Supervised Clustering – Algorithms and Benefits

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

Abstract. This paper centers on a novel data mining technique we term *supervised clustering*. Unlike traditional clustering, supervised clustering assumes that the examples are classified. The goal of supervised clustering is to identify class-uniform clusters that have high probability densities. Four representative–based algorithms for supervised clustering are introduced: a greedy algorithm with random restart, named SRIDHCR, that seeks for solutions by inserting and removing single objects from the current set of cluster representatives, SPAM (a variation of the popular clustering algorithm PAM), an evolutionary computing algorithm named SCEC, and a fast medoid-based top-down splitting algorithm, named TDS. The four algorithms were evaluated using a benchmark consisting of four UCI machine learning data sets. Surprisingly, SPAM performed quite poorly in the experiments. TDS, on the other hand, considering its high speed, performed quite well on two datasets and quite poorly on the other two datasets. The experimental results suggest that SRIDHCR is a good choice if resources are limited or data sets are very large, whereas, SCEC, although runtime intensive, finds the best clusters in almost all experiments conducted. In general, it seems that "greedy" algorithms, such as SPAM, SRIDHCR, and TDS, do not perform particularly well for supervised clustering and seem to terminate prematurely too often. We also present experimental results that illustrate the benefits of supervised clustering for creating summaries of datasets and for enhancing existing classification algorithms.

Keywords: supervised clustering, clustering classified examples, clustering for classification, representative-based clustering algorithms.

1. Introduction

This paper centers on a novel data mining technique we term *supervised clustering*. Clustering is typically applied in an unsupervised learning framework using particular error functions, e.g. an error function that minimizes the distances inside a cluster keeping clusters tight. *Supervised clustering*, on the other hand, deviates from traditional clustering in that it is applied on classified examples with the objective of identifying clusters that have high probability density with respect to a single class. Moreover, in supervised clustering, we also like to keep the number of clusters small, and objects are assigned to clusters using a notion of closeness with respect to a given distance function.

Fig. 1 illustrates the differences between traditional and supervised clustering. Let us assume that the black examples and the white examples in the figure represent subspecies of Iris plants named Setosa and Virginica, respectively. A traditional clustering algorithm would, very likely, identify the four clusters depicted in Figure 1.a. The reason is that traditional clustering is ignorant with respect to class membership of examples. If our objective is to generate summaries for the Virginica and Setosa classes of the Iris Plants, the clustering in Figure 1.a would not be very attractive since it combined Setosa and Virginica objects in cluster **A** and put examples of the Virginica class in two different clusters **B** and **C**.

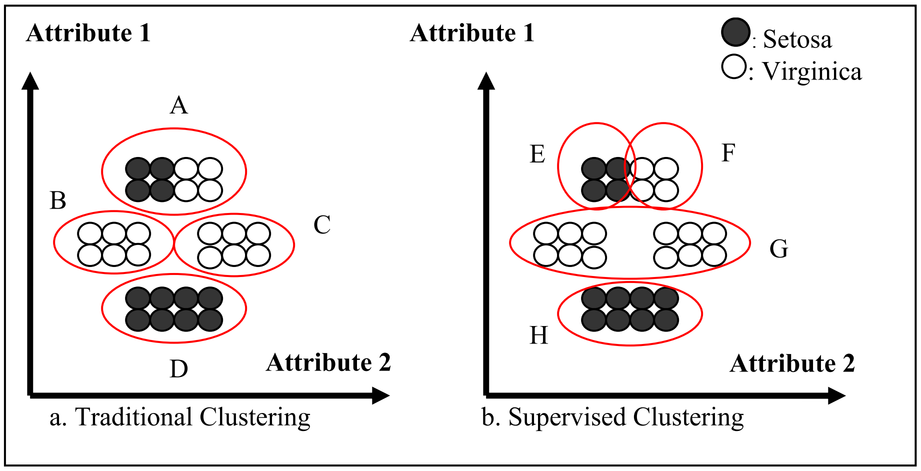

Figure 1: Differences between Traditional Clustering and Supervised Clustering.

A supervised clustering algorithm that maximizes class purity, on the other hand, would split cluster **A** into two clusters **E** and **F**. Another characteristic of supervised clustering is that it tries to keep the number of clusters low. Consequently, clusters **B** and **C** would be merged into one cluster without compromising class purity while reducing the number of clusters. A supervised clustering algorithm would identify cluster **G** as the union of clusters **B** and **C** as illustrated by Figure 1.b.

The remainder of this paper will center on the discussion of algorithms for supervised clustering and on the empirical evaluation of the performance of these algorithms as well as the benefits of supervised clustering. Section 2 discusses related work. Section 3 characterizes the class of the algorithms investigated and Section 4 presents the algorithms themselves. Section 5 provides an experimental evaluation of the algorithms. Section 6 summarizes the results of experiments that evaluate the benefits of supervised clustering. Section 7 summarizes the results of the paper.

2. Related Work

There has been some work that has some similarity with our research under the heading of semi-supervised clustering. The idea of semi-supervised clustering is to enhance a clustering algorithm by using side information in the clustering process that usually consists of a "small set" of classified examples; the objective of the clustering process, then, is to optimize class purity (examples with different class labels should belong to different clusters) in addition to the traditional objectives of a clustering algorithm. 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 [BBM03]). 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.

[XNJ03] (and similarly [BHSW03]) 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 data set 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 [KKM02] proposes a shortest path algorithm to modify a Euclidian distance function based on prior knowledge.

Demiriz [DBE99] proposes 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 [CCM00] modifies the popular EM algorithm so that it is capable of incorporating similarity and dissimilarity constraints. Finally, Basu et. al. [BBM02] centers on modifying the *k*-means clustering algorithm to cope with prior knowledge.

However, there are two approaches that can be viewed as supervised clustering approaches. Sinkkonen et al., [SKN02], propose a very general approach called *discriminative clustering* that minimizes distortion within clusters. Distortion, in their context, represents the loss of mutual information between the auxiliary data (e.g., classes) and the clusters caused by representing each cluster by a prototype. The technique seeks to produce clusters that are internally as homogeneous as possible in conditional distributions $p(c|x)$ of the auxiliary variable, i.e., belong to a single class. Similarly, Tishby et. al. introduced the *information bottleneck method* [TPB99]. Based on that method, they proposed an agglomerative clustering algorithm, [ST99], that minimizes information loss with respect to P(C|A) with C being a class and A being an attribute.

3. Representative-based Algorithms for Supervised Clustering

As mentioned earlier, the fitness functions used 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).* 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.

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

Notation	Description
$O=\{o_1, \ldots, o_n\}$	Objects in a data set
n	Number of objects in the data set
$d(o_i,o_j)$	Distance between objects o_i & o_j
c	The number of classes in the dataset
C_i	Cluster associated with the i-th representative
$X=\{C_1, \ldots, C_k\}$	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, see formula (1)

Table 1: Notations Used in the Paper.

3.1 A Fitness Function for Supervised Clustering

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

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

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

with n being the total number of examples and c being the number of classes in a data set. The parameter β ($0< \beta \le 2.0$) determines the penalty that is associated with the numbers of clusters, k, in a clustering: higher values for β imply larger penalties for a higher number of clusters.

Two special cases of the above fitness function should be mentioned; the first case is a clustering X1 that uses only c clusters; the second case is a clustering X2 uses n clusters and assigns a single object to each cluster, therefore, each cluster is pure. We obtain: q(X1)=Impurity(X1) and q(X2)≈β.

3.2 Representative-Based Supervised Clustering Algorithms

As there are many possible algorithms for clustering, there are, also, a lot of algorithms for supervised clustering. Our work centers on the development of representative-based clustering algorithms. Representative-based clustering aims at finding a set of k representatives that best characterize a dataset. Clusters are created by assigning each object to the closest representative. Representative-based supervised clustering algorithms seek to accomplish the following goal: *Find a subset O_R of O such that the clustering X obtained by using the objects in O_R as representatives minimizes $q(X)$.*

One might ask why our work centers on developing *representative-based* supervised clustering algorithms. The reason is that representatives (such as medoids) are quite useful for data summarization. Moreover, clustering algorithms that restrict representatives to objects belonging to the dataset, such as the *k-medoid* algorithm [KR90], explore a smaller solution space if compared with centroid–based clustering algorithms, such as the *k-means* algorithm[1]. Furthermore, representative based algorithms are, also, useful for dataset compression (more on this in section 6.2). Finally, when using representative-based clustering algorithms, only an inter-object distance matrix is needed and no "new" distances have to be computed, as it is the case with *k-means*.

4. Clustering Algorithms Investigated

As part of our research, we have designed and evaluated several representative-based supervised clustering algorithms as well as the traditional clustering algorithm PAM. This section describes these algorithms.

4.1 Partitioning Around Medoids (PAM)

This traditional clustering algorithm, sometimes also call *k-medoid* [KR90], seeks to find *k representative objects* among the objects in the data set minimizing the fitness function given in formula (2):

$$\text{Tightness(X)}=\frac{1}{|\,\text{objects}\,|}\sum_{\text{objects}}\text{distance (object, medoid(object))} \qquad (2)$$

where *medoid(object)* is the medoid (representative) of the cluster that *object* belongs to. The number of clusters, *k,* is an input parameter for the algorithm.

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

PAM is subdivided into two algorithms. The first algorithm, called the *BUILD* algorithm, starts with a set of representatives that initially contains the medoid of the complete data set, and greedily inserts new representatives into this set while minimizing the above fitness function. The second algorithm of PAM, called *SWAP*, tries to improve the clustering obtained by BUILD by exploring all possible replacements of medoids by non-medoids picking the replacement that enhances the fitness function the most. If no such fitness improving replacement can be found, the algorithm terminates.

4.2 Supervised Partitioning Around Medoids (SPAM)

This algorithm is a variation of the algorithm PAM that uses the fitness function q(X) instead of Tightness(X). The number of clusters *k* is an input parameter to the algorithm. It consists of two sub-algorithms. Sub-algorithm *SBUILD* starts by selecting the medoid of the members of the most frequent class in the data set as the first representative. After that, it repeatedly and *greedily* adds to the current set of representatives a non-representative object that, if added to the set of representatives, would generate a clustering that produces the minimum value for the fitness function q(X). 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. SPAM terminates if no replacement can be found that leads to a clustering with a better (lower) fitness value with respect to q(X).

4.3 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: Pseudo Code for Algorithm SRIDHCR.

Set of medoids after adding one non-medoid	q(X)	Set of medoids after removing a medoid	q(X)
5 30 48 72 150	0.098	30 48 72 150	0.095
5 30 48 72 150 1	0.101	5 48 72 150	0.095
5 30 48 72 150 2	0.101	5 30 72 150	0.095
…….....	…….	5 30 48 150	0.341
5 30 48 72 150 110	***0.054***	5 30 48 72	0.341
………	…….		
5 30 48 72 150 148	0.114		
5 30 48 72 150 149	0.100		
Trials in first part (add a non-medoid)		Trials in second part (drop a medoid)	

Table 2: Solutions Explored in the First Iteration.

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

Table 3: Set of Representatives Explored.

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 obtained 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. Table 2 depicts the solutions that are evaluated in the first iteration. The 150 (e.g., 145+5) clusterings (that were generated from the solutions that are partially listed in Table 2) are then evaluated, and the solution whose clustering has the lowest value with respect to q(X) is selected, highlighted in ***italic bold*** font in Table 2. 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.

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 3 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.

4.4 Top Down Splitting Algorithm (TDS)

This is a very simple algorithm that aims at creating clusters quickly. It starts by assigning all data objects to a root cluster. After that the algorithm recursively splits clusters by replacing the medoid of the cluster with two medoids: the medoid of the most frequent class in the cluster and the medoid of the second most frequent class in the cluster. Clusters are only split if q(X) does not increase as a result of the split. This splitting procedure is recursively applied to newly generated clusters.

4.5 Supervised Clustering using Evolutionary Computing (SCEC).

This algorithm uses evolutionary computing techniques to seek for the "optimal" set of representatives by evolving population of solutions over a fixed number *N* of generations. The size of the population is fixed to a predetermined number *PS* 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*. The key features of this approach include:

1) Chromosomal Representation: A solution consists of a set of representatives that are a subset of the objects 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:

    1. Include all representatives that occur in both parent in the offspring
    2. Include representatives that occur in a single parent with probability 50% in the offspring

    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-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:
    1. Create clusters by assigning the remaining objects to the closest representative in the solution to be evaluated.
    2. Evaluate the so obtained clustering X using q.

5. Experimental Comparison

In order to study the performance of the clustering algorithms presented in section 4, we applied those algorithms to a benchmark consisting of 4 data sets that were obtained from UCI Machine Learning Repository [UCIMLR]. Table 4 gives a summary for the four data sets we used. All data sets were normalized using a linear interpolation function that assigns 1 to the maximum value and 0 to the minimum value. Manhattan distance was used to compute the distance between two objects.

Data set name	**# of objects**	**# of attributes**	**# of classes**
Iris Plants	150	4	3
Image Segmentation	2100	19	7
Vehicle silhouettes	846	18	4
Pima-Indians Diabetes	768	8	2

Table 4: Additional Data Sets Used in the Experimental Evaluation.

In the experiments, 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 which PAM and SPAM were run. In particular, PAM and SPAM were run for the same value of β. The number of clusters *k* was set to the number of representatives of the best solution that was found by SRIDHCR.

SCEC creates the initial population randomly which is decomposed of solutions having between *c*+1 and 2*c* representatives. The algorithm SCEC has several other input parameters. Table 5 shows the parameters and their corresponding values used in our experiments.

Parameter	Value
Population size (PS)	400
Tournament selection (K)	2
Crossover rate (CRR)	0→0.95
Mutation rate (MR)	0.95→0
Copy rate (CR)	0.05
Number of generations (N)	1500

Table 5: Input Parameters for the Algorithm SCEC.

In summary, 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 0 at the beginning of the run 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 values of those and other parameters were determined experimentally. During this parameter selection process we observed (for more details see [Z04]):

- 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 lead to better solution quality.
- Using lower selective pressure lead to better solution quality.

In summary, our results suggest that parameter settings that allow for a more randomized exploration of the search space seem to be more suitable for supervised clustering.

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

- Cluster Purity.
- Value of the fitness function q(X) --- see formula (1).
- Average dissimilarity between all objects and their representatives. (abbreviated as Avg. Diss.) --- see formula (2).
- Wall-Clock Time (WCT). 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.

5.1 Traditional Versus Supervised Clustering

Table 6 and Table 7 display the different performance measures for the five algorithms when applied on the four data sets for β=0.1 and β=0.4, respectively. As expected, looking at Table 6 we observe that traditional clustering algorithm PAM produces tighter clusters, characterized by smaller intra-cluster distances (Avg. Diss in Table 6 & 7). On the other hand, the supervised clustering algorithms SCEC and SRIDHCR, produce better cluster purity. The improvement in cluster purity ranges from 8% to 25% for the four datasets. Algorithm SCEC produces class purity that is 10% to 25% better than class purity produced by the traditional clustering algorithm PAM, obtained when clustering data sets *Iris-Plants* and *Vehicle*, respectively. Algorithm SRIDHCR produces class purity that is 8% to 19% better than that produced by PAM, obtained by clustering *Iris-Plants* and *Vehicle* data sets, respectively.

5.2 Performance of the Supervised Clustering Algorithms

Algorithm	k	Cluster Purity	q(X)	Avg. Diss.	WCT (Sec.)
Iris-Plants Dataset					
PAM	3	0.907	0.093	0.081	0.06
TDS	3	0.927	0.073	0.082	0.002
SCEC	5	0.993	0.018	0.107	1019
SRIDHCR	3	0.980	0.020	0.113	13
SPAM	3	0.973	0.027	0.133	0.32
Vehicle Dataset					
PAM	65	0.701	0.326	0.044	372.00
TDS	126	0.852	0.186	0.081	0.03
SCEC	132	0.923	0.116	0.049	31989
SRIDHCR	65	0.835	0.192	0.062	1666
SPAM	65	0.764	0.263	0.097	1090
Segmentation Dataset					
PAM	53	0.880	0.135	0.027	4073
TDS	55	0.952	0.062	0.043	0.02
SCEC	60	0.989	0.026	0.046	22598
SRIDHCR	53	0.980	0.035	0.050	9868
SPAM	53	0.944	0.071	0.061	1422
Pima-Indian Diabetes Dataset					
PAM	45	0.763	0.237	0.056	186.00
TDS	3	0.732	0.272	0.103	0.003
SCEC	64	0.893	0.135	0.074	12943
SRIDHCR	45	0.859	0.164	0.076	660
SPAM	45	0.822	0.202	0.086	58

Table 6: Comparative Performance of the Different Algorithms (β=0.1)

Algorithm	k	Cluster Purity	q(X)	Avg. Diss.	WCT (Sec.)
IRIS-Plants Dataset					
PAM	3	0.907	0.093	0.081	0.06
TDS	3	0.927	0.073	0.082	0.002
SCEC	3	0.987	0.013	0.125	618
SRIDHCR	3	0.987	0.013	0.125	9
SPAM	3	0.973	0.027	0.133	0.3
Vehicle Dataset					
PAM	56	0.681	0.418	0.046	505
TDS	111	0.831	0.311	0.071	0.02
SCEC	61	0.857	0.247	0.059	10947
SRIDHCR	56	0.835	0.265	0.067	1130
SPAM	56	0.754	0.345	0.100	681
Segmentation Dataset					
PAM	32	0.875	0.169	0.032	1529
TDS	38	0.950	0.096	0.045	0.01
SCEC	28	0.969	0.069	0.050	11533
SRIDHCR	32	0.970	0.074	0.051	8450
SPAM	32	0.940	0.103	0.065	1053
Pima-Indians-Diabetes Dataset					
PAM	2	0.656	0.344	0.101	1.0
TDS	3	0.732	0.283	0.103	0.003
SCEC	9	0.819	0.219	0.103	1700
SRIDHCR	2	0.776	0.224	0.139	228
SPAM	2	0.772	0.227	0.125	2.7

Table 7: Comparative Performance of the Different Algorithms (β=0.4)

The results is Tables 6 & 7 clearly show that all supervised clustering algorithms produce better cluster purity than PAM. Among the four supervised clustering algorithms, SCEC consistently finds the best solutions for all four data sets, whereas SRIDHCR finds "good" solutions, and somewhat surprisingly SPAM performs quite poorly in the experiments. However, the wall-clock time for the algorithm SCEC is the highest among all four supervised clustering algorithms, followed by algorithm SRIDHCR. Bearing in mind that the wall-clock time reported for SRIDHCR is the time needed to execute 50 runs, makes a single run of SRIDHCR be 1.4 to 25 times faster than a single run of algorithm SCEC. This makes SRIDHCR a good choice if resources are somewhat limited. Finally, algorithm TDS, considering its high speed, performed quite well for two datasets but quite poorly for the other two datasets.

As β increases from 0.1 to 0.4 (see Table 7), the penalty for increasing the number of clusters increases as well, see formula (1). Consequently, the algorithm will produce solutions with a smaller number of clusters. Due to the fact that more smaller-sized solutions (solutions with less 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 7 are slightly lower than the corresponding numbers in Table 6 which is not surprising because using a larger number of clusters is punished more severely by the fitness function with β being set to 0.4.

Algorithm	**q(X)**		**WCT (sec.)**
Iris-Plants			
PAM		0.0933	0.06
TDS		0.0733	0.02
SCEC		0.0182	93.42
SRIDHCR	Min	0.018	0.11
	Max	0.045	0.38
	Mean	0.029	0.24
	STDV	0.005	0.064
SPAM		0.0267	0.321
Vehicle			
PAM		0.3260	372.00
TDS		0.3950	0.16
SCEC		0.1570	7368.70
SRIDHCR	Min	0.181	17.00
	Max	0.291	56.00
	Mean	0.228	33.00
	STDV	0.027	9.28
SPAM		0.263	1090.00
Segmentation			
PAM		0.0350	4073.00
TDS		0.0780	1.30
SCEC		0.0300	11087.20
SRIDHCR	Min	0.032	122.00
	Max	0.084	264.00
	Mean	0.050	197.34
	STDV	0.012	33.91
SPAM		0.0710	1422.00
Pima-Indian Diabetes			
PAM		0.2370	186.00
TDS		0.2710	0.19
SCEC		0.1660	2015.20
SRIDHCR	Min	0.164	3.00
	Max	0.223	24.00
	Mean	0.191	12.12
	STDV	0.013	4.64
SPAM		0.2020	58.00

Table 8: Average Performance of SRIDHCR Compared to other Algorithms (β=0.1).

Table 8: … (continued)

Table 8 displays the different performance measures of the five algorithms when applied on all data sets for β=0.1 except that measures reported for algorithm SRIDHCR include mean, minimum, maximum, and standard deviation for the 50 (re-)runs of the algorithm. Comparing the results of the algorithm SPAM with those for algorithm SRIDHCR in Table 8, we can clearly see that SRIDHCR not only outperforms SPAM in its "best out of 50 runs" but in its *average* performance as well (with the exception of the Iris Plant data set).

SRIDHCR, if compared with SPAM, has the interesting characteristic that when attempting to enhance a solution with *k* representatives, it looks for better solutions with *k*–1 and *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; for example, SRIDHCR might add v1 and v2 to {u1,u2,u3,u4} obtaining {u1,u2,u3,u4,v1,v2} and would next remove u1, and u2, obtaining a better solution {u3,u4,v1,v2} whereas SPAM might terminate with the suboptimal solution {u1,u2,u3,u4}, if neither the replacement of u1 by v1 nor the replacement of u2 by v2 enhances q(X). Furthermore, encouraged by its smaller average runtime, SRIDHCR can be restarted up to 33 times (as it is the case for the Vehicle data set for example in Table 8) 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 search space, which we believe is a second contributing factor for the significantly better performance of algorithm SRIDHCR.

In summary, SCEC found much better solutions than the other 3 algorithms. 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 and plateau-like structures (clusterings X1 and X2 with q(X1)=q(X2)). It seems that greedy algorithms, such as PAM, SPAM, TDS do not perform particularly well in this fitness landscape terminating prematurely too often. We conducted a small experiment in which we estimated the probability that two solutions with 10/50 representatives have the same fitness value with respect to q(X) and Tightness(X), respectively. The results of this experiment, summarized in Table 9, show that ties are much more likely when q(X) is used.

Dataset	k	β	Ties % using q(X)	Ties % using Tightness(X)
Iris-Plants	10	0.00001	5.8	0.0004
Iris-Plants	10	0.4	5.7	0.0004
Iris-Plants	50	0.00001	20.5	0.0019
Iris-Plants	50	0.4	20.9	0.0018
Vehicle	10	0.00001	1.04	0.000001
Vehicle	10	0.4	1.06	0.000001
Vehicle	50	0.00001	1.78	0.000001
Vehicle	50	0.4	1.84	0.000001
Segmentation	10	0.00001	0.220	0.000000
Segmentation	10	0.4	0.225	0.000001
Segmentation	50	0.00001	0.626	0.000001
Segmentation	50	0.4	0.638	0.000000
Diabetes	10	0.00001	2.06	0.0
Diabetes	10	0.4	2.05	0.0
Diabetes	50	0.00001	3.43	0.0002
Diabetes	50	0.4	3.45	0.0002

Table 9: Tie Distribution for the Four Datasets.

6. Benefits of Supervised Clustering

This section briefly discusses the benefits of supervised clustering. In general, supervised clustering can be used for many different tasks that include:

1. Create background knowledge for a dataset
2. Dataset compression and editing
3. To learn subclasses and to use these subclasses to enhance classification algorithms [VAE04]
4. To evaluate distance functions in distance function learning [ERCBV04]

Due to space limitations we center on discussing the first two applications in this section.

6.1 Using Supervised Clustering to Create Background Knowledge

Figure 3 shows how cluster purity and the number of clusters for the best solution found, *k*, change as the value of parameter β increases for the Vehicle and the Diabetes data sets (the results were obtained by running algorithm SRIDHCR)**.** As can be seen in Figure 3, as β increases, more penalty is associated with using the same number of clusters and the algorithm tries to use a lower number of clusters resulting in decreasing cluster purity. It is interesting to note that the Vehicle data set 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 data set. The Diabetes data set, 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 data set).

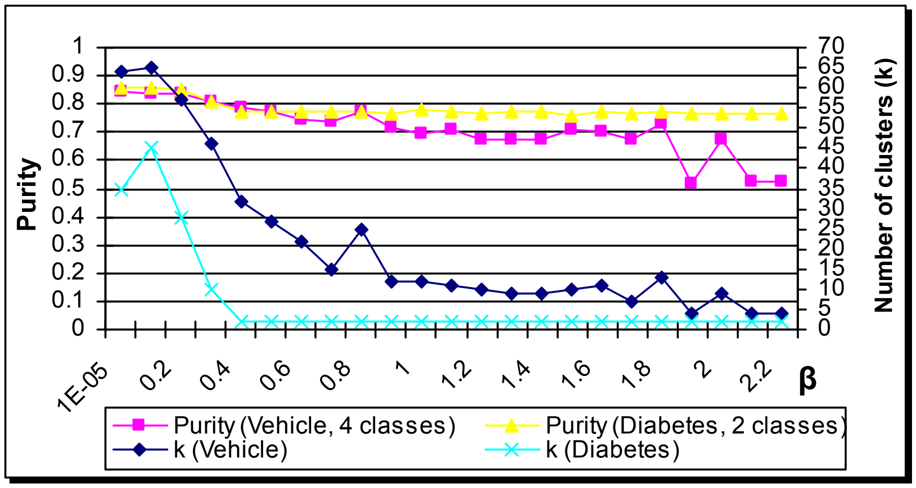

Figure 3. How Purity and *k* Change as the Value of β Increases.

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

- It shows how instances of a particular class distribute in the attribute space; this information is of value for "discovering" subclasses of particular classes.
- 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.
- Statistical summaries can be created for each cluster.
- Meta attributes, such as various radiuses, distances between representatives, etc. can be generated, and their usefulness for enhancing classifiers can be explored.

6.2. Using Cluster Prototypes for Dataset Editing

The objective of *dataset editing* [W72] is to remove examples from a training set in order to enhance the accuracy of a classifier. In this paper, we propose using supervised clustering as a tool for *editing* a dataset *O* to produce a reduced subset O_r. The reduced subset O_r consists of cluster representatives that have been selected by a supervised clustering algorithm. A 1-NN classifier, that we call nearest-representative (NR) classifier, is then used for classifying new examples using subset O_r

instead of the original dataset O. We call this approach *supervised clustering editing* (SCE for short). Figure 4 presents the classification algorithm of the NR classifier. A NR classifier can be viewed as a compressed 1-Nearest-Neighbor classifier because it uses only k ($k << n$) examples out of the n examples in the dataset O.

PREPROCESSING

A. Apply a representative-based supervised clustering algorithm (e.g. SRIDHCR) on dataset O to produce a set of k prototypical examples.

B. *Edit* dataset O by deleting all non-representative examples to produce subset O_r.

CLASSIFICATION RULE

Classify a new example q by using a 1-NN classifier with the edited subset O_r.

Figure 4: Nearest Representative (NR) Classifier.

Figure 5 gives an example that illustrates how supervised clustering is used for dataset editing. Figure 5.a shows a dataset that has not been clustered yet. Figure 5.b shows a dataset that was partitioned into 6 clusters using a supervised clustering algorithm. Cluster representatives are marked with circles around them. Figure 5.c shows the result of supervised clustering editing.

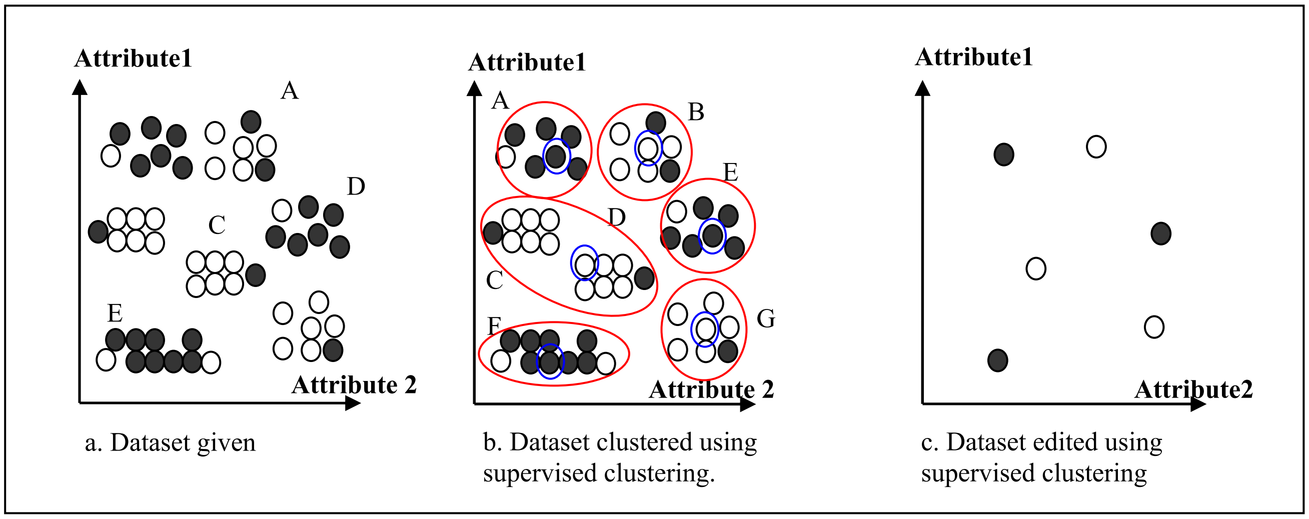
Attribute1
A
D
C
E
Attribute 2
a. Dataset given
Attribute1
A
B
E
D
C
F
G
Attribute 2
b. Dataset clustered using supervised clustering.
Attribute1
Attribute2
c. Dataset edited using supervised clustering

Figure 5: Editing a Dataset Using Supervised Clustering.

To evaluate the benefits of supervised clustering editing, we applied our editing technique to a benchmark involving 8 datasets; the properties of the four additional datasets used are summarized in Table 10.

Dataset name	# of objects	# of attributes	# of classes
Glass	214	9	6
Heart-Statlog	270	13	2
Heart-Disease-Hungarian (Heart-H)	294	13	2
Waveform	5000	21	3

Table 10: Additional Datasets Used in the Dataset Editing Experiment.

The parameter β has a strong influence on the number *k* of representatives chosen by the supervised clustering algorithm; i.e., the size of the edited subset O_r. If high β values are used, clusterings with a small number of representatives are likely to be chosen. On the other hand, low values for β are likely to produce clusterings with a large number of representatives. In general, an editing technique reduces the size of a dataset, *n*, to a smaller size *k*. We define the dataset compression rate of an editing technique as:

$$\text{Compression Rate } = 1 - \frac{k}{n} \tag{3}$$

In order to explore different compression rates for supervised clustering editing, three different values for parameter β were used in the experiments: 1.0, 0.4, and 0.1. Prediction accuracies were measured using 10-fold cross-validation throughout the experiments for the eight datasets tested. Representatives for the nearest representative (NR) classifier were computed using a version of the SRIDHCR supervised clustering algorithm that was introduced in Section 4.3. In our experiments, SRIDHCR was restarted 50 times ($r = 50$), and the best solution (i.e., set of representatives) found in the 50 runs was used as the edited subset for the NR classifier. We also computed prediction accuracy for a traditional 1-NN classifier that uses all training examples when classifying a new example. Table 11 reports the accuracies obtained using the edited subset and the original dataset as well as the average dataset compression rates for supervised clustering editing. Due to the fact that the supervised clustering algorithm has to be run 10 times, once for each fold, different numbers of representatives are usually obtained for each fold. Consequently, Table 11 also reports the average, minimum, and maximum number of representatives found on the 10 runs. For example, when running the NR classifier for the Diabetes dataset with β set to 0.1 the (rounded) average number of representatives was 27, the maximum number of representatives during the 10 runs was 33 and the minimum number of representatives was 22; supervised clustering editing reduced the size of the original dataset *O* by an average of 96.5%, as displayed in Table 11. The NR classifier classified 73.6% of the testing examples correctly, as indicated in Table 11.

β	Avg. *k* [Min-Max] for SCE	SCE Compression Rate (%)	NR Prediction Accuracy	1-NN Prediction Accuracy
Glass* (214*)				
0.1	34 [28-39]	84.3	0.636	0.692
0.4	25 [19-29]	88.4	0.589	0.692
1.0	6 [6 – 6]	97.2	0.575	0.692
***Heart-Stat Log* (270)**				
0.1	15 [12-18]	94.4	0.796	0.767
0.4	2 [2 – 2]	99.3	0.833	0.767
1.0	2 [2 – 2]	99.3	0.838	0.767
***Diabetes* (768)**				
0.1	27 [22-33]	96.5	0.736	0.690
0.4	9 [2-18]	98.8	0.736	0.690
1.0	2 [2 – 2]	99.7	0.745	0.690
***Vehicle* (846)**				
0.1	57 [51-65]	97.3	0.667	0.700
0.4	38 [26-61]	95.5	0.667	0.700
1.0	14 [9-22]	98.3	0.665	0.700
***Heart-H* (294)**				
0.1	14 [11-18]	95.2	0.755	0.783
0.4	2	99.3	0.793	0.783
1.0	2	99.3	0.809	0.783
***Waveform* (5000)**				
0.1	104 [79-117]	97.9	0.834	0.768
0.4	28 [20-39]	99.4	0.841	0.768
1.0	4 [3-6]	99.9	0.837	0.768
***Iris-Plants* (150)**				
0.1	4 [3-8]	97.3	0.947	0.947
0.4	3 [3 – 3]	98.0	0.973	0.947
1.0	3 [3 – 3]	98.0	0.953	0.947
Segmentation (2100)				
0.1	57 [48-65]	97.3	0.938	0.956
0.4	30 [24-37]	98.6	0.919	0.956
1.0	14	99.3	0.889	0.956

Table 11: Dataset Compression Rates for SCE and Prediction Accuracy for the NR and 1-NN Classifiers.

If we inspect the results displayed in Table 11, we can see that the SCE approach accomplished significant improvement in accuracy for the Heart-Stat Log, Diabetes, Waveform, and Iris-Plants datasets, outperforming the traditional 1-NN-classifier. Further inspecting the second and third columns of Table 11, we notice that with the exception of the Glass and the Segmentation datasets, SCE accomplishes compression rates of more than 94% without a significant loss in prediction accuracy for the other 6 datasets. For example, for the Waveform dataset, a 1-NN classifier that uses only 28 representatives outperforms the traditional 1-NN classifier that uses all 4500 training examples[2] by 7.3% in accuracy, increasing the accuracy from 76.8% to 84.1%. Similarly, for the Heart-StatLog dataset, a 1-NN classifier that uses just one representative for each class outperforms traditional 1-NN classifier by more than 6%.

7. Summary and Conclusion

In the paper a novel data mining technique we named *supervised clustering* was introduced. Supervised clustering algorithms aim at producing class-uniform density clusters. We claim that, to the best of our knowledge, only the work of Sinkkonen et al. [SKN02]. and Tishby et al. [TPB99] has some resemblance with our work. However, these approaches are targeted towards information retrieval applications, whereas our work centers on mining data that are commonly found in relational databases. Furthermore, our approach relies on distance metrics and representative-based clustering, whereas their approach is probabilistic and based on mutual information maximization.

Four representative-based supervised clustering algorithms were presented in the paper: TDS, SCEC, SRIDHCR and SPAM. Experiments were conducted that compare the performance of these four algorithms with a popular traditional clustering algorithm named PAM. Our experimental results show that supervised clustering enhances class purity by 9% to 25% over the traditional clustering algorithm PAM. Surprisingly, SPAM, a variation of PAM, performed quite poorly in the experiments, raising doubts with respect to PAM being a "good" algorithm for traditional clustering. TDS, on the other hand, considering its high speed, performed quite well on two datasets and quite poorly on the other two datasets. SRIDHCR is a good choice if resources are limited or data sets are very large, whereas, SCEC, although runtime intensive, finds the best clusters in almost all experiments conducted. In general, it seems that "greedy" algorithms, such as SPAM, SRIDHCR, and TDS, do not perform particularly well for supervised clustering and seem to terminate prematurely. Our experimental results suggest that algorithms that center on a more randomized exploration of the search space, such as SCEC, seem to find significantly better solutions at an average.

We argued that supervised clustering is useful for enhancing our understanding of datasets (e.g., summaries could generated for each cluster). We also presented empirical results that show how supervised clustering could be used for editing a dataset

[2] Due to the fact that we use 10-fold cross-validation, training sets contain 0.9*5000=4500 examples.

for the purpose of improving classification accuracy. For example, a 1-NN classifier that uses just one representative for each class of the Heart-StatLog dataset outperforms traditional 1-NN classifier that uses all 768 examples of the dataset, by more than 6% in accuracy.

References

[BBM02] Basu, S., Banerjee, A., Mooney, R. J. "*Semi-supervised Clustering by Seeding",* in Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002), pp. 19-26, Sydney, Australia, July 2002.
[BBM03] Basu, S., Bilenko,M., Mooney, R. "*Comparing and Unifying Search-based and Similarity-Based Approaches to Semi-Supervised Clustering*", in Proc. ICML03 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, pp. 42-29, Washington DC, August 2003.
[BHSW03] Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D. "*Learning Distance Functions Using Equivalence Relations*", in Proc. ICML03, Washington DC, August 2003.
[CCM00] Cohn, D., Caruana, R., McCallum, A. "*Semi-supervised Clustering with User Feedback*", unpublished manuscript, available at www-2.cs.smu.edu/~mccallum/, 2000.
[DBE99] Demiriz, A., Benett, K.-P., Embrechts, M.J. "*Semi-supervised Clustering using Genetic Algorithms*", in Proc. ANNIE'99.
[ERCBV04] Eick, C., Rouhana, A., Chen, C., Bagheriran, A., Vilalta, R. "*Using Clustering to Learn Distance Functions for Supervised Similarity Assessment*", submitted for publication.
[KKM02] 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.
[KR90] Kaufman L. and Rousseeuw P. J. "*Finding Groups in Data: an Introduction to Cluster Analysis*", John Wiley & Sons, 1990.
[UCIMLR] University of California at Irving, Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html
[SKN02] Sinkkonen, J., Kaski, S., and Nikkila, J., "*Discriminative Clustering: Optimal Contingency Tables by Learning Metrics", ECML'02.*
[ST99] Slonim, N. and Tishby, N., "*Agglomerative Information Bottleneck"*, Neural Information Processing Systems (NIPS-1999).
[TPB99] Tishby, N., Periera, F.C., and Bialek, W., "*The Information Bottleneck Method*", In proceedings of the 37th Allerton Conference on Communication and Computation, 1999.
[TS00] Tishby, N. and Slonim, N., "*Document Clustering using Word Clusters via the Information Bottleneck Method*", In the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2000.
[VAE04] Vilalta, R., Achari, M., Eick C. "*Piece-Wise Model Fitting Using Local Data Patterns*", to appear in Proceedings European Conference on Artificial Intelligence (ECAI), Valencia, Spain, August 2004.
[W72] Wilson, D.L., "*Asymptotic Properties of Nearest Neighbor Rules Using Edited Data*", IEEE Transactions on Systems, Man, and Cybernetics, 2:408-420, 1972.
[XNJ03] 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.
[Z04] Zhao, Z., "*Evolutionary Computing and Splitting Algorithms for Supervised Clustering*", Master's Thesis, University of Houston, Department of Computer Science, May 2004, http://www.cs.uh.edu/~zhenzhao/ZhenghongThesis.zip.