

An Algorithm for Non-distance Based Clustering in High Dimensional Spaces

Shenghuo Zhu
zsh@cs.rochester.edu

Tao Li
taoli@cs.rochester.edu

Technical Report 763

January 2002

UNIVERSITY OF

ROCHESTER

COMPUTER SCIENCE

An Algorithm for Non-distance Based Clustering in High Dimensional Spaces

Shenghuo Zhu
zsh@cs.rochester.edu

Tao Li
taoli@cs.rochester.edu

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 763

January 2002

Abstract

The clustering problem has been widely studied since it arises in many application domains in engineering, business and social science. It aims at identifying the distribution of patterns and intrinsic correlations in large data sets by partitioning the data points into similarity clusters. Traditional clustering algorithms use distance functions to measure similarity and are not suitable for high dimensional spaces. In this paper, we propose a non-distance based clustering algorithm for high dimensional spaces. Based on the maximum likelihood principle, the algorithm is to optimize parameters to maximize the likelihood between data points and the model generated by the parameters. Experimental results on both synthetic data sets and a real data set show the efficiency and effectiveness of the algorithm.

KEYWORD: clustering, non-distance based, high dimensional space, maximum likelihood

1 Introduction

Clustering problems arise in many disciplines including engineering, business and social science, and have a wide range of applications, such as data compression, information retrieval, pattern recognition, trend analysis, customer segmentation and classification. The problem of clustering has been studied extensively in the database [Zhang *et al.*, 1996; Guha *et al.*, 1998; Ng and Han, 1994; Ester *et al.*, 1995a; Ester *et al.*, 1995c; Ester *et al.*, 1995b; Li *et al.*, 2001], statistics [Brito *et al.*, 1997; Berger and Rigoutsos, 1991; Duda and Hart, 1973; Dubes and Jain, 1980; Lee, 1981; Murtagh, 1983] and machine learning communities [Cheeseman *et al.*, 1988; Fisher, 1987; Fisher, 1995; Lebowitz, 1987; Liu *et al.*, 2000] with different approaches and different focuses.

The clustering problem can be described as follows: let W be a set of n multi-dimensional data points, we want to find a partition of W into clusters such that the points within each cluster are “similar” to each other. Various distance functions have been widely used to define the measure of similarity.

Most clustering algorithms do not work efficiently in high dimensional spaces due to the *curse of dimensionality*. It has been shown that in a high dimensional space, the distance between every pair of points is almost the same for a wide variety of data distributions and distance functions [Beyer *et al.*, 1999]. Many feature selection techniques have been applied to reduce the dimensionality of the space [Kohavi and Sommerfield, 1995]. However, as demonstrated in [Aggarwal *et al.*, 1999], the correlations in the dimensions are often specific to data locality; in other words, some data points are correlated with a given set of features and others are correlated with respect to different features.

As pointed out in [Hastie *et al.*, 2001], all methods that overcome the dimensionality problems have an associated and often implicit or adaptive-metric for measuring neighborhoods. Our algorithm, as you will see in the following sections, is based on an adaptive metric.

In this paper, we present a non-distance based algorithm for clustering in high dimensional spaces. The main idea of the algorithm is as follows: given a data set W and the feature set S , we want to cluster the data set into K clusters. Based on the maximum likelihood principle, the algorithm is to optimize parameters to maximize the likelihood between the data set W and the model generated by the parameters. The parameters in the model are the data map D and the feature map F , which are functions from W and S to $\{0, C_1, C_2, \dots, C_K\}$ ¹, respectively. Then several approximation methods are applied to iteratively optimize D and F . Our algorithm can also be easily adapted to estimate the number of clusters instead of using K as an input parameter. In addition, interpretable descriptions of the resulting clusters can be generated by the algorithm since it produces an explicit feature map.

The rest of the paper is organized as follows: section 2 introduces the core idea and presents the details of the algorithm; section 3 shows our experimental results on both the synthetic data sets and a real data set; section 4 surveys the related work; finally our conclusions and directions for future research are presented in section 5.

¹The number 0 represents the outlier set. C_1, \dots, C_K represent K different clusters.

2 Clustering Algorithm

In this section, we introduce the core idea and present the details of the algorithm. We first introduce the algorithm for binary data sets. Later on we will show how to extend the algorithm to continuous or non-binary categorical data sets in section 2.6.

2.1 The Model

Consider the zoo data set, where most animals are in the cluster of “chicken”, “crow” and “dove”, have “two legs”. Intuitively, when given an animal with “two legs”, we would say it has a large chance of being in the cluster. Therefore, we regard the feature “two legs” as a *positive* (characteristic) feature of the cluster.

Based on the above observation, we want to find out whether each feature is a positive feature of some cluster or not. *Feature map* F is defined as a function from feature set \mathcal{S} to $\{0, C_1, \dots, C_K\}$. $F(j) = k (k > 0)$ if feature j is a positive feature of cluster k . $F(j) = 0$ if feature j is not a positive feature of any cluster, *i.e.*, an outlier feature. Similarly, *data map* D is a function from data set W to $\{0, C_1, \dots, C_K\}$. $D(i) = C_k (k > 0)$ if data point i is an instance of cluster k . $D(i) = 0$ if data point i is an outlier.

Let N be the total number of data points, and d be the number of features. W can be represented as a data-feature matrix. By assuming all features are binary, if W_{ij} is 1, feature j is said *active* in data point i . Also, data point i is said to be an *active* data point of feature j if W_{ij} is 1.

The motivating idea behind the algorithm is maximum likelihood principle, that is, to find data map \hat{D} and its correspondent feature map \hat{F} from which the data-feature matrix W is most likely generated. Let $\hat{W}(D, F)$ be the model generated from the data map D and the feature map F . The values of \hat{W}_{ij} is interpreted as the consistence of $D(i)$ and $F(j)$. We only consider the cases that $D(i)$ is consistent with $F(j)$, inconsistent with $F(j)$, or an outlier. $P(W_{ij}|\hat{W}_{ij}(D, F))$ is the probability whether the feature j of data point i is active in the real data, given the feature j of data point i is active (or not) in the ideal model of maps D and F . $P(W_{ij}|\hat{W}_{ij}(D, F))$ has the same value for different i 's and j 's if W_{ij} and $\hat{W}_{ij}(D, F)$ have the same values, denoted by w and $\hat{w}(D, F)$ respectively. Therefore, we count the number of instances of w and $\hat{w}(D, F)$, which is $dNP(w, \hat{w}(D, F))$. Now we have Eq. (1).

$$\begin{aligned} \log L(D, F) &= \log \prod_{i,j} P(W_{ij}|\hat{W}_{ij}(D, F)) \\ &= \log \prod_{w,\hat{w}} P(w|\hat{w}(D, F))^{dNP(w,\hat{w}(D, F))} \\ &= dN \sum_{w,\hat{w}} P(w, \hat{w}(D, F)) \log P(w|\hat{w}(D, F)) \\ &\equiv -dNH(W|\hat{W}(D, F)) \end{aligned} \tag{1}$$

$$\hat{D}, \hat{F} = \arg \max_{D, F} \log L(D, F) \tag{2}$$

We apply the hill-climbing method to maximize $\log L(D, F)$, *i.e.*, alternatively optimizing one of D and F by fixing the other. First, we try to optimize F by fixing D . The problem of optimizing F over all data-feature pairs can be approximately decomposed into subproblems of optimizing each $F(j)$ over all data-feature pairs of feature j , *i.e.*, minimizing the conditional entropy $H(W_{\cdot j}|\hat{W}_{\cdot j}(D, F))$. If D and $F(j)$ are given, the entropies can be directly estimated. Therefore, $F(j)$ is assigned to the cluster, given which the conditional entropy of $W_{\cdot j}$ is minimum. To optimize D by fixing F is a dual problem of the problem above. Hence, we only need minimize $H(W_i|\hat{W}_i(D, F))$ for each i . A straightforward approximation method to minimize $H(W_{\cdot j}|\hat{W}_{\cdot j}(D, F))$ is to assign $F(j)$ to $\arg \max_k |\{i|W_{ij} = k\}|$. This method also applies to minimize $H(W_i|\hat{W}_i(D, F))$ by assigning $D(i)$ to $\arg \max_k |\{j|W_{ij} = k\}|$.

A two-layer neural network can be used to implement the approximation algorithm. The lower level of the network consists of data nodes, each of which represents a data point. The upper level of the network consists of feature nodes, each of which represents a feature. The connection between these two levels are the data-feature matrix, W . Figure 1 depicts the structure of the network.

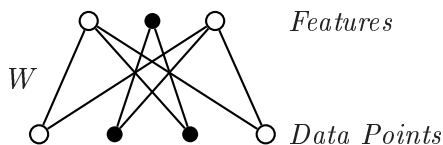


Figure 1: Conceptual View of the Network Structure

Initially each data node is randomly assigned to a class. In figure 1, different classes are colored in different colors. Each feature node is assigned to a class according to the classes of lower-level nodes that are connected to it. In our experiments, we use *winner-take-all* method to assign classes to feature nodes, *i.e.*, the class with most data nodes that connected to the feature node is assigned to it. Another promising method is to pick a class with a probability proportional to the number of data nodes assigned to the class. We call this procedure as *upward updating*. Because of the dual property of the problem, each data node can also be assigned to a class according to the classes of feature nodes that connected to it. The same method mentioned above can be used. This procedure is named as *downward updating*. The upward and downward updating procedures run alternatively until no class assignment is different from the previous one.

2.2 Algorithm Description

There are two auxiliary procedures in the algorithm: EstimateFeatureMap and EstimateDataMap. EstimateFeatureMap is to estimate the feature map from the data map. For each feature j , the procedure is to find a cluster k which minimizes conditional entropy $H(W_{\cdot j}|\hat{W}_{\cdot j}(D, F))$. If the feature is almost equally active in more than one cluster, the feature is said to be an outlier feature. EstimateDataMap is to estimate the data map from the feature map. It is a dual procedure of EstimateDataMap. For each data point i , the procedure is to find a cluster k which minimizes conditional entropy $H(W_i|\hat{W}_i(D, F))$.

The clustering algorithm proceeds in two steps. The first step is to select seed data points. The algorithm randomly draws nK distinct data points from the set of data points, and assigns each n of them to a cluster, where n is a small integer number, for example, 5. The algorithm estimates the feature map from the seed points by applying procedure EstimateFeatureMap.

The second step is an iterative routine. The aim is to find a best cluster by an iterative process similar to EM algorithm. In this step, the algorithm iteratively estimates the data and feature maps based on previous estimations, until no more change occurs in the feature map. The pseudo code of the algorithm is shown in Figure 2.

However, computing entropies is still time consuming. Here, we propose an approximating version of EstimateFeatureMap and EstimateDataMap (Figure 3). In EstimateFeatureMap, we calculate the cardinality of the set $\{i|D(i) = C \wedge W_{ij} = 1\}$, *i.e.*, the number of data points in cluster C whose j -th feature is active, and the cardinality of the set $\{i|W_{ij} = 1\}$, *i.e.*, the number of data points whose j -th feature is active, to approximate the conditional entropies. A similar method is also applied to EstimateDataMap.

It can be observed from the pseudo-code description that the time complexities of both procedures EstimateFeatureMap and EstimateDataMap are $O(K \times N \times d)$. The number of iterations in algorithm Cluster is not related to N or d .

2.3 An Example

To illustrate the algorithm, an example is given below. Suppose we have a data set as Table 1. Initially, data points 2 and 5 are chosen as seed points. Say, data point 2 is in cluster A , data point 5 in cluster B , *i.e.*, $D(2) = A$, $D(5) = B$. EstimateFeatureMap returns that features a , b and c are positive in cluster A ; features e and f are positive in cluster B ; features d and g are outliers. That is, $F(a) = F(b) = F(c) = A$, $F(e) = F(f) = B$, $F(d) = F(g) = 0$. After applying EstimateDataMap, data points 1, 2 and 3 are assigned to cluster A ; data points 4, 5 and 6 are assigned to cluster B . After applying EstimateFeatureMap, features a , b and c are positive in cluster A ; features d , e and f are positive in cluster B ; feature g is an outlier. After the next iteration, the result does not change. Therefore, we have a clustering result that cluster A contains data points 1, 2, and 3 and cluster B contains data points 4, 5, and 6.

feature data point	a	b	c	d	e	f	g
1	1	1	0	0	1	0	0
2	1	1	1	1	0	0	1
3	1	0	1	0	0	0	0
4	0	1	0	0	1	1	0
5	0	0	0	1	1	1	1
6	0	0	0	1	0	1	0

Table 1: An example of data set

```

Procedure EstimateFeatureMap(data points:  $W$ , data map:  $D$ )
begin
  for  $j$  in  $1 \cdots d$  do
    if  $\min_C H(W_{\cdot j} | \hat{W}_{\cdot j}(D, C)) / H(W_{\cdot j}) \ll 1$  then
       $F(j) = \arg \min_C H(W_{\cdot j} | \hat{W}_{\cdot j}(D, C));$ 
    else
       $F(j) = \text{outlier};$ 
    endif
  end
  return  $F$ 
end

```

```

Procedure EstimateDataMap(data points:  $W$ , feature map:  $F$ )
begin
  for  $i$  in  $1 \cdots N$  do
    if  $\min_C H(W_{i \cdot} | \hat{W}_{i \cdot}(C, F)) / H(W_{i \cdot}) \ll 1$  then
       $D(i) = \arg \min_C H(W_{i \cdot} | \hat{W}_{i \cdot}(C, F));$ 
    else
       $D(i) = \text{outlier};$ 
    endif
  end
  return  $D;$ 
end

```

```

Algorithm Cluster(data points:  $W$ , the number of clusters:  $K$ )
begin
  let  $W1$  be the set of randomly chosen  $nK$ 
  distinct data points from  $W$ ;
  assign each  $n$  of them to one cluster,
  say the map be  $D1$ ;
  assign EstimateFeatureMap( $W1, D1$ ) to  $F$ ;
  repeat
    assign  $F$  to  $F1$ ;
    assign EstimateDataMap( $W, F$ ) to  $D$ ;
    assign EstimateFeatureMap( $W, D$ ) to  $F$ ;
  until conditional entropy  $H(F|F1)$  is zeros;
  return  $D$ ;
end

```

Figure 2: Clustering algorithm

Procedure EstimateFeatureMap(data points: W , data map: D)
begin
 for j **in** $1 \cdots d$ **do**
 if $\max_C \frac{|\{i|D(i)=C \wedge W_{ij}=1\}|}{|\{i|W_{ij}=1\}|} \gg 1/K$ **then**
 $F(j) = \arg \max_C \frac{|\{i|D(i)=C \wedge W_{ij}=1\}|}{|\{i|W_{ij}=1\}|}$;
 else
 $F(j) = \text{outlier}$;
 endif
 end
 return F
end

Procedure EstimateDataMap(data points: W , feature map: F)
 $\{TC_{outlier} : \text{If no outliers, } TC_{outlier} = 0. \text{ Otherwise, } 1/K. \}$
begin
 for i **in** $1 \cdots N$ **do**
 if $\max_C \frac{|\{j|F(j)=C \wedge W_{ij}=1\}|}{|\{j|W_{ij}=1\}|} \gg TC_{outlier}$ **then**
 $D(j) = \arg \max_C \frac{|\{j|F(j)=C \wedge W_{ij}=1\}|}{|\{j|W_{ij}=1\}|}$;
 else
 $D(i) = \text{outlier}$;
 endif
 end
 return D ;
end

Figure 3: Approximation algorithm

2.4 Refining

The clustering results are sensitive to the initial seed points. Randomly chosen seed points may result trapping into local minima. We present a refining method (Figure 4). The idea is to use conditional entropy to measure the similarity between a pair of clustering results. The algorithm is to find a best clustering result which has smallest average conditional entropy to all others.

Algorithm Refine(data points: W , the number of clusters: K)

begin

do m times of Cluster(W, K) and

assign the results to C_i for $i = 1, \dots, m$;

compute the average conditional entropies

$$T(C_i) = \frac{1}{m} \sum_{j=1}^m H(C_j|C_i) \text{ for } i = 1, \dots, m;$$

return $\arg \min_{C_i} T(C_i)$;

end

Figure 4: Clustering and refining algorithm

Clustering on a large data set may be time consuming. To speed up the algorithm, we focus on reducing the number of iterations. A small set of data points, for example, 1% of the entire data set, is selected as the *bootstrap* data set. First, the clustering algorithm runs on the bootstrap data set. Then, we run the clustering algorithm on the entire data set using the data map obtained from clustering on the bootstrap data set (instead of using randomly generated seed points).

2.5 Informal Description

The algorithm presents an effective method for finding clusters in high dimensional spaces without explicit distance functions. The clusters are defined as the group of points that have many features in common. The algorithm iteratively selects features with high accuracy and assigns data points into clusters based on the selected features. A feature f with high accuracy means that there is a “large” subset V of data set W such that f is present in most data points of set V . In other words, feature f has a small variance on set V . A feature f with low accuracy means that there is no such a “large” subset V of data set W on which feature f has a small variance. That is, f spreads largely within data set W . Hence our algorithm repeatedly projects the data points to the subspaces defined by the selected features of each cluster, assigns them to the clusters based on the projection, recalculate the accuracies of the features, then selects the features. As the process moves on, the selected features tend to converge to the set of features which have small variances among all the features.

The motivating idea behind the algorithm is maximum likelihood principle. The algorithm is an iterative (EM-type) process, which tries to find data map D and its correspondent feature map F to maximize $L(W|\hat{W}(D, F))$. Several approximation methods have been used in our algorithm to speed up the process. For a rigorous proof of convergence in

EM-type algorithms, please refer to [Selim and Ismail, 1984; Ambroise and Govaert, 1998; Kaski, 2000].

Our algorithm can be easily adapted to estimate the number of clusters instead of using K as an input parameter. We observed that the best number of clusters results the smallest average conditional entropy between clustering results obtained from different random seed point sets. Based on the observation, the algorithm of guessing the number of clusters is described in Figure 5.

Algorithm GuessK(data points: W)
 { L is the estimated maximum number of clusters;}
begin
 for K in $2 \cdots L$ **do**
 do m times of Cluster(W, K) and
 assign the results to C_{K_i} for $i = 1, \cdots, m$;
 compute the average conditional entropies
 $T(C_{K_i}) = \frac{1}{m} \sum_{j=1}^m H(C_{K_j} | C_{K_i})$ for $i = 1, \cdots, m$;
 end
 return $\arg \min_K \min_i T(C_{K_i})$;
end

Figure 5: Algorithm of guessing the number of clusters

2.6 Extending to Non-binary Data Sets

In order to handle non-binary data sets, we first translate raw attribute values of data points into binary feature spaces.

If an attribute is categorical, our translation scheme is similar to the method described in [Srikant and Agrawal, 1996]. We use as many features as the number of attribute values. The value of a binary feature corresponding to $\langle attribute_1, value_1 \rangle$ would be 1 if $attribute_1$ had $value_1$ in the raw attribute space, and 0 otherwise.

If an attribute is continuous, the method presented in [Srikant and Agrawal, 1996] can also be applied as the translation scheme. However, in the experiments we use Gaussian mixture models to fit each attribute of the original data sets, since most of them are generated from Gaussian mixture models, *i.e.*, each value is generated by one of many Gaussian distributions. The number of Gaussian distributions n can be obtained via maximizing Bayesian information criterion of the mixture model [Schwarz, 1978]. Then the value is translated into n feature values in the binary feature space. The j -th feature value is 1 if the probability that the value of the data point is generated from the j -th Gaussian distribution is the largest. Considering that some value is generated from a uniform distribution because of outliers, the attribute value is not mapped into any binary feature.

3 Experimental Results

There are many ways to measure how accurately the algorithm performs. One is the *confusion matrix* which is described in [Aggarwal *et al.*, 1999]. Entry (o, i) of a confusion matrix is the number of data points assigned to output cluster o and generated from input cluster i .

For input map I , which maps data points to input clusters, the entropy $H(I)$ measures the information of the input map. The task of clustering is to find out an output map O which recover the information. Therefore, the condition entropy $H(I|O)$ is interpreted as the information of the input map given the output map O , *i.e.*, the portion of information which is not recovered by the clustering algorithm. Therefore, the *recovering rate* of a clustering algorithm is defined as $1 - H(I|O)/H(I) = MI(I, O)/H(I)$, where $MI(I, O)$ is mutual information between I and O .

To test the performance of our algorithm, we did three experiments. Two experiments ran on synthetic data sets, and one ran on a real data set. The simulations were performed on a 700MHz Pentium-III IBM Thinkpad T20 computer with 128M of memory, running on octave 2.1.34 ² on Linux 2.4.10.

3.1 A Binary Synthetic Data Set

First we generate a binary synthetic data set to evaluate our algorithm. $N = 400$ points are from $K = 5$ clusters. Each point has $d = 200$ binary features. Each cluster has $l = 35$ positive features, 140 negative features. The positive features of a point have a probability of 0.8 to be 1 and 0.2 to be 0; the negative features of a point have a probability of 0.8 to be 0 and 0.2 to be 1; the rest features have a probability of 0.5 to be 0, and 0.5 to be 1.

Table 2 shows the confusion matrix of this experiment.

Input Output	A	B	C	D	E
1	0	0	0	83	0
2	0	0	0	0	81
3	0	0	75	0	0
4	86	0	0	0	0
5	0	75	0	0	0

Table 2: Confusion matrix for Experiment 1

In this experiment, $p(D|1) = 83/83 = 1$, $p(E|2) = 1$, $p(C|3) = 1$, $p(A|4) = 1$, $p(B|5) = 1$, and the rest are zeros. So we have $H(I|O) = \sum p(i|o) \log p(i|o) = 0$, *i.e.* the recovering rate of the algorithm is 1, *i.e.*, all data points are correctly recovered.

²GNU Octave is a high-level language, primarily intended for numerical computations. The software can be obtained from <http://www.octave.org/>.

3.2 A Continuous Synthetic Data Set

The second experiment is to cluster a continuous data set. We use the method described in [Aggarwal *et al.*, 1999] to generate a data set. The data set has $N = 100,000$ data points in a 20-dimensional space, with $K = 5$. All input clusters were generated in some 7-dimensional subspace. And 5% data points were chosen to be outliers, which were distributed uniformly at random throughout the entire space. Using the translation scheme described in section 2.6, we map all the data point into a binary space with 41 features.

Then, 1000 data points are randomly chosen as the bootstrap data set. Running Cluster and Refine algorithm on the bootstrap data set, we have a clustering result of the bootstrap data set. Using the bootstrap data set as the seed points, we run the algorithm on the entire data set. Table 3 shows the confusion matrix of this experiment. About 99.65% of data points are recovered. The conditional entropy $H(I|O)$ is 0.0226 as respect to the input entropy $H(I) = 1.72$. The recovering rate of this algorithm is $1 - H(I|O)/H(I) = 0.987$.

Input Output	A	B	C	D	E	O.
1	1	0	1	17310	0	12
2	0	15496	0	2	22	139
3	0	0	0	0	24004	1
4	10	0	17425	0	5	43
5	20520	0	0	0	0	6
Outliers	16	17	15	10	48	4897

Table 3: Confusion matrix for Experiment 2

We made a rough comparison with the result reported in [Aggarwal *et al.*, 1999]. Computing from the confusion matrix reported in their paper, their recovering rate is 0.927.

3.3 Zoo Database

We also evaluate our algorithm on the zoo database from UCI machine learning repository [UCI]. The database contains 100 animals, each of which having 15 Boolean attributes and 1 categorical attribute³.

In our experiment, all Boolean attributes are translated into two features, which are “true” and “false” features of the attributes. The numeric attribute, “legs”, is translated into six features, which represents 0, 2, 4, 5, 6, and 8 legs respectively.

Table 3 shows the confusion matrix of this experiment. The conditional entropy $H(I|O)$ is 0.317 while the input entropy $H(I)$ is 1.64. The recovering rate of this algorithm is $1 - H(I|O)/H(I) = 0.807$.

In the confusion matrix, we found that the clusters with a large number of animals are likely correctly clustered, for example, cluster 1, which contains “aardvark”, “antelope”,

³In the raw data, there are 101 instances and 18 attributes. But there are two instances of “frog”, which are only counted once in our experiment. Also, the attributes “animal name” and “type” are not counted.

Input Output	1	2	3	4	5	6	7
A	0	0	0	0	0	0	1
B	0	20	0	0	0	0	0
C	39	0	0	0	0	0	0
D	0	0	2	0	0	0	0
E	2	0	1	13	0	0	4
F	0	0	0	0	0	8	5
G	0	0	2	0	3	0	0

Table 4: Confusion matrix of Experiment Zoo

etc., is mapped into cluster *C*; cluster 2, which contains “chicken”, “crow”, *etc.*, is mapped into cluster *B*; cluster 4, which contains “bass”, “carp”, *etc.*, is mapped into cluster *E*; cluster 5, which contains “frog”, “gnat”, *etc.*, is mapped into cluster *F*; and cluster 6, which contains “flea”, “gnat”, *etc.*, is mapped into cluster *F*.

Our algorithm comes with an important by-product that the resulting clusters can be easily described in terms of features, since the algorithm produces an explicit feature map. For example, the positive features of cluster *A* are “no eggs”, “no backbone”, “venomous” and “8 legs”; the positive features of cluster *B* are “feather”, “airborne”, and “2 legs”; the positive features of cluster *C* are “hair”, “milk”, “domestic” and “catsize”; the positive feature of cluster *D* is “0 legs”; the positive feature of cluster *E* are “aquatic”, “no breathes”, “fins” and “5 legs”; the positive features of cluster *F* are “6 legs” and “no tail”; the positive feature of cluster *G* is “4 legs”. Hence, cluster *B* can be described as animals having feather and 2 legs, and being airborne, which are the representative features of the animals called *birds*. Cluster *E* can be described as animals being aquatic, having no breathes, having fins and 5 legs.

Animals “dolphin” and “porpoise” are in cluster 1, but were clustered into cluster *E*, because their attributes “aquatic” and “fins” make them more like animals in cluster *E* than their attribute “milk” does for cluster *C*.

3.4 Scalability Results

In this subsection we present the computational scalability results for our algorithm. The results were averaged over five runs on each case to eliminate the randomness effect. We report the scalability results in terms of the number of points and the dimensionality of the space.

Number of points: The data sets we tested have 200 features (dimensions). They all contained 5 clusters and each cluster has an average 35 features. Figure 6 shows the scalability result of the algorithm in terms of the number of data points. The value of the *y* coordinate in figure 6(a) is the running time in seconds. We implemented the algorithm using octave 2.1.34 on Linux 2.4.10. If it were implemented in *C*, the running time could be reduced considerably. Figure 6(a) contains two curves: one is the scalability result with bootstrap and one without. It can be seen from figure 6(a) that with bootstrap, our

algorithm scales sublinearly with the number of points and without bootstrap, it scales linearly with the number of points. The value of the y coordinate in figure 6(b) is the ratio of the running time to the number of points. The linear and sublinear scalability properties can also be observed in figure 6(b).

Dimensionality of the space: The data sets we tested have 10,000 data points. They all contained 5 clusters and the average features (dimensions) for each cluster is about $\frac{1}{6}$ of the total number of features (dimensions). Figure 7 shows the scalability result of the algorithm in terms of dimensionality of the space. The value of the y coordinate in figure 7(a) is the running time in seconds. It can be seen from figure 7(a) that our algorithm scales linearly with the dimensionality of the space. The value of the y coordinate in figure 7(b) is the ratio of the running time to the number of features.

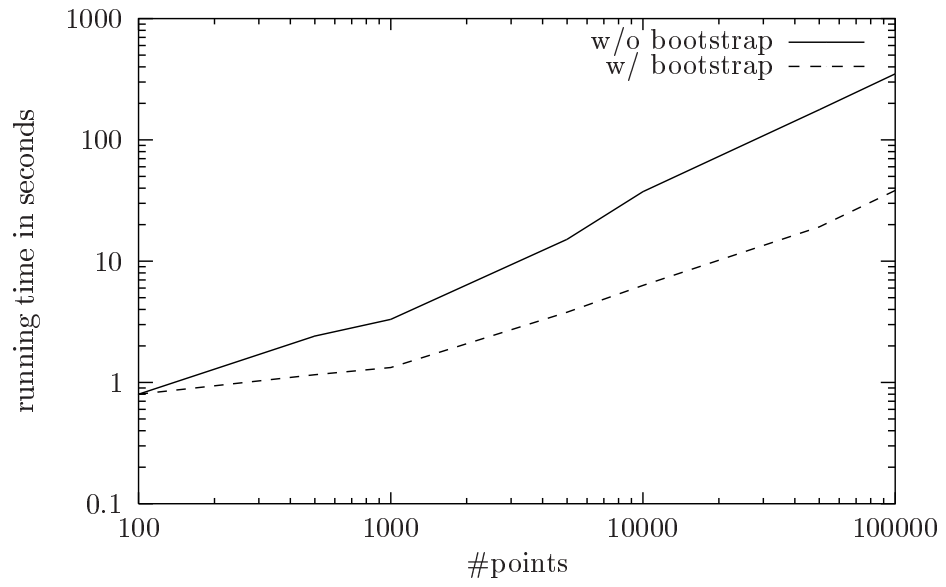
4 Relative Work

Traditional clustering techniques can be broadly classified into partitional clustering, hierarchical clustering, density-based clustering and grid-based clustering [Han and Kamber, 2000]. Partitional clustering attempts to directly decompose the data set into K disjoint clusters such that the data points in a cluster are nearer to one another than the data points in other clusters. Hierarchical clustering proceeds successively by building a tree of clusters. It can be viewed as a nested sequence of partitioning. The tree of clusters, often called *dendrogram*, shows the relationship of the clusters and a clustering of the data set can be obtained by cutting the dendrogram at a desired level. Density-based clustering is to group the neighboring points of a data set into clusters based on density conditions. Grid-based clustering quantizes the object space into a finite number of cells that form a grid-structure and then performs clustering on the grid structure. Most of the traditional clustering methods use the distance functions as objective criteria and are not effective in high dimensional spaces.

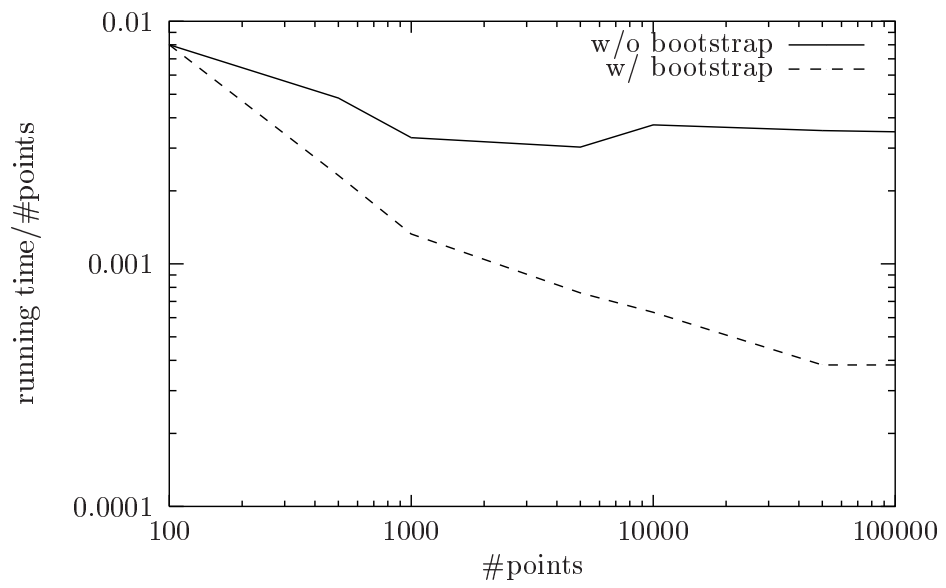
Next we review some recent clustering algorithms which have been proposed for high dimensional spaces or without distance functions and are largely related to our work.

CLIQUE [Agrawal *et al.*, 1998] is an automatic subspace clustering algorithm for high dimensional spaces. It uses equal-size cells and cell density to find dense regions in each subspace of a high dimensional space. Cell size and the density threshold need to be provided by the user as inputs. CLIQUE does not produce disjoint clusters and the highest dimensionality of subspace clusters reported is about 10. Our algorithm produces disjoint clusters and does not require the additional parameters such as the cell size and density. Also it can find clusters with higher dimensionality.

In [Aggarwal and Yu, 2000; Aggarwal *et al.*, 1999], the authors introduced the concept of projected clustering and developed algorithms for discovering interesting patterns in subspaces of high dimensional spaces. The core idea of their algorithm is a generalization of feature selection which allows the selection of different sets of dimensions for different subsets of the data sets. However, their algorithms are based on the Euclidean distance or Manhattan distance and their feature selection method is a variant of singular value decomposition (SVD). Also their algorithms assume that the number of projected dimensions

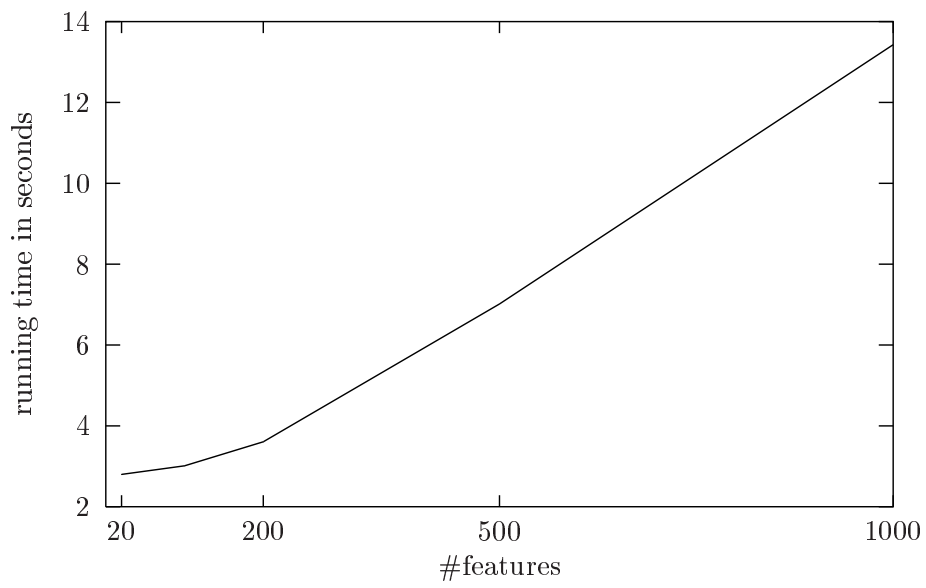


(a) running time

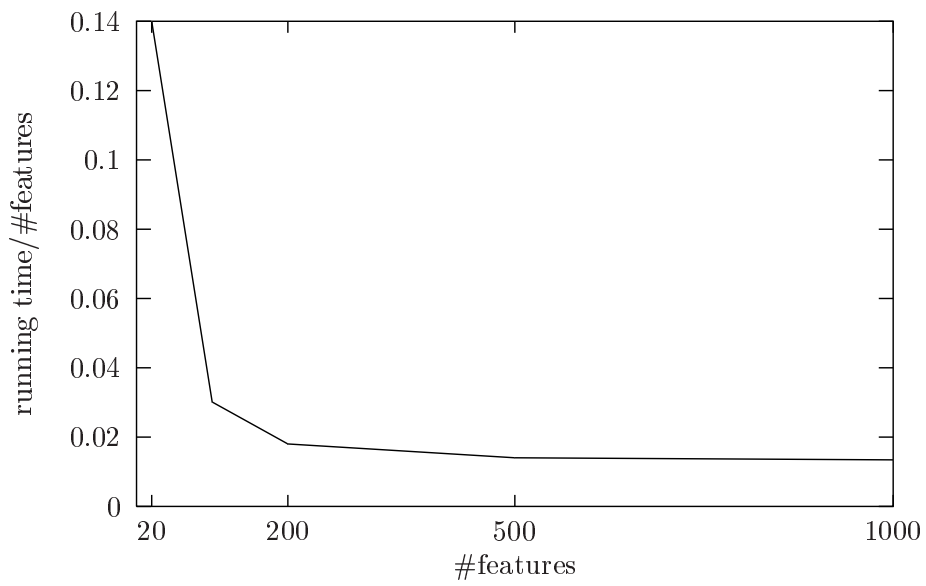


(b) running time/#points

Figure 6: Scalability with number of points



(a) running time



(b) running time/#features

Figure 7: Scalability with number of features

are given beforehand. Our algorithm does not need the distance measures and the number of dimensions for each cluster. Also it does not require all projected clusters to have the same number of dimensions.

Ramkumar and Swami [Ramkumar and Swami, 1998] proposed a new method for clustering without distance function. Their method is based on the principles of co-occurrence maximization and label minimization which are normally implemented using the association rules and classification techniques. Their method does not consider the correlations of dimensions with respect to data locality and critically depend on the input parameters like parameter w and thresholds for association. Our algorithm is quite different. We consider the fact that the correlations of dimensions are often specific to data distribution.

Liu, Xia and Yu [Liu *et al.*, 1998] proposed a method to find sparse and data regions using decision tree. The method is a grid-based method and it does not work well for high dimensional space. It is also hard to decide the size of the grids and the density threshold. In [Liu *et al.*, 2000], a clustering technique based on decision tree construction is presented. The technique first introduces non-existing points to the data space and then performs the decision tree algorithm to partitioning the data space into dense and sparse regions.

Cheng [Cheng *et al.*, 1999] proposed an entropy-based subspace clustering for mining numerical data. The method is motivated by the fact that a subspace with clusters typically has lower entropy than a subspace without clusters.

Strehl and Ghosh [Strehl and Ghosh, 2000] proposed OPOSSUM, a similarity-based clustering approach based on constrained, weighted graph-partitioning. OPOSSUM is based on *Jaccard Similarity* and is particularly attuned to real-life market baskets, characterized by high-dimensional sparse customer-product matrices.

Banfield and Raftery [Banfield and Raftery, 1993] described the approach based on *mixture model* for clustering. As pointed out in [Fasulo, 1999], there are several problems with this approach. First the approach does not focus on efficiency. Second, a large degree of manual intervention is required. Last but not the least, mixture models rely on the assumption that the data fits Gaussian distribution which may be not be the case in many applications.

In [Kleinberg, 1997], an approach based on dynamical systems is proposed. The approach relies on a simple process which iteratively computes weights on the vertices of a graph until a fixed point is reached. Ben-Dor etc. [Ben-Dor *et al.*, 1999] proposed a clustering algorithm based on *corrupted clique model*. The basic idea of this model is that ideally all elements within each cluster should be similar to one another and not similar to any elements of other clusters. Hence, the similarity graph for the data should appear as a set of vertex-disjoint cliques. [Fasulo, 1999] also gave a detailed survey on clustering approaches based on mixture models, dynamical systems and clique graphs.

A Self-Organizing Map (SOM), also called Kohonen clustering, can be used to clarify relations in a complex set of data by revealing some inherent order [Kohonen, 1990]. SOM is self-organized and the only parameter needs to be specified is the number of clusters. SOM is particularly well suited to the task of identifying a small number of prominent classes in a data set with multidimensionality. However, the "centroids" with huge dimensionality are hard to interpret in SOM. Moreover the robustness of SOM is also a problem especially when the number of clusters is unknown or the data set is sparse.

Kearns *et al.* [Kearns *et al.*, 1998] gave an information-theoretic analysis of *hard assignments* (used by *K*-means) and *soft assignments* (used by EM algorithm) for clustering and proposed a *posterior* partition algorithm which is close to the soft assignments of EM for clustering. The relationship between maximum likelihood and clustering is also discussed in [Jordan, 2001].

5 Conclusions

In this paper, we proposed a novel clustering method which does not require the distance function for high dimensional spaces. The algorithm performs clustering by iteratively optimize the data map and the feature map. We have adopted several approximation methods to maximize the likelihood between the given data set and the generated model. Extensive experiments have been conducted and the results show that the algorithm is both efficient and effective.

Our contributions are:

- We propose a non-distance based clustering algorithm in high dimensional spaces based on maximum likelihood principle.
- We come up with a new perspective of viewing the clustering problem by interpreting it as the dual problem of optimizing the feature map.
- As a by-product, the algorithm also produces the feature map which can provide interpretable descriptions of the resulting clusters.
- Our algorithm also provides a method to select the number of clusters based on the conditional entropy.
- We introduce the recovering rate, an accuracy measure of clustering, to measure the performance of clustering algorithms and to compare clustering results of different algorithms.

Our future work includes developing more direct methods to optimize the data map and the feature map, designing the parallel and distributed versions of our algorithm and the incremental clustering algorithm based on our algorithm.

References

- [Aggarwal and Yu, 2000] C. Aggarwal and P. S. Yu, “Finding generalized projected clusters in high dimensional spaces,” In *SIGMOD-00*, 2000.
- [Aggarwal *et al.*, 1999] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park, “Fast algorithms for projected clustering,” In *ACM SIGMOD Conference*, pages 61–72, 1999.
- [Agrawal *et al.*, 1998] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, “Automatic subspace clustering for high dimensional data for data mining applications,” In *SIGMOD-98*, 1998.
- [Ambroise and Govaert, 1998] C. Ambroise and G. Govaert, “Convergence of an EM-type algorithm for spatial clustering,” *Pattern Recognition Letters*, 19:919–927, 1998.
- [Banfield and Raftery, 1993] J. Banfield and A. Raftery, “Model-based Gaussian and non-Gaussian clustering,” *Biometrics*, 49:803–821, 1993.
- [Ben-Dor *et al.*, 1999] A. Ben-Dor, R. Shamir, and Z. Yakhini, “Clustering Gene Expression Patterns,” *J. of Comp. Biology*, 6(3/4):281–297, 1999.
- [Berger and Rigoutsos, 1991] M. Berger and I. Rigoutsos, “An Algorithm for Point Clustering and Grid Generation,” *IEEE Trans. on Systems, Man and Cybernetics*, 21(5):1278–1286, 1991.
- [Beyer *et al.*, 1999] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is nearest neighbor meaningful?,” In *ICDT Conference*, 1999.
- [Brito *et al.*, 1997] M.R. Brito, E. Chavez, A. Quiroz, and J. Yukich, “Connectivity of the Mutual K-Nearest-Neighbor Graph for Clustering and Outlier Detection,” *Statistics and Probability Letters*, 35:33–42, 1997.
- [Cheeseman *et al.*, 1988] P. Cheeseman, J. Kelly, and M. Self, “AutoClass: A Bayesian Classification System,” In *ICML’88*, 1988.
- [Cheng *et al.*, 1999] C-H Cheng, A. W-C Fu, and Y. Zhang, “Entropy-based Subspace Clustering for Mining Numerical Data,” In *KDD-99*, 1999.
- [Dubes and Jain, 1980] R. Dubes and A.K. Jain, “Clustering Methodologies in Exploratory Data Analysis,” In M.C. Yovits, editor, *Advances in Computers*, volume 19. Academic Press, New York, 1980.
- [Duda and Hart, 1973] Richard Duda and Peter E. Hart, *Pattern Classification and Scene Analysis*, Wiley, 1973.
- [Ester *et al.*, 1995a] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu, “A Database Interface for Clustering in Large Spatial Databases,” In *Proc. of 1st Int’l Conf. on KDD*, 1995.

- [Ester *et al.*, 1995b] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu, “A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Nosies,” In *Proc. of 1st Int’l Conf. on KDD*, 1995.
- [Ester *et al.*, 1995c] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu, “Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification,” In *Proc. of 4th Int’l Symposium on Large Spatial Databases*, 1995.
- [Fasulo, 1999] D. Fasulo, “An analysis of recent work on clustering algorithms,” TR 01-03-02, U. of Washington, Dept. of Comp. Sci. & Eng., 1999.
- [Fisher, 1987] Douglas H. Fisher, “Knowledge Acquisition via Incremental Conceptual Clustering,” *Machine Learning*, 2(2), 1987.
- [Fisher, 1995] Douglas H. Fisher, “Iterative Optimization and Simplification of Hierarchical Clusterings,” Tr, Vanderbilt U., Dept. of Comp. Sci., 1995.
- [Guha *et al.*, 1998] S. Guha, R. Rastogi, and K. Shim, “CURE: An Efficient Clustering Algorithm for Large Database,” In *Proceedings of the 1998 ACM SIGMOD Conference*, pages 73–84, 1998.
- [Han and Kamber, 2000] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2000.
- [Hastie *et al.*, 2001] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2001.
- [Jordan, 2001] M. I. Jordan, *Graphical Models: Foundations of Neural Computation*, MIT Press, 2001.
- [Kaski, 2000] Samuel Kaski, “Convergence of a stochastic semisupervised clustering algorithm,” TR A62, Helsinki University of Technology, Espoo, Finland, 2000.
- [Kearns *et al.*, 1998] M. Kearns, Y. Mansour, and A. Y. Ng, “An Information-theoretic Analysis of Hard and Soft Assignment Methods for Clustering,” In *Learning in Graphical Models*. Kluwer AP, 1998.
- [Kleinberg, 1997] J. M. Kleinberg, “Two Algorithms for Nearest-Neighbor Search in High Dimensions,” In *ACM Symp. on Theory of Computing*, 1997.
- [Kohavi and Sommerfield, 1995] R. Kohavi and D. Sommerfield, “Feature subset selection using the wrapper method: overfitting and dynamic search space technology,” In *KDD-95*, 1995.
- [Kohonen, 1990] Teuvo Kohonen, “The Self-Organizing Map,” In *New Concepts in Computer Science*, 1990.
- [Lebowitz, 1987] Michael Lebowitz, “Experiments with Incremental Concept Formation: UNIMEM,” *Machine Learning*, 1987.

- [Lee, 1981] R. Lee, “Clustering Analysis and its Applications,” In J. Toum, editor, *Advances in Information Systems Science*, volume 8, pages 169–292. Plenum Press, New York, 1981.
- [Li *et al.*, 2001] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara, “Mining Patterns from Case Base Analysis,” In *Workshop on Integrating Data Mining and Knowledge Management (ICDM’01)*, San Jose, CA, 2001.
- [Liu *et al.*, 1998] B. Liu, K. Wang, L. Mun, and X. Qi, “Using decision tree for discovering,” In *PRICAI-98*, 1998.
- [Liu *et al.*, 2000] Bing Liu, Yiyuan Xia, and Philip S. Yu, “Clustering Through Decision Tree Construction,” In *SIGMOD-00*, 2000.
- [Murtagh, 1983] F. Murtagh, “A Survey of Recent Advances in Hierarchical Clustering Algorithms,” *The Computer Journal*, 1983.
- [Ng and Han, 1994] Raymond T. Ng and Jiawei Han, “Efficient and Effective Clustering Methods for Spatial Data Mining,” In *Proc. of VLDB*, 1994.
- [Ramkumar and Swami, 1998] G.D. Ramkumar and A. Swami, “Clustering Data Without Distance Function,” *IEEE Data(base) Engineering Bulletin*, 21:9–14, 1998.
- [Schwarz, 1978] G. Schwarz, “Estimating the dimension of a model,” *Annals of Statistics*, 6:461–464, 1978.
- [Selim and Ismail, 1984] S. Z. Selim and M. A. Ismail, “K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality,” *IEEE Trans.*, PAMI-6(1), January 1984.
- [Srikant and Agrawal, 1996] R. Srikant and R. Agrawal, “Mining Quantitative Association Rules in Large Relational Tables,” In *SIGMOD-96*, 1996.
- [Strehl and Ghosh, 2000] A. Strehl and J. Ghosh, “A Scalable Approach to Balanced, High-dimensional Clustering of Market-baskets,” In *HiPC-2000*, 2000.
- [UCI] UCI, “UCI machine learning repository,” <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [Zhang *et al.*, 1996] T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH: An Efficient Data Clustering Method for Very Large Databases,” In *ACM SIGMOD Conference*, 1996.