighbor dataset editing [8]. The experimental procedure is to the apply adaptive clustering to several datasets from the UCI machine learning repository [2]. Each test of adaptive clustering performs 10 runs of cross-validation on each of the datasets. Each run of cross-validation consists of 20 iterations of 50 steps in the learning process and retains the weight vector with the best reward value to be passed as a parameter to the two classifiers. Each iteration re-clusters the data and clears the open list. Each step expands state-action pairs based on value and follow the search process described in Section III-B.

In particular, we tested the following parameter settings for each of the 1-NN and NNCC classifiers:

Key	k	$\|L\|$
2	C	1
3	C	30
4	C	60
5	$5C$	60

V. Experimental Results

TABLE I

RESULTS COMPARING THE 1-NN CLASSIFIER WITH AND WITHOUT WEIGHTS FROM ADAPTIVE CLUSTERING.

Data Set	(1)	(2)	(3)	(4)	(5)
breast-cancer	68.58±1.82	68.58± 1.82	68.58± 1.82	68.58± 1.82	68.58± 1.82
credit-rating	81.58±0.65	81.86± 0.83○	81.74± 0.57	81.77± 0.72	81.84± 0.87
diabetes	70.62±0.84	69.29± 0.97●	69.75± 1.18●	70.12± 1.11	69.91± 1.28
german-credit	71.63±0.68	71.28± 0.73●	70.92± 0.80●	71.20± 0.64●	71.42± 0.60
glass	69.95±0.93	68.89± 2.00	72.20± 1.89○	72.01± 2.56○	76.26± 2.18○
heart-c	75.70±0.84	76.62± 1.03○	76.59± 1.21○	76.65± 0.88○	76.72± 1.07○
heart-h	78.33±1.06	78.32± 1.34	78.40± 1.11	77.86± 1.62	78.43± 1.67
heart-statlog	76.15±0.88	76.63± 1.59	77.41± 1.34○	77.26± 1.11○	77.37± 0.86○
ionosphere	87.10±0.49	87.24± 0.88	88.24± 0.88○	87.21± 0.91	88.52± 1.12○
sonar	86.17±0.84	85.79± 1.93	86.12± 1.85	86.07± 1.48	86.22± 1.05
vehicle	69.59±0.67	69.14± 0.89	68.83± 1.09●	68.59± 1.25●	70.55± 1.12○
vote	92.23±0.50	92.23± 0.50	92.23± 0.50	92.23± 0.50	92.23± 0.50
vowel	99.05±0.14	98.22± 0.51●	99.15± 0.19○	99.27± 0.26○	99.05± 0.24
wisconsin-breast-cancer	95.65±0.34	95.38± 0.57	95.49± 0.39	95.28± 0.32	95.64± 0.44
zoo	96.55±0.50	96.55± 0.50	96.55± 0.50	96.55± 0.50	96.55± 0.50
Averages	81.26	81.07	81.48	81.38	81.95

○, ● statistically significant improvement or degradation

TABLE II

RESULTS COMPARING THE NNCC CLASSIFIER WITH AND WITHOUT WEIGHTS FROM ADAPTIVE CLUSTERING.

Data Set	(1)	(2)	(3)	(4)	(5)
breast-cancer	72.25±1.52	68.58± 1.82●	68.58± 1.82●	68.58± 1.82●	68.58± 1.82●
credit-rating	79.33±0.69	81.61± 0.68○	81.77± 0.94○	81.55± 0.68○	81.48± 1.01○
diabetes	70.18±0.99	69.36± 1.11	69.78± 1.21	69.87± 2.09	69.77± 0.62
german-credit	69.47±0.52	71.31± 0.86○	71.17± 0.70○	71.37± 0.92○	71.09± 1.09○
glass	64.53±2.69	69.18± 1.61○	72.94± 2.49○	73.50± 2.49○	75.95± 1.73○
heart-c	78.29±1.94	76.62± 0.78●	76.65± 0.83●	76.79± 0.61	76.39± 1.30●
heart-h	81.10±1.25	78.43± 1.78●	78.09± 0.97●	78.14± 1.61●	77.95± 1.72●
heart-statlog	78.74±1.74	75.81± 1.05●	77.04± 1.23●	77.48± 1.21	76.70± 1.30●
ionosphere	86.21±0.87	87.19± 0.87○	88.38± 1.13○	87.72± 1.47○	88.23± 0.94○
sonar	63.21±2.06	85.64± 1.33○	86.61± 1.42○	86.36± 1.53○	86.27± 1.46○
vehicle	54.74±1.57	69.03± 1.28○	68.47± 1.03○	68.25± 1.74○	70.66± 1.52○
vote	90.53±0.68	92.23± 0.50○	92.23± 0.50○	92.23± 0.50○	92.23± 0.50○
vowel	12.28±0.94	98.02± 0.40○	99.18± 0.28○	99.18± 0.25○	99.11± 0.21○
wisconsin-breast-cancer	96.41±0.33	95.58± 0.38●	95.48± 0.27●	95.58± 0.48●	95.75± 0.34●
zoo	92.78±1.32	96.55± 0.50○	96.55± 0.50○	96.55± 0.50○	96.55± 0.50○
Averages	72.67	81.01	81.53	81.54	81.78

○, ● statistically significant improvement or degradation

Tables I and II show the average and standard deviation of the accuracy results for the parameters tested in the experiment. The open and closed circles indicate statistical significance using the paired t-test of significance

for the 10 runs. These results demonstrate that adaptive clustering can improve the quality of the clusters and the distance metric to improve two instance-based classifiers.

Table I compares the 1-NN classifier with the same classifier after learning weights. First, the glass dataset had consistent improvement in variations with open list size greater than 1. The diabetes dataset which was significantly worse with open list size 1 became not significantly worse with a larger open list. The results improve on average over all the data sets; furthermore, the results tend to improve with the size of the open list. For the cluster size of $5C$, the average results continued to improve.

Table II compares the NNCC classifier with the same classifier after learning weights. Based on the results, adaptive clustering dramatically affects this classifier. Since the classifier makes its decision based only on the centroid of the cluster, even a small change in the distance between points and centroids can make a difference. Most datasets significantly improved with an open list size of 1. With a larger open list size, two of the datasets with significantly worse performance with open list size 1 became not significantly worse. Several of the datasets had a very large difference. Notably, the performance of the vowel dataset improved from 12% to over 99%. The tables also show that averaged across all datasets, NNCC with learned weights outperforms the 1-NN classifier. The most important conclusion is that adaptive clustering improves the clustering so that using only the representatives, one can achieve better accuracy than with the 1-NN classifier. For a training set of n examples with $C < n$ classes, the NNCC classifier compares each incoming point with $O(C)$ points instead of the $O(n)$ comparisons usually necessary with the 1-NN classifier without more sophisticated indexing methods.

VI. Conclusion

Adaptive clustering learns attribute weights for clustering under externally defined objectives. Users of clustering algorithms often have some idea as to the composition and representatives of a clustering of a dataset. Unfortunately, users often cannot specify their objectives directly to the clustering algorithm. Adaptive clustering uses external feedback in the form of a reward function to find clusterings that better meet these objectives. Recent work in semi-supervised and supervised clustering demonstrate that the alteration of attribute weights in a distance metric can meet this challenge. Most of those techniques, however, require the user to assess cluster quality at a very low level of granularity. In adaptive clustering, on the other hand, it is sufficient to assign a reward to the clustering as a whole.

Given a clustering, adaptive clustering alters the vector of attribute weights along one dimension and receives a new clustering and reward. With this environment, adaptive clustering resembles a reinforcement learning task. Since the environment is not under the temporal constraints often associated with reinforcement learning tasks, the evaluation of multiple paths in the environment is possible. Adaptive clustering supports memorizing clusterings that worked well in a given context, which provides the ability to adapt the weights more quickly if new data were to arrive. In conjunction with value estimation, it can efficiently search for actions that have led and will lead to better clusterings.

The goal of adaptive clustering is to improve the clustering of a set of objects in a dataset. As a means to evaluate this goal, the experiments use information gain with respect to class labels as an objective function. With this objective, adaptive clustering can enhance existing instance-based classifiers. Using the learned weights, the 1-NN classifier achieves a statistically significant improvement in accuracy over some datasets in the UCI machine learning repository [2]. With the learned distance weights, even a classifier that only uses the cluster representatives to classify incoming instances can, on average, meet or beat the more costly 1-NN classifier. Based on its improvements to the accuracy, the technique can find more informative clusters and representatives in this domain.

Future work will apply adaptive clustering to other, less contrived domains. Interesting potential applications in online clustering include such diverse applications as information retrieval, spatial summary data mining, and multi-agent coordination. The results provide interesting evidence that adaptive clustering can find better representatives with reinforcement learning and deserves further work.

References

[1] Bram Bakker and JÃ$\frac{1}{4}$rgen Schmidhuber. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In A. Bonarini E. Yoshida F. Groen, N. Amato and B. KrÃ¶se, editors, *Proceedings of the 8-th Conference on Intelligent Autonomous Systems, IAS-8*, 2004.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] James L. Carroll, Todd Peterson, and Kevin Seppi. Reinforcement learning task clustering. In *ICMLA*, 2003.

[4] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. Technical Report 2003-1892, Cornell University, 2003.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[6] C. Domeniconi and D. Gunopulos. Adaptive nearest neighbor classification using support vector machines. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[7] C. Eick and N. Zeidat. Using supervised clustering to enhance classifiers. In *Proc. 15th International Symposium on Methodologies for Intelligent Systems (ISMIS)*, 2005. To appear.

[8] C. Eick, N. Zeidat, and R. Vilalta. Using representative-based clustering for nearest neighbor dataset editing. In *Proc. IEEE International Conference on Data Mining (ICDM)*, 2004.

[9] Fernando Fernández and Daniel Borrajo. Vector quantization applied to reinforcement learning. In *Proceedings of the Fifth Workshop on RoboCup*, pages 97–102, 1999.

[10] Barbara Hammer and Thomas Villmann. Generalized relevance learning vector quantization. *Neural Netw.*, 15(8-9):1059–1068, 2002.

[11] Lelie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[12] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[13] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ PAK: The learning vector quantization program package. Technical report, Helsinki University of Technology, 1996.

[14] J. McQueen. Some methods for the classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Stat., Prob.*, number 1, pages 281–297, 1967.

[15] A. W. Moore and C. G. Atkeson. Prioritized sweeping–reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[16] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[17] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

[18] G. Michael Youngblood, Edwin O. Heierman, Diane J. Cook, and Lawrence B. Holder. Automated hierarchical POMDP construction through data-mining techniques in the intelligent environment domain. *AAAI*, 2004.