A Fast Algorithm to Cluster High Dimensional Basket Data

Carlos Ordonez, Edward Omiecinski, Norberto Ezquerra
Georgia Institute of Technology

***Abstract*—Clustering is a data mining problem that has received significant attention by the database community. Data set size, dimensionality and sparsity have been identified as aspects that make clustering more difficult. This work introduces a fast algorithm to cluster large binary data sets where data points have high dimensionality and most of their coordinates are zero. This is the case with basket data transactions containing items, that can be represented as sparse binary vectors with very high dimensionality. An experimental section shows performance, advantages and limitations of the proposed approach.**

I. Introduction

Clustering algorithms identify those regions that are more densely populated than others in multidimensional data [8], [13]. In general clustering algorithms partition the data set into several groups such that points in the same group are close to each other and points across groups are far from each other. It has been shown that high dimensionality [10], data sparsity [1] and noise [2] make clustering a harder problem.

In this work we focus on the problem of efficiently clustering binary data sets that are sparse and have very high dimensionality. This is precisely the case with basket data transactions, where transactions contain combinations of a few items out of thousands of items. Our approach can be used as an alternative data mining technique to association rule discovery [3].

A. Overview

We introduce a fast clustering algorithm for sparse high dimensional binary data (basket data) based on the well-known Expectation-Maximization (EM) clustering algorithm [7], [17], [6], [13]. The EM algorithm is a general statistical method of maximum likelihood estimation [7], [14], [17]. In particular it can be used to perform clustering. In our case we will use it to fit a mixture of Normal distributions to a sparse binary data set.

Our algorithm is designed to efficiently handle large problem sizes as typically encountered in modern database systems and it is guaranteed to produce high quality solutions as it will be shown by our experiments.

http://doi.ieeecomputersociety.org/10.1109/ICDM.2001.989586

The proposed clustering algorithm builds a statistical model so that the user can understand transactions at a high level. Items are mapped to binary dimensions and transactions are thus mapped to binary data points. The basic idea is to group similar transactions. Clusters of transactions can have different interpretations. Each cluster can tell us what the typical transaction looks like; this is precisely the mean or average of transactions per cluster. Each cluster describes which items commonly appear together in each transaction. Since cluster centroids are averages of binary numbers the mean of a certain dimension can be interpreted as a probability or a percentage. If transactions are not well clustered in certain dimensions this can be explained by the deviation they have from the mean. The user will be able to compare several cluster models by looking at a quantity measuring their quality.

B. Contributions and paper outline

This is a summary of our contributions. We introduce a novel algorithm to cluster very high dimensional and sparse binary data sets. The proposed solution does not require complex data structures to store patterns or model parameters, but only matrices that in general can fit in memory. From a quality point of view the algorithm computes highly accurate clusters. From a performance point of view the algorithm is fast, having linear time complexity in data set size, in transaction size and in the desired number of clusters.

The rest of this paper is organized as follows. Section II provides definitions and statistical background. Section III contains the algorithm to cluster high dimensional and sparse binary data sets. Section IV contains a brief experimental evaluation. Section V discusses related work. The paper concludes with section VI.

II. Definitions and statistical background

This section provides formal definitions that will be used throughout this work. First, basic statistical background on EM and the mixture problem are described. Second, additional definitions relating transactions and multidimensional binary vectors are introduced.

The multivariate normal density function for a d-dimensional vector $x = [x_1, x_2, \ldots, x_d]^t$ is:

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp[-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)],$$

Matrix	size	contents
C	$d \times k$	means
R	$d \times k$	covariances
W	$k \times 1$	weights

TABLE I
OUTPUT MATRICES

Matrix	size	contents
N	$k \times 1$	$\|D_j\|$
M	$d \times k$	$M_j = \sum_{i=1}^{N_j} t_i, \forall t_i \in D_j$

TABLE II
SUFFICIENT STATISTICS

where μ is called the mean vector and Σ is called the covariance matrix; μ is a d-dimensional vector and Σ is a $d \times d$ matrix. Our algorithm uses diagonal covariance matrices.

The input to EM are n d-dimensional points and k, the desired number of clusters. These n points are modeled as a mixture of normal distributions as defined above. This mixture has 3 parameters, namely, the means, the covariances and the weights. Data set size, i.e. number of points, is n. The desired number of clusters is k. Dimensionality is d. The parameters computed by the EM algorithm are stored in the matrices described in table I. In the statistical literature all parameters are used as a single set called Θ, i.e. $\Theta = \{C, R, W\}$. To refer to one column of C or R we use the j subscript (i.e. C_j, R_j).

Since it is our intention to cluster basket data we will combine our previous definitions with additional definitions commonly used for association rules [3], [4]. Let $D = \{T_1, T_2, \ldots, T_n\}$ be a set of n transactions containing items, and let $\mathcal{I}$ be a set of d items, $\mathcal{I} = \{i_1, i_2 \ldots i_d\}$, where each item will be identified by its index, that is, an integer in $\{1, 2, \ldots, d\}$. Let $D_1, D_2, \ldots, D_k$ be k subsets of D (i.e. $D_j \subseteq D, j = 1 \ldots k$). s.t. $D_j \cap D_{j'} = \emptyset, j \neq j'$ (i.e. they are a partition of D induced by clusters). Each subset D_j represents one cluster. We use T_i to avoid confusion with t_i that will be used as a binary vector: T_i will be a set of integers and t_i will be a binary vector. Items will be mapped to binary dimensions. For each item $i_1, i_2, \ldots, i_d$ there will be a corresponding dimension b_l. Each transaction T_i will be given as a set of integers (items), $T_i = \{i_1, i_2, \ldots, i_K\}$, where $i_l \in \{1, \ldots, d\}$ and i (without subscript) denotes the number of transaction; $i \in \{1, 2, \ldots, n\}$. Then the notation t_i is used, meaning a binary vector, where each entry corresponds to one dimension (item). Then $(t_i)_l = 1$ for $l = i_1, i_2, \ldots, i_K$ and $(t_i)_l = 0$ otherwise. Each transaction becomes a sparse binary vector having d entries, but only K of them different from zero. So D in this case can be considered a huge and sparse $d \times n$ matrix. Each item i_l will be an integer, $i_l \in \{1, 2, \ldots, d\}$ to index matrices to refer to one dimension. Mathematically transactions will be points in $[0, 1]^d$ space, but for the algorithm they will be sets of integers.

III. A CLUSTERING ALGORITHM FOR BINARY DATA SETS WITH VERY HIGH DIMENSIONALITY

A. Improvements

We propose several improvements and changes on EM to deal with very high dimensionality, sparsity, null covariances, large data set size and slow convergence. Such improvements include suitable initialization for high dimensional data, sufficient statistics, covariance matrix regularization techniques, sparse distance computation and learning steps.

Input: $T_1, T_2, \ldots, T_n$ and k.
Output $\Theta = \{C, R, W\}$ and $L(\Theta)$
$\alpha \leftarrow (dk)^{-1}, L \leftarrow 50$
FOR $j = 1$ TO k DO /* Initialize */
  $C_j \leftarrow \mu \pm \alpha r$ diag$[\sigma]$, $R_j \leftarrow I$, $W_j \leftarrow 1/k$
  $\Delta_j = \delta(\bar{0}, C_j, R_j) = C_j^t R_j^{-1} C^j$
  $M_j \leftarrow C_j$, $N_j \leftarrow 1$
ENDFOR
FOR $scan = 1$ TO 2 DO
  $L(\Theta) = 0$
  FOR $i = 1$ TO n DO
    $t_i \leftarrow vect[T_i]$
    FOR $j = 1$ TO k DO /* E step */
      $\delta_{ij} \leftarrow \delta(t_i, C_j, R_j)$,
      $p_{ij} \leftarrow ((2\pi)^d |R_j|)^{-1/2} exp(-\delta_{ij}/2)$
    ENDFOR
    Let m be s.t.$p_{im} \geq p_{ij} \forall j \in 1 \ldots k$
    $M_m \leftarrow M_m + t_i$, $N_m \leftarrow N_m + 1$
    $L(\Theta) \leftarrow L(\Theta) + ln(p_{ij})$
    IF(i mod $(n/L) = 0$) THEN /* M step */
      FOR $j = 1$ TO k DO
        $C_j \leftarrow M_j / N_j$
        $R_j \leftarrow M_j/N_j - M_j M_j^t / N_j^2 + I$
        $W_j \leftarrow N_j / \sum_{J=1}^{k} N_J$
        $\Delta_j \leftarrow C_j^t R_j^{-1} C_j$
      ENDFOR
    ENDIF
  ENDFOR /* Reset sufficient statistics */
  IF scan=1 THEN $M_j \leftarrow C_j, N_j \leftarrow 1$ ENDIF
ENDFOR

Fig. 1. Clustering algorithm for sparse high dimensional binary data

Initialization is based on the global statistics of the data set: the global mean and the global covariance. They are computed in a one-time pass over the data set and are available thereafter. Seed centroids are initialized based on the global mean and standard deviation of the data. Sufficient statistics [12], [11] (table II) are used to summarize information about clusters; this reduces I/O time by avoiding repeated passes over the data and by allowing to make parameter estimation periodically as transactions are being read. The E step is executed for every transaction and the M step is executed a fixed number of times making convergence to the solution fast. The algorithm uses sparse distance computation and sparse matrix additions to make the E step faster. It uses regularization techniques [15] to deal with zero covariances, common with sparse data and specially with basket data. The algorithm requires two scans over the data per run. The main input parameter is only the desired number of clusters.

B. Algorithm to cluster sparse binary data sets with very high dimensionality

The pseudo-code of our clustering algorithm is in figure 1. This is a high-level description. The input is a set of transactions $D = \{T_1, T_2, \ldots, T_n\}$ and k, the desired number of

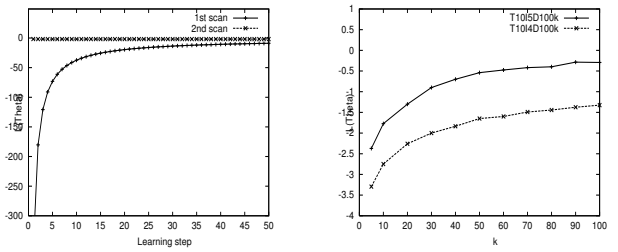

Fig. 2. Quality of results

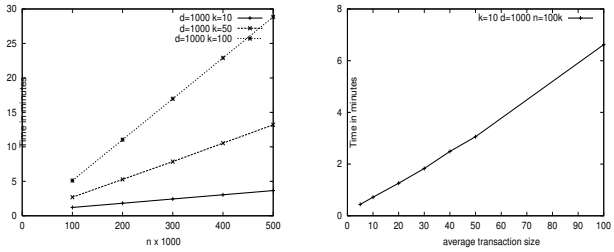

Fig. 3. Performance

clusters. The output is $\Theta = \{C, R, W\}$, describing the mixture model, $L(\Theta)$ measuring model quality and a partitioning of D into k subsets. The constant α is used to seed C based on d and k. The global statistics μ and Σ are computed in a one-time scan and are available thereafter. Standard deviations are computed as $\sigma_{ll} = \sqrt{\Sigma_{ll}}$. The E step is executed for every transaction (n times). δ_{ij} is efficiently computed using Δ_j. The M step is periodically executed every n/L transactions (L times). L is typically a number between 10 and 100. The update formulas for C, R, W are based on sufficient statistics [12] M, N, shown in table II, and regularization techniques [15]. M, N are the multidimensional version of the univariate sufficient statistics shown in [12] when points are binary; due to lack of space we do not explain how to derive their formulas. Sufficient statistics are reset at the end of the first scan. The goal of the first scan is to get accurate cluster centroids C_j and accurate covariances R_j. The goal of the second scan is to tune Θ and recompute $L(\Theta)$. Dimensions (items) are ranked within each cluster by their value in C_j to make output easier to understand.

IV. Experimental evaluation

This section includes experimental evaluation of our algorithm. All experiments were performed on a Sun Machine running at 600 MHz with 256 Mb of memory. This machine had several Gb of available disk space. Our algorithms were implemented in the C++ language and compiled with the GNU C++ compiler.

Our algorithm was evaluated with large transaction test files created by the well-known IBM synthetic data generator [4]. Test files are named after the parameters with which they were created. The standard way [4] is to use T (average transaction size), I (pattern length) and D (for us n) to label files since those are the most common parameters to change. The algorithm parameters were set as follows. $L = 50$ and $\alpha = 1/(dk)$.

In this paragraph we explain quality of results. The left graph in figure 2 shows how our algorithm converges on the 1st scan. The 2nd scan just tunes the solution without decreasing $L(\Theta)$. The right graph in figure 2 shows how model accuracy increases as k increases; the behavior is clearly asymptotic.

In this paragraph we describe performance with large data sets. Note that $d = 1000$ is a very high dimensionality. The left graph in figure 3 shows running as time as we vary n for several typical transaction files; the algorithm scales linearly. The right graph in figure 3 shows the impact of average transaction size (T) on performance; the algorithm is linear. Times varying k are also linear; this graph is not shown.

V. Related work

There has been so much work on both clustering and association rule mining that it is impossible to compare our approach with everybody else's. To the best of our knowledge there is no previous work on clustering *high dimensional and sparse large* binary data sets using EM. We do not know work where there are experiments with 1,000 or more dimensions [9], [10], [1], [2], [6]. Also, we believe that the idea of building a statistical model based on clustering for basket data has not been explored before. The only work that has analyzed how to cluster basket data transactions is [16]; their approach goes in the opposite direction since they mine associations and from them clusters are generated. We are not the first to propose a scalable and faster version of EM for data mining applications. One important work that also studied how to construct a faster and Scalable EM algorithm (SEM) is [6]. This work extended previous work on scaling K-means [5]. The authors present an algorithm, also based on sufficient statistics [12], that makes compression in two phases for dense and quasi-dense regions. The authors use it to build several models concurrently. SEM is significantly different from ours. It is designed for low dimensional continuous numerical data without zero covariance problems, and then it is not suitable for very high dimensional sparse binary data. It does not incorporate sparse distance computation, regularization techniques. Initialization is done by sampling and it keeps sufficient statistics on many subsets of the data, many more than k. Also, it uses an iterative K-means algorithm [14] to cluster data points in memory and then it does not make a fixed number of computations. One advantage over ours it that it only requires one scan over the data, but it makes heavier CPU use and it requires careful buffer size tuning.

VI. Conclusions

This paper presented a new clustering algorithm. The proposed algorithm is designed to work with *large* binary data sets having *very high* dimensionality. The algorithm only requires two scans over the data to cluster transactions and construct a statistical model. Each cluster is a summary of a group of similar transactions and thus represents one significant pattern discovered in the data. Experimental evaluation showed transactions can be clustered with high accuracy. Model quality mainly depends on k, the desired number of clusters. The

algorithm makes its best effort to get a high quality model given data characteristics. Performance is linear and it is mainly affected by n, k and transaction size, and minimally by dimensionality since data sets are sparse. The algorithm is restricted to problem sizes whose model can fit in main memory.

A summary of future work follows. Evidently some of our results can be applied to cluster high dimensional numerical data, but data skew, noise and cluster overlap make the problem different and to some extent more difficult. We plan to adapt and modify our algorithm to cluster continuous numeric data and compare it with the simplification of Scalable K-means [5] discussed in [9]. We would like to analyze the possibility of mining association rules from the model without scanning transactions.

Acknowledgments

This work was supported by grant LM 06726 from the National Library of Medicine.

REFERENCES

[1] C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD Conference*, pages 70–81, 2000.

[2] R. Agrawal, J. Gehrke, D. Gunopolos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD Conference*, pages 94–105, 1998.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference*, pages 207–216, 1993.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB Conference*, pages 487–499, 1994.

[5] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. ACM KDD Conference*, pages 9–15, 1998.

[6] P. Bradley, U. Fayyad, and C. Reina. Scaling EM clustering to large databases. Technical report, Microsoft Research, 1999.

[7] A.P. Dempster, N.M. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of The Royal Statistical Society*, 39(1):1–38, 1977.

[8] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. J. Wiley and Sons, New York, 1973.

[9] F. Fanstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *SIGKDD Explorations*, 2(1):51–57, June 2000.

[10] A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality. In *VLDB Conference*, pages 506–517, 1999.

[11] A. Mood, F. Graybill, and D. Boes. *Introduction to the Theory of Statistics*. McGraw Hill, NY, 1974.

[12] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. Technical report, Dept. of Statistics, University of Toronto, 1993.

[13] C. Ordonez and P. Cereghini. SQLEM: Fast clustering in SQL using the EM algorithm. In *Proc. ACM SIGMOD Conference*, pages 559–570, 2000.

[14] S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11:305–345, 1999.

[15] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton. SMEM algorithm for mixture models. In *Neural Information Processing Systems*, 1998.

[16] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *ACM CIKM Conference*, pages 483–490, 1999.

[17] L. Xu and M. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.