Feature Selection for Clustering – A Filter Solution

Manoranjan Dash Kiseok Choi Peter Scheuermann
Dept of Elect & Comp Eng
Northwestern University
Evanston, IL 60208
{manoranj,kchoi,peters}@ece.northwestern.edu

Huan Liu
Dept of Comp Sci & Eng
Arizona State University
Tempe, AZ 85287
hliu@asu.edu

Abstract

Processing applications with a large number of dimensions has been a challenge to the KDD community. Feature selection, an effective dimensionality reduction technique, is an essential pre-processing method to remove noisy features. In the literature there are only a few methods proposed for feature selection for clustering. And, almost all of those methods are 'wrapper' techniques that require a clustering algorithm to evaluate the candidate feature subsets. The wrapper approach is largely unsuitable in real-world applications due to its heavy reliance on clustering algorithms that require parameters such as number of clusters, and due to lack of suitable clustering criteria to evaluate clustering in different subspaces. In this paper we propose a 'filter' method that is independent of any clustering algorithm. The proposed method is based on the observation that data with clusters has very different point-to-point distance histogram than that of data without clusters. Using this we propose an entropy measure that is low if data has distinct clusters and high otherwise. The entropy measure is suitable for selecting the most important subset of features because it is invariant with number of dimensions, and is affected only by the quality of clustering. Extensive performance evaluation over synthetic, benchmark, and real datasets shows its effectiveness.

1 Introduction

Many real-world applications deal with high dimensional data. Feature selection that chooses the important original features is an effective dimensionality reduction technique. An important feature for a learning task can be defined as one whose removal degrades the learning accuracy. By removing the unimportant features, data sizes reduce, while learning accuracy and comprehensibility improve. Learning can be supervised or unsupervised. In supervised learning a class label is specified for each instance while unsupervised learning uses no class label. The literature for feature selection for classification, which is a supervised learning task, is very vast (see a review in [7]). On the other hand, there are only a few, mostly recent, feature selection methods for clustering, which is a unsupervised task. Arguably, the reason behind this gap in research is due to the fact that it is easier to select features for classification than for clustering.

Feature selection for clustering is the task of selecting important features for the underlying clusters. Among the methods proposed [8, 9, 11, 17, 19] most of them are 'wrapper' methods for feature selection. A wrapper method for feature selection evaluates the candidate feature subsets by the learning algorithm itself which uses the selected features later for efficient learning. The term wrapper was extensively used in [15] for feature selection for classification. In clustering, a wrapper method evaluates the candidate feature subsets by a clustering algorithm. For example in [8, 17] the K-means clustering algorithm is used for evaluation, while in [11] EM – Expectation Maximization – is used for evaluation. Although wrapper methods for classification have several disadvantages such as lack of robustness across different learning algorithms, they are still preferable in applications where accuracy is an important criterion. But, unlike classification which has a very quantifiable way of evaluating accuracy, there is no generally acceptable criterion to estimate the accuracy of clustering (see [14] for a partial list of clustering criteria). To make matters worse, feature selection for clustering requires that the evaluation functions be able to distinguish among clustering in different subspaces. The above limitations make wrapper methods for clustering disadvantageous.

In this paper we propose and evaluate a 'filter' method for feature selection. A filter method, by definition, is independent of clustering algorithms, and thus completely avoids the issue about lack of unanimity in the choice of clustering criterion. The proposed method is based on the observation that data with clusters has very different point-to-point distance histogram than data without clusters. Using this observation we propose an entropy measure that is low if data has distinct clusters and high otherwise. The entropy measure is suitable for selecting the most important subset of features because it is invariant with number of dimensions, and is affected only by the quality of clustering. Experiments over real, benchmark, and synthetic datasets show the effectiveness of the proposed method.

2 Properties of Feature Selection

Let us assume that our dataset consists of N data points or instances each with M dimensions or features. We shall denote X_i as the i^{th} data point, X_{i_k} as the k^{th} feature value of the i^{th} point, and F_k as the k^{th} feature for $i = 1...N$ and $k = 1...M$. Also $D_{i1,i2}$ denotes the distance between points X_{i1} and X_{i2}, and χ_j denotes the j^{th} cluster for $j = 1...c$ where c is the number of clusters. Below we discuss two important properties of unsupervised data that affect feature selection.

Importance of Features over Clustering Typically, while gathering information for a particular application, one tends to gather as much information as possible without considering the significance of each feature over the underlying clusters. It is an essential data mining pre-processing task to remove unwanted features before performing other KDD tasks such as clustering.

In Figure 1(a,b,c) we show an example using synthetic data in (3,2,1)-dimensional spaces respectively. There are a total of 75 points in three dimensions; there are three clusters in the $F1$-$F2$ dimensions with each cluster having 25 points. Values in $F1$ and $F2$ features follow Gaussian distribution within each of the three clusters while values in feature $F3$, that does not define any cluster, follow a uniform distribution. When we take all three features the clusters are unnecessarily complex (see Figure 1(a)), whereas no clusters can be found when we visualize using only one feature, say $F1$ (Figure 1(c)). Figure 1(b) with $F1$-$F2$ features shows three well-formed clusters. Selecting features $F1$ and $F2$ reduces the dimensionality of the data while forming well separated clusters. This is basically the goal of this paper, i.e., to select the important original features for clustering thus reducing the data size (and the computation time) and at the same time improving the knowledge discovery performance and comprehensibility.

In a single dimensional dataset clusters can be formed if the single feature takes values in separate ranges. In a multi-dimensional dataset, however, clusters can be formed from combination of feature values although the single features by themselves alone may not define clusters. Below we have noted down two distinct scenarios.

Scenario 1 *A single feature defines clusters independently.* Consider Figure 1(e) where feature $F2$ is uniformly distributed while $F1$ takes values in two separate ranges. It can be clearly seen that $F1$ is more important for defining the two clusters than $F2$.

Scenario 2 *Individual features do not define clusters but correlated features do.* Consider Figure 1(f) where features $F1$ and $F2$ are uniformly distributed. Note that $F1$ and $F2$ alone cannot define the clusters, but by their correlation they define two distinct clusters.

Any feature selection algorithm for clustering must take into account these two scenarios while selecting features.

Distance Histogram In Figure 2 we show the histogram of point-to-point distances for two datasets: one with clusters and the other without clusters. The histogram has b (=100) buckets. Distance are normalized in the interval $[0...1]$ and the bucket# is determined by multiplying the normalized value by b in order to select a bucket. A counter is maintained for each bucket by incrementing it each time a distance belongs to the particular bucket. In the histogram plots, each plotted point corresponds to a single bucket. X value of the plotted point is the bucket number and Y value is the frequency or the counter value of bucket X.

An important distinction between the two histograms is that histogram for data without clusters has a predictable shape similar to a bell shape (see Figure 2(b)). But the histogram for data with clusters has a very different distribution (see Figure 2(a) where the dataset has 20 clusters in 2-D). The early buckets (or distances) are mostly intra-cluster while the latter ones are mostly inter-cluster. *Typically, if the dataset consists of some clusters then, the majority of the intra-cluster distances will be smaller than the majority of the inter-cluster distances.* This observation is true for a wide range of data types. When clusters are very distinct intra-cluster and inter-cluster distances are quite distinguishable. In this paper we propose a method that, without doing clustering, can distinguish between data with clusters and data without clusters. In the next section we introduce such a method which is then applied in Section 4 to select a subset of important features.

3 Distance-based Entropy Measure and Its Efficient Calculation

As stated, we want to develop a method that can distinguish between data with clusters and data without clusters. Entropy theory says that entropy of a system measures the disorder in the system. Mathematically:

$$E = -\sum_{X_i} p(X_{i_1}, ..., X_{i_M}) \log p(X_{i_1}, ..., X_{i_M}) +$$

$$-\sum_{X_i} (1 - p(X_{i_1}, ..., X_{i_M})) \log(1 - p(X_{i_1}, ..., X_{i_M}))$$

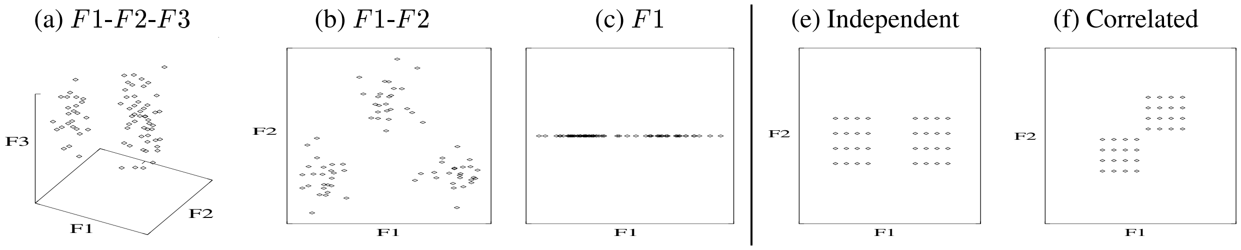

Figure 1. Effect of Features on Clustering

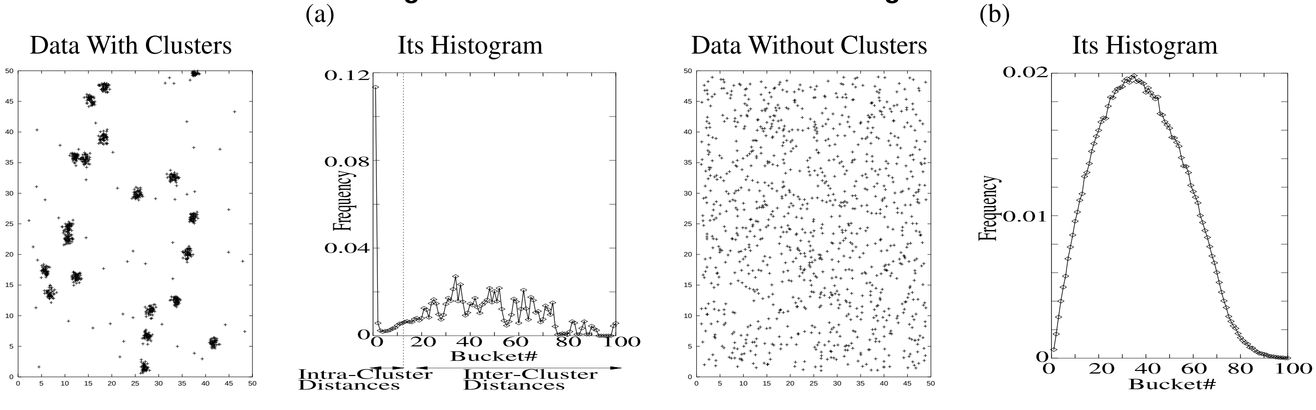

Figure 2. Distance histograms of data *with* and *without* clusters

where $p(X_{i_1}, ..., X_{i_M})$ is the probability or density at the point $(X_{i_1}, ..., X_{i_M})$. The second term in the expression inside summation is used to make the expression symmetric. If the probability of each point is equal we are most uncertain about the outcome, and entropy is the 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. So, the entropy can be used to distinguish between data with clusters and data without clusters.

But as we do not know the probability of points, we propose the following proxy method to estimate the entropy. *The* **goal** *of this method is to assign low entropy to intra- and inter-cluster distances, and to assign a higher entropy to noisy distances.* A straight-forward method is obtained by substituting probability with distance:

$$E = -\sum_{X_i}\sum_{X_j}[D_{ij}\log D_{ij} + (1 - D_{ij})\log(1 - D_{ij})] \quad (1)$$

where D_{ij} is the normalized distance [1] in the range [0.0 – 1.0] between instances X_i and X_j. E is normalized to the range [0.0 – 1.0]. Entropy is 0.0 for the minimum (0.0) or the maximum (1.0) distance and it is 1.0 for the mean distance (0.5). A plot showing entropy-distance relationship looks similar to the dotted curve of Figure 3(a). Although, to some extent, it works well in distinguishing data with clusters from data without clusters, considering its stated goal it suffers from the following two drawbacks. (a) The mean distance of 0.5, the meeting point (μ) of the two sides (i.e., left and right) of the plot, can be an inter-cluster distance, but still it is assigned the highest entropy. (b) Entropy increases rapidly for very small distances thus assigning very different entropy values for intra-cluster distances. In summary, this measure does not fulfill its stated goal of assigning small entropy for both intra- and inter-cluster distances. The second drawback can be easily overcome by incorporating a coefficient (β) in the equation and by using an exponential function in place of logarithmic function. Regarding the first drawback, we can set the meeting point (μ) so as to separate the intra-cluster and inter-cluster distances more accurately. Considering all these we propose the following method:

$$\mathbf{E} = \sum_{X_i}\sum_{X_j} E_{ij} \quad (2)$$

$$E_{ij} = \begin{cases} \frac{exp(\beta * D_{ij}) - exp(0)}{exp(\beta*\mu) - exp(0)} & : \quad 0 \le D_{ij} \le \mu \\ \frac{exp(\beta*(1.0 - D_{ij})) - exp(0)}{exp(\beta*(1.0-\mu)) - exp(0)} & : \quad \mu \le D_{ij} \le 1.0 \end{cases} \quad (3)$$

where E_{ij} is normalized to the range [0.0 – 1.0].

Setting β and μ Figure 3(a) shows that increasing the β value will decrease the entropy. An effective β is a pos-

[1] We use Euclidean measure although other distances such as Manhattan can be used.

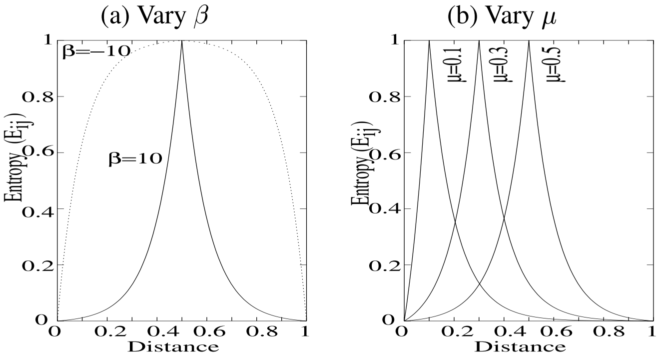

Figure 3. Relationship between entropy and distance with varying β and μ values

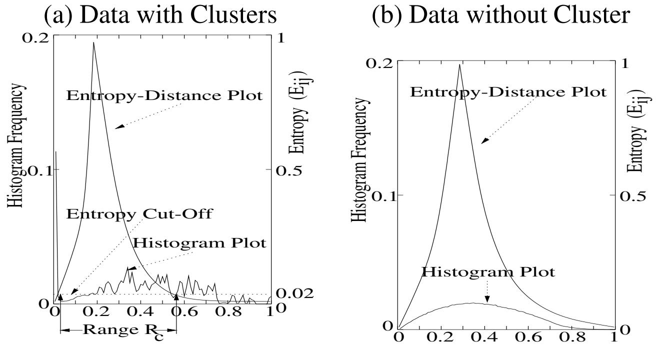

Figure 4. Illustrating the effectiveness of the proposed method

itive value (> 0) because a -ve β will lead to the second drawback. A very large +ve β will fail to distinguish between intra- and inter-cluster distances. A very small +ve β will not assign sufficiently small entropy to intra- and inter-cluster distances. Among different +ve β's that we experimented, it works well when it is set to a value around 10 (as shown in the Figure 3(a) bold curve.

Varying μ has the effect of shifting the meeting point of the two sides of the entropy – distance plot (see Figure 3(b)). The value of μ affects the total entropy. According to the histogram plots of Figure 2, it is easy to see that if the μ is set properly then we can distinguish between data with and data without clusters. The method we propose here is based on the observation that it is easy and more accurate to estimate the range of intra-cluster than inter-cluster distances. This is due to the fact that intra-cluster distances typically occupy the lowest portion of the complete range of distances. Below we describe an easy and robust estimation of μ using the intra-cluster distance range (R_I).

1. Starting with the first bin of the histogram, reject all bins of frequencies less than a very low frequency Q_{Min} until a bin with higher frequency occurs.

2. Starting from the bin with frequency more than Q_{Min} find the bin (B_I) with highest frequency in range R_I.

3. Calculate the μ value by setting other unknowns in the Equation 3 for the range $0 \leq D_{ij} \leq \mu$ as follows: set E to a small value E_T, set β, and set distance D_{ij} corresponding to the bin B_I.

Step 1 This step is useful when dimensionality (M) is very large. As shown in [4], for data with large M, if the points are uniformly distributed then all distances approach the maximum distance, i.e., the histogram plot will be effectively shifted towards the right forcing the frequencies of the lower bins to be very small. So, to make a more effective estimation of μ we reject these initial bins with frequency less than a very small frequency Q_{Min}. In the experiments we set Q_{Min} to 0.5% of the total frequency 1.

Step 2 We tested for various R_Is. It is found that the results remain insensitive if it is set to the initial 5% to 30% of the over-all distance range. The reason is if the data has clusters then the intra-cluster distances are very small and the maximum intra-cluster frequency will typically occur for a very small distance.

Step 3 E_T is set to 2% of the maximum entropy 1, i.e. 0.02. Setting of β is discussed earlier in this section.

We explain the above procedure pictorially in Figure 4. Datasets with and without cluster are taken from Figure 2. Figure 4(a) superimposes the entropy-distance relationship plot over histogram plot for the data with clusters (do not be distracted by the entropy cut-off line and label "Range R_c" in the figure whose significances are explained in the next sub-section). For super-imposition, the bucket numbers of the histogram plot are converted to a range [0.0 – 1.0]. The step 1 is not required for this low-dimensional data. The step 2 yields bin B_I as 1 corresponding to the maximum frequency in the range R_I. Finally step 3 calculates μ as 0.185. Using this information the total entropy (E) for the data with clusters is calculated to be 93433.9. Similarly Figure 2(b) shows the estimation of μ for data without clusters. Using this information E is calculated to be 174723.0 which is much larger than 93433.9. A notable difference between the two figures is: in case of data with clusters entropy is low for many bins (or distances) having considerably high frequency counts (signifying intra- and inter-cluster distances), but for data without clusters entropy is large for most of the bins with considerably high frequency counts. *The proposed method is able to exploit the shape of the histogram plots to assign a low entropy for data with clusters and a high entropy otherwise.*

A Fast Method Number of distance calculations required to calculate the entropy for a subset of features is $O(\frac{N^2-N}{2})$ where N is the number of points and computation required to calculate the distance between a pair of points is taken as unit. Obviously quadratic run times are impractical for large datasets.

A much faster method to calculate entropy is described here. The basic idea is as follows. Until now, in this and the previous sections, a major concern was how to assign low entropy to intra-cluster and inter-cluster distances. In other words, *this approach tries to minimize the entropy for data with clusters.* So, typically a large portion of distances have very low entropy whose total make a less than significant contribution to the over-all entropy. By considering only those distances having entropy higher than a threshold entropy E_{thres} we are able to find a range of distance R_C with higher entropy than E_{thres}, that spreads equally on both sides of the meeting point μ. Entropy for any distance outside this range can be set to a very low constant value C_E less than or equal to E_{thres}. Figure 4(a) illustrates the above observation. A range of distance R_C is shown having entropy above a minimum threshold E_{thres} which is set to 2% of the maximum entropy 1. Note that only 62% of the distances fall in the range R_C, i.e., approximately 38% entropy calculations can be avoided. The total entropy is little affected if, for distances outside R_C, we replace all entropy calculations by a constant entropy (C_E) value less than or equal to E_{thres}.

To exploit this observation, we propose an algorithm based on grid-blocks where data space is partitioned into equal size grid-blocks by dividing each dimension using axis-parallel partitions.

1. For data size that can fit into the main memory we read the data into main memory and separate them into grid-blocks. For large data residing in disk we create an index such that data can be read by grid-blocks.
2. For each pair of blocks, minimum distance is calculated and stored either in memory or disk depending on its size.
3. For each point whose entropy (or distance) is being calculated, its block is determined. For each of the remaining blocks, minimum distance from this block is obtained. If this distance falls in the range R_C then entropy between the candidate point and all points in the block are calculated in the usual way. Otherwise, for all points in the block entropy is pre-assigned to C_E.

For very high-dimensional data where most grid-blocks will be sparse or empty, we consider only those blocks that have considerable number of data points. This way the number of grid-blocks will not grow in an exponential manner which has been a curse for high-dimensional partitioning. For uniformly distributed data, most grid blocks may be sparse. We decide to go for grid block computation if total number of sparse blocks contain less than a threshold number of points, otherwise a full computation is done.

For very large datasets where even the above fast method is prohibitive, we use sampling. Sampling works well because the entropy measure, to work well, requires the underlying cluster structure to be retained, which a sample is particularly good at.

4 Feature Selection Algorithm

In the previous section we discussed how to distinguish between data with and data without clusters by using an entropy measure. In this section we propose a feature selection algorithm based on this measure. Feature selection process has two main steps: search or generation and evaluation of subsets of features. The entropy measure can be directly used as an evaluation technique to compare subsets of features. This is possible because it is independent of the cardinality of the subsets and comparison is made on how well the subspaces define clusters. So, irrespective of the cardinalities of subsets, low entropy is output for subset defining well-formed clusters while high entropy is output otherwise. Regarding the other important step of feature selection, namely search method, efficiency is measured by *optimality*. In the present context, optimality is defined as that subset for which the entropy is minimum. In the literature, particularly for classification, many search techniques are proposed. See [7] for a list of these techniques. Prominent among these methods are exhaustive, heuristic, random, or some hybrid of these techniques. Exhaustive methods guarantee optimality but are impractical due to their exponential complexity in number of features M, i.e., $O(2^M)$. Random methods that generate subsets randomly are anytime algorithm in that they can return the best subset at any point of time, and moreover they asymptotically approach the optimal subset. A variation of pure random methods is probabilistic method where the probability of generating a subset varies by some rules. Examples of such rules are genetic algorithm and simulated annealing. Commonly used heuristic methods for feature selection are forward or backward selection or some combination of both. A forward selection method first finds the best feature among all features, and then using the already selected features finds the next best feature, and so on. The subset that outputs the least entropy is output as the best subset. Backward selection algorithm is the opposite of the forward selection algorithm. There are many other search techniques that can be applied to feature selection. As our goal in this paper is to propose an evaluation method that can correctly compare data with and data without clusters, we do not go into the details about these search methods. In the experiments we use a forward selection algorithm (ForwardSelect) and compare its output with the exhaustive method. A forward selection algorithm has two loops: the outer loop iterates M – total number of

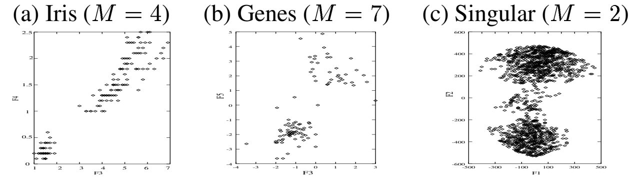

Figure 5. Subspace showing clusters

features – times, and selects the overall best subset of features while the inner loop selects the best feature for each iteration of the outer loop.

5 Experimental Evaluation

Evaluation of a unsupervised learning task such as clustering is arguably more difficult than a supervised task such as classification for the simple reason that in clustering there is no commonly accepted evaluation approach unlike classification where accuracy of the classifier is a commonly accepted evaluation measure. Feature selection for clustering has the additional disadvantage that the clusters depend on the dimensionality of the selected features and that any given feature subset may have its own clusters, which may well be incompatible with those formed based on different feature subsets. Considering these aspects, the best way of evaluating a feature selection method for clustering is to check the correctness of the selected features, i.e., how well the selected features match with the actual important features. According to this evaluation criterion, we first evaluate the proposed method over synthetic datasets for which we know the important features. Next we evaluate over benchmark and real datasets for which important features are known or can be found out by visualization. We run forward selection method and validate its result by running exhaustive search method to check if ForwardSelect selected the optimal subset that has the overall minimum entropy. If not stated otherwise, the results of the forward selection matches that of exhaustive method, i.e., the forward selection method outputs the optimal subset. Next, we compare a wrapper method with the proposed filter method. In all experiments we follow the guidelines of Section 3 on how to set the two parameters β and μ.

5.1 Synthetic Datasets

Synthetic datasets are generated by using a simple data generation algorithm which can be run from the web-site http://i2s.kisti.re.kr/c̃hoi/choi/proj/gen-data.html. Clusters can be defined by two distributions: Gaussian and Uniform. If k features define a Gaussian cluster, then each feature will have a mean and a standard deviation randomly chosen

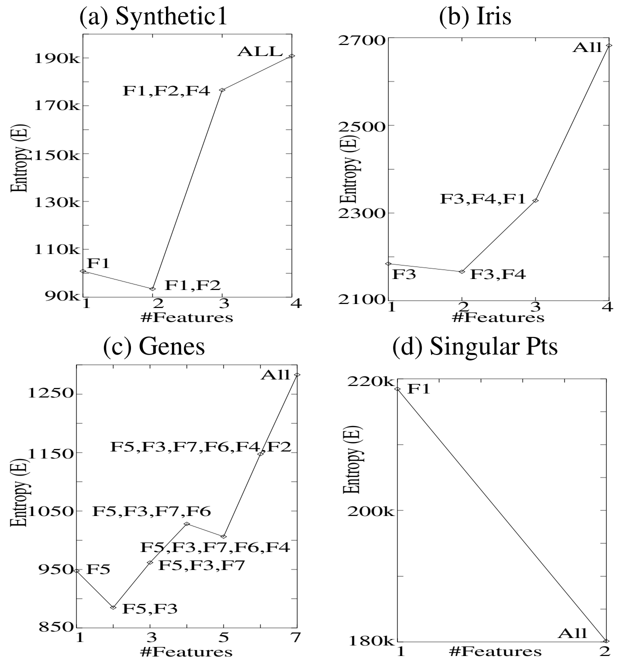

Figure 6. Results of ForwardSelect

from a given range. Similarly, for a uniformly distributed cluster, the lower and the higher ranges are randomly chosen from the range. A noisy point will have all of its features uniformly distributed. The above information is kept in a configuration file that also contains information about the number of clusters, the number of points per each cluster, and the number of noisy points.

An Example The first synthetic dataset (Synthetic1) has 4 features and 1050 points in 20 clusters with 5% noise. Features {F1,F2} define the clusters while {F3,F4} are noise. This is taken from Figure 2(a) after adding {F3,F4}. Figure 6(a) shows that {F1,F2} has the minimum entropy. When we re-checked using exhaustive search, the entropy is also minimum for {F1,F2}.

Other Datasets Other datasets with varying number of features and clusters are generated similarly. Experiments are conducted for M = 4, 10, 20, and 100 features in 2, 5, 10, 20 and 50 clusters. First half of the features i.e. $\{F1,...,F_{\frac{M}{2}}\}$ are important whereas the last half, i.e. $\{F_{\frac{M}{2}+1},...,F_M\}$, are noisy features. For example, for data with 4 features, {F1,F2} is the best subset. Each data has 5-10% noisy points. Out of these 20 (4×5) datasets, except for 10 features – 10 clusters, 10 features – 50 clusters, and 20 features – 20 clusters datasets, the results are correct for the remaining 17 datasets, i.e. subset $\{F1,...,F_{\frac{M}{2}}\}$ has the lowest entropy. The results for these three datasets are almost correct as well as the selected features are all important and it missed out only one important feature.

5.2 Benchmark and Real Datasets

Iris dataset, popularly used for testing clustering and classification algorithms, is taken from UCI ML repository [5]. It contains 3 classes of 50 instances each, where each class refers to a type of Iris plant. One class is linearly separable from the other two; the latter are not linearly separable from each other (see Figure 5(a)). Out of the four it is known that F3 (petal length) and F4 (petal width) features are more important for the underlying clusters because of their very high correlation with the three classes. Before performing experiments we removed the class labels. In Figure 6(b) we plot the entropy values for the subsets of features selected by ForwardSelect. For the subset consisting of the two most important features ($\{$F3,F4$\}$) the entropy is minimum and is correctly selected by the algorithm.

Genes dataset is taken from the recently publicized clustering software CLUTO available from the web site http://www-users.cs.umn.edu/k̃arypis/cluto/. In this, there is a dataset called Genes2 which has 99 yeast genes (or data points) described using 7 profiles (or features). When ForwardSelect is run over this data, it shows the minimum entropy for subset $\{$F3,F5$\}$ (see Figure 6(c)). When we checked the data, as shown in Figure 5(b), it showed two clusters in this two-dimensional subspace. Exhaustive search also returned this subset as the best. With the help of domain experts, we are in the process of finding the significance of these clusters in this subspace. *This experiment shows that the proposed method can be used to uncover clusters from datasets with previously unknown clusters hidden in subspaces of features.*

Singular Points of Parallel Manipulator We obtained data of an experiment conducted in a robotics laboratory to test the singularity points of a parallel manipulator. A parallel manipulator is a closed-loop mechanism in which a moving platform is connected to the base by at least two serial kinematics chains (legs). Singularity is a point inside the workspace where the manipulator moves in a random direction even if all joints are locked. In order to avoid it one approach is to cluster them and find paths in the empty space among the clusters. The data, shown in Figure 5(c), is a good example of *scenario 2* (Section 2) where clusters are defined by the correlation of features and not by individual features. The result shows that the entropy for the subset consisting of both features is smaller than the entropy for any other subset (see Figure 6(d)).

5.3 Comparison with Wrapper Method

We compared the results with a wrapper method described in [11]. Data points are first clustered using a partitional clustering algorithm EM – Expectation Maximization – and then evaluated by using an invariant criterion *trace*. The $trace(S_w^{-1} S_b)$ is the chosen criterion because it is invariant under any non-singular linear transformation [10]. S_w ($\sum_{j=1}^{c} \sum_{X_i \in \chi_j} (X_i - m_j)(X_i - m_j)^t$) is the within-cluster scatter matrix and S_b ($\sum_{j=1}^{c} (m_j - m)(m_j - m)^t$) is the between cluster scatter matrix where χ_j is the j^{th} cluster, m is the total mean vector of the data and m_j is the mean vector for j^{th} cluster, $(X_i - m_j)^t$ is the matrix transpose of the column vector $(X_i - m_j)$, and *trace* of a matrix is the sum of its diagonal elements. The larger the $trace(S_w^{-1} S_b)$, the larger the normalized distance between clusters which results in better cluster discrimination. In order to test whether the wrapper method is able to select features correctly, we evaluate subsets exhaustively.

For all datasets we set the number of clusters correctly to run EM. Results show that (1) for the example dataset with clusters, (shown in Figure 2), the maximum trace occurs for the subset $\{$F1$\}$ while the trace for the important subset $\{$F1,F2$\}$ is much lower than that for $\{$F1$\}$, (2) for the Iris dataset the maximum trace occurs for the subset $\{$F2,F3,F4$\}$ which is higher than the actual important subset $\{$F3,F4$\}$, (3) for the Genes dataset the maximum trace occurs for the subset $\{$F1,F2$\}$ (visualization shows no clusters in this subspace) which is much higher than the trace for the important subset $\{$F3,F5$\}$, and (4) for the singular points the maximum trace occurs for the subset $\{$F2$\}$ although the clusters are most distinct in $\{$F1,F2$\}$.

Sensitivity of Parameters Among the parameters of the proposed algorithm, μ affects the outcome more directly than β. Setting of μ depends on finding the maximum intra-cluster frequency which in turn depends on the setting of the higher range of intra-cluster distances. In the experiments we set it to the smallest 10% of the total range of distances. For all experiments we varied it from 5% to 30% and still the results are same as 10%. But note that by varying the number of clusters even very slightly for wrapper methods, the results change drastically. We can explain the robustness of the proposed method as follows. If a subspace contains some clusters then, typically, majority of the intra-cluster distances will be smaller than majority of the inter-cluster distances. So setting any value less than 30% is robust enough because the maximum frequency found are the same most of the time.

6 Related Work and Conclusion

Earliest methods for reducing dimensionality for unsupervised learning are feature extraction methods such as Principal Components Analysis, Karhunen-Loeve transformation, or Singular Value Decomposition [10, 13]. These methods do not reduce the number of the original features, instead they create extracted features or principal components from the original ones. In the last several years a number of methods for feature selection for clustering are proposed most of which are 'wrapper' in approach. In clustering, a wrapper method uses a clustering algorithm to evaluate the candidate feature subsets. Wrapper methods can be categorized based on whether they select features for the whole data (*global type*) or just for a fraction of the data in a cluster (*local type*). The global type assumes a subset of features to be more important than others for the whole data while the local type assumes each cluster to have a subset of important features. Examples of global methods are [8, 9, 11, 17, 19, 20] and the proposed method in this paper. The method described in [17] uses K-means for evaluation of subsets of features. In [11] EM (Expectation–Maximization) and trace measure are used for evaluation. The authors also propose visual aids for the user to decide the optimal number of features. In [9, 18] features are ranked and selected for categorical data. Forward and backward search techniques are used to generate candidate subsets. To evaluate each candidate subset, these methods measure the category utility of the clusters by applying COBWEB [12]. [2] In [20] authors proposed an objective function for choosing the feature subset and finding the optimal number of clusters for a document clustering problem using a Bayesian statistical estimation framework.

Examples of local wrapper methods are [1, 2, 6]. Projected clustering (ProClus [1]) finds subsets of features defining (or important for) each cluster. ProClus first finds clusters using K-medoid [16] considering all features and then finds the most important features for each cluster using Manhattan distance. The algorithm called CLIQUE in [2] divides each dimension into a user given divisions. It starts with finding dense regions (or clusters) in 1-dimensional data and works upward to find j-dimensional dense regions using candidate generation algorithm Apriori [3].

In this paper we proposed a filter method to evaluate feature subsets and choose the best subset for clustering by considering their effect on the underlying clusters. Earlier methods proposed for clustering were mostly wrapper methods which require some clustering algorithm and some invariant clustering criterion to evaluate feature subsets. A main drawback of this approach is the lack of unanimous agreement in evaluating the clusters. Furthermore, running a clustering algorithm is very sensitive on some parameters such as the number of clusters or some equivalent of it. For real-world data this information is usually hard to obtain, making it unusable in most cases. But in contrast the proposed method largely depends on a parameter (range of intra-cluster distance) which is easier to set because the proposed method is quite insensitive to it. We performed experiments over benchmark, synthetic and real datasets and results show that the proposed method correctly finds the most important subsets. Comparison with a wrapper method showed the superior evaluation accuracy of the proposed method. Without performing clustering, the proposed method can discover clusters in subspaces even if no *a priori* information about such clusters is available. This is evident in the experimental study of the Genes dataset.

[2] COBWEB is a hierarchical clustering algorithm for categorical data.

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 20th International Conference on Very Large Databases (VLDB)*, Santiago, Chile, 1994.

[4] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the International Conference on Database Theory (ICDT)*, pages 217–235, 1999.

[5] C. Blake and C. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[6] C. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*, 1999.

[7] M. Dash and H. Liu. Feature selection for classification. *International Journal of Intelligent Data Analysis*, 1(3), 1997.

[8] M. Dash and H. Liu. Feature selection for clustering. In *Proceedings of Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining, (PAKDD)*, 2000.

[9] M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 92–97, 1997.

[10] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, chapter Unsupervised Learning and Clustering. John Wiley & Sons, 1973.

[11] J. G. Dy and C. E. Brodley. Visualization and interactive feature selection for unsupervised data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 360–364, 2000.

[12] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[13] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[14] A. K. Jain and R. C. Dubes. *Algorithm for Clustering Data*, chapter Clustering Methods and Algorithms. Prentice-Hall Advanced Reference Series, 1988.

[15] G. H. John. *Enhancements to the data mining process*. PhD thesis, Department of Computer Science, Stanford University, March 1997.

[16] L. Kaufman and P. Rousseuw. *Finding Groups in Data - An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics, 1990.

[17] Y. S. Kim, W. N. Street, and F. Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proceedings of ACM SIGKDD International Conference on Knowledge and Discovery*, pages 365–369, 2000.

[18] L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of International Conference on Machine Learning (ICML)*, 1999.

[19] L. Talavera. Feature selection and incremental learning of probabilistic concept hierarchies. In *Proceedings of International Conference on Machine Learning (ICML)*, 2000.

[20] S. Vaithyanathan and B. Dom. Model selection in unsupervised learning with applications to document clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 433–443, 1999.