Adaptive Clustering: Obtaining Better Clusters Using Feedback and Past Experience

Abraham Bagherjeiran, Christoph F. Eick, Chun-Sheng Chen, Ricardo Vilalta
abagherj@cs.uh.edu, ceick@cs.uh.edu, lyons19tw@sbcglobal.net, vilalta@cs.uh.edu
University of Houston
Department of Computer Science
Houston, Texas 77204-3010

Abstract

Adaptive clustering uses external feedback to improve cluster quality; past experience serves to speed up execution time. An adaptive clustering environment is proposed that uses Q-learning to learn the reward values of successive data clusterings. Adaptive clustering supports the reuse of clusterings by memorizing what worked well in the past. It has the capability of exploring multiple paths in parallel when searching for good clusters. In a case study, we apply adaptive clustering to instance-based learning relying on a distance function modification approach. A distance function adaptation scheme that uses external feedback is proposed and compared with other distance function learning approaches. Experimental results indicate that the use of adaptive clustering leads to significant improvements of instance-based learning techniques, such as k-nearest neighbor classifiers. Moreover, as a by-product a new instance-based learning technique is introduced that classifies examples by solely using cluster representatives; this technique shows high promise in our experimental evaluation.

1 Introduction

A clustering algorithm finds groups of objects in a predefined attribute space. Since the objects have no known prior class membership, clustering is an unsupervised learning process. 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-means clustering algorithm [11]. A large amount of work, as will be discussed in Section 2, details the limitations of these algorithms; the main criticism is that these criteria are excessively simplistic and do not accurately capture the user's understanding of the true nature of the data.

This paper introduces a novel data mining technique we term adaptive clustering. The central idea is to use feedback and past experience to guide the search toward better clusters. An adaptive clustering environment is proposed that modifies the weights of a distance function based on environmental feedback and employs Q-learning [15] to learn the reward values of successive data clusterings. In particular our approach employs a traditional reinforcement learning framework, in which states consist of a set of cluster representatives and a set of distance function weights and actions are modifications of distance function weights.

Since the application of a reinforcement learning (RL) algorithm to clustering is unusual, it is worthwhile to discuss our reasons for this fusion of ideas. The goal of weight-based distance function learning is to find the weights that induce a clustering that best meets externally defined objectives. Since this is clearly an optimization problem, researchers have applied optimization algorithms, such as hill-climbing, to this task. The objective of adaptive clustering, however, extends the objective of distance function learning to support relearning. In most optimization tasks, the search algorithm terminates when the best weights are found. An adaptive clustering environment, however, is a continuously running process that solves different but similar clustering tasks in an environment with dynamic data and potentially interactive evaluations.

An adaptive clustering algorithm should relearn in response to changes in evaluation or data, concentrate on actions that have high expected value, and assign incoming reward to the states and actions that led to the reward. Algorithms with these properties are the goal of reinforcement learning (RL) research [8, 12, 15].

Reinforcement learning algorithms are often applied in environments in which state transitions and external rewards are subject to change. [13]. Moreover, RL algorithms assume that an agent can control which actions to apply in a particular state but cannot control which state it visits and reward it receives as the result of applying a particular action. Agents receive rewards when traversing the state space, and RL algorithms, such as Q-learning, learn which actions lead to the best long-term expected reward [15].

In summary, this paper centers on adaptive clustering. Our proposed adaptive clustering framework modifies the distance function based on external feedback with the goal to obtain better clusters. A novel distance function modification scheme is proposed for this purpose and is compared with other distance function learning approaches. Moreover, given the similarities between the goal of adaptive clustering and reinforcement learning, we investigate the application and modification of techniques that originate in reinforcement learning for the task of adaptive clustering. The paper is organized as follows. Section 2 discusses related work. Section 3 introduces the idea of adaptive clustering in more detail, and introduces an environment for adaptive clustering that implements this idea. Section 4 introduces a framework that applies adaptive clustering to instance-based learning and Section 5 provides an experimental evaluation of the proposed framework. Section 6 concludes the paper.

2 Related Work

2.1 Clustering

2.1.1 Unsupervised Clustering

A traditional unsupervised clustering algorithm maximizes a known criterion function, such as the squared summed error. Given a set of n objects O the algorithm seeks for a set of k clusters $X = \{c_1, \ldots, c_k\}$ each with a cluster representative $\bar{r}_i$ that minimize the following error function:

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

[1]

The k-means algorithm [11] uses centroids as cluster representatives, whereas the k-medioids algorithm [9] uses objects that belong to O as representatives.

[1] I think that the formula is correct because it refers to the squared difference between two vectors rather than using a distance function we have yet to define. We could write it as: $\langle o - \bar{r_i} \rangle \cdot \langle o - \bar{r_i} \rangle$ referring to the inner (dot) product between the difference of the two vectors.

The k-means and other partitioning algorithms typically use 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:

$$d(o_i, o_j) = \sum_{l=1}^{d} w_l \, |o_{i_l} - o_{j_l}|$$

where o_i and o_j are d-dimensional objects and $\mathbf{w} = (w_1, ..., w_d)$ is a weight vector whose components are non-negative and $\sum_{l=1}^{d} w_l = 1$. The formula defines the distance between two objects as a weighted sum of the differences between two objects with respect to their attributes; when all the weights are equal the distance is exactly the Manhattan distance.

2.1.2 Supervised Clustering

Supervised clustering is applied to classified examples using criterion functions that emphasize class purity. Most approaches seek clusters that are pure (i.e., all or at least most examples in a cluster belong to the same class). One group of approaches adapt distance functions with the goal of obtaining purer clusters. The LVQ algorithm modifies a distance function based on the performance of the classifier [10]. Since a support vector machine finds separating hyperplanes in a larger feature space, the hyperplane function can be transformed into a distance function [5, 4]. Another group of approaches directly integrate supervised learning with a traditionally unsupervised clustering algorithm using fitness functions that are based on the class purity within a cluster [6].

2.1.3 Semi-Supervised Clustering

Semi-supervised clustering algorithms adapt the clustering based on background knowledge that usually consists of a small set of classified examples. Most work in this area relies on the preferences of a user to inform the algorithm whether or not the clustering is useful. One application uses semi-supervised clustering for content-based image retrieval, requesting a user to tell if two images should or should not belong to the same cluster. Based on a series of evaluation examples, the algorithm adapts the clusters to satisfy the constraints that were imposed by the user [3]. Other 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, the distance function is adapted so that the shape of the clusters satisfies these constraints [16].

2.2 Reinforcement Learning

Reinforcement learning (RL) lies between unsupervised and supervised learning because it expects feedback for its solutions rather than the correct answer [8]. 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 which 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 [15]. 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 essentially an exponential average of rewards that the learner has received. By learning the rewards of (state, action) pairs, the learner takes action based on an estimate of the long-term reward it expects to receive given its current knowledge. Maintaining this table presents a practical difficulty when the state space is large. The amount of memory that the learner must maintain grows with the number of states s and actions a; the space complexity is $O(sa)$ for Q-learning. These problems greatly impede the use of this simple RL algorithm in large state spaces.

Since RL algorithms like Q-learning are expensive to implement in large state spaces, a more complicated RL algorithm called prioritized sweeping has been proposed [12]. This algorithm presumes that in a large state space, maintaining a value estimate for individual states makes learning unnecessarily slow. Its solution is to only apply Q-learning to selected states during each interaction with the environment. The states are updated based on a priority related to their value and recent frequency of activation.

3 Adaptive Clustering

In brief, adaptive clustering uses the prioritized-sweeping variant of the Q-learning algorithm to guide the search for better clusters based on expected external feedback. This section details our approach.

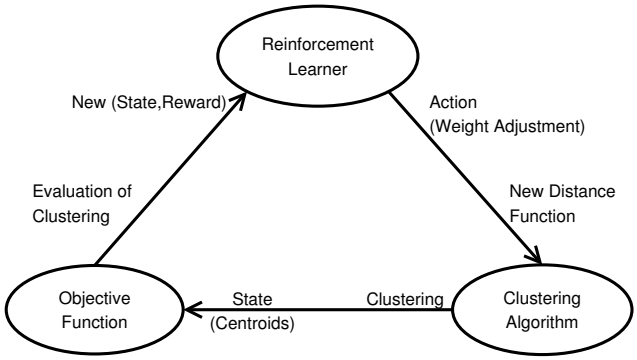


Figure 1. The adaptive clustering environment.

3.1 Clustering Environment Model

To apply an algorithm intended for reinforcement learning environments, we must transform the adaptive clustering environment into a reinforcement learning environment. One can make clustering a reinforcement learning problem with states, actions, and rewards in a possibly non-deterministic environment as illustrated in Figure 1. In the environment, the clustering algorithm uses its current distance function to form a clustering on the data, and the centroids of the resulting clustering are state information to an RL algorithm. The objective function evaluates the clustering and returns a reward. Given its current state and reward, the algorithm selects a weight-changing action to apply to the distance function. It then performs a few iterations of the clustering algorithm using the new distance function. This process repeats for a fixed number of iterations. In general, the objective function must combine the influence of multiple objectives into a single scalar value. The current implementation uses information gain as described in Section 4.1.

The state space consists of pairs of cluster representatives and distance function weights. Clustering algorithms usually operate in real-valued domains, but the Q-learning algorithm we employ requires discrete states. The discretization step divides the range of a 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 each of the coordinates lie between 0 and 1. Given k cluster representatives each of which is a point in a d-dimensional space with m attribute values, the number of states is $m^{d(k+1)}$; the additional 1 is for the weight

vector of the distance function. The length of the state vector $\mathbf{s}$ is $d(k+1)$ and has the following form:

$$\begin{aligned}\mathbf{s} &= (\bar{r}_1, \ldots, \bar{r}_k; \mathbf{w}) \\ &= (\bar{r}_{1_1}, \ldots, \bar{r}_{1_d}; \ldots; \bar{r}_{k_1}, \ldots, \bar{r}_{k_d}; w_1, \ldots, w_d)\end{aligned}$$

where $r_{i,j}$ is the jth coordinate value of the ith centroid and the optional weights. This large state space presents some practical difficulties that we address in Section 3.3.

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

$$\begin{aligned}\hat{w}_i &= \begin{cases} w_i \pm \Delta w_i & i = i^\star \\ w_i & i \neq i^\star \end{cases} \\ w'_i &= \frac{\hat{w}_i}{\sum_{l=1}^{d} \hat{w}_i}\end{aligned}$$

where $i^\star$ is the index of the attribute whose weight will be changed, the constant $\Delta \in [.25, .5]$ is a randomly chosen percent change of the target weight, and the last equation is the renormalization of the weights. Given d attributes, the learner chooses one of $2d$ actions that increases or decreases the weight of an attribute by adding or subtracting Δw_i.

3.2 Search Strategy

In a traditional reinforcement learning environment, agents are only allowed to perform actions in the current state; thus, 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 forms of exploration that perform actions on previously visited states.

In our current implementation, the search process uses the Q-learning algorithm to estimate the value of applying a particular weight change action to a particular state. The search algorithm maintains an open list $L = \{S_1, \ldots, S_{|L|}\}$ of search states each of which consists of:

$$S_i = (\mathbf{s}, a, v)$$

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 algorithm reads the Q-value of each state-action pair in the search state and keeps the top $|L|$ most valuable states for execution. This search algorithm is an example of a local beam search in which the algorithm expands states based on their values rather than rewards [14].

As an example, Figure 2 further illustrates the search strategy. The figure assumes that the open list is of size 2. We assume that the open-list currently contains states S_1 and S_2 corresponding to state vectors $\mathbf{s}_1$ and $\mathbf{s}_2$, respectively. There is only one attribute in the example, so there are only two actions–increase or decrease the one weight. 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 that state. The algorithm looks up the Q-value of each of the $2d$ bottom-row states and then keeps the states having the largest value (shown in bold). The reinforcement learning agent will apply the selected action in each of the two states that remain in the opening list. 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 it retains the state with the highest reward value as the current best solution.

3.3 Complexity Concerns

From the previous discussion, it is obvious that the state space is very large. However, this does not present significant difficulty for the adaptive clustering algorithm. The reason for this is that the state does not change much. A change in the weights will only change the clustering when it causes one or more objects to be assigned to different clusters. The clustering only changes the cluster representatives–to the precision of the discretization–when the new objects cause a significant shift in the distribution of the objects in the cluster. The result of these infrequent changes is that only a very small portion of the state space is actually reachable. Adaptive clustering emphasizes exploiting states that have been visited before and received high rewards rather than generating novel states.

Despite the small portion of the state space that the algorithm explores, its size is still exponential in the dimensionality and number of values in the discretization of an attribute. As noted in Section 2.2, the table lookup method of the original Q-learning algorithm does not scale well in space or time for these large state spaces. Therefore, we use the prioritized sweeping algorithm that employs a priority update heuristic to give preference to updating the utility of frequently visited paths that have a high surprise value [12]. A path with a high surprise value is a path that leads to a state for

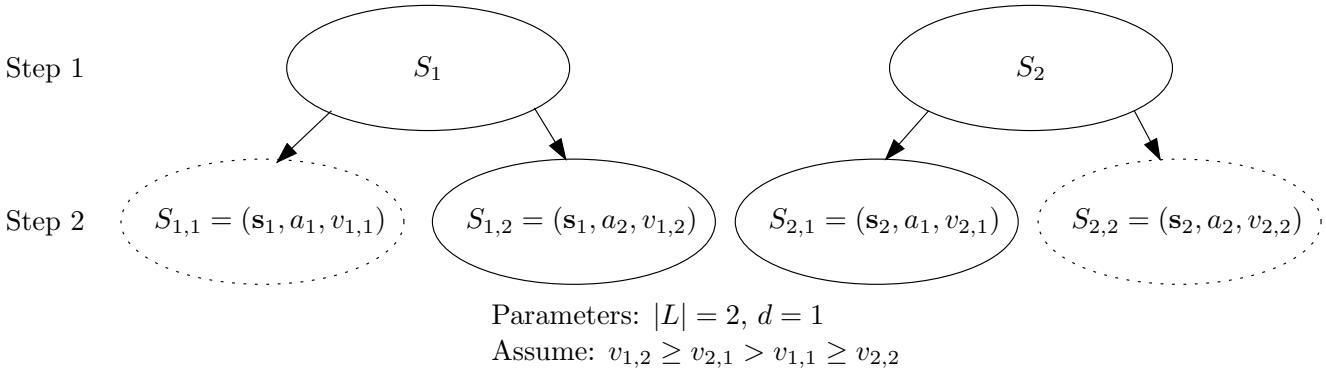


Figure 2. The adaptive clustering search strategy.

which the actual reward is significantly different from the estimated reward. By employing this heuristic, the algorithm does not waste time updating the utilities of state-action pairs whose predicted reward matches the actual reward or that rarely occur.

The priority heuristic of the prioritized sweeping algorithm complements our beam search strategy well. It is important to note that the likelihood of performing value iteration on a state depends on the absolute difference between the change in the expectation of reward given the obtained reward. Therefore, for example, if the obtained reward causes the expectation to change from 5 to 3 or 7, the algorithm will update states with the same priortity. This makes sense in the context of the beam search strategy we employ: if there are significant changes in the utility of currently explored paths, we would prefer to search the paths that suddenly appear to be better rather than worse and not to waste time performing value iteration on paths for which our predictions are correct or for paths that are not currently explored.

4 Experimental Evaluation

The experiments investigate if adaptive clustering can find better clusterings with respect to an externally defined objective function. Our experimental evaluation uses a popular objective function known as 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 accuracy of the enhanced classifiers with the original classifiers will demonstrate the benefit of adaptive clustering. As a by-product, a new instance-based classifier will be introduced that, as our experimental results will show, is generally useful for instance-based learning.

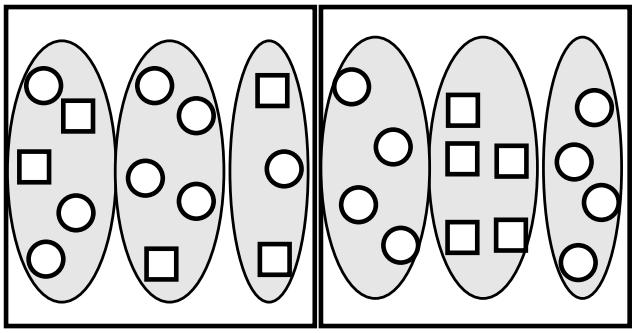
4.1 Objective Function

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 [11]. The objective function R determines the worth of a particular clustering as the percent information gain of the objects in the clusters with respect to the original unclustered data according to the following formula [14]:

$$
\begin{aligned}
R(X) &= R(\{c_1, \ldots, c_k\}) \\
&= \frac{H(CL, O) - \sum_{i=1}^{k} \frac{|c_i|}{|O|} * H(CL, c_i)}{H(CL, O)} \\
\text{where} \quad & O = \bigcup_{i=1}^{k} c_i \text{ and } c_i \cap c_{j \neq i} = \emptyset \\
H(CL, V) &= -\sum_{cl \in CL} p(cl, V) \log_2 p(cl, V) \\
\text{where} & \\
p(cl, V) &= \frac{|\{o \mid o \in V \wedge o \text{ is of class } cl \in CL\}|}{|\{o \mid o \in V\}|}
\end{aligned}
$$

where O is the original dataset, X is a clustering on O, c_i is one of the k mutually exclusive and exhaustive subsets of O that result from the clustering algorithm, and $H(CL, V)$ is the entropy with respect to the distribution of classes $cl \in CL$ in V. $H(CL, V)$ is computed by iterating over the individual classes in the data set. In summary, information gain computes the weighted average of the entropy with respect to the class distribution in each cluster.

Let us assume that we have 2 distance functions d_{init} and d_{better} for the dataset illustrated in Figure 3 that



(a) d_{init} (b) d_{better}

Figure 3. Clusterings for two different distance functions.

consists of 13 examples of which 5 belong to the class identified by a square and 8 belong to the class identified by a circle. We apply the k-means clustering algorithm with $k = 3$ to the dataset and obtain the 2 depicted clusterings. Obviously, the information gain for the second clustering that used distance function d_{better} is much higher than the information gain for the first clustering, because all clusters only contain examples of a single class. Most importantly, using a distance function with high information gain function in conjunction with a k-nearest neighbor classifier allows us to obtain a classifier with high predictive accuracy. For example, if we use a 3-nearest neighbor classifier with distance function d_{better} it will have 100% accuracy with respect to leave-one-out cross-validation, whereas several examples are misclassified if d_{init} is used.

4.2 Procedure

The experiments in this section test the adaptive clustering algorithm with several different parameter settings. The first parameter k, is the numbers of clusters generated by the clustering algorithm and is a multiple of the number of classes, C. The second parameter is the size of the open list $|L|$. If the size is set to 1, the search model reduces to a traditional reinforcement learning algorithm, in which actions are applied to the current state.

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. The Q-learning algorithm that we use assumes that the states are discrete and distinct from each other. To accommodate the algorithm, 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 making the number of values per attribute $m = 100$.

After the process of building the model on the training data, testing it is relatively simple. The experiment learns the attribute weights and then uses them with one of two instance-based classifiers: 1-NN [14] and NCC. The 1-NN classifier is the one nearest-neighbor classifier. The Nearest Centroid Classifier (NCC) 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 NCC finds the nearest centroid and assigns its class to the point. This algorithm seems simplistic but has been shown to be useful for certain datasets [7]. The experimental procedure is to apply adaptive clustering to several datasets from the UCI machine learning repository [1]. 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 each of which is run for a randomly created initial set of clusters. The weight vector with the highest reward is is sent as a parameter to the 1-NN and NCC classifiers to classify the examples in the test set. After an iteration is completed, the open list is cleared. Each iteration consists of 50 steps in the learning process. Each step expands state-action pairs based on value and follow the search process described in Section 3.2.

In particular, we tested the parameter settings listed in Table 1 for the 1-NN and NCC classifiers; e.g. key "1.60" indicates that in the particular experiment k was set to the number of classes C in the dataset and the open list had size 60. We also show the average accuracy obtained using two other distance function learning algorithms that do not use adaptive clustering. The first distance function learning algorithm [7] employs a weight-updating heuristic that changes weights based on class density information in clusters. The second distance function learning algorithm [2] uses randomized hill climbing to search for good distance function weights. The approach selects a randomly created weight vector as the current weight vector, creates m ($m = 30$ was used in the experiments) new weight vectors in the neighborhood of the current weight vector, and selects the weight vector with the highest value for

Key	Clusters k	Open List Size $\|L\|$
1,1	C	1
1,30	C	30
1,60	C	60
5,60	$5C$	60

Table 1. Legend for interpreting the results.

the objective function R as the new current weight vector. This process terminates if there is no improvement with respect to objective function R. The algorithm is then restarted with a new randomly created weight vector until a predefined time limit is reached.

5 Experimental Results

Tables 2, 3, 4, 5, and 6 show the average and standard deviation of the accuracy results for the parameters tested in the experiment compared against several base classifiers for a benchmark consisting of UCI datasets [1]. The open and closed circles indicate statistical significance using the paired t-test of significance with significance level 0.10 for the 10 runs.

Table 2 compares a 1-NN classifier having a distance function that considers all attributes to have the same weight with the same classifier with learned weights. First, adaptive clustering achieved consistent improvement in accuracy for open list sizes greater than one. The diabetes dataset which was significantly worse with open list size 1 became not significantly worse with a larger open list. On average, the results improve over all the data sets; furthermore, the results tend to improve with the size of the open list. In summary, the capability to explore multiple paths in parallel when searching for good distance functions leads to better accuracies for the tested classifiers. Moreover, increasing the number of clusters from C to $5C$ increases the average accuracy for all datasets further.

Tables 3 and 4 compare the traditional 1-NN classifier with learned weights from an algorithm that employs a density-based weight-updating heuristic [7], a randomized hill-climbing algorithm (RHC), and the adaptive clustering algorithm. The results show a substantial improvement in several of the tested datasets over the two distance function learning algorithms. For the vehicle and glass datasets, the improvement is most obvious.

Table 5 compares the NCC classifier using equal weights with the same classifier after learning weights. Interestingly, using adaptive clustering leads to a quite dramatic increase in accuracy: the average accuracy over the datasets tested improved by more than 8%. Since this classifier makes its decision based only on the centroid of the cluster, even a small change in the distance between points and centroidshas a significant impact. Several of the datasets had a very large difference. Notably, the accuracy for the vowel dataset improved from 12% to over 99%.

Most importantly, the NCC classifier with learned weights using $k = 5C$ and $|L| = 60$ achieved an average accuracy of 81.78% which is 0.52% higher than the average accuracy for the traditional 1-NN classifier of 81.26%. This interesting result suggests that adaptive clustering sufficiently improves cluster quality so that by just using cluster representatives instead of all objects in the dataset, one can achieve a better accuracy than the 1-NN classifier as shown in Table 6. For a training set of n examples with $C \ll n$ classes, the NCC 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. In the glass dataset, for example, the traditional 1-NN classifier that uses all $214 \times 0.9 = 193$ training examples to classify an example (10-fold cross validation only uses 90% of the examples for training) achieves an accuracy of 69.95%, whereas the NCC classifier that uses the learned distance function with just six training examples achieves accuracies of 72.94% and 73.50% for open list sizes of 30 and 60.

In summary the experimental results demonstrate that adaptive clustering can improve the quality of the clusters and the distance metric to improve two instance-based classifiers. Compared to the 1-NN algorithm without learned weights, adaptive clustering yields significantly better results for several datasets. On average, it outperforms the traditional 1-NN classifier. As a distance function learning algorithm for classification, adaptive clustering outperforms two other distance function learning approaches. Finally, on a more direct evaluation of utility of the clustering, adaptive clustering substantially improves the NCC classifier in all but a few datasets. The improvement makes this relatively simple classification algorithm comparable in accuracy with the 1-NN classifier.

6 Conclusion

Adaptive clustering learns attribute weights for clustering under externally defined objectives. Users of clustering algorithms often have some idea what a good clustering of a dataset should be. Unfortunately, users often have problems in quantifying their ideas. Adaptive clustering uses external feedback to find clusterings that better meet these objectives. Recent work

Dataset	1-NN	1,1	1,30	1,60	5,60
breast-cancer	68.58±1.82	68.58±1.82	68.58±1.82	68.58±1.82	68.58±1.82
breast-cancer-w	95.65±0.34	95.38±0.57	95.49±0.39	95.28±0.32	95.64±0.44
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
zoo	96.55±0.50	96.55±0.50	96.55±0.50	96.55±0.50	96.55±0.50
Average	81.26	81.07	81.48	81.38	81.95

○, ● statistically significant improvement or degradation

Table 2. Results comparing the 1-NN classifier with and without weights from adaptive clustering.

Dataset	[7]	1,1	1,30	1,60	5,60
diabetes	72.95±1.13	69.29±0.97●	69.75±1.18●	70.12±1.11●	69.91±1.28●
glass	49.41±3.06	68.89±2.00○	72.20±1.89○	72.01±2.56○	76.26±2.18○
heart-c	78.83±1.66	76.62±1.03●	76.59±1.21●	76.65±0.88●	76.72±1.07●
heart-h	79.36±1.60	78.32±1.34	78.40±1.11	77.86±1.62	78.43±1.67
heart-statlog	82.78±1.20	76.63±1.59●	77.41±1.34●	77.26±1.11●	77.37±0.86●
ionosphere	85.19±1.30	87.24±0.88○	88.24±0.88○	87.21±0.91○	88.52±1.12○
vehicle	53.54±2.73	69.14±0.89○	68.83±1.09○	68.59±1.25○	70.55±1.12○

○, ● statistically significant improvement or degradation

Table 3. Results comparing improvements to the 1NN classifier with weights learned by a related impurity-based weight updating algorithm and adaptive clustering [7].

Dataset	RHC	1,1	1,30	1,60	5,60
diabetes	74.51±0.78	69.29±0.97●	69.75±1.18●	70.12±1.11●	69.91±1.28●
glass	51.55±1.75	68.89±2.00○	72.20±1.89○	72.01±2.56○	76.26±2.18○
heart-c	79.18±1.70	76.62±1.03●	76.59±1.21●	76.65±0.88●	76.72±1.07●
heart-h	80.15±1.27	78.32±1.34●	78.40±1.11	77.86±1.62●	78.43±1.67
heart-statlog	82.63±1.47	76.63±1.59●	77.41±1.34●	77.26±1.11●	77.37±0.86●
ionosphere	83.74±1.58	87.24±0.88○	88.24±0.88○	87.21±0.91○	88.52±1.12○
vehicle	57.06±3.67	69.14±0.89○	68.83±1.09○	68.59±1.25○	70.55±1.12○

○, ● statistically significant improvement or degradation

Table 4. Results comparing improvements to the 1NN classifier with weights learned by a randomized hill-climbing algorithm and adaptive clustering.

Dataset	NCC	1,1	1,30	1,60	5,60
breast-cancer	72.25±1.52	68.58±1.82●	68.58±1.82●	68.58±1.82●	68.58±1.82●
breast-cancer-w	96.41±0.33	95.58±0.38●	95.48±0.27●	95.58±0.48●	95.75±0.34●
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○
zoo	92.78±1.32	96.55±0.50○	96.55±0.50○	96.55±0.50○	96.55±0.50○
Average	72.67	81.01	81.53	81.54	81.78

○, ● statistically significant improvement or degradation

Table 5. Results comparing the NCC classifier with and without weights from adaptive clustering.

Dataset	1-NN	NCC	1,1	1,30	1,60	5,60
breast-cancer	68.58±1.82	72.25±1.52○	68.58±1.82	68.58±1.82	68.58±1.82	68.58±1.82
breast-cancer-w	95.65±0.34	96.41±0.33○	95.58±0.38	95.48±0.27	95.58±0.48	95.75±0.34
credit-rating	81.58±0.65	79.33±0.69●	81.61±0.68	81.77±0.94	81.55±0.68	81.48±1.01
diabetes	70.62±0.84	70.18±0.99	69.36±1.11●	69.78±1.21	69.87±2.09	69.77±0.62●
german-credit	71.63±0.68	69.47±0.52●	71.31±0.86●	71.17±0.70●	71.37±0.92	71.09±1.09●
glass	69.95±0.93	64.53±2.69●	69.18±1.61	72.94±2.49○	73.50±2.49○	75.95±1.73○
heart-c	75.70±0.84	78.29±1.94○	76.62±0.78○	76.65±0.83○	76.79±0.61○	76.39±1.30
heart-h	78.33±1.06	81.10±1.25○	78.43±1.78	78.09±0.97	78.14±1.61	77.95±1.72
heart-statlog	76.15±0.88	78.74±1.74○	75.81±1.05	77.04±1.23	77.48±1.21○	76.70±1.30
ionosphere	87.10±0.49	86.21±0.87	87.19±0.87	88.38±1.13○	87.72±1.47	88.23±0.94○
sonar	86.17±0.84	63.21±2.06●	85.64±1.33	86.61±1.42	86.36±1.53	86.27±1.46
vehicle	69.59±0.67	54.74±1.57●	69.03±1.28	68.47±1.03●	68.25±1.74●	70.66±1.52
vote	92.23±0.50	90.53±0.68●	92.23±0.50	92.23±0.50	92.23±0.50	92.23±0.50
vowel	99.05±0.14	12.28±0.94●	98.02±0.40●	99.18±0.28	99.18±0.25○	99.11±0.21
zoo	96.55±0.50	92.78±1.32●	96.55±0.50	96.55±0.50	96.55±0.50	96.55±0.50

○, ● statistically significant improvement or degradation

Table 6. Results comparing the 1NN classifier against the NCC classifier and its improvements with adaptive clustering.

in semi-supervised and supervised clustering demonstrates that the alteration of attribute weights in a distance metric can help to address user needs. 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.

We discussed the results of a case study that applied adaptive clustering to learn distance function weights to maximize externally defined objectives. In particular, adaptive clustering was used to enhance existing instance-based classifiers based using information gain as the objective function. Using the learned weights, the 1-NN classifier achieves a statistically significant improvement in accuracy for several datasets in the UCI machine learning repository [1]. With the learned weights, even a classifier that only uses the cluster representatives to classify incoming instances can, on average, meet or beat the much 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 less contrived domains. Interesting potential applications include diverse applications such as information retrieval, spatial data mining, and multi-agent meta-learning.

References

[1] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[2] C.-S. Chen. Using weight updating heuristics and randomized hill climbing distance function learning. Master's thesis, University of Houston, 2005.

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

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

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

[6] 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.

[7] C. F. Eick, A. Rouhana, A. Bagherjeiran, and R. Vilalta. Using clustering to learn distance functions for supervised similarity assesment. In *IAPR International Conference on Machine Learning and Data Mining*, 2005.

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

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

[10] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. LVQ PAK: The learning vector quantization program package. Technical report, Helsinki University of Technology, 1996.

[11] 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.

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

[13] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, 1994.

[14] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[15] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[16] 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.