Supervised Clustering: Algorithms and Application

Nidal Zeidat, Christoph F. Eick, and Zhenghong Zhao
Department of Computer Science, University of Houston, Houston, Texas 77204-3010
{ceick, nzeidat, zhenzhao}@cs.uh.edu

submitted to PAMI on May 23, 2006

ABSTRACT

This work centers on a novel data mining technique we term *supervised clustering*. Unlike traditional clustering, supervised clustering assumes that the examples are classified and has the goal of identifying class-uniform clusters that have high probability densities. Three representative–based algorithms for supervised clustering are introduced: two greedy algorithms SRIDHCR and SPAM, and an evolutionary computing algorithm named SCEC. The three algorithms were evaluated using a benchmark consisting of UCI machine learning datasets. Experimental results suggest that SCEC outperforms the other two algorithms for almost all data sets in the benchmark. Moreover, a study of the fitness function used by supervised clustering shows that the landscape seems to have a "Canyonland" shape with many hills and plateaus, thereby, increasing the difficulty of the clustering task for the greedy algorithms. Potential applications of supervised clustering are discussed as well. We discuss how supervised clustering can be used for class decomposition and demonstrate with experimental results how it enhances the performance of simple classifiers. We also present a dataset editing technique, we call supervised clustering editing (SCE), which replaces examples of a learned cluster by the cluster representative. Our experimental results demonstrate how dataset editing techniques in general, and SCE technique in particular, enhance the performance of NN classifiers. We also discuss how supervised clustering could be used for regional learning.

1. INTRODUCTION

Clustering is a popular *descriptive* data mining task where one seeks to identify a finite set of groups (or clusters) to describe the data. A group contains objects that are similar while objects belonging to different groups are dissimilar to each other with respect to a particular similarity metric. Clustering techniques can be further subdivided into three groups, see Figure 1. *Traditional clustering*, Figure 1.a, focuses on finding tight clusters and would generally produce a clustering with four clusters A, B, C, and D. *Semi-supervised clustering* algorithms [3], Figure 1.b, not only focus on obtaining tight clusters but also on satisfying constraints with respect to a small set of pre-classified objects. Objects belonging to different classes should belong to different clusters while objects belonging to the same class should belong to the same cluster. Consequently, a semi-supervised clustering algorithm would generate clusters E, F, G, and H. Finally, a *supervised clustering* algorithm [12] that uses a fitness function which maximizes the purity of the clusters while keeping the number of clusters low would produce clusters I, J, K, and L, as depicted in Figure 1.c.

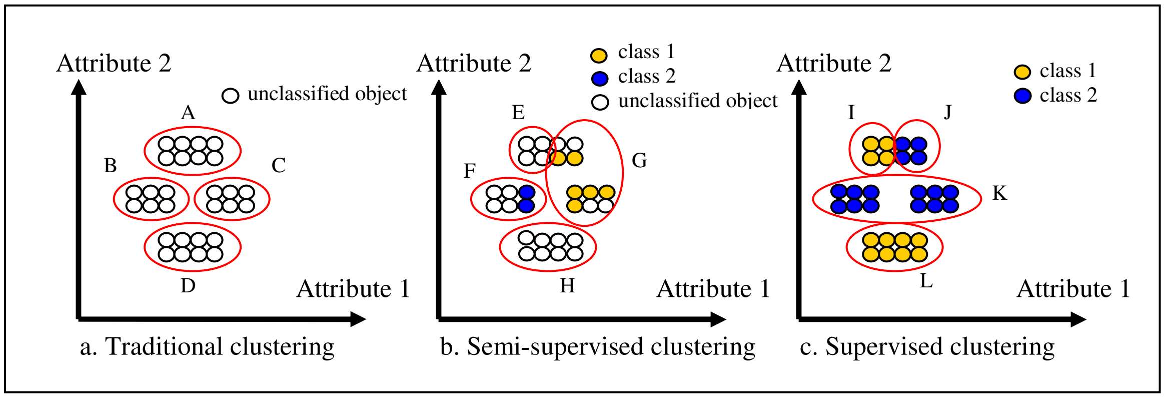

Figure 1: Traditional, semi-supervised, and supervised clustering

In this paper, a novel framework for supervised clustering is introduced. Three supervised clustering algorithms are introduced and analyzed with respect to their performance using a benchmark of UCI datasets as well as a set of 2-dimensional datasets. Furthermore, applications of supervised clustering are discussed as well.

This paper is organized as follows. Section 2 introduces supervised clustering. Section 3 of this paper presents related work with respect to both supervised clustering as well as representative based clustering. Section 4 presents and discusses the supervised clustering algorithms developed and investigated during this work. Analysis of the experimental results related to the performance of the developed algorithms is presented in Section 5 while discussion about the application of supervised clustering in dataset editing is presented in Section 6. Section 7 discusses the use of supervised clustering for regional learning. Finally, Section 8 draws conclusions for this research and presents potential future research directions.

2 SUPERVISED CLUSTERING

A summary of the notations used throughout this paper is given in Table 1.

Table 1: A summary of the notations used throughout the paper

Notation	Description
$O=\{o_1,\ldots, o_n\}$	Objects in a dataset
n	Number of objects in the dataset
CL_i	Class number i
$CL=\{CL_1,\ldots, CL_c\}$	The set of classes in the dataset
$d(o_i,o_j)$	Distance between objects o_i & o_j
c	The number of classes in the dataset
C_i	Cluster number i where i=1, …, k
$X=\{C_1,\ldots, C_k\}$	A clustering solution consisting of clusters C_1 to C_k
$k=\|X\|$	The number of clusters (or representatives) in a solution X
q(X)	A fitness function that evaluates a clustering X, e.g., Equation (1)
d	The dimensionality of a dataset, i.e., number of attributes.

As stated earlier, supervised clustering is applied on classified examples with the aim of producing clusters that have high probability density with respect to individual classes. Consequently, the fitness functions suitable for *supervised* clustering are significantly different from the fitness functions used by *traditional* clustering algorithms. Supervised clustering evaluates a clustering based on the following two criteria:

- *Class impurity, Impurity(X).* This is measured by the percentage of minority examples in the different clusters of a clustering X. A minority example is an example that belongs to a class different from the most frequent class in its cluster.
- *Number of clusters, k.* In general, we like to keep the number of clusters low.

In particular, we used the following fitness function in our experimental work (lower values for q(X) indicate a 'better' solution).

$$q(X) = \text{Impurity}(X) + \beta * \text{Penalty}(k) \tag{1}$$

$$\textit{where}\quad \textit{Impurity}(X) = \frac{\#\ \textit{of Minority Examples}}{n},\ \textit{and}\ \ \textit{Penalty}(k) = \begin{cases} \sqrt{\dfrac{k-c}{n}} & k \geq c \\ 0 & k < c \end{cases}$$

with *n* being the total number of examples and *c* being the number of classes in a dataset. The parameter β (0< β ≤ 5.0) determines the penalty that is associated with the number of clusters, *k*, in a clustering: higher values for β imply larger penalties for a higher number of clusters. The objective of Penalty(*k)* is to dampen the algorithm's natural tendency to increase the number of clusters. However, we believe this dampening ought not to be linear because the effect of increasing the number of clusters from *k* to *k*+1 has much stronger effect on the clustering outcome when *k* is small than when *k* is large.

As we will discuss later in Section 5, finding the best, or even a good, clustering X with respect to the fitness function *q*(X) is a challenging task for a supervised clustering algorithm due to the following reasons:

1. The search space is very large, even for small datasets.
2. The fitness landscape of q contains a large number of local minima.
3. There is a significant number of ties (i.e., clusterings X_1 and X_2 with $q(X_1)=q(X_2)$) in the fitness landscape creating plateau-like structures that present a major challenge for most search algorithms, especially hill climbing and greedy algorithms.

As there are many possible algorithms for supervised clustering, our work centers on the development of *representative-based* supervised clustering algorithms. Representative-based clustering aims at finding a set of *k* representative examples from the dataset that best characterize the whole dataset. Clusters are created by assigning each example in the dataset to the closest representative using some distance metric. Following are the reasons that motivated us to focus on representative based style clustering algorithms.

- Representatives are quite useful for applications such as data summarization, dataset editing to improve classifiers, and class decomposition to improve simple classifiers.
- Clustering algorithms that restrict cluster representatives to objects belonging to the dataset explore a smaller solution space if compared with centroid–based clustering algorithms, such as the *k-Means* algorithm[1].
- Representative-based clustering algorithms are more robust with respect to outliers.
- When using representative-based clustering algorithms, only an inter-example distance matrix is needed and no "new" distances have to be computed, as it is the case with *k-means*.
- Furthermore, representative based algorithms that use actual dataset examples as cluster representatives are more applicable when the data is represented by attributes that include ordinal and nominal type data. For example, the *k*-Means clustering algorithm which uses centroids as representatives (points in the input space), which might not be actual examples in the datasets, can only be applied when the mean of the cluster is defined, which is not the case for ordinal and nominal attributes.

[1] There are 2^n possible centroids for a dataset containing *n* objects.

3 RELATED WORK

There are two approaches that can be viewed as supervised clustering approaches. Sinkkonen et al. [19] proposed a probabilistic approach based on discriminative clustering that minimizes distortion within clusters. Distortion, in their context, is represented (approximated) by the average Kullback-Leibler divergence of the conditional probability of the auxiliary data and the distributional cluster representatives over the Voronoi regions that cover the primary data. Similarly, Tishby et al. [22] introduced the information bottleneck method. Based on this method, they proposed an agglomerative clustering algorithm [20] that minimizes mutual information loss with respect to class information (auxiliary data). The work of Tishby et al. [22] and of Sinkkonen et al. [19] is geared towards information retrieval type of applications. Our approach is partitioning-based and is heuristic in nature.

There has, also, been some work that has some similarity with our research under the heading of semi-supervised clustering. The existing research on semi-supervised clustering can be subdivided into 2 major groups: similarity-based methods and search-based methods (for more details see [1]). Similarity-based methods create a modified distance function that incorporates the knowledge with respect to the classified examples and use a traditional clustering algorithm to cluster the data. Search-based methods, on the other hand, modify the clustering algorithm itself but do not change the distance function. Cohn et al. [3] use constraints, labeled data, to help the algorithm learn a distance function. Furthermore, Basu et al. [4] investigate determining the most informative set of constraints to get improved clustering performance. Bilenko et al. [5] present a new semi-supervised clustering algorithm that integrates both the distance-function-learning based as well as the algorithm-modification based techniques in a uniform principled framework. Although the work of Basu et al. ([3], [4], and

[5]) aims at producing clusterings with high probability density with respect to a single class, they use only few classified examples while we assume that the whole dataset is classified. Another difference between our work and the work of Basu et al. is the fitness function that we use to evaluate clusterings. Our fitness function evaluates a clustering based on how pure clusters are as well as the number of clusters proposed (see Equation (1)) while Basu et al. evaluate clusterings based on traditional clustering fitness functions, i.e., mean distance between examples and their cluster representatives.

Xing et al. [27] (and similarly [2]) take the classified training examples and transform those into constraints (points that are known to belong to different classes need to have a distance larger than a given bound) and derive a modified distance function that minimizes the distance between points in the dataset that are known to be similar with respect to these constraints using classical numerical methods. The K-means clustering algorithm in conjunction with the modified distance function is then used to compute clusters. Klein et al. [17] propose a shortest path algorithm to modify a Euclidian distance function based on prior knowledge.

Demiriz et al. [8] propose an evolutionary clustering algorithm in which solutions consist of *k* centroids and the objective of the search process is to obtain clusters that minimize (the sum of) cluster dispersion and cluster impurity. Cohn et al. [6] modify the popular EM algorithm so that it is capable of incorporating similarity and dissimilarity constraints.

4 REPRESENTATIVE-BASED SUPERVISED CLUSTERING ALGORITHMS

We have designed and evaluated three representative-based supervised clustering algorithms SCEC, SRIDHCR, and SPAM. The performance of these algorithms was compared with that of the traditional clustering algorithm PAM [16]. This Section describes these algorithms.

4.1 Supervised Partitioning Around Medoids (SPAM)

This algorithm is a variation of the popular unsupervised clustering algorithm *Partitioning Around Medoids* (PAM) [16]. The objective in PAM is to find *k* representative examples in a data set that minimize the following objective function

$$\text{Tightness}(X) = \frac{1}{|O|}\sum_{i=1}^{n} d(o_i, medoid(o_i)) \qquad (2)$$

where X is a clustering solution (i.e., a set of cluster representatives or medoids) and $d(o_i, medoid(o_i))$ is the distance between example o_i and its medoid. Unlike PAM, *SPAM* opts to minimize the fitness function in Equation (1). The number of clusters *k* is an input parameter to the algorithm. The algorithm consists of two sub-algorithms. Sub-algorithm *SBUILD* intelligently selects the initial solution. The second sub-algorithm, *SSWAP*, tries to improve the clustering produced by *SBUILD* by exploring all possible replacements of a single representative by a single non-representative. Like PAM, SPAM terminates if no replacement can be found that leads to a clustering with a better (lower) fitness value with respect to q(X).

As stated earlier, we also implemented the traditional clustering algorithm PAM as part of our research. We use PAM as a representative of traditional representative-based clustering algorithms for comparison with supervised clustering algorithms. PAM was implemented based on a FORTRAN version of the algorithm that we obtained from the authors of [16].

4.2 Single Representative Insertion/Deletion Steepest Decent Hill Climbing with Randomized Restart (SRIDHCR)

This greedy algorithm starts by randomly selecting a number of examples from the dataset as the initial set of representatives. Clusters are then created by assigning examples to the cluster of their closest representative. Starting from this randomly generated set of representatives, the algorithm tries to improve the quality of the clustering by adding a single non-representative example to the set of representatives as well as by removing a single representative from the set of representatives. The algorithm terminates if the solution quality (measured by *q*(X)) does not show any improvement. Moreover, we assume that the algorithm is run *r* (input parameter) times starting from a randomly generated initial set of representatives each time, reporting the best of the *r* solutions as its final result. The pseudo-code of algorithm SRIDHCR that was used for the evaluation of supervised clustering is given in Figure 2. It should be noted that the number of clusters *k* is not fixed for SRIDHCR; the algorithm searches for "good" values of *k*.

```
REPEAT r TIMES
      curr := a randomly created set of representatives (with size between
      c+1 and 2c)
      WHILE NOT DONE DO
         1. Create new solutions S by adding a single non-representative
            to curr and by removing a single representative from curr
         2. Determine the element s in S for which q(s) is minimal (if
            there is more than one minimal element, randomly pick one)
         3. IF q(s)<q(curr) THEN curr:=s
            ELSE IF q(s)=q(curr) AND |s|>|curr| THEN curr:=s
            ELSE terminate and return curr as the solution for this run.
Report the best out of the r solutions found.
```

Figure 2: Pseudocode for algorithm SRIDHCR.

To illustrate how the algorithm works let us have a closer look at a run of the algorithm for the Iris-Plant data set that consists of 150 flowers, numbered 1 through 150. The algorithm starts with a randomly generated set of representatives, e.g. {5, 30, 48, 72, 150}. Firstly, the algorithm creates clusterings by adding a single non-representative to the current set of representatives. Secondly, the algorithm creates clusterings obtained by removing a single representative from the current set of representatives. The 150 (e.g., 145+5) clusterings are then evaluated, and the solution whose clustering has the lowest value with respect to q(X) is selected, depicted as the solution for iteration # 1. The search now continues using {5, 30, 48, 72, 150, 110} as the new set of representatives. In the second iteration the solution {5, 30, 48, 72, 150, 110, 52} (flower 52 was added to the set of representatives) turned out to be the best solution, leading to an improvement in fitness from 0.054 to 0.043.

Table 2: Set of Representatives Explore

Iter. #	Set of medoids producing lowest q(X) in the run	q(X)	Purity
0	5 30 48 72 150 (Initial set)	0.098	0.913
1	5 30 48 72 150 110	0.054	0.960
2	5 30 48 72 150 110 52	0.043	0.973
3	5 30 48 72 150 110 52 86	0.038	0.980
4	5 30 48 72 150 110 52 86 73	0.033	0.987
5	30 48 72 150 110 52 86 73	0.031	0.987
6	48 72 150 110 52 86 73	0.030	0.987
7	48 150 110 52 86 73	0.027	0.987

The algorithm continues iterating as long as there is an improvement in fitness function q(X). The algorithm terminates after 7 iterations with the solution {48, 150, 110, 52, 86, 73,}. Table 2 illustrates how the set of representatives changed during the iterations. It is worth mentioning that in iterations 5, 6, & 7, the class purity did not improve any further. Nevertheless, the algorithm did not stop. This is because the fitness function q(X) does not only try to maximize the class purity, but also minimizes the number of clusters; the algorithm, therefore, continued and found a clustering that uses a smaller number of clusters but still achieved the same class purity of 0.987.

4.3 Supervised Clustering using Evolutionary Computing (SCEC)

This algorithm uses evolutionary computing techniques to seek for the “optimal” set of representatives by evolving a population of solutions over a fixed number of generations *N*. The size of the population (PS) is fixed to a predetermined number when running SCEC. The initial generation is created randomly. The subsequent generations are generated by applying three different genetic operators to members of the current generation that are selected based on the principles of *survival of the fittest*. Figure 3 presents a flowchart of the SCEC algorithm, (more details about SCEC are discussed in Section 5.2) whose key features include:

1. Chromosomal Representation: A solution consists of a set of representatives that are a subset of the examples to be clustered.
2. Genetic Operators:

   Mutation: replace a representative by a non-representative

   Crossover: take 2 "parent" solutions and create an offspring solution as follows:

   A. Include all representatives that occur in both parents in the offspring

   B. Include representatives that occur in a single parent with a probability of 50%.

   Copy: Copy a member of the current generation into the next generation.
3. Selection: K-tournament selection is used to select solutions for generating the next generation through mutation, crossover, and copying. K-tournament randomly selects K solutions from the current population, and uses the solution with the lowest q(X) value to be added to the mating pool for the breeding of the next generation.
4. Transformation of the Chromosomal Representation into Clusters and Evaluation:

   A. Create clusters by assigning the remaining examples to the closest representative in the solution to be evaluated.

   B. Evaluate the so obtained clustering X using q.

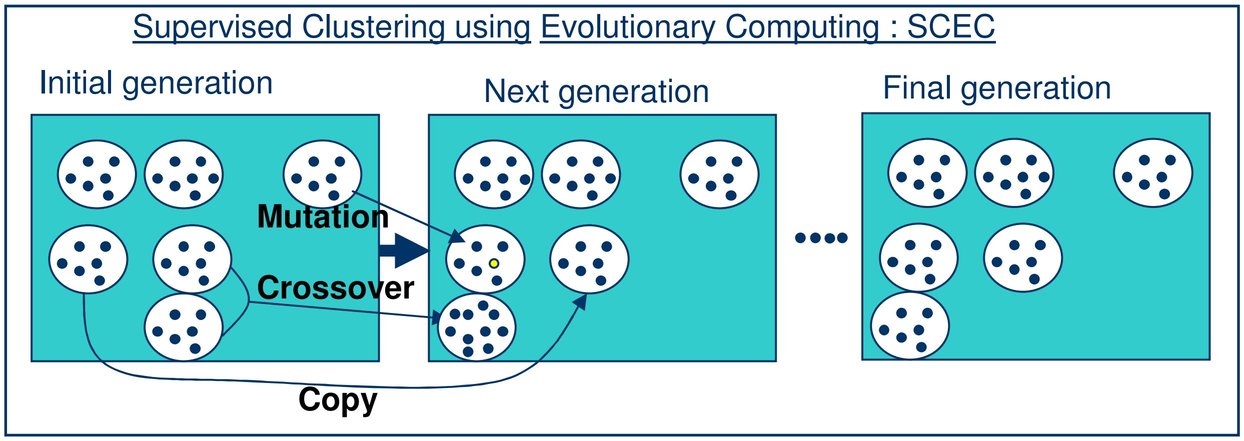

Figure 3: Key features of the SCEC algorithm

5. EXPERIMENTAL EVALUATION OF SUPERVISED CLUSTERING ALGORITHMS

This Section describes the experiments conducted to evaluate and analyze the clustering algorithms developed in this research.

5.1 Datasets Used in the Experiments

In order to study the performance of the clustering algorithms presented in the previous Section, we applied those algorithms to a benchmark consisting of the datasets listed in Table 3.

Table 3: Datasets used in the benchmark

Dataset Name	**No. of Examples**	**No. of Attributes**	**No. of Classes**
IRIS plants	150	4	3
Glass	214	9	6
Diabetes	768	8	2
Waveform	5000	21	3
Ionosphere	351	34	2
Heart-H	294	13	2
Image Segmentation	2100	19	7
Vehicle Silhouettes	846	18	4
Vote	435	16	2
Vowel	990	13	11
Heart-StatLog	270	13	2
Complex9	3031	2	9
Oval10	3359	2	10

The first 11 datasets were obtained from University of California at Irving Machine Learning Repository [23]. The last 2 datasets, we named Complex9 and Oval10, are two dimensional spatial datasets whose examples distribute in different shapes. These two 2D datasets were obtained from the authors of [18] and some of them seem to be similar to proprietary datasets used in [15]. These datasets are used mainly for visualization purposes presented later in this Section.

5.2 How the Experiments were Conducted: Parameters, and Preprocessing

In the experiments, algorithm SRIDHCR was run 50 times, each time with a different set of initial representatives and the quality of the best solution found was reported. Moreover, the results of running SRIDHCR for a fixed value of β were used to determine the *k*-value with wh [ich PAM and SPAM were run. Algorithm SCEC creates the initial population randomly which is composed of solutions having between c+1 and 2c representatives. SCEC was run for 1500 generations using a population size of 400. The mutation probability is initially set to 0.95 when creating generation 1 and is linearly decreased to 0 at generation 1500, whereas the initial crossover probability is initially 0 and is linearly increased to 0.95 at the end of the run. The remaining 5% of a new generation are created by copying solutions from the previous generation. The algorithm uses tournament selection with a tournament size of K=2. When it is time to apply one of the genetic operators on members of the current generation, 2 members (i.e., solutions) are randomly picked from the pool and the solution that produces the smaller q(X) is selected to participate in the breeding of the next generation. Values of parameters used were determined experimentally. During this parameter selection process we observed that [29]:

- Using large population sizes lead to finding better solutions; e.g. using PS=400, N=1500 performed better than PS=200, N=3000.
- Using higher mutation rates and lower crossover rates leads to better solution quality.
- Using lower selective pressure (smaller values for k) leads to better solution quality.

The investigated clustering algorithms were evaluated based on the following measures:

1. Cluster purity, defined as the ratio of the number of examples inside a cluster that belong to the majority class to the total number of examples in that cluster.
2. Value of the fitness function q(X), see Equation (1).
3. Tightness (X), see Equation (2).

4. Wall-Clock Time (WCT). This is the actual time, in seconds, that the algorithm took to finish the clustering task. The algorithms were run on a computer that has a Pentium 4 processor and 512 MB of main memory.

During preprocessing, a dissimilarity matrix is created using the Manhattan distance function. Prior to that, all attribute values are normalized using a normalizing function that gives a value of 1.0 to the largest attribute value and 0.0 to lowest attribute value

5.3 Experimental Results

This section presents and analyzes the experimental results we obtained. It is important to note that each set of results presented in a table was obtained independently through a separate set of experiments. The results in Tables 4 & 5 show that among the three supervised clustering algorithms, SCEC consistently finds the best solutions for all four datasets, whereas SRIDHCR finds "good" solutions, and, somewhat surprisingly, SPAM performs quite poorly in the experiments.

Table 4: Comparative performance of the different algorithms (□=0.1)

Algorith m	***k***	**Cluster Purity**	**q(X)**	**Tight ness (X)**	**Wall Clock Time**
Iris-Plants Dataset					
PAM	3	0.907	0.093	0.081	0.060
SCEC	5	0.993	0.018	0.107	1019.0
SRIDHCR	3	0.980	0.020	0.113	13.000
SPAM	3	0.973	0.027	0.133	0.320
Vehicle Dataset					
PAM	65	0.701	0.326	0.044	372.00
SCEC	132	0.923	0.116	0.049	31989.
SRIDHCR	65	0.835	0.192	0.062	1666.0
SPAM	65	0.764	0.263	0.097	1090.0
Segmentation Dataset					
PAM	53	0.880	0.135	0.027	4073.0
SCEC	60	0.989	0.026	0.046	22598.
SRIDHCR	53	0.980	0.035	0.050	9868.0
SPAM	53	0.944	0.071	0.061	1422.0
Pima-Indian Diabetes Dataset					
PAM	45	0.763	0.237	0.056	186.00
SCEC	64	0.893	0.135	0.074	12943.
SRIDHCR	45	0.859	0.164	0.076	660.00
SPAM	45	0.822	0.202	0.086	58.000

As β increases from 0.1 (in Table 4) to 0.4 (in Table 5), the penalty for increasing the number of clusters increases as well, see

Equation (1). Consequently, the algorithms produce solutions with smaller number of clusters. Due to the fact that more smaller-sized solutions (solutions with a smaller number of clusters) will be searched when β increases, the algorithms take less time for β=0.4. It should be noted that the purity values in Table 5 are slightly lower than the corresponding numbers in Table 4 because using a larger number of clusters (larger value for *k* in Equation 1) is penalized more severely by the fitness function q(X) when β is set to 0.4.

Table 5: Comparative performance of the different algorithms (□=0.4)

Algorithm	***k***	**Cluster Purity**	**q(X)**	**Tightness(X)**	**WCT (Sec.)**
IRIS-Plants Dataset					
PAM	3	0.907	0.093	0.081	0.06
SCEC	3	0.987	0.013	0.125	618.00
SRIDHCR	3	0.987	0.013	0.125	9.00
SPAM	3	0.973	0.027	0.133	0.30
Vehicle Dataset					
PAM	5	0.681	0.418	0.046	505.00
SCEC	6	0.857	0.247	0.059	10947.00
SRIDHCR	5	0.835	0.265	0.067	1130.00
SPAM	5	0.754	0.345	0.100	681.00
Segmentation Dataset					
PAM	3	0.875	0.169	0.032	1529.00
SCEC	2	0.969	0.069	0.050	11533.00
SRIDHCR	3	0.970	0.074	0.051	8450.00
SPAM	3	0.940	0.103	0.065	1053.00
Pima-Indians-Diabetes Dataset					
PAM	2	0.656	0.344	0.101	1.00
SCEC	9	0.819	0.219	0.103	1700.00
SRIDHCR	2	0.776	0.224	0.139	228.00
SPAM	2	0.772	0.227	0.125	2.70

Comparing the results of the algorithm SPAM with those for algorithm SRIDHCR, we can clearly see that SRIDHCR outperforms SPAM. A factor that we believe contributes to this observation is that the fitness landscape induced by q(X) contains many local minima (i.e., local hills). SPAM is not particularly good in coping with those "canyon like" structures in the fitness landscape. Another contributing factor, we believe, is that SRIDHCR has the interesting characteristic that when attempting to enhance a solution with *k* representatives, it looks for better solutions with *k*–1 as well as *k*+1 representatives, whereas SPAM looks for better solutions with exactly *k* representatives. We believe that the capability of algorithm SRIDHCR to add new elements to the solution set, some of which are removed again at a later stage of the search contributes to its better performance. Furthermore, encouraged by its smaller average runtime, SRIDHCR can be restarted up to 33 times in the same time that SPAM needs to complete a single run. These re-runs with different initial sets of representatives allows SRIDHCR to explore different regions of the solution space, which we believe is a third contributing factor to the significantly better performance of algorithm SRIDHCR compared to SPAM.

On the other hand, SCEC found much better solutions than the other 2 algorithms followed by algorithm SRIDHCR. We attribute this observation to the fact that the fitness landscape induced by q(X) for a particular dataset contains a large number of local minima. It seems that greedy algorithms, such as SPAM and SRIDHCR, do not perform particularly well in this fitness landscape terminating prematurely too often.

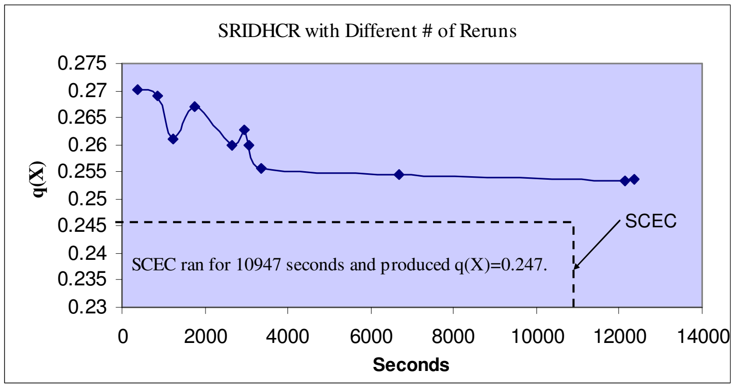

Figure 4: Performance of SRIDHCR for different number of reruns

Finally, we wanted to compare the performance of SRIDHCR and algorithm SCEC when both algorithms are given the same time to perform a clustering task. From Table 5, we can see that algorithm SCEC took 10947 seconds to finish its clustering of the Vehicle dataset and produced a clustering with q(X)=0.247. We ran an experiment on the Vehicle dataset with β=0.4 in which we changed the number of reruns (*r*) from 25 up to 675 times to make the time used by SRIDHCR comparable to that used by algorithm SCEC.

SRIDHCR reported the best results out of these *r* reruns. Each experiment was repeated 3 times each time reshuffling the dataset examples. Figure 4 shows a plotting of the values of the fitness function q(X) produced by SRIDHCR versus the time the algorithm used to produce that value. Notice from Figure 4 that algorithm SRIDHCR was unable to produce a value of q(X) as low as the value produced by algorithm SCEC even when it ran for as long as SCEC did. Actually, it failed to do so even when it ran for a longer time.

5.4 A Visualization of Characteristics of Representative Based Supervised Clustering

This section uses visualization in an attempt to shed light on the way supervised clustering algorithms work.

5.4.1 How Cluster Representatives are Selected

Multiple factors play a role in selecting good representatives. Among these factors are the density of the examples in the clusters, the location of the cluster with respect to other adjacent clusters that belong to the same or different class, and cluster shape. Figure 5 gives examples of three cases. Figure 5.a shows two clusters that belong to two different classes, depicted as black and white. Notice that choosing any representative in either cluster does not affect cluster purities (i.e., does not attract objects from the other cluster) because there is enough space between the two clusters. Two suggested cluster representatives for Figure 5.a are encloses in rectangles. On the other hand, Figure 5.b shows three clusters that have the same convex shape, same size and density, and are close to each others. In this case a SC algorithm would choose a representative that is in the middle of the cluster, such as the examples that have circles around them, in order not to attract examples of neighboring clusters that belong to a different class. Choosing the examples with rectangles around them as cluster representatives, for example, will attract two of the white examples from cluster C to cluster D. This will degrade the overall average cluster purity and, consequently, the quality of the clustering. Finally, for more complex cluster shapes and sizes, as illustrated in Figure 5.c, the clustering algorithm will choose the representatives lined up against each others and properly spaced to create the best possible decision boundaries that produce good cluster purities. Referring to Figure 5.c, cluster representatives are enclosed in dotted rectangles. Improving the quality of a clustering in a situation similar to the examples in Figures 5.b and 5.c is challenging because replacing any of the cluster representatives with another example necessitates that other representatives be modified as well to keep the lining up and spacing proper to minimize misclassification.

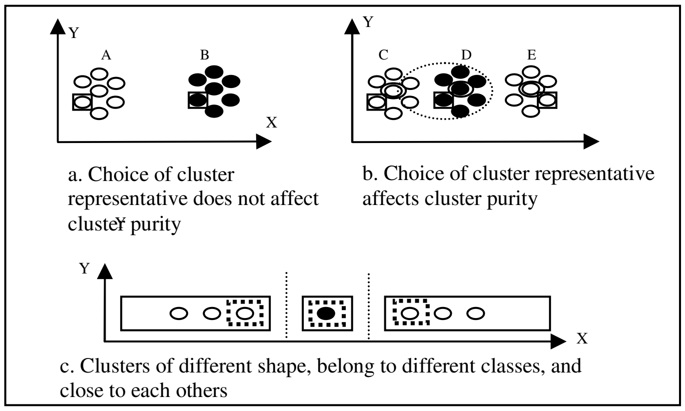
Y
A
B
X
a. Choice of cluster representative does not affect cluster purity
Y
C
D
E
b. Choice of cluster representative affects cluster purity
Y
X
c. Clusters of different shape, belong to different classes, and close to each others

Figure 5: How cluster shape and relative location affects the choice of its representative

Figure 6 shows a clustering of the dataset Complex9. Examples of the same shape were initially labeled with the same class. The dataset was then clustered by the representative-based SC algorithm SRIDHCR. Representatives selected are characterized by (red) solid squares. To minimize class impurity, the clustering algorithm sometimes selects representatives located in the middle of the cluster (lower left rectangle clusters in Figure 6) and other times located on the edge of their clusters (the clusters that make the oval shaped class). Notice how the (red) solid square symbols in the two rectangle clusters in the lower left part of Figure 6 are lined up with proper spacing. If either representative is moved further up or down, it will cause some of the examples of one of the clusters to be attracted to the other cluster and, consequently, be misclassified. This explains the bad performance of PAM-style algorithms that do not follow this behavior and move representatives of adjacent clusters independently with no regard to the relative location of the representatives of the neighboring clusters.

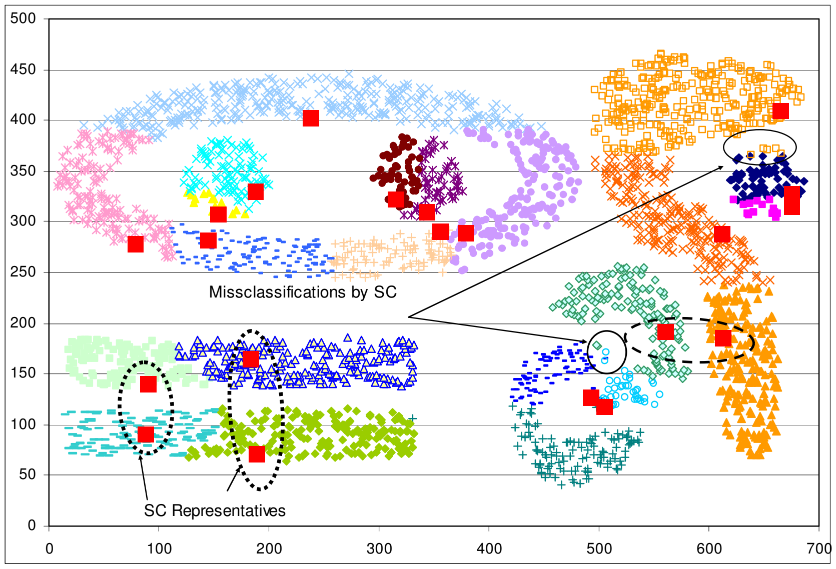

Figure 6: Complex9 dataset clustered using SC algorithm SRIDHCR

A challenge that a representative-based clustering algorithm (i.e., an algorithm that selects one representative for a cluster) faces is that it can not discover clusters of non-convex shapes. However, clusters having non-convex shapes can be approximated by a union of several neighboring clusters that are dominated by the same class. An example of such case is the large circular cluster in Figure 6, which was approximated using 5 neighboring clusters without any loss in purity; cluster representatives are represented in solid (red) squares.

Although we claim that the representative-based supervised clustering algorithm shows a good performance[2], the obtained purity in the experiment depicted in Figure 6 was 99.4% and not 100% due to minor errors (highlighted in the figure) in the right-most cluster and in the two interleaved U-shaped clusters near the bottom center.

[2] The Nearest Representative classifier only uses 22 of the 3031 examples in the Complex9 dataset

5.4.2 Adjacent Clusters for the Same Class

Another characteristic of the fitness function q(X) is its penalizing for the number of clusters *k*. This penalty makes the clustering algorithm reluctant to increase the number of clusters unless it is justified by a balancing decrease in the "Impurity(X)" part of q(X). To verify if supervised clustering algorithms take advantage of this property, we ran the following experiment. We ran SC algorithm SRIDHCR on the Oval10 dataset. Before clustering, the examples were assigned to 4 classes as shown in Figure 7. The result of the clustering is shown in Figure 8. Expectedly so, the algorithm chose to separate the right most two clusters into two separate clusters to avoid attracting examples from Class2. On the other hand, it combined the top left 2 clusters into one bigger cluster with 1 representative because it did not affect purity, but at the same time decreased the number of clusters *k* (see Equation (1)). Cluster representatives are denoted by solid black rectangles.

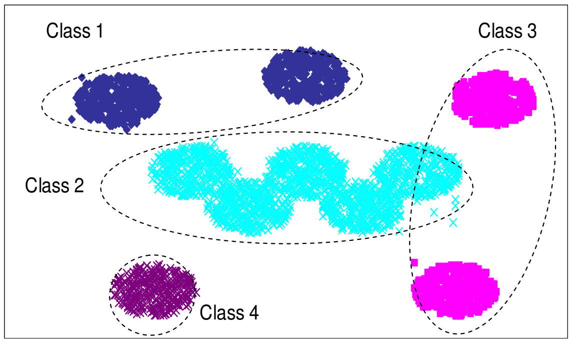

Figure 7: The 10-Ovals dataset with 4 classes

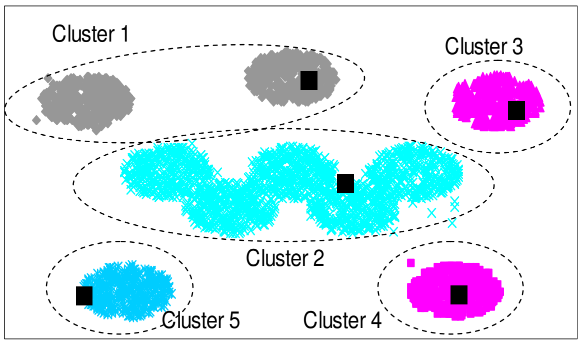

Figure 8: The 10-Ovals dataset with 4 classes clustered using SRIDHCR, □=0.25

5.5 Characteristics of the Fitness Function and Fitness Landscape

As part of the supervised clustering research, we wanted to study the landscape of the solution space characterized by the fitness function q(X). The objective was to study how a supervised clustering algorithm behaves during its search for the best clustering when it uses q(X) to evaluate a candidate clustering X. We expect the study to help us explain some of the observations gathered from the experimental results as well as to help us in changing the navigation part of a SC algorithm to make it more effective in covering larger area of the solution landscape which might increase the chances of finding better solutions.

Experiments were conducted using SC algorithm SRIDHCR on the Vehicle dataset with β set to a value of 0.4. Three initial solution sizes were used for the Vehicle dataset, namely 5, 75, and 300 representatives. Clustering process was repeated 1000 times, each time starting with a randomly selected initial solution of the same size (i.e., 5, 75, or 300). The 1000 solutions were grouped according to the number of clusters in the solution. Figure 9 shows the number of solutions produced for the different solution sizes. Graphs A, B, and C represent results for the initial solutions sizes of 5, 75, and 300 representatives, respectively. Notice how in Graphs A and B, the best solution has a size that is not among the most frequent solution sizes, but higher. Results in Graphs A and B indicate that the optimal solution lies somewhere to the right of both graphs. On the other hand, the best solution for Graph C lies in the lower range of the graph. What we learn from these experiments is that although the best solution the SC algorithm produces seems to be in the direction of the optimal solution, the algorithm seems to always get stuck within a certain k-range that is determined by the number of representatives in the initial solution. For example, in Graph A, algorithm SRIDHCR produced a

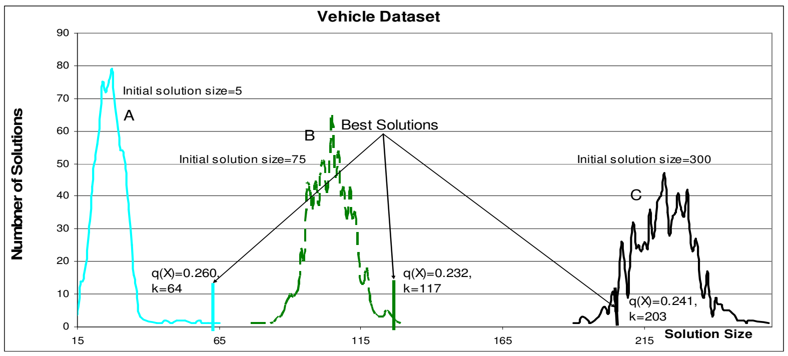

Figure 9: Number of solutions versus solution size for the Vehicle dataset

best solution of q(X)=0.26 and *k*=64. It could not reach the area of the solution space that produced a better q(X)=0.232 for *k*=117 in Graph B which started with an initial solution of 75 representatives. The same could be said about the case when the initial solution was set to 300 in Graph C. The best solution was of size *k*=203 and q(X)=0.241. The algorithm could not extend its set of explored solutions to include the solution size of 117 that produced the best q(X) among all three graphs. The results in this experiment might explain the observations presented earlier in Figure 4 Section 5.3 where algorithm SRIDHCR was incapable of reaching the quality of solution that algorithm SCEC produced even when it was given amble resources.

We also wanted to find how difficult the job of a supervised clustering algorithm is by counting the number of different solutions the algorithm produces as it is run *r* number of times. To do that we ran algorithm SRIDHCR 1000 times for the Vehicle dataset for different initial solution sizes. We assumed that two different clusterings X_1 and X_2 are different (i.e., $X_1 \neq X_2$) if the following is true:

$$X_1 \neq X_2 \Leftrightarrow (q(X_1) \neq q(X_2)) or (|X_1| \neq |X_1|)$$

where $|X_1|$ and $|X_2|$ are the number of clusters for solutions X_1 and X_2, respectively. The number of different solutions as defined above is a lower bound because even if $q(X_1)=q(X_2)$ and $|X_1|=|X_2|$ the clusters might still contain different examples as long as the number of minority examples in both clusterings (i.e., solutions) are the same.

The number of different solutions gives an idea about the number of local minima the solution space contains within the solution space area visited by the supervised clustering algorithm during the *r* reruns. The higher the number of different solutions, the more difficult it is to reach the area of the solution space where the optimal solution lies.

Table 6: Percentage of different solutions for 1000 reruns

Dataset, β, Size of initial solution	Lower bound on % of different solutions produced
Vehicle, 0.00001, 5	74.5
Vehicle, 0.00001, 75	52.8
Vehicle, 0.4, 5	61.8
Vehicle, 0.4, 75	51.7
Vehicle, 0.4, 300	49.6
Heart-H, 0.4, 4	7.7 %
Heart-H, 0.4, 25	21.2 %
Heart-H, 0.4, 150	19.9 %

As can be seen from Table 6, a difficult clustering task such as the Vehicle dataset, for example, generated 50% to 75% different solutions from the 1000 solutions produced. Another way to interpret results in Table 6 is to notice that more than 50% of the time, the hill climber ends up on a different hill and that the same hill is reached at an average less than 2 times. This also explains quite well why SRIDHCR performs quite poorly for the Vehicle dataset: it gets stuck on the "wrong" hill too quickly. Table 6 shows better performance for the algorithm SRIDHCR on the Heart-H as the chance of producing a different solution is much less than that that for the Vehicle dataset. This further emphasizes what we mentioned before that greedy algorithms, such as SRIDHCR, have difficulties producing good solutions in such "Canyonland" like fitness landscape compared to non-greedy algorithms such as SCEC.

6 SUPERVISED CLUSTERING FOR DATASET EDITING

Since its introduction by Fix and Hodges in 1951 [14], the Nearest Neighbor (NN) rule and its generalizations have received considerable attention from the research community. Most research aimed at producing time-efficient versions of the NN rule (for a detailed survey see Toussaint [21]). Although the NN rule continues to be one of the most popular non-parametric classification techniques, it has some drawbacks. First, if the number of objects in the dataset is very large and with high dimensionality, computing the nearest neighbors becomes quite time consuming. Second, if the original training dataset contains erroneously labeled instances, the prediction accuracy will be greatly decreased.

Condensing and Editing are two techniques that have been proposed to address these problems. Condensing aims at selecting the minimal subset of examples that lead to the same behavior of the NN-rule as using the original dataset. Editing, on the other hand, seeks to remove noise (i.e., miss-labeled) examples from the original training set with the goal of producing smooth decision boundaries and, consequently, improve prediction accuracy.

In this Section, we study a set of dataset editing techniques and compare their performance using a benchmark consisting of a set of UCI datasets as well as a set of two dimensional spatial datasets. Furthermore, we also propose a new dataset editing technique based on supervised clustering. Moreover, the benefits of editing techniques have not been systematically analyzed in the literature which is the second motivation of this work[3].

6.2 Editing Algorithms Investigated

6.2.1 Wilson Editing

Wilson editing [26] removes all instances that have been misclassified by the NN-rule applied on a training dataset. Wilson editing cleans interclass overlap regions, thereby leading to smoother boundaries between classes. Figure 10.a shows a hypothetical dataset where examples that are misclassified using the 1-NN-rule are marked with circles around them. Figure 10.b shows the edited dataset after applying Wilson editing. The pseudocode for Wilson editing technique is presented in Figure 11.

[3] Preliminary results from research we conducted on using supervised clustering for dataset editing were published in [13]

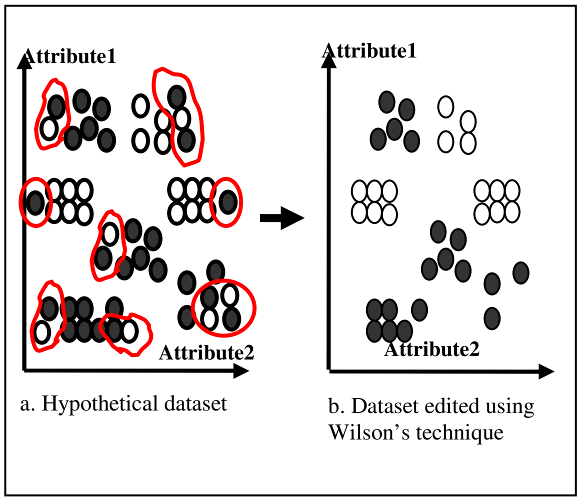

Figure 10: Wilson editing for 1-NN classifier

```
PREPROCESSING
A: For each example $o_i$ in the
  dataset $O$,
  1:Find the K-Nearest Neighbors of
     $o_i$ in $O$(excluding $o_i$)
  2:Label $o_i$ with the class
     associated with the largest
     number of examples among the $K$
     nearest neighbors (breaking
     ties randomly)
B: Edit Dataset $O$ by deleting all
  examples that were misclassified
  in step A.2 resulting in subset $O_r$

CLASSIFICATION RULE:
Classify new example q using K-NN
rule applied on the edited subset
$O_r$
```

Figure 11: Pseudocode for Wilson editing algorithm

6.2.4 Supervised Clustering Editing (SCE)

In supervised clustering (SC) editing, a SC algorithm is used to cluster a training dataset *O*. Then *O* is replaced by subset O_r, which consists of cluster representatives that have been discovered by the SC algorithm. We call a 1-NN based classifier that uses the set of cluster representatives as training set the *Nearest Representative (NR)* classifier. Figure 12 gives an example that illustrates how SC editing works. Figure 12.a shows a dataset that has not been clustered yet. Figure 12.b

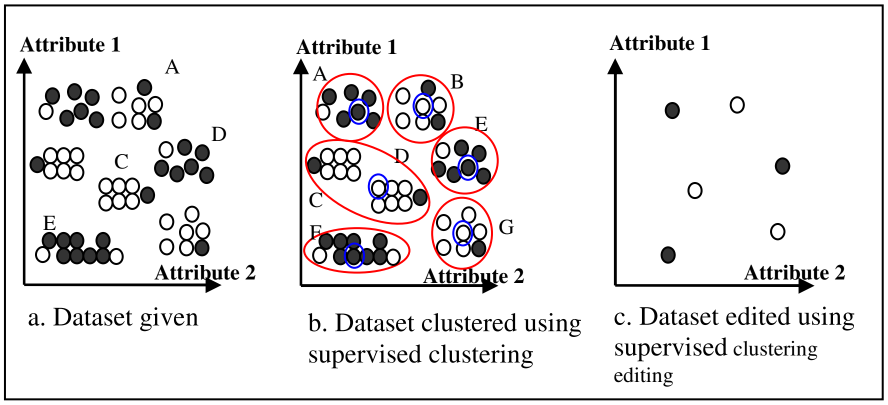
Attribute 1
A
D
C
E
Attribute 2
a. Dataset given
Attribute 1
A
B
E
D
C
F
G
Attribute 2
b. Dataset clustered using supervised clustering
Attribute 1
Attribute 2
c. Dataset edited using supervised clustering editing

Figure 12: Illustration of supervised clustering editing (SCE)

shows a dataset that was clustered into 6 clusters using a SC algorithm. Cluster representatives are marked with (blue) circles around them. Figure 12.c shows the edited subset O_r resulting from removing all examples except for the cluster representatives. To classify a new unseen example the NR classifier finds the nearest representative for the new example. The new example is, then, given the class of that representative.

The reader might ask whether the same objective of SCE could be accomplished by clustering examples of each class separately using a traditional clustering algorithm, such as PAM? Figure 13 illustrates why this is not a good idea. Examples in the figure belong to one of two classes 'X' and 'O'. If this dataset is edited using SCE, the (red) underlined **O** example and the (purple) underlined **X** example would be picked as representatives. On the other hand, if examples of each class are clustered separately, the (blue) italic ***O*** example and the (purple) underlined **X** example would be picked as the representatives. Note that the (blue) italic ***O*** representative is **not** a good choice for dataset editing, because it "attracts" examples belonging to the class 'X' which leads to misclassifications.

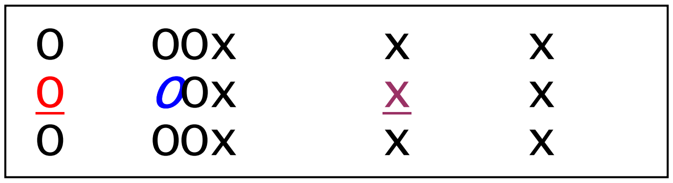

Figure 13: Supervised clustering editing versus clustering each class separately

6.3 Experimental Results

In this section, the performance of the two editing techniques are analyzed for a benchmark consisting of 11 UCI datasets [23] using the performance measures of compression rate and class prediction accuracy. Furthermore, we analyze how differently the editing techniques cope with artificial noise of different forms and degrees using the 2D spatial datasets.

We used 5-fold cross validation to determine the accuracies of the editing techniques. Furthermore, we repeated each experiment three times every time reshuffling the examples in the dataset. In the experiments, Wilson editing was run with *K* (neighborhood size) equal to 1. As for supervised cluster editing, we used the algorithm SRIDHCR for clustering with r=10 and β=0.4.

6.3.1 Class Prediction Accuracy for the Investigated Editing Techniques

To comparatively study Wilson and SC editing techniques with respect to improvement in classification accuracy we applied both editing techniques on 11 datasets. We also computed the prediction accuracy of traditional 1-NN classifier to compare its performance to the performance of the techniques under study. Experimental results are shown in Table 7. Next to each accuracy value, Table 7 also gives the standard deviation calculated based on the three reruns of each experiment. We also calculated statistical significance using paired T-test for a significance level of 90% comparing the two editing techniques against the traditional 1-NN technique. Calculation of the statistical significance was based on the 15 different accuracy values reported in each experiment (i.e., 3 reruns each with 5 cross-validation folds). Prediction accuracy values that show statistically significant improvement over the corresponding 1-NN values are marked with a plus (**+**) next to them while values that show statistically significant degradation compared to 1-NN are marked with a minus (**-**) next to them.

Looking at the performance of the editing techniques in Table 7 we see that in 6 out of 11 datasets (Iris, Waveform, Diabetes, Heart-H, Vote, and Heart-StatLog) 1-NN classifier was outperformed by both editing techniques. Improvement was also found to be statistically significant for 5 of these 6 cases for both Wilson and SCE/NR (Supervised Clustering

Editing/Nearest Representative) classifiers. These results clearly show that dataset editing does have the potential of improving classification accuracy.

Table 7: Class prediction accuracies of UCI datasets.
(Statistically significant improvement (+) or degradation (-) compared to 1-NN)

Dataset Name	1-NN	Wilson Editing	SCE/NR Classifier
Iris	93.8±0.40	95.3±0.65 +	94.9±0.75
Glass	71.2±0.29	67.4±0.71 -	71.5±1.70
Diabetes	70.0±0.35	74.0±1.05 +	75.1±0.78 +
Waveform	76.9±0.74	79.1±0.38 +	83.3±0.23 + (●)
Heart-H	0.77±0.04	0.82±0.03 +	0.82±0.05 +
IonoSphere	90.9±0.30	88.6±0.98 -	89.4±1.70
Segmentation	97.5±0.21	96.9±0.26 -	93.3±0.10 - (o)
Vehicle	71.1±0.40	70.6±0.31	67.2±0.98 -
Vote	92.6±0.76	93.0±0.80	93.5±1.29 + (●)
Vowel	98.1±1.0	96.8±0.67 -	78.1±2.31 - (o)
Heart StatLog	77.9±3.14	81.0±1.40 +	81.7±2.26 +

Comparing the performance of the editing techniques among themselves, Table 7 shows that the NR classifier (based on SCE) outperformed Wilson editing in 6 of the 11 datasets. In 2 out of these 6 datasets, the better performance of the NR classifier was statistically significant (marked with (●) next to them). Nevertheless, the Wilson editing based classifier showed a strong performance as well. Out of the 5 datasets for which WE classifier outperformed SCE classifier, 2 of these cases were statistically significant (marked with (o) next to them)

6.3.5 Training Set Compression Rate (TSCR)

A second factor that we compared the performance of the different editing techniques according to is the "training set compression rate (TSCR). TSCR measures the percentage of reduction in the training set size achieved by the editing technique. In general, if an editing technique reduces the size of a training set from *n* examples down to *r* examples, we calculate the TSCR as:

$$\text{Training Set Compression Rate} = 1 - \frac{r}{n} \qquad (3)$$

Table 8: TSCR for the UCI datasets

Dataset Name	Wilson Editing	SC Editing
IRIS	6.0	96.7
Glass	27.1	85.6
Diabetes	30.2	84.5
Waveform	23.4	92.6
Heart_H	41.8	77.2
Ionosphere	9.1	99.7
Vote	8.3	97.2
Vehicle	30.5	80.3
Segmentation	2.3	98.2
Vowel	0.7	85.5
Heart_StatLog	22.6	87.2

Table 8 reports the training set compression rates for the UCI datasets. We clearly notice from the table that the highest training set compression rates are reported by supervised clustering editing for all datasets. Coupling this fact with the good relative performance of supervised clustering editing with respect to classification accuracy reported in the previous subsection, the overall performance of SCE moves up the ladder. For example, supervised clustering editing reduced the Vote training set from 348 examples to only 10 representative examples, at the average. On the other hand, Wilson editing reduced the Vote dataset to 319 examples. When applying traditional 1-NN classification on a training set containing only the 10 cluster representatives selected by SCE, we achieved better classification accuracy than the classification accuracies obtained when applying traditional 1-NN classifier on the edited training set Wilson editing produced.

As mentioned earlier, Wilson editing reduces the size of a dataset by removing examples that have been misclassified by a *k*-NN classifier. Consequently, the training set compression rates are quite low for Wilson editing on "easy" classification tasks for which high prediction accuracies are normally achieved. For example, Wilson editing produces dataset reduction rates of only 0.7%, 2.3%, and 6.0% for the Vowel, Segmentation, and Iris datasets, respectively.

7 SUPERVISED CLUSTERING FOR REGIONAL LEARNING

In the previous section, we discussed the use of supervised clustering for enhancing local learning tools such as NN classifiers. This Section discusses the usage of supervised clustering for regional learning. The parameter β in Equation (1), see Section 2, plays key role in determining whether the patterns identified by supervised clustering are local (i.e., low value for β) or more regional (i.e., higher values for β). Figure 14 shows how cluster purity and the number of clusters for the best solution found, *k*, changes as the value of parameter β increases for the Vehicle and the Diabetes datasets (the results were obtained by running algorithm SRIDHCR). As can be seen in Figure 14, as β increases, the fitness function q(X) imposes more penalty for using the same number of clusters and the clustering algorithm tries to use a lower number of clusters, resulting in reduced cluster purity.

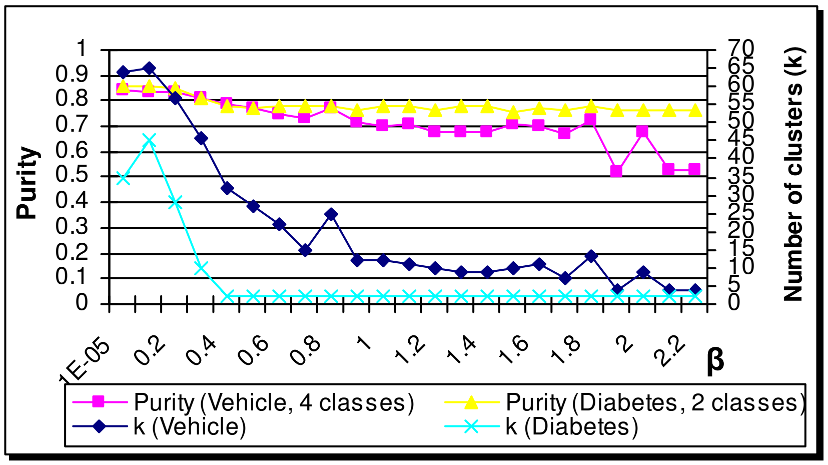

Figure 14: How Purity(*k*) and *k* change as the value of increases.

It is interesting to note that the Vehicle dataset seems to contain smaller regions with above average purities. Consequently, even if β increases beyond 0.5 the value of *k* remains quite high for that dataset. The Diabetes dataset, on the other hand, does not seem to contain such localized patterns; as soon as β increases beyond 0.5, *k* immediately reaches its minimum value of 2 (there are only two classes in the Diabetes dataset).

In general, we claim that supervised clustering is useful for enhancing our understanding of datasets [28]. Examples include:

1. It shows how instances of a particular class distribute in the attribute space; this information is of value for “discovering” subclasses of particular classes.

2. Maps for domain experts can be created that depict class densities in clusters and that identify which clusters share decision boundaries with which other clusters.[4]
3. Statistical summaries can be created for each cluster.
4. Meta attributes, such as various radiuses, distances between representatives, etc. can be generated, and their usefulness for enhancing classifiers can be explored.

7.2 Using Supervised Clustering to Enhance Simple Classifiers via Class Decomposition

Simple classifiers, such as linear classifiers, are known to exhibit low variance and high bias [9]. Since these models start off with simple representations, increasing their complexity is expected to improve their generalization performance while still retaining their ability to output models amenable to interpretation.

Figure 15.a shows two examples of classifiers. The simple linear classifier clearly produces high error rate while the high polynomial classifier improves classification accuracy on the expense of variance. Figure 15.b shows the result of applying supervised clustering to the dataset. Referring to Figure 15.b, we could transform the problem of classifying examples belonging to the two classes, black examples and white examples, into the "simpler" problem of classifying the examples that belong to clusters A, B, C, and D. The reduced complexity can be attributed to the fact that those 4 "clusters" are linearly separable (note the dotted lines in Figure 15.b) whereas the original 2 classes are not.

Vilalta et al. [25] proposed a scheme where the examples of each class are clustered separately using a traditional clustering algorithm. After clustering, a tuning step of cluster

[4] Triangulation techniques, such as Delaunay and Gabriel graphs, can be used for this purpose; a research subject that is currently being investigated by the UH-DMML group.

merging is executed to discover the combination of clusters that produces the least classification error. This “cluster merging” step is quite expensive.

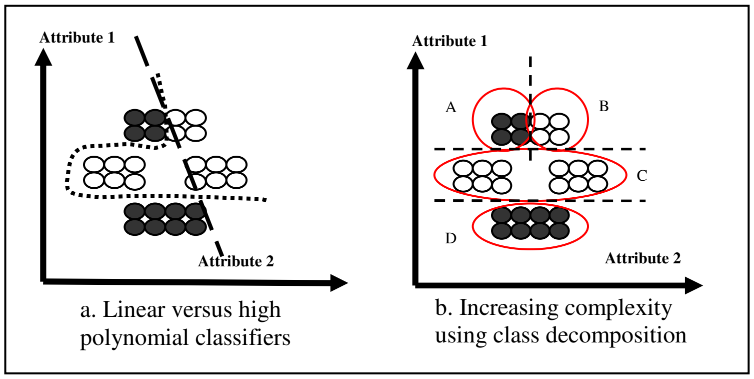
Attribute 1
Attribute 2
a. Linear versus high polynomial classifiers
Attribute 1
A
B
C
D
Attribute 2
b. Increasing complexity using class decomposition

Figure 15: Class decomposition to enhance simple classifiers

We propose to use supervised clustering for class decomposition applied on the whole dataset. Supervised clustering has the tendency to merge clusters with the same majority class if found close to each others (see C in Figure 15.b) and, consequently, achieve good clusterings while possibly eliminating the need for the expensive “merging” step used by Vilalta et al. [25]. To test this idea, we ran an experiment where we compared the average prediction accuracy of a traditional Naïve Bayes (NB) classifier with a NB classifier that treats each cluster as a separate class. We used SRIDHCR supervised clustering algorithm (with β set to 0.25) to obtain the clusters. We used 4 UCI datasets as a benchmark. The results of the experiment are reported in Table 9. These results indicate that using class decomposition improved the average prediction accuracy for 3 of the 4 datasets.

Table 9: Average prediction accuracy of Naive Bayes (NB) with and without using class decomposition

Dataset	**NB *without* Class Decomposition**	**NB *with* Class Decomposition**	**Improvement**
Diabetes	76.56	77.08	0.52%
Heart-H	79.73	70.27	–9.46%
Segment	68.00	75.045	7.05%
Vehicle	45.02	68.25	23.23%

Analyzing the results further, we see that the accuracy improvement for the Vehicle dataset (23.23%) is far higher than that for the Diabetes dataset (0.52%). This result is consistent with the analysis presented at the beginning of Section 7; namely that the Vehicle dataset contains groups of small size of examples that belong to the same class while the

Diabetes dataset does not. Consequently, the Vehicle dataset benefits more from class decomposition than the Diabetes dataset.

8. CONCLUSIONS AND FUTURE RESEARCH

Unlike traditional clustering, supervised clustering (SC) is applied on classified data and seeks to find clusters with high probability density with respect to a single class while keeping the number of clusters low. We have implemented and investigated three representative-based supervised clustering algorithms SPAM, SRIDHCR, and SCEC. Experimental results showed that the evolutionary computing algorithm SCEC produced the best results followed by SRIDHCR. Experimental results also showed that the fitness landscape has high number of local minima, thereby increasing the chance that greedy algorithms, such as SRIDHCR and SPAM, get stuck on the "wrong" hill too quickly and, consequently, produce poorer results compared to a non-greedy clustering algorithm, such as SCEC.

Our experimental results also demonstrated how supervised clustering is able to enhance NN classifiers through dataset editing. For example, for the Waveform dataset, a 1-NN classifier that uses just 296 representatives leads to an accuracy of 83.3%, whereas a 1-NN classifier that uses all 4000 examples in the training set achieves a lower accuracy of 76.9%.

The *UH Data Mining and Knowledge Discovery Group* [24] is currently investigating using supervised clustering for learning distance functions for classification tasks, see Eick et al. [10], and for discovering interesting regions in spatial data sets, see [11]. Other research efforts focus on devising SC algorithms that are capable of discovering clusters of arbitrary shapes and not only convex shapes [7]. Furthermore, we are currently exploring the use of grid-based supervised clustering algorithms that can cope with large datasets, ranging from 10000 to millions of examples [11].

Finally, this research shed some light on the challenges of designing efficient representative-based clustering algorithms. We believe that this knowledge will be beneficiary to designers of representative-based clustering algorithms. Furthermore, our study suggests that using evolutionary computing techniques for traditional representative-based clustering deserves more attention by future research.

REFERENCES

[1] Basu, S., Bilenko,M., Mooney, R., "*Comparing and Unifying Search-based and Similarity-Based Approaches to Semi-Supervised Clustering*", in Proc. of ICML03 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning & Data Mining, Washington, DC. August 21, 2003

[2] Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D., "*Learning Distance Functions Using Equivalence Relations*", in Proc. ICML03, Washington DC, August 2003.

[3] Basu, S., Bilenko,M., Mooney, R., "*A Probabilistic Framework for Semi-Supervised Clustering*", in Proc. 10th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD-2004), Seattle, WA, August 2004.

[4] Basu, S., Bilenko,M., Mooney, R. "*Active Semi-Supervision for Pairwise Constrained Clustering*", in Proc. 4th SIAM Intl. Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004.

[5] Bilenko, M., Basu, S., and Mooney, R., J., "*Integrating Constraints and Metric Learning in Semi-Supervised Clustering*", in Proc. ICML'04, Banff, Alberta, Canada, July 4-8, 2004.

[6] Cohn, D., Caruana, R., McCallum, A. "*Semi-supervised Clustering with User Feedback*", unpublished manuscript, available at www-2.cs.smu.edu/~mccallum/, 2000.

[7] Choo, Ji Yeon, "*Using Proximity Graphs to enhance representative-based clustering algorithms*", Master's thesis, University of Houston, to appear Summer 2006.

[8] Demiriz, A., Benett, K.-P., and Embrechts, M.J., "*Semi-supervised Clustering using Genetic Algorithms*", in Proc. ANNIE'99, November 7 - 10, 1999, St. Louis, Missouri.

[9] Duda, R., Hart, P., and Stork, D. G., "*Pattern Classification*", 2001, John Wiley & Sons, New York, NY, pp. 465-470

[10] Eick, C., Rouhana, A., Bagherjeiran, A., and Vilalta, R., "*Using Clustering to Learn Distance Functions for Supervised Similarity Assessment*", in Proc. Int. Conf. on Machine Learning and Data Mining (MLDM), Leipzig, Germany, July 2005.

[11] Eick, C., Vaezian B., Jiang, D., and Wang, J., "*Discovery of Interesting Regions in Spatial Data Sets Using Supervised Clustering*", submitted for publication, May 2006 .

[12] Eick, C.F., Zeidat, N., and Zhenghong, Z., "*Supervised Clustering – Algorithms and Benefits*", In proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI04) , Boca Raton, Florida, November 2004, pp. 774-776

[13] Eick, C.F., Zeidat, N., and Vilalta, R.: "*Using Representative-Based Clustering for Nearest Neighbor Dataset Editing*". ICDM 2004: 375-378

[14] Fix, E. and Hodges, J., "*Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties*", Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[15] Karypis, G., Han, E.H., and Kumar, V.: "*Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling*", IEEE Computer, Vol 32, No 8, pp 68-75, August 1999

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

[17] Klein, D., Kamvar, S.-D., Manning, C. "*From instance-level Constraints to Space-level Constraints: Making the Most of Prior Knowledge in Data Clustering*", in Proc. ICML'02, Sydney, Australia.

[18] Salvador, S., and Chan, P., "*Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms*", in Proc. ICTAI, Boca Raton, FL, November 2004

[19] Sinkkonen, J., Kaski, S., and Nikkila, J., "*Discriminative Clustering: Optimal Contingency Tables by Learning Metrics", in Proc. ECML'02.*

[20] Slonim, N. and Tishby, N., "*Agglomerative Information Bottleneck"*, Neural Information Processing Systems (NIPS-1999).

[21] Toussaint, G., "*Proximity Graphs for Nearest Neighbor Decision Rules: Recent Progress*", in Proc. 34th Symposium on the INTERFACE, Montreal, Canada, April 17-20, 2002.

[22] Tishby, N., Periera, F.C., and Bialek, W., "*The Information Bottleneck Method*", In proceedings of the 37th Allerton Conference on Communication and Computation, 1999.

[23] University of California at Irving, Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html

[24] University of Houston, Machine Learning and Data Mining Group, http://www. cs.uh.edu/~sujingwa/PKDD05/

[25] Vilalta, R., Achari, M., and Eick, C., "*Class Decomposition Via Clustering: A New Framework For Low-Variance Classifiers",* in Proc. Third IEEE International Conference on Data Mining (ICDM03), Melbourne, Florida.

[26] Wilson, D.L., "*Asymptotic Properties of Nearest Neighbor Rules Using Edited Data*", IEEE Transactions on Systems, Man, and Cybernetics, 2:408-420, 1972.

[27] Xing, E.P., Ng A., Jordan, M., Russell, S. "*Distance Metric Learning with Applications to Clustering with Side Information*", Advances in Neural Information Processing 15, MIT Press, 2003.

[28] Zeidat, N., "*Supervised Clustering: Algorithms and Applications"*, PhD dissertation, Dept. of Computer Science, University of Houston, August 2005, www.cs.uh.edu/~nzeidat/publications/phd_dissert.html

[29] Zhao, Z., "*Evolutionary Computing and Splitting Algorithms for Supervised Clustering*", Master's Thesis, Department of Computer Science, University of Houston, May 2004, http://www.cs.uh.edu/~zhenzhao/ZhenghongThesis.zip.