

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

Shenghuo Zhu, Tao Li, and Mitsunori Ogihara

Department of Computer Science
University of Rochester
Rochester, NY 14620
{zsh,taoli,ogihara}@cs.rochester.edu

Abstract. The clustering problem, which aims at identifying the distribution of patterns and intrinsic correlations in large data sets by partitioning the data points into similarity clusters, has been widely studied. Traditional clustering algorithms use distance functions to measure similarity and are not suitable for high dimensional spaces. In this paper, we propose *CoFD* algorithm, which is a non-distance based clustering algorithm for high dimensional spaces. Based on the maximum likelihood principle, *CoFD* 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 *CoFD*.

1 Introduction

Clustering problems arise in many disciplines and have a wide range of applications. Intuitively, 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. The problem of clustering has been studied extensively in the database [20,13], statistics [5,7] and machine learning communities [8,12] with different approaches and different focuses.

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 [6]. Many feature selection techniques have been applied to reduce the dimensionality of the space [17]. However, as demonstrated in [2], 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.

* The project is supported in part by NIH Grants 5-P41-RR09283, RO1-AG18231, and P30-AG18254 and by NSF Grants EIA-0080124, NSF CCR-9701911, and DUE-9980943.

As pointed out in [15], all methods that overcome the dimensionality problems have an associated and often implicit or adaptive-metric for measuring neighborhoods.

In this paper, we present *CoFD*¹, a non-distance based algorithm for clustering in high dimensional spaces. The *CoFD* algorithm described here is an improvement over our previous method [22] and it contains several major extensions and revisions. The main idea of *CoFD* is as follows: Suppose that a data set W with feature set S needs to be clustered into K classes, C_1, \dots, C_K with the possibility of recognizing some data points to be outliers. The clustering of the data then is represented by two functions, the *data map* $D : W \rightarrow \{0, 1, \dots, K\}$ and the *feature map* $F : S \rightarrow \{0, 1, \dots, K\}$, where $1, \dots, k$ correspond to the clusters and 0 corresponds to the set of outliers. Accuracy of such representation is measured using the log likelihood. Then, by the *Maximum Likelihood Principle*, the best clustering will be the representation that maximizes the likelihood. In *CoFD*, several approximation methods are used to optimize D and F iteratively. The *CoFD* algorithm can also be easily adapted to estimate the number of classes when the value K is not given as a part of the input. An added bonus of *CoFD* is that it produces interpretable descriptions of the resulting classes 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 *CoFD*; 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.

2 The CoFD Algorithm

This section describes *CoFD* and the core idea behind it. We first present the *CoFD* algorithm for binary data sets. Then we will show how to extend it to continuous or non-binary categorical data sets in Section 2.4.

2.1 The Model of CoFD

Suppose we wish to divide W into K classes with the possibility of declaring some data as outliers. Such clustering can be represented by a pair of functions, (F, D) , where $F : S \rightarrow \{0, 1, \dots, K\}$ is the *feature map* and $D : W \rightarrow \{0, 1, \dots, K\}$ is the *data map*.

Given a representation (F, D) we wish to be able to evaluate how good the representation is. To accomplish this we use the concept of *positive features*. Intuitively, a positive feature is one that best describes the class it is associated with. Suppose that we are dealing with a data set of animals in a zoo, where the vast majority of the animals is the monkey and the vast majority of the animals is the four-legged animal. Then, given an unidentified animal having four legs in the zoo, it is quite natural for one to guess that the animal is a monkey

¹ *CoFD* is the abbreviation of Co-training between Feature maps and Data maps.

because the conditional probability of that event is high. Therefore, we regard the feature “having four legs” as a *positive* (characteristic) feature of the class. In most practical cases, characteristic features of a class do not overlap with those of another class. Even if some overlaps exist, we can add the combinations of those features into the feature space.

Let N be the total number of data points and let d be the number of features. Since we are assuming that the features are binary, W can be represented as a 0-1 matrix, which we call the data-feature matrix. Let $1 \leq i \leq N$ and $1 \leq j \leq d$. We say that the j th feature is *active* in the i th point if and only if $W_{ij} = 1$. We also say that the i th data point *possesses* the i th feature if and only if $W_{ij} = 1$.

The key idea behind the *CoFD* algorithm is the use of the Maximum Likelihood Principle, which states that the best model is the one that has the highest likelihood of generating the observed data. We apply this principle by regarding the data-feature matrix as the observed data and the representation (D, F) as the model. Let the data map D and the feature map F be given. Consider the $N \times K$ matrix, $\hat{W}(D, F)$, defined as follows: for each i , $1 \leq i \leq N$, and each j , $1 \leq j \leq d$, the ij entry of the matrix is 1 if $1 \leq D(i) = F(j) \leq K$ and 0 otherwise. This is the model represented by F and D , interpreted as the consistence of $D(i)$ and $F(j)$. Note that $\hat{W}(D, F) = 0$ if $0 = D(i) = F(j)$. For all i , $1 \leq i \leq N$, j , $1 \leq j \leq d$, $b \in \{0, 1\}$, and $c \in \{0, 1\}$, we consider $P(W_{ij} = b \mid \hat{W}_{ij}(D, F) = c)$, the probability of the j th feature being active in the i th data point in the real data conditioned upon the j th feature being active in the i th data in the model represented by D and F . We assume that this conditional probability is dependent only on the values of b and c . Let $Q(b, c)$ denote the probability of an entry being observed as b while the entry in the model being equal to c . Also, let $p(b, c)$ denote the proportion of (i, j) such that $W_{ij} = b$ and $\hat{W}_{ij}(D, F) = c$. Then the likelihood of the model can be expressed as follows:

$$\begin{aligned} \log L(D, F) &= \log \prod_{i,j} P(W_{ij} \mid \hat{W}_{i,j}(D, F)) = \log \prod_{b,c} Q(b, c)^{dNp(b,c)} \\ &= dN \sum_{b,c} p(b, c) \log Q(b, c) \equiv -dNH(W \mid \hat{W}(D, F)) \end{aligned} \quad (1)$$

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

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 $H(W_{\cdot j} \mid \hat{W}_{\cdot j}(D, F))$. If D and $F(j)$ are given, the entropies can be directly estimated. Therefore, $F(j)$ is assigned to the class in which the conditional entropy of $W_{\cdot j}$ is the smallest. Optimizing D while fixing F is the dual. Hence, we only need to minimize $H(W_{i \cdot} \mid \hat{W}_{i \cdot}(D, F))$ for each i . A straightforward approximation method for minimizing $H(W_{\cdot j} \mid \hat{W}_{\cdot j}(D, F))$ is to assign $F(j)$ to $\arg \max_k \|\{i \mid W_{ij} = k\}\|$. This approximation method is also applicable to minimization of

$H(W_i | \hat{W}_i(D, F))$ where we assign $D(i)$ to $\arg \max_k \|\{j | W_{ij} = k\}\|$. A direct improvement of the approximation is possible by using the idea of entropy-constrained vector quantizer [10] and assigning $F(j)$ and $D(i)$ to $\arg \max_k (\|\{i | W_{ij} = k\}\| + \log(\frac{d(k)}{d}))$ and $\arg \max_k (\|\{j | W_{ij} = k\}\| + \log(\frac{N(k)}{N}))$, respectively, where $d(k)$ and $N(k)$ are the number of features and the number of points in class k .

There are two auxiliary procedures in the algorithm: **EstimateFeatureMap** estimates the feature map from the data map; **EstimateDataMap** estimates the data map from the feature map. Chi-square tests are used for deciding if a feature is an outlier or not. The main procedure, **CoFD**, attempts to find a best class by an iterative process similar to the EM algorithm. Here the algorithm iteratively estimates the data and feature maps based on the estimations made in the previous round, until no more changes occur in the feature map. The detailed pseudo-code of **EstimateFeatureMap** and **EstimateDataMap** can be found in [21]. It can be observed from the pseudo-code description in Fig 1 that the time complexities of both **EstimateFeatureMap** and **EstimateDataMap** are $O(K \times N \times d)$. The number of iterations in **CoFD** is not related to N or d .

```

Algorithm CoFD(data points:  $W$ , # of classes:  $K$ )
begin
  let  $W1$  be the set of randomly chosen  $nK$  distinct data points from  $W$ ;
  assign each  $n$  of them to one class, 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

```

Fig. 1. Clustering algorithm

2.2 Refining

Clustering results are sensitive to initial seed points. Randomly chosen seed points may make the search trapped in local minima. We use a refining procedure, whose idea is to use conditional entropy to measure the similarity between a pair of clustering results. *CoFD* attempts to find a best clustering result having the smallest average conditional entropy against all others. 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 2). The idea is to use conditional entropy to measure the similarity between a pair of clustering results. *CoFD* is to find a best clustering result which has smallest average conditional entropy to all others.

Clustering 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, may be selected as the *bootstrap* data set. First, the clustering algorithm is executed on the bootstrap data set. Then, the clustering algorithm is run on the entire data set using the data map obtained from clustering on the bootstrap data set (instead of using randomly generated seed points).

```

Algorithm Refine(data points:  $W$ , the number of clusters:  $K$ )
begin
  do  $m$  times of  $\text{CoFD}(W, K)$  and
    assign the results to  $C_i$  for  $i = 1, \dots, m$ ;
  compute the average conditional entropies
     $T(C_i) = \frac{1}{m} \sum_{j=1}^m H(C_j|C_i)$  for  $i = 1, \dots, m$ ;
  return  $\arg \min_{C_i} T(C_i)$ ;
end

```

Fig. 2. Clustering and refining algorithm

2.3 Informal Description

CoFD 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. *CoFD* 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.

CoFD 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, *CoFD* of guessing the number of clusters is described in Figure 3.

2.4 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. The translation scheme in [18] can be

```

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 CoFD( $W, K$ ) and
      assign the results to  $C_{K_i}$  for  $i = 1, \dots, m$ ;
    compute the average conditional entropies
       $T(C_{K_i}) = \frac{1}{m} \sum_{j=1}^m H(C_{K_j} | C_{K_i})$  for  $i = 1, \dots, m$ ;
    end
  return  $\arg \min_K \min_i T(C_{K_i})$ ;
end

```

Fig. 3. Algorithm of guessing the number of clusters

used to discretize categorical and continuous attributes. However, this type of translation is vulnerable to outliers that may drastically skew the range. In our algorithm, we use two other discretization methods. The first is combining *Equal Frequency Intervals* method with the idea of CMAC [4]. Given m instances, the method divides each dimension into k bins with each bin containing $\frac{m}{k} + \gamma$ adjacent values². In other words, with the new method, each dimension is divided into several overlapped segments and the size of overlap is determined by γ . An attribute is then translated into a binary sequence having bit-length equal to the number of the overlapped segments, where each bit represents whether the attribute belongs to the corresponding segment. We can also use Gaussian mixture models to fit each attribute of the original data sets, since most of them are generated from Gaussian mixture models. The number of Gaussian distributions n can be obtained by maximizing the Bayesian information criterion of the mixture model. 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.

3 Experimental Results

There are many ways to measure how accurately *CoFD* performs. One is the *confusion matrix* which is described in [2]. 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

² The parameter γ can be a constant for all bins or different constants for different bins depending on the distribution density of the dimension. In our experiment, we set $\gamma = \lfloor m/5k \rfloor$.

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 experiments on both synthetic and real data sets. The simulations were performed on a 700MHz Pentium-III IBM Thinkpad T20 computer with 128M of memory, running on octave 2.1.34³ on Linux 2.4.10.

3.1 A Continuous Synthetic Data Set

In this experiment we attempted to cluster a continuous data set. We used the method described in [2] to generate a data set W_1 . W_1 has $N = 100,000$ data points in a 20-dimensional space, with $K = 5$. All input classes were generated in some 7-dimensional subspace. Five percent of the data points was chosen to be outliers, which were distributed uniformly at random throughout the entire space. Using the second translation method described in Section 2.4, we mapped all the data point into a binary space with 41 features. Then, 1000 data points were randomly chosen as the bootstrap data set. By running CoFD algorithm on the bootstrap data set, we obtained clustering of the bootstrap data set. Using the bootstrap data set as the seed points, we ran the algorithm on the entire data set. Figure 4 shows the confusion matrix of this experiment. About 99.65% of the data points were recovered. The conditional entropy $H(I|O)$ is 0.0226 while the input entropy $H(I)$ is 1.72. The recovering rate is thus $1 - H(I|O)/H(I) = 0.987$. We made a rough comparison with the result reported in [2]. From the confusion matrix reported in the paper, their recovering rate is calculated as 0.927, which seems to indicate that our algorithm is better than theirs in terms of recovering rate.

Output	Input	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

Fig. 4. Confusion matrix for Continuous Synthetic Data Set

Output	Input	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

Fig. 5. Confusion matrix of Zoo

We made a rough comparison with the result reported in [2]. Computing from the confusion matrix reported in their paper, their recovering rate is 0.927.

³ GNU Octave is a high-level language, primarily intended for numerical computations. The software can be obtained from <http://www.octave.org/>.

3.2 Zoo Database

We also evaluated the performance of the *CoFD* algorithm on the zoo database available at the UC Irvine Machine Learning Repository. The database contains 100 animals, each of which has 15 boolean attributes and 1 categorical attribute⁴. We translated each boolean attribute into two features, whether the feature is active and whether the feature is inactive. We translated the numeric attribute, “legs”, into six features, which correspond to 0, 2, 4, 5, 6, and 8 legs, respectively. Figure 4 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 to be correctly clustered.⁵

CoFD comes with an important by-product that the resulting classes can be easily described in terms of features, since the algorithm produces an explicit feature map. For example, the positive features of class *B* are “feather,” “airborne,” and “two legs.” Hence, class *B* can be described as animals having feather and two legs, and being airborne, which are the representative features of the *birds*.⁶

3.3 Clustering of Technical Reports

We applied our *CoFD* algorithm to cluster the collection of technical reports published in the years 1989–2000 in our department based on the words in their abstracts. We obtained data points from the abstracts by removing stop words and then applying words stemming operations. Then we computed the frequency of each remaining word. We selected the top 500 frequent words and transformed the original dataset into a categorical data set according to the occurrence of these 500 words. The recovering rate is 0.5762 and the confusion matrix is shown in Table 6. We loosely divided the research areas into four groups: Symbolic-AI, Spatial-AI, Systems, and Theory. Thus there are four clusters in the result. The columns of the confusion matrix are the four groups in the order they are mentioned. The result shows that Systems and Theory are much different from each other, and are different both from Symbolic-AI and from Spatial-AI, that Symbolic-AI and Spatial-AI are similar to each other, and that Symbolic-AI is more similar to Spatial-AI. We also used *K-means* to this task, whose recovering rate was about 0.27.

⁴ The original data set has 101 data points but one animal, “frog,” appears twice. So we eliminated one of them. We also eliminated two attributes, “animal name” and “type.”

⁵ For example, cluster no. 1 is mapped into cluster *C*; cluster no. 2 is mapped into cluster *B*; *etc.*

⁶ Animals “dolphin” and “porpoise” are in class 1, but were clustered into class *E*, because their attributes “aquatic” and “fins” make them more like animals in class *E* than their attribute “milk” does for class *C*.

3.4 Dermatology Database

We also evaluated our algorithms on the dermatology database from the UC Irvine Machine Learning Repository. The database is used for the differential diagnosis of erythema-squamous diseases. These diseases all share the clinical features of erythema and scaling, with very little differences. The dataset contains 366 data points over 34 attributes and was previously used for classification. In our experiment, we use it to demonstrate our clustering algorithm. The first translation method described in Section 2.4 was used for preprocessing. The confusion matrix of *CoFD*, on this dataset is presented in Figure 7.

Output \ Input	1	2	3	4
A	8	9	5	109
B	9	3	83	5
C	7	44	0	3
D	74	8	6	3

Fig. 6. Confusion matrix of technical reports.

Output \ Input	1	2	3	4	5	6
A	0	5	0	1	0	20
B	112	13	0	2	4	0
C	0	0	72	1	0	0
D	0	14	0	15	5	0
E	0	7	0	4	43	0
F	0	22	0	26	0	0

Fig. 7. Confusion matrix of Dermatology Database.

Our algorithm scales linearly with the number of points and features. More detailed experiment results are presented in [21].

4 Relative Work

Traditional clustering techniques can be broadly classified into partitional clustering, hierarchical clustering, density-based clustering and grid-based clustering [14]. 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 [3] 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. CLIQUE does not produce disjoint clusters and the highest dimensionality of subspace clusters reported is about 10. *CoFD* produces disjoint clusters.

In [1,2], 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 are given beforehand. *CoFD* 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.

Cheng[9] proposed an entropy-based subspace clustering for mining numerical data. There are also some recent work on clustering categorical datasets, such as ROCK and CACTUS, and on clustering based on decision tree. Our algorithm can be applied to both categorical and numerical data.

Strehl and Ghosh[19] 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.

Fasulo[11] also gave a detailed survey on clustering approaches based on mixture models, dynamical systems and clique graphs. The relationship between maximum likelihood and clustering is also discussed in [16]. Similarity based on shared features has also been analyzed in cognitive science such as the *Family resemblances* study by Rosch and Mervis.

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 *CoFD* is both efficient and effective.

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

1. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD-00*, 2000.
2. C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. Soo Park. Fast algorithms for projected clustering. In *ACM SIGMOD Conference*, 1999.
3. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering for high dimensional data for data mining applications. In *SIGMOD-98*, 1998.
4. J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Trans. of the ASME, J. Dynamic Systems, Measurement, and Control*, 97(3):220–227, sep 1975.
5. 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.

6. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? In *ICDT Conference*, 1999.
7. 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.
8. P. Cheeseman, J. Kelly, and M. Self. AutoClass: A bayesian classification system. In *ICML'88*, 1988.
9. C-H Cheng, A. W-C Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD-99*, 1999.
10. P. A. Chou, T. Lookabaugh, and R. M. Gray. Entropy-constrained vector quantization. *IEEE Trans.*, ASSP-37(1):31, 1989.
11. D. Fasulo. An analysis of recent work on clustering algorithms. Technical Report 01-03-02, U. of Washington, Dept. of Comp. Sci. & Eng., 1999.
12. Douglas H. Fisher. Iterative optimization and simplification of hierarchical clusterings. Technical Report CS-95-01, Vanderbilt U., Dept. of Comp. Sci., 1995.
13. S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large database. In *Proceedings of the 1998 ACM SIGMOD Conference*, 1998.
14. Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.
15. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.
16. Michael I. Jordan. *Graphical Models: Foundations of Neural Computation*. MIT Press, 2001.
17. R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper method: overfitting and dynamic search space technology. In *KDD-95*, 1995.
18. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD-96*, 1996.
19. A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *HiPC-2000*, 2000.
20. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *ACM SIGMOD Conference*, 1996.
21. Shenghuo Zhu and Tao Li. An algorithm for non-distance based clustering in high dimensional spaces. Technical Report 763, University of Rochester, Computer Science Department, Rochester, NY, 2002.
22. Shenghuo Zhu and Tao Li. A non-distance based clustering algorithm. In *Proc. of IJCNN 2002*, 2002. To appear.