Feature Selection for Clustering

Manoranjan Dash and Huan Liu

School of Computing, National University of Singapore, Singapore.

Abstract. Clustering is an important data mining task. Data mining often concerns large and high-dimensional data but unfortunately most of the clustering algorithms in the literature are sensitive to largeness or high-dimensionality or both. Different features affect clusters differently, some are important for clusters while others may hinder the clustering task. An efficient way of handling it is by selecting a subset of important features. It helps in finding clusters efficiently, understanding the data better and reducing data size for efficient storage, collection and processing. The task of finding original important features for unsupervised data is largely untouched. Traditional feature selection algorithms work only for supervised data where class information is available. For unsupervised data, without class information, often principal components (PCs) are used, but PCs still require all features and they may be difficult to understand. Our approach: first features are ranked according to their importance on clustering and then a subset of important features are selected. For large data we use a scalable method using sampling. Empirical evaluation shows the effectiveness and scalability of our approach for benchmark and synthetic data sets.

1 Introduction

Clustering is an important data mining task that groups similar objects together [8,11,10,4,2]. Similarity between a pair of data points is due to different features. If similarity is distance-based then for a pair of data points in a cluster there exist at least a few features on which the points are close to each other. Most clustering methods assume all features to be equally important for clustering, or in other words they do not distinguish among different features. This is one of the reasons why most clustering algorithms may not perform well in the face of high-dimensional data. Another reason of the poor performance is the inherent sparsity of data in high-dimensional space. In reality different features have varying effects on clustering. An important feature helps in creating clusters while an unimportant feature may not help in creating clusters and, in contrary, it may affect the clustering algorithms adversely by blurring the clusters. Unimportant features are *noisy or irrelevant* and can be removed to reduce the data size for more efficient clustering. It also reduces the noise and helps in data storage, collection, and processing.

As clustering is done on unsupervised data without class information, traditional feature selection algorithms for classification [6] do not work. Little work has been done on feature selection for unsupervised data. Dimensionality reduction or feature extraction methods (e.g., Principal Components Analysis, Karhunen-Loeve transformation, or Singular Value Decomposition) are commonly used [8]. They have drawbacks such as: (1) it is difficult to understand the data (and the found clusters) using the extracted features, and (2) the original features remain as they are required to determine the extracted features.

Some recent works on clustering try to handle high-dimensionality by selecting important features. In [2] and later in [5] it is observed that dense regions may be found in subspaces of high dimensional data. The algorithm called CLIQUE in [2] divides each dimension into a user given divisions. It starts with finding dense regions in 1-dimensional data and works upward to find k-dimensional dense regions using candidate generation algorithm Apriori [3]. This approach is different from the conventional clustering that partitions the whole data. In [1] a new concept is presented called "projected clustering" to discover interesting patterns in subspaces of high-dimensional data. It finds the clusters first and then selects a subset of features for each cluster. It searches for the subset of features by putting a restriction on the minimum and the maximum number of features.

We address the problem of selecting a subset of important features for clustering for the *whole data and not just for clusters* unlike in [1, 2]. This helps in knowing the important features before doing clustering and the clustering task becomes more efficient and focused as only the important features can be used. Finding the important original features for the whole data helps in understanding the data better unlike principal components. Data storage, collection and processing tasks become more efficient and noise is reduced as the data is pruned.

Our approach is a 2-step method: we first rank and then select a subset of important features. Ranking of features is done according to their importance on clustering. An entropy-based ranking measure is introduced. We then select a subset of features using a criterion function for clustering that is invariant with respect to different numbers of features. A novel scalable method based on random sampling is introduced for large data commonly found in data mining applications.

2 Importance of Features on Clustering

Notations used in the paper are as follows: X_i is i^{th} data point, X_{ik} is k^{th} feature value of i^{th} point, F_k is k^{th} feature where $i = 1...N$ and $k = 1...N$; D_{i_1,i_2} and S_{i_1,i_2} are distance and similarity between points X_{i_1} and X_{i_2}; χ_j is j^{th} cluster where $j = 1...c$. We start by showing visually the effects of features on clustering. In Figure 1(a,b,c) we show a synthetic data in (3,2,1)-d feature spaces respectively. There are 75 points with 3 clusters in $F1$-$F2$ dimensions, with each cluster having 25 points. Values in $F1$ and $F2$ features follow Gaussian distribution within each of the 3 clusters while values in feature $F3$ are uniformly random. When we take 3 features the clusters are unclear and unnecessarily complex (see Figure 1(a)), whereas no clusters can be found when we visualize using only 1 feature $F1$ (Figure 1(c)). Figure 1(b) with $F1$-$F2$ features shows 3 well-formed clusters. Selecting features $F1$ and $F2$ reduces the dimensionality of the data while forming well separated clusters.

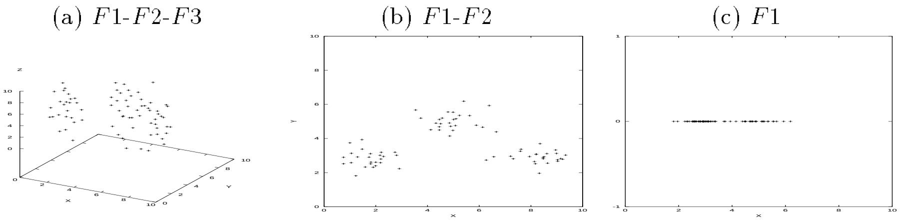

Fig. 1. Effect of features on clustering.

In a single dimensional data set clusters can be formed if the single feature takes values in separate ranges. In a multi-dimensional data set clusters can be formed from combination of feature values although the single features by themselves alone may take uniform values. We have noted down 2 distinct scenarios in the following.

Scenario 1: *A single feature is important by itself only*: Consider Figure 2(a) where there are 2 features. Feature $F2$ is uniformly distributed while $F1$ takes values in 2 separate ranges. It can be clearly seen that $F1$ is more important for creating clusters than $F2$.

Scenario 2: *Two features are necessarily important and any individual feature is useless in defining clusters*: Consider Figure 2(b) where there are 2 features. Both $F1$ and $F2$ are uniformly distributed. It can be clearly seen that both $F1$ and $F2$ are necessary for clustering and any one alone is useless.

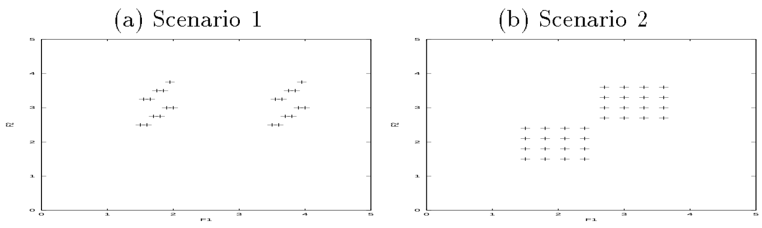

Fig. 2. Effects of features on clusters: scenario 1 shows effect of individual feature while scenario 2 shows the combined effect of 2 features.

3 Entropy-based Feature Ranking

Consider each feature F_i as a random variable while f_i as its value. From entropy theory we know that, entropy is:
$E(F_1, ..., F_M) = -\sum_{f_1} ... \sum_{f_M} p(f_1, ..., f_M) \log p(f_1, ..., f_M)$ where $p(f_1, ..., f_M)$ is the probability or density at the point $(f_1, ..., f_M)$. If the probability is uniformly distributed we are most uncertain about the outcome, and entropy is maximum. This will happen when the data points are uniformly distributed in the feature space. On the other hand, when the data has well-formed clusters the uncertainty is low and so also the entropy. As we do not have a priori information about clusters, calculation of $p(f_1, ..., f_M)$ is not direct. But we can use the following way to calculate entropy without any cluster information.
Entropy Measure: Usually in a real-world data there may be a few not very well-formed clusters and some noise (points not belonging to any cluster properly). Two points belonging to the same cluster or 2 different clusters will contribute to the total entropy less than if they were uniformly sepatated. Similarity S_{i_1,i_2} between 2 instances X_{i_1} and X_{i_2} is high if the 2 instances are very close and S_{i_1,i_2} is low if the 2 are far away. Entropy E_{i_1,i_2} will be low if S_{i_1,i_2} is either low or high and E_{i_1,i_2} will be high otherwise. The following mathematical formulation is based on this idea.

Our similarity measure is applicable to both numeric and nominal data. Similarity is based on distance, i.e., for numeric data we use *Euclidean* distance while for nominal data we use *Hamming* distance. Mathematically similarity for numeric data is given as: $S_{i_1,i_2} = e^{-\alpha \times D_{i_1,i_2}}$ where α is a parameter. In a multi-dimensional space, distance D_{i_1,i_2} for numeric data is defined as: $D_{i_1,i_2} = [\sum_{k=1}^{M} (\frac{x_{i_1k} - x_{i_2k}}{max_k - min_k})^2]^{1/2}$. The interval in the k^{th} dimension is normalized by dividing it by the maximum interval $(max_k - min_k)$ before calculating the distance. If we plot similarity against distance, the curve will have a bigger curvature for a larger α. The insight is we assign a very high similarity for points 'very close' together but assign a low similarity for points 'not close' or 'far away'.

Similarity for *nominal* features is measured using the Hamming distance. The similarity between two data points is given as: $S_{i_1,i_2} = \frac{\sum_{k=1}^{M} |x_{i_1k} = x_{i_2k}|}{M}$ where $|x_{i_1k} = x_{i_2k}|$ is 1 if x_{i_1k} equals x_{i_2k} and 0 otherwise. For data with both numeric and nominal features, we can discretize numeric values first before applying our measure. For two points X_{i_1} and X_{i_2}, entropy is: $E = -S_{i_1,i_2} \log S_{i_1,i_2} - (1 - S_{i_1,i_2}) \log (1 - S_{i_1,i_2})$ which assumes the maximum value of 1.0 for $S_{i_1,i_2} = 0.5$, and the minimum value of 0.0 for $S_{i_1,i_2} = 0.0$ and $S_{i_1,i_2} = 1.0$. For a data set of N data points entropy is given as: $E = -\sum_{i_1=1}^{N} \sum_{i_2=1}^{N} (S_{i_1,i_2} \times \log S_{i_1,i_2} + (1 - S_{i_1,i_2}) \times \log(1 - S_{i_1,i_2}))$ where S_{i_1,i_2} takes values in [0.0-1.0]. In this work, α is calculated automatically by assigning 0.5 in Equation: $\overline{S} = e^{-\alpha \times \overline{D}}$ at which entropy is maximum; so we get: $\alpha = \frac{-\ln 0.5}{\overline{D}}$ where $\overline{D}$ is the average distance among the data points.
Algorithm to Rank Features: If the removal of feature F_1 causes more disorder than the removal of feature F_2 then $E_{-F_1} > E_{-F_2}$ where E_{-F_1} and E_{-F_2}

are entropy after removing F_1 and F_2 respectively. In scenario 1 (Figure 2(a)) $E_{-F_1} > E_{-F_2}$. For scenario 2 (Figure 2(b)) we added one more feature F_3 which takes uniformaly random values and does not take part in forming the 2 clusters. As expected we got $E_{-F_1} > E_{-F_3}$ and $E_{-F_2} > E_{-F_3}$. *Secenario 2 suggests that our entropy measure works for dependent features also.*

For ranking of features we can use E in the following way: Each feature is removed in turn and E is calculated. If the removal of a feature results in minimum E the feature is the least important; and *vice versa*. In the algorithm CalcEnt(F_k) calculates E of the data after discarding feature F_k.

Algorithm (RANK):

$P = E$ values for M features
For $k = 1$ to M
  $P_k = \text{CalcEnt}(F_k)$
OutputRank(P)

Scalable Feature Ranking: Data mining generally concerns data with large number of data points. For a large number of data points our ranking measure may not be practical *as it is* (the complexity of RANK is $O(MN^2)$ if we take the similarity measure between 2 points as unit). There are different approaches available in the literature for handling large data sets for a given algorithm. Our scalable method is based on *random sampling*. We observed that a reasonably small random sample retains the original cluster information in most cases. This phenomenon was also observed in [9] in their work on initialization of partitional clustering algorithms. Notice that for entropy measure to work well the cluster structure needs to be retained and it is largely independent of the number of data points. So, random sampling is a good choice for scalability. The algorithm is simple. Initially all features are ranked 0. Random samples are generated and RANK is run over each sample to produce the rankings of features. The feature rankings are added correspondingly. At the end of all random samples p (we suggest the use of at least 35 samples as 35 is often considered the minimum number of samples for large sample procedures [7]) we obtain the final rankings of the features.

Algorithm for Scalable Ranking, SRANK

for all features F_k's *O*verall *R*ank, $OR_k = 0$
for $l = 1$ to p
  take a sample L_l
  run RANK to find rankings R_l
  for $k = 1$ to M
    $OR_k = OR_k + R_{l_k}$
output overall rankings OR

Selecting a Subset of Important Features:

A problem is how many features we should choose from a ranked list. A natural expectation is that entropy would fall initially with removal of unimportant features, but would stop falling at some point. This is not the case as the entropy is not invariant with respect to different numbers of features. Hence we are left with finding the different alternatives of selecting a subset of features: (1) If one knows the number of important features required, just pick them starting with the most important one, or (2) we can choose a clustering algorithm and choose the subset that maximizes the clustering quality. The first option is not practical without any a priori knowledge. The second option is a wrapper method. Wrapper method is a feature selection method that wraps around clustering algorithm which is a standard way for supervised feature selection with a classification algorithm [12]. The difference is that *in our case features are already ranked according to their importance and so the task of searching through the feature subset space of 2^M is avoided.* The idea is to run a clustering algorithm on the selected features and choose the subset that produces best cluster quality.

We choose k-means clustering algorithm which is very popular and simple to implement. It is iterative, provides results fast (converges fast to local maxima), has a time complexity of $\#Iter * |Data| * c$ where $\#Iter$ is is the number of iterations, $|Data|$ is the size of the data $(N * M)$, and c is the number of clusters. Once the clustering is done we need to measure the cluster quality. There are numerous criterion functions for clustering in literature to measure cluster quality. We select scattering criterion which is invariant under non-singular transformation of data. **Scattering Criteria**: These criteria consider the scatter matrices used in multiple discriminant analysis. Scatter matrix for j^{th} cluster: $P_j = \sum_{X_i \in \chi_j} (X_i - m_j)(X_i - m_j)^t$ Within-cluster scatter matrix: $P_W = \sum_{j=1}^{c} P_j$ Between-cluster scatter matrix: $P_B = \sum_{j=1}^{c} (m_j - m)(m_j - m)^t$ where m is the total mean vector and m_j is the mean vector for j^{th} cluster and $(X_i - m_j)^t$ is the matrix transpose of the column vector $(X_i - m_j)$. Among different scattering criteria we briefly describe here an 'Invariant Criterion'. One invariant criterion is: $tr({P_W}^{-1} P_B)$ where tr is *trace* of a matrix which is the sum of its diagonal elements. It is invariant under nonsingular linear transformations of the data. It measures the ratio of between-cluster to within-cluster scatter. The higher the $tr({P_W}^{-1} P_B)$, the higher the ratio of between-cluster scatter to within-cluster one and hence, and hence, the higher the cluster quality. We use $tr({P_W}^{-1} P_B)$ to compare the cluster quality for different subsets of important features. The algorithm for selecting a subset of features is as follows:

Algorithm SELECT:

run RANK to get rankings $R_k, k = 1...M$
for k=1 to M
  run K-means to find clusters using subset $(R_1, ..., R_k)$
  calculate $tr(P_W^{-1} P_B)$
  if stopping criterion satisfy break

In case of large data run SRANK instead of RANK. $Tr(P_W^{-1} P_B)$ will increase with the addition of features if the ratio of between-cluster to within-cluster increases, otherwise it decreases or remains relatively unchanged. As is found by the experiments, $tr(P_W^{-1} P_B)$ increases initially and once all important features are added, it either goes down or remains relatively unchanged for any addition of unimportant features. The point at which $tr(P_W^{-1} P_B)$ goes down or remains unchanged is not difficult to detect visually, hence the stopping criterion is manually decided.

4 Experiments

We empirically tested our feature selection method on different scenarios that one may find in various data mining applications. First, tests are conducted on benchmark and synthetic data sets to check the correctness of our claim that our feature selection method can select correct features as we know well about these data sets. Tests are then conducted on a large high-dimensional data to test the performance of SRANK. We used a MATLAB random function to generate synthetic data. For synthetic data sets a few features are chosen as important and these features follow Gaussian distribution. Each cluster is of equal size if not mentioned otherwise. Clusters are usually overlapping. Unimportant features are added which take uniformly random values. Each data has 5% noisy data points.

Benchmark and Synthetic Data: Three synthetic data sets are generated with different numbers of clusters and features. Benchmark data sets (both numeric and nominal) are selected from UCI machine-learning repository [13]. See Table 1 for the details about data sets. We have chosen those data sets from the repository for which prior information is available regarding importance of features. Although for these benchmark data sets class information is available, in our experiments we have removed the class labels. Parity3+3 has 3 relevant, 3 redundant, and 6 irrelevant features.

The results for ranking are shown in Table 1. Our method is able to rank the important features in the top ranks for all data. For CorrAL our method ranks feature $F6$ higher. $F6$ is correlated to the class label 75% of the data points. This shows our ranking measure favors features that are correlated to the class. Although for CorrAL this is not desired but for real-world data this may be acceptable. For Parity3+3 ranking was correct although the redundant features could not be detected. This can be removed if after selecting a subset of features we check for redundancy between the features in pair. For a small subset of selected features this may not be extermely prohibitive.

The results for selecting a subset of features are shown for the data sets with a known number of clusters and numeric data. We use k-means and $tr({P_W}^{-1}P_B)$ to evaluate the subsets of important features. Initialization of k-means is done by randomly choosing points from the data. Once a set of points are chosen for the whole data the same set is used for different subsets of features. The results are summarized in Figure 3. The X-axis of the plots is for number of most important features and Y-axis is for $tr(P_W^{-1}P_B)$ value for the corresponding subset of most important features. For Iris data set *trace* value was the maximum for the two most important features. For D3C, D4C and D6C data *trace* value increases with addition of important features in a fast rate but slows down to almost a halt after all the important features are added. For a practical application it will not

Data Set	M	#Clusters/Classes	Important Features	Ranking (Descending Order)
Iris	4	3	3,4	**{3,4}**,1,2
ChemicalPlant	5	∞	1,2,3	**{3,1,2}**,4,5
Non-linear	4	∞	1,2	**{2,1}**,3,4
Parity3+3	12	2	{1,7},{2,8},{3,9}	**{9,3,8,2,7,1}**,4,10,...
CorrAL	6	2	1,2,3,4	**{3**,6,**1,2,4}**,5
Monk3	6	2	2,4,5	**{5,2,4}**,1,6,3
D3C	4	3	1,2	**{2,1}**,4,3
D4C	15	4	1-5	**{1,3,5,2,4}**,13,9,11,...
D6C	22	6	1-7	**{3,6,5,4,2,1,7}**,10,9,...

Table 1. Ranking of features: ∞ – Class is continuous, bold font is used to show the correctness of the ranking.

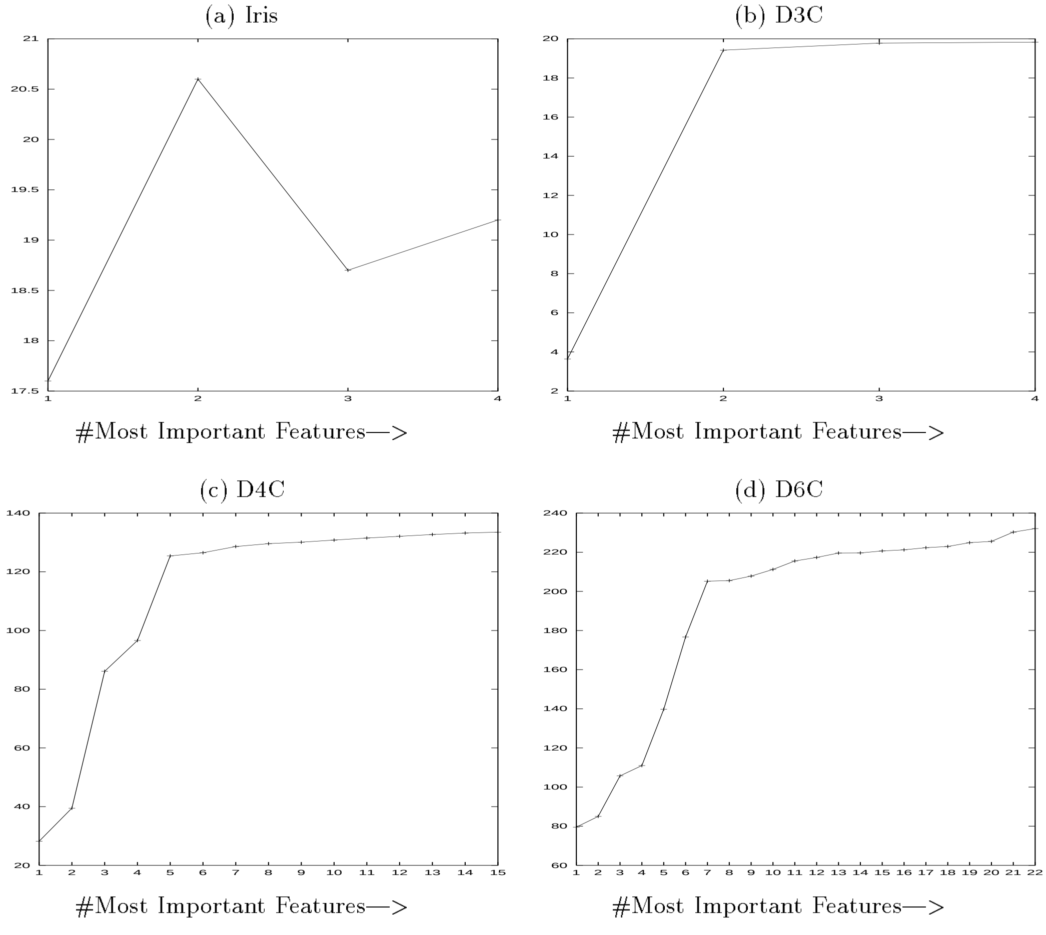

Fig. 3. $tr(P_W^{-1} P_B)$ of Iris and Synthetic data.

be difficult to notice these trends, and hence selecting a subset of features can be an easy task.

Large and High-Dimensional Data: We show the results of our feature selection on a synthetic large and high-dimensional data. The data has 100 features (first 20 features are important and the next 80 features unimportant), 5 clusters, each cluster created by Gaussian distribution, unimportant features take uniformly random values. Each cluster has 20,000 points and the data has 5000 (approximately 5%) noisy data points. Sample sizes chosen are 0.25%, 0.50% and 1.0%.

<u>Results</u>: For space constraint we have shown results of SRANK and SELECT for 5 samples. SRANK results are shown in Table 2. The last row in Table 2 is the over all rankings after 5 runs. In all the runs the 20 important features are ranked at the top, and hence, they are ranked at the top in over all ranking as well. SELECT results are shown in Figure 4 and Table 3. We have shown average results for 5 sample runs for 0.25%. In Table 3 impurity is "number of misclassifications not including the noise"[1]. Notice that impurity is low or zero when only important features are used and it grows with addition of unimportant features. It further confirms our suspicion that k-means (and probably other clustering algorithms) get confused with useless features and removing them can contribute to the cluster quality. This result shows satisfactory scalability of SRANK and SELECT.

5 Conclusion

We tested RANK over a real-world textual finance data. As many as 423 phrases or words are used as features for each textual financial data taken on a daily basis from reliable and standard sources such as Wall Stree Journal. The feature values are the frequencies of the corresponding phrases in that day's reports. After running RANK over this high-dimensional data we showed the results to a domain expert. He was satisfied regarding the top rankings given to important phrases such as: blue chip, property lost, banking retreat, etc. Efforts are on to use this ranking for prediction purposes. Another application yet untested is Reuters text categorization data which has hundereds of thousands of words as features. It may be useful to pick up a few hundred words for further classification. We studied a number of related issues such as high-dimensional data, noisy data, large data, redundant/correlated features, and hill-climbing *vs.* exhaustive. Handling high dimensional data is a prominent desirable characteristic of our method. Experiments show that in the face of high-dimensional data k-means algorithm perform poorly, but removal of unimportant features significantly improved its performance. Our method is able to handle noisy data. To handle very large data sets we used random samples. Our ranking measure works well consistently for

[1] As we have generated the data, the data points that group together in a cluster are known and it enables us to find impurity after clustering. This may not be the case for real-world data, and hence $tr({P_W}^{-1}P_B)$ is more useful practically.

#Run	Sample Size		
	0.25%	0.50%	1.0%
1	{15,5,20,14,9,12,2, 7,18,11,17,1,19,10,3, 6,16,8,4,13},42,57,...	{20,5,19,3,14,17,9, 2,11,10,13,1,16,7,4, 8,15,6,18,12},43,81,...	{15,19,3,16,13,10,9,5, 11,17,12,4,14,20,2,1, 8,18,6,7},71,23,...
2	{5,6,14,20,7,10,12, 17,16,18,15,13,8,9,19, 2,11,1,4,3},63,25,...	{19,14,16,13,3,6,15, 18,17,2,11,8,1,4,7, 9,5,12,10,20},55,23,...	{12,13,7,6,4,1,19,3, 9,20,10,11,15,18,8,2, 14,17,16,5},42,29,...
3	{14,6,17,13,12,9,20,15, 10,5,2,19,1,16,8,7, 11,3,18,4},29,92,...	{19,10,15,2,18,3,8, 13,16,7,17,14,12,5,11, 20,9,4,1,6},68,39,...	{9,4,5,2,16,14,1,3, 12,19,20,6,17,15,18,10, 13,7,8,11},71,92,...
4	{11,12,17,1,4,9,8,3, 5,18,16,2,6,19,14,13, 7,20,15,10},32,77,...	{8,1,3,19,15,18,12,7,11, 2,4,20,10,13,5,14,17, 6,9,16},44,83,...	{10,19,12,6,1,14,8, 7,20,17,18,13,16,15,2, 4,11,5,3,9},26,70,...
5	{13,19,9,16,20,18,10, 6,8,4,12,5,15,14,17, 2,1,11,3,7},42,37,...	{15,16,13,2,10,8,19,11,14, 4,3,6,1,9,17,12,7, 20,18,5},37,24,...	{4,15,14,13,3,9,10,19, 1,7,8,12,5,18,2,16, 17,6,20,11},74,53,...
OverAll Ranking	{12,5,9,14,20,17, 6,13,15,11,18,10,16,19, 2,8,1,7,3,4},23,98...	{19,3,15,2,13,8,14, 16,10,11,18,17,1,7,12, 20,4,5,9,6},45,87,...	{19,4,12,13,10,1,3, 9,15,14,16,6,7,5,20, 2,17,8,18,11},62,54,...

Table 2. Ranking for 5 random samples of 3 different sizes

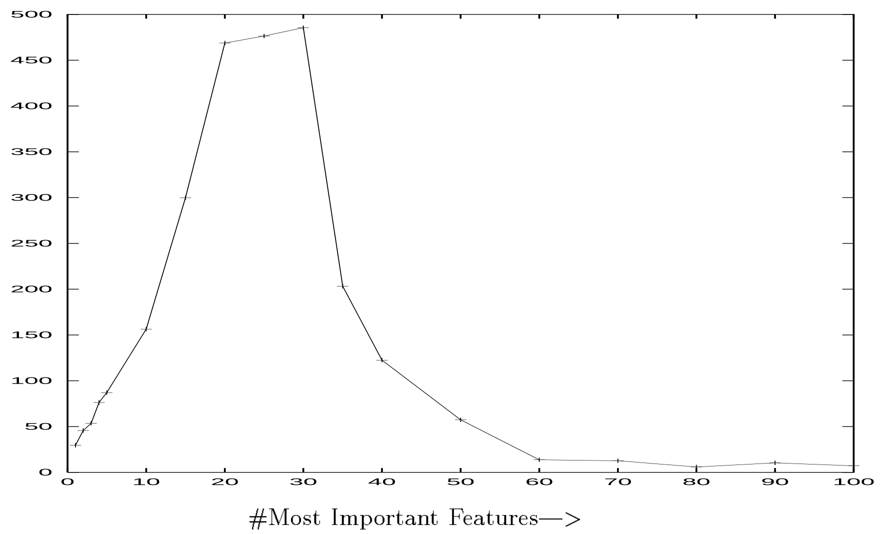

Fig. 4. Average $tr(P_W^{-1} P_B)$ of 5 samples of size 0.25% of a Large and High-Dimensional Data.

#MostImpFea	$tr(P_W^{-1}P_B)$	Impurity (%)	#MostImpFea	$tr(P_W^{-1}P_B)$	Impurity (%)
1	29.56	10.4	35	203.18	10.2
5	87.13	8.2	40	122.5	17.4
10	156.4	0.0	50	57.4	36.8
15	299.67	0.0	60	13.71	62.2
20	468.81	0.0	70	5.87	56.0
25	476.36	0.0	80	10.37	66.4
30	485.57	2.0	100	7.13	73.0

Table 3. Average $tr(P_W^{-1}P_B)$ and Impurity of 5 samples of 0.25% of a large and high dimensional Data

the different runs with different sizes of random samples. Our method only requires the cluster structure be retained which a reasonably small random sample is expected to maintain. We studied the issue of redundant/correlated features. We did an experimental study of comparing our hill-climbing feature selection method *vis-a-vis* exhaustive method. Hill-climbing method performed reliably while consuming much less time.

Testing our feature selection method for clustering algorithms other than k-means is an ongoing work. But as shown by the experiments over data sets with known important features, it can be expected that our algorithm would perform equally well for other clustering algorithms. Another area to explore is subspace clustering (CLIQUE [2]) which is the task of finding dense regions in subspaces of features instead of whole space. It can help find interesting data hidden in subspaces of features where clusters may not be defined by all features. A problem encountered in CLIQUE concerns scalability with respect to number of features. Their experiments exhibited a quadratic behavior in the number of features. It may be interesting to check the effectiveness of our approach in reducing the dimensionality thereby making the search for subspace clusters more efficient.

References

1. C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 61–72, 1999.
2. R Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1998.
3. R Agrawal and R. Srikant. Fast algorithm for mining association rules. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.
4. P. S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of the 4th International Conference on Knowledge Discovery & Data Mining (KDD'98)*, pages 9–15, 1998.

5. C. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of Internationl Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999.
6. M. Dash and H. Liu. Feature selection for classification. *International Journal of Intelligent Data Analysis, http://www.elsevier.com/locate/ida*, 1(3), 1997.
7. J. L. Devore. *Probability and Statistics for Engineering and Sciences.* Duxbury Press, 4th edition, 1995.
8. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, chapter Unsupervised Learning and Clustering. John Wiley & Sons, 1973.
9. U. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinment clustering algorithms. In *Proceedings of the 4th International Conference on Knowledge Discovery & Data Mining (KDD'98)*, pages 194–198, 1998.
10. V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS - clustering categorical data using summaries. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999.
11. A. K. Jain and R. C. Dubes. *Algorithm for Clustering Data*, chapter Clustering Methods and Algorithms. Prentice-Hall Advanced Reference Series, 1988.
12. R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs.* PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 1995.
13. C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1996.