K-medoid-style Clustering Algorithms for Supervised Summary Generation

Nidal Zeidat
Dept. of Computer Science
University of Houston
E-mail: nzeidat@cs.uh.edu

Christoph F. Eick
Dept. of Computer Science
University of Houston
E-mail: ceick@cs.uh.edu

Abstract

This paper centers on the discussion of k-medoid-style clustering algorithms for supervised summary generation. This task requires clustering techniques that identify class-uniform clusters. This paper investigates such a novel clustering technique we term supervised clustering. Our work focuses on the generalization of k-medoid-style clustering algorithms. We investigate two supervised clustering algorithms: SRIDHCR (Single Representative Insertion/Deletion Hill Climbing with Restart) and SPAM, a variation of PAM. The solution quality and run time of these two algorithms as well as the traditional clustering algorithm PAM are evaluated using a benchmark consisting of four data sets. Experiments show that supervised clustering algorithms enhance class purity by 7% to 19% over the traditional clustering algorithm PAM, and that SRIDHCR finds better solutions than SPAM.

Key Words: supervised summary generation, clustering classified examples, k-medoid clustering algorithms, data mining.

1. Introduction

This paper centers on a novel data mining technique we term *supervised summary generation*. The objective of supervised summary generation is the creation of class centered summaries that recognize the patterns that are typical for one class; it also identifies how those patterns deviate from patterns that characterize other classes. In order to create such summaries, supervised summary generation requires the recognition of class-uniform clusters that have high probability density. This paper proposes *supervised clustering* algorithms for this purpose. 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. *Supervised clustering*, on the other hand, deviates from traditional clustering in that it is applied on classified examples where the objective is to identify 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 distance function.

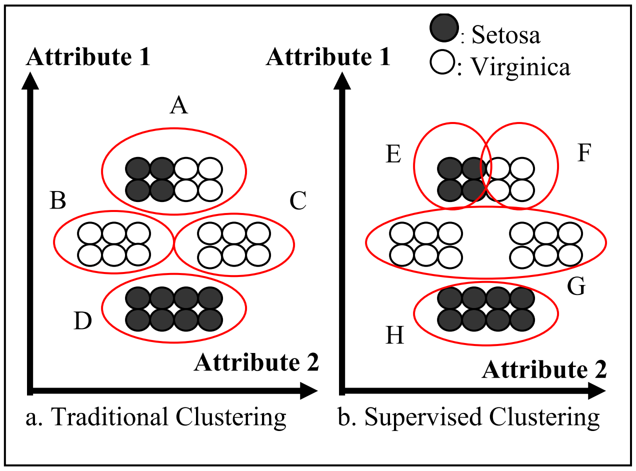

Figure 1: Differences between Traditional Clustering and Supervised Clustering

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 focuses on minimizing intra-cluster dissimilarities regardless of the classes that the objects belong to. 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**. 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 with respect to speed and solution quality. Section 2 discusses related work. Section 3 talks about K-medoid-style clustering. Section 4 introduces the clustering algorithms investigated. Section 5 presents experimental results and Section 6 concludes the paper and sheds light on our future work.

2. Related Work

Although we are not aware of research that directly centers on supervised summary generation and supervised clustering, there has been some work that has some similarity to our research under the heading of semi-supervised clustering. Semi-supervised clustering attempts to enhance a clustering algorithm by using side information in the clustering process that usually consists of a "small set" of classified examples. Xian [4] (and similarly [1]) take the classified training examples and transform those into constraints 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 [2] proposes a shortest path algorithm to modify a Euclidian distance function based on prior knowledge. Demiriz [6] proposes an evolutionary clustering algorithm to obtain clusters that minimize (the sum of) cluster dispersion and impurity.

Although some similarity in algorithms and techniques can be observed with respect to the work reviewed above, it should be stated clearly that, although our research investigates clustering algorithms, its focus is not on traditional clustering, but rather on using clustering as a preprocessing step to enhance existing classification algorithms and as a framework for summary generation.

3. K-mdoid-style Algorithms for Supervised Clustering

K-medoid-style clustering aims at finding a set of k representatives among all objects in the data set that best characterize the objects in the data set. Clusters are created by assigning each object to the cluster of the representative (medoid) that is closest to that object.

One might wonder why our work centers on developing k-medoid-style supervised clustering algorithms. The reason is that we believe that medoids are quite useful for class-specific data summarization, because it is the most prototypical object of the members of a cluster. Moreover, algorithms that restrict representatives to objects belonging to the data set, such as *k-medoid*, explore a smaller solution space if compared with centroid–based clustering algorithms, such as *k-means,* which searches a much larger set of representatives. Finally, when using k-medoid style clustering algorithms, ***only*** an inter-object distance matrix is needed and no "new" distances have to be computed during the clustering process as is the case with *k-means*.

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; trivially, having clusters that only contain a single example is not desirable, although it maximizes class purity.

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

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

$$\text{where Impurity}(X) = \frac{\text{\# of Minority Examples}}{n},$$

$$\text{and Penalty}(k) = \begin{cases} \sqrt{\dfrac{k-c}{n}} & k \geq c \\ 0 & k < c \end{cases}$$

with *n* being the total number of examples and *c* being the number of classes in a data set. The parameter β (0< β ≤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.

4. Clustering Algorithms Investigated

As stated earlier, our work investigates *k-medoid* style supervised clustering algorithms. In a traditional k-medoid algorithm, such as PAM [3], the major computations involve the determination if replacing a medoid by a non-medoid is beneficiary with respect to the evaluation function that evaluates a clustering. Due to the fact that finding the best number of clusters is part of the search process, this work extends this idea by having the proposed algorithms not only consider replacements of representatives by non-representatives, but also adding a non-representative to a set of representatives, and removing a representative from a set of representatives.

The following subsections describe the clustering algorithms that we investigated in our research.

4.1 Partitioning Around Medoids (PAM).

Due to its similarity to the investigated supervised clustering algorithms, PAM was selected as a representative of traditional clustering. We use PAM to compare the performance of traditional clustering and supervised clustering. PAM seeks to find *k representatives* minimizing the fitness function given in formula (2):

$$\text{Tightness(X)} = \frac{1}{n}\sum_{n} dissimilarity(obj_i, medoid(obj_i)) \quad (2)$$

where $medoid(obj_i)$ is the medoid (representative) of the cluster that object "obj_i" belongs to. The number of clusters, *k,* is an input parameter for the algorithm. As can be seen looking at formula (2), PAM evaluates a clustering by computing the average dissimilarity between all objects in the data set and their medoids. PAM is divided into two parts. The first part, called the *BUILD* algorithm, starts with a set of representatives that initially contains only the medoid of the complete data set. *BUILD*, then, greedily inserts new representatives into this set while minimizing the above fitness function. The second part of PAM, algorithm *SWAP*, tries to improve the clustering by exploring all possible (medoid, non-medoid) pairs in the data set; each time replacing the medoid with the non-medoid and evaluating the new clustering looking for better solutions.

We implemented PAM according to the algorithm described in section 2.4 of the book "Finding Groups in Data", [3].

4.2 Supervised Partitioning Around Medoids (SPAM); a generalization of PAM.

This algorithm is a modification of algorithm PAM. SPAM uses the fitness function q(X). The number of clusters (k) is an input parameter to the algorithm. SPAM consists of two sub-algorithms. Sub-algorithm *SBUILD* builds an initial solution (i.e., clustering) and 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 *X* that produces the minimum value for the fitness function *q(X)*. The second sub-algorithm, SSWAP, tries to improve the initial clustering produced by SBUILD by exploring all possible replacements of a single representative by a single non-representative. SSWAP continues its attempts to improve the current solution as long as it is able to find a solution that produces a lower value for the fitness function q(X) than the current solution. SSWAP terminates if no replacement can be found that leads to a clustering with a lower fitness value.

4.3 Single Representative Insertion/Deletion Steepest Decent Hill Climbing with Randomized Starting (SRIDHCR)

This algorithm starts by randomly selecting a number of objects from the data set as an initial set of representatives. Starting from this randomly generated set of representatives, the algorithm tries to improve the clustering by adding a single non-representative object to the set of representatives as well as trying to remove a single representative from the set. The algorithm terminates if the solution quality (measured by q(X)) does not improve. Moreover, we assume that the algorithm is run *r* (input parameter) times starting from different initial set of representatives each time, reporting the best of the *r* solutions found as its final clustering solution. Unlike PAM and SPAM, the number of clusters k is not fixed for SRIDHCR; the algorithm searches for "good" values of k.

To illustrate how the algorithm works let us have a closer look at a run of the algorithm for the Iris-Plants data set that consists of 150 flowers, numbered 1 through 150. The algorithm starts with a randomly generated set of representatives, e.g. {8, 42, 62, 148}. 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 1 depicts the solutions that are evaluated in the first iteration.
The 150 (146+4) clusterings (that were generated from the solutions that are partially listed in Table 1) 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 1. The search now continues using {8, 42, 62, 148, 52} as the new set of representatives.

Set of Medoids after adding one non-medoid	**q(X)**	**Set of Medoids after removing a medoid**	**q(X)**
8 42 62 148 1	0.091	42 62 148	0.086
8 42 62 148 2	0.091	8 62 148	0.073
…….....	…….	8 42 148	0.313
8 42 62 148 52	***0.065***	8 42 62	0.333
………	…….		
8 42 62 148 150	0.0715		
Trials in first part (add a non-medoid)		Trials in second part (drop a medoid)	

Table 1: Solutions Explored in the First Iteration

In the second iteration the solution {8, 42, 62, 148, 52, 122} (flower 122 was added to the set of representatives) turned out to be the best solution, leading to an improvement in fitness from 0.065 to 0.041. The program continues iterating as long as there is an improvement in fitness function q(X). The algorithm terminates after 7 iterations with the final solution {8, 62, 122, 117, 87}. Table 2 below illustrates how the set of representatives changed during the iterations.

Run	**Set of Medoids producing lowest q(X) in the run**	**q(X)**	**Purity**
1	8 42 62 148 52	0.065	0.947
2	8 42 62 148 52 122	0.041	0.973
3	42 62 148 52 122 117	0.030	0.987
4	8 62 148 52 122 117	0.021	0.993
5	8 62 148 52 122 117 87	0.016	1.000
6	8 62 52 122 117 87	0.014	1.000
7	8 62 122 117 87	0.012	1.000

Table 2: Set of Representatives Explored

Notice that in iteration 5, the class purity already reached 100%. 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 minimize the number of clusters; the algorithm therefore continued and found a clustering that uses only 5 clusters but still achieves 100% class purity.

5. Experimental Evaluation

In order to evaluate the comparative performance of the clustering algorithms presented in section 4, we ran them on a benchmark, consisting of four data sets that were obtained from UCI Machine Learning repository [5]. Table 3 gives a summary for the four data sets we used. This section presents and analyzes the outcome of these experiments**.**

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 3: Data Sets Used in the Experimental Evaluation

All data was 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 and an inter-object distance matrix was generated for each normalized data set. In the experiments the SRIDHCR algorithm was run (empirically) 50 times, each time with a different set of initial representatives and the quality of the best solution found was reported.

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. (Tightness(X), 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 memory.

Since algorithm SRIDHCR searches for a good k value, it was run first for a certain β value. After that, algorithms PAM & SPAM were run for the same value of β and the number of clusters (k) that algorithm SRIDHCR determined to be the best value.

5.1 Traditional Versus Supervised Clustering

Table 4 presents results from clustering the four data sets using the traditional clustering algorithm, PAM, as well supervised clustering algorithms SPAM and SRIDHCR. Looking at Table 4 we observe that, although traditional clustering using PAM produces tighter clusters, characterized by smaller intra-cluster distances (Tightness(X) in Table 4), supervised clustering algorithms, represented by algorithms SRIDHCR and SPAM, produce better cluster purity. The improvement in cluster purity ranges from 7% to 19% improvement for the four datasets. Data sets that have objects with clear class discriminating characteristics, like the Iris-Plants and Image-Segmentation data sets, produce better class purity than other, more challenging, data sets, such as *Vehicle* and *Diabetes*.

Algorithm	Purity	q(X)	Tightness(X).
Iris-Plants data set, # clusters=3			
PAM	0.907	0.0933	0.081
SRIDHCR	0.981	0.0200	0.093
SPAM	0.973	0.0267	0.133
Vehicle data set, # clusters =65			
PAM	0.701	0.326	0.044
SRIDHCR	0.835	0.192	0.072
SPAM	0.764	0.263	0.097
Image-Segment data set, # clusters =53			
PAM	0.880	0.135	0.027
SRIDHCR	0.980	0.035	0.050
SPAM	0.944	0.071	0.061
Pima-Indian Diabetes data set, # clusters =45			
PAM	0.763	0.237	0.056
SRIDHCR	0.859	0.164	0.093
SPAM	0.822	0.202	0.086

Table 4: Traditional vs. Supervised Clustering (β=0.1)

Looking at these results from *class summary generation* prospective, the notable improvement in clusters' class purity in supervised clustering greatly aids in producing class summaries that are far more accurate.

5.2 Performance of the Supervised Clustering Algorithms

Table 5 presents the different performance measures of the three algorithms when applied on all data sets for β=0.1. The results clearly show that the supervised clustering algorithms produce better cluster purity than PAM. Nevertheless, Wall-Clock time for algorithm SRIDHCR is the highest among all three because its results are the best of 50 runs, as explained earlier. Table 6 presents the different performance measures of the three algorithms for β=0.4. Unlike Table 5, Table 6 gives the *average value per run* for the performance measures for algorithm SRIDHCR. Table 6 shows that not only the *average performance* of algorithm SRIDHCR is better than that of algorithm PAM by 6% (Iris-Plants) to 21% (Pima-Indian Diabetes) but also the *average* Wall-Clock Time for SRIDHCR is 0.04 and 0.11 of that of PAM for data sets Vehicle and Segmentation, respectively.

On the other hand, comparing the performance of the two supervised clustering algorithms with each other, we clearly see that algorithm SRIDHCR produces better cluster purity than SPAM in all experiments, especially for the *Vehicle* data set: SRIDHCR produces cluster purity that is 10% higher than SPAM.

Algorithm	q(X)	Purity	Tightness (X)	WCT (Sec.)
IRIS-Flowers Dataset, # clusters=3				
PAM	0.0933	0.907	0.081	0.06
SRIDHCR	0.0200	0.980	0.113	11.00
SPAM	0.0267	0.973	0.133	0.32
Vehicle Dataset, # clusters=65				
PAM	0.326	0.701	0.044	372.00
SRIDHCR	0.192	0.835	0.062	1715.00
SPAM	0.263	0.764	0.097	1090.00
Segmentation Dataset, # clusters=53				
PAM	0.135	0.880	0.027	4073.00
SRIDHCR	0.035	0.980	0.050	11250.00
SPAM	0.071	0.944	0.061	1422.00
Pima-Indians-Diabetes, # clusters=45				
PAM	0.237	0.763	0.056	186.00
SRIDHCR	0.164	0.859	0.076	6600.00
SPAM	0.202	0.822	0.086	58.00

Table 5: Comparative Performance of the Different Algorithms, β=0.1

Algorithm	Avg. Purity	Tightness(X)	Avg.WCT (Sec.)
IRIS-Flowers Dataset, # clusters=3			
PAM	0.907	0.081	0.06
SRIDHCR	0.959	0.104	0.18
SPAM	0.973	0.133	0.33
Vehicle Dataset, # clusters=56			
PAM	0.681	0.046	505.00
SRIDHCR	0.762	0.081	22.58
SPAM	0.754		681.00
Segmentation Dataset, # clusters=32			
PAM	0.875	0.032	1529.00
SRIDHCR	0.946	0.054	169.39
SPAM	0.940	0.065	1053.00
Pima-Indians-Diabetes, # clusters=2			
PAM	0.656	0.104	0.97
SRIDHCR	0.795	0.109	5.08
SPAM	0.772	0.125	2.70

Table 6: *Average* Comparative Performance of the Different Algorithms, β=0.4

But this better performance of SRIDHCR does not come for free. In the same experiment that SRIDHCR produces cluster purity 10% better than SPAM, it takes SRIDHCR 70% more time to do the clustering task than SPAM. This is due to the fact that SRIDHCR uses 50 restarts.

One reason we attribute to the fact that SRIDHCR finds significantly better solutions than SPAM is that the fitness landscape induced by q(X) contains many ties (different clusterings X1 and X2 with q(X1)=q(X2)); i.e., two solutions might have the same cardinality as well as same number of minority class examples although they cluster the data set differently. SPAM is not particularly good in coping with those plateau structures in the fitness landscape: SPAM just terminates if no better solution can be found through a single replacement of a representative. SRIDHCR, on the other hand, has the interesting characteristic that when enhancing 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.

To further investigate this observation we ran an experiment where for a certain combination of (dataset, *β* value, and *k* value), we created 5000 different sets containing *k* representatives for the dataset. For each of the 5000 sets of objects, we calculated q(X) as well as Tightness(X). Then we calculated how many ties with respect to q(X) and Tightness(X) we have. The result was divided by maximum number of possible ties among the 5000 sets, using the formula below, before it is reported.

$$\text{Maximum Number of Ties (n)} = \frac{n*(n-1)}{2} \qquad (3)$$

Table 7 shows the percentage of ties for all 4 datasets with respect q(X) and Tightness(X) for β=0.00001 and 0.4. Notice that the probability of ties increases dramatically when using q(X), which, we believe, is one reason for the fact that SPAM does not seem to find solutions of good quality, if compared with SRIDHCR.

We believe that SRIDHCR's capability to add and remove representatives also 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 faster average runtime, SRIDHCR can be re-run up to 30 times (as it is the case for the Vehicle data set for example in Table 6) with different initial solutions in the same time that SPAM needs to complete a single run. These re-runs with different initial sets of representatives allow SRIDHCR to explore different regions of the search space, which we believe is a third explanation for SRIDHCR's significant better performance.

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 7: Ties distribution

5.3 Relationships between β, k, and Purity

Figure 2 shows how 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 result were obtained by running the *SRIDHCR* algorithm).

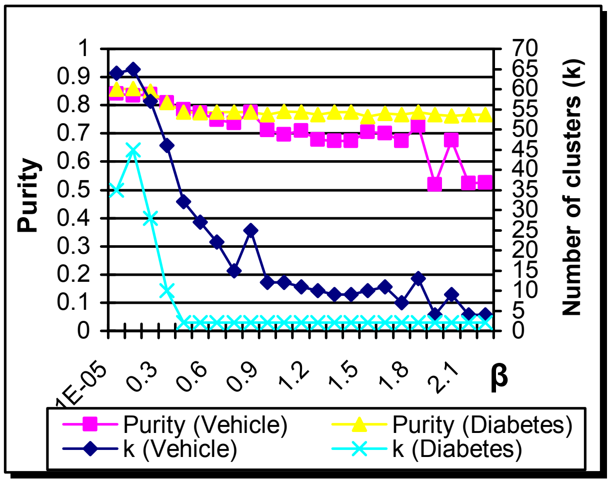

Figure 2: How Purity and k Change as β Increases

As can be seen in Figure 2, 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 a decreasing cluster purity as β increases.

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).

The previous example shows one advantage of our approach of associating a penalty function with the number of clusters instead of using fixed values for *k*. The parameter β narrows the search space for the values of *k* that "good" solutions can take, but does not restrict it to a single value. Consequently, a supervised clustering algorithm still tries to find the best value for *k* within the boundaries induced by β. This explains why in the previous example algorithm SRIDHCR selected clusterings with higher *k*-values for the Vehicle data set than for the Diabetes dataset.

6. Summary

A novel data mining technique we term supervised summary generation was introduced. Supervised summary generation utilizes supervised clustering algorithms aiming at producing class-uniform density clusters. Two proposed supervised clustering algorithms were presented: SRIDHCR and SPAM. Experiments were conducted that compare the results of these two algorithms with a popular traditional clustering algorithm PAM. Both, SRIDHCR and SPAM, were found to produce significantly better cluster purity than PAM. Improvement ranges between 7% and 19% for different data sets.

Moreover, SRIDHCR outperforms SPAM, finding better solutions with respect to q(X) for all four data sets for all β-values tested. We also tried to explain SPAM's unexpectedly bad performance. Reasons for its bad performance include the presence of a large number of local minima, the presence of plateaux like structures in the fitness landscape of q(X), and the lack of a restart feature. We also believe that our results raise some serious doubts about the quality of solutions that PAM finds, a topic that is investigated by our current research.

In addition to developing efficient, and scalable supervised clustering algorithms our current and future work centers on using supervised clustering for enhancing simple classifiers and for distance function learning.

References

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

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

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

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

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

[6] A. Demiriz, K.P. Benett, and M.J. Embrechts, "Semi-supervised Clustering using Genetic Algorithms", *in Proc. ANNIE'99*.