Data Mining using Matrices and Tensors (DMMT'09)

Proceedings of a Workshop held in Conjunction with
The 15th ACM SIGKDD International Conference on
Knowledge Discovery and Data Mining (SIGKDD 2009)

Paris, France, June 28, 2009

Edited by

Chris Ding
University of Texas at Arlington, USA

Tao Li
Florida International University, USA

ISBN 978-1-60558-673-1

Table of Contents

Preface

The ***2009 Workshop on Data Mining using Matrices and Tensors (DMMT'09)*** is the second workshop on this theme held annually with the SIGKDD Conference. Through the workshop, we expect to bring together leading researchers on many topic areas (e.g., computer scientists, computational and applied mathematicians) to assess the state-of-the-art, share ideas and form collaborations. We also wish to attract practitioners who seek novel ideals for applications. In summary, this workshop will strive to emphasize the following aspects:

- Presenting recent advances in algorithms and methods using matrix and scientific computing/applied mathematics

- Addressing the fundamental challenges in data mining using matrices and tensors

- Identifying killer applications and key industry drivers (where theories and applications meet)

- Fostering interactions among researchers (from different backgrounds) sharing the same interest to promote cross-fertilization of ideas

- Exploring benchmark data for better evaluation of the techniques

The field of pattern recognition, data mining and machine learning increasingly adapt methods and algorithms from advanced matrix computations, graph theory and optimization. Prominent examples are spectral clustering, non-negative matrix factorization, Principal component analysis (PCA) and Singular Value Decomposition (SVD) related clustering and dimension reduction, tensor analysis such as 2DSVD and high order SVD, L-1 regularization, etc. Compared to probabilistic and information theoretic approaches, matrix-based methods are fast, easy to understand and implement; they are especially suitable for parallel and distributed-memory computers to solve large scale challenging problems such as searching and extracting patterns from the entire Web. Hence the area of data mining using matrices and tensors is a popular and growing are of research activities. This workshop will present recent advances in algorithms and methods using matrix and scientific computing/applied mathematics for modeling and analyzing massive, high-dimensional, and nonlinear-structured data.

The Topic areas for the workshop include (but are not limited to) the following:

Methods and algorithms:

- Principal Component Analysis and Singular value decomposition for clustering and dimension reduction

- Nonnegative matrix factorization for unsupervised and semi-supervised learning
- Spectral graph clustering
- L-1 Regularization and Sparsification
- Sparse PCA and SVD
- Randomized algorithms for matrix computation
- Web search and ranking algorithms
- Tensor analysis, 2DSVD and high order SVD
- GSVD for classification
- Latent Semantic Indexing and other developments for Information Retrieval
- Linear, quadratic and semi-definite Programming
- Non-linear manifold learning and dimension reduction
- Computational statistics involving matrix computations
- Feature selection and extraction
- Graph-based learning (classification, semi-supervised learning and unsupervised learning)

Application areas

- Information search and extraction from Web
- Text processing and information retrieval
- Image processing and analysis
- Genomics and Bioinformatics
- Scientific computing and computational sciences
- Social Networks

This year, we selected 5 papers to include in the workshop program. All submissions were reviewed and discussed by two reviewers and workshop co-chairs.

We are very indebted to all program committee members who helped us organize the workshop and reviewed the papers very carefully. We would also like to thank all the authors who submitted their papers to the workshop; they provided us with an excellent workshop program. More information about the workshop can be found at: http://www.cs.fiu.edu/~taoli/kdd09-workshop/.

June 2009

Workshop Co-chairs

Chris Ding
Tao Li

Organization Committee

Workshop Co-Chairs

Chris Ding, University of Texas at Arlington, chqding@uta.edu
Tao Li, Florida International University, taoli@cs.fiu.edu

Publicity Chair

Fei Wang, Florida International University, feiwang@cs.fiu.edu

Program Committee

- Jesse Barlow, Penn State University
- Michael Berry, University of Tennessee
- Yun Chi, NEC Laboratories America
- Lars Elden, Linkping University, Sweden
- Christos Faloutsos, Carnegie Mellon University
- Estratis Gallopoulos, University of Patras
- Joydeep Ghosh, University of Texas at Austin
- Ming Gu, University of Califonia, Berkeley
- Michael Jordan, University of California, Berkeley
- Michael Ng, Hong Kong Baptist University
- Haesun Park, Georgia Tech
- Wei Peng, Xerox Research
- Robert Plemmons, Wake Forest
- Yousef Saad, University of Minnesota
- Horst Simon, Lawrence Berkeley National Laboratory
- Gang Wang, Microsoft Research
- Fei Wang, Florida International University
- Jieping Ye, Arizona State University
- Kai Yu, NEC Laboratories America
- Zhongyuan Zhang, Central University of Finance & Economics
- Shenghuo Zhu, NEC Laboratories America

Workshop Website: http://www.cs.fiu.edu/~taoli/kdd09-workshop/

Accuracy of Distance Metric Learning Algorithms

Frank Nielsen
École Polytechnique
Route de Saclay
91128, Palaiseau cedex, France
Sony CSL
Tokyo, Japan
nielsen@lix.polytechnique.fr

Aurélien Sérandour
École Polytechnique
Route de Saclay
91128, Palaiseau cedex, France
aurelien.serandour@polytechnique.org

ABSTRACT

In this paper, we wanted to compare distance metric-learning algorithms on UCI datasets. We wanted to assess the accuracy of these algorithms in many situations, perhaps some that they were not initially designed for. We looked for many algorithms and chose four of them based on our criteria. We also selected six UCI datasets. From the data's labels, we create similarity dataset that will be used to train and test the algorithms. The nature of each dataset is different (size, dimension), and the algorithms' results may vary because of these parameters. We also wanted to have some robust algorithms on dataset whose similarity is not perfect, whose the labels are no well defined. This occurs in multi-labeled datasets or even worse in human-built ones. To simulate this, we injected contradictory data and observed the behavior of the algorithms. This study seeks for a reliable algorithm in such scenarios keeping in mind future uses in recommendation processes.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering Algorithms, Similarity measures; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra, Singular value decomposition; G.1.6 [**Optimization**]: Constrained optimization

Keywords

Metric learning, Mahalanobis distance, Bregman divergences

1. INTRODUCTION

In many unsupervised learning problems, algorithms tend to find cluster to separate data, to label them. However there is sometimes no perfect boundaries between these assumed groups. They can easily overlap. For example, in music, even if some labels exist (classical, jazz, rock etc.), one can see each song as a combination of them. This can be annoying in a recommendation process because it becomes impossible to rely on these labels. In order to remove the definition of labels and avoid song clustering during the recommendation, one can see the problem as a similarity one. Given an entry point, a user can jump from a song to another in a logical way, respecting some similarity constrains. If a user is listening to a song, the most similar song can be recommended as the next one in an automatically generated playlist. Of course the area is not bound to music and can be applied to any recommendation problem.

The similarity is a binary evaluation: similar or dissimilar. Representing the similarity between two data as a distance function and a threshold is the most convenient way. Above a threshold, pair is a dissimilar one, under it, it is a similar one. So learning this distance function should give a solution to the problem. Many algorithms have been proposed, focusing on Mahanalobis distance. We decided to compare them on several datasets.

The similarity between data is not a mathematically defined attribute in music for example. It is more about feeling. So the sets have to be human-built ones. Unfortunately the human factor ensures that some randomness and inconsistencies will occur. This is an essential parameter in the recommendation process and it shouldn't be neglected.

2. NOTATION USED

We will use these notations all along:

$$
\begin{aligned}
d &= \text{dimension of the space}\\
\mathcal{S} &= \{(x_i, x_j) | x_i \text{ and } x_j \text{ are similar}\}\\
\mathcal{D} &= \{(x_i, x_j) | x_i \text{ and } x_j \text{ are dissimilar}\}\\
N &= \text{total number of points}\\
\mathcal{M}_d(\mathbb{R}) &= d \times d \text{ matrices in } \mathbb{R}\\
\mathcal{S}_d(\mathbb{R}) &= \left\{ M \in \mathcal{M}_d(\mathbb{R}) | M = M^T \right\}\\
\mathcal{S}_d^+(\mathbb{R}) &= \left\{ M \in \mathcal{S}_d(\mathbb{R}) | \forall X \in \mathbb{R}^d,\, X^T M X \geqslant 0 \right\}\\
\mathcal{S}_d^{++}(\mathbb{R}) &= \left\{ M \in \mathcal{S}_d(\mathbb{R}) | \forall X \in \mathbb{R}^d,\, X^T M X > 0 \right\}\\
\|\cdot\|_F &= \text{ Frobenius norm on matrices}\\
\|\cdot\|_A &= \text{ Mahalanobis distance with matrix A}\\
A \succeq 0 &\text{ means } A \in \mathcal{S}_d^+(\mathbb{R})
\end{aligned}
$$

3. MODUS OPERANDI

3.1 Datasets

DMMT'09, June 28, 2009, Paris.

We chose six datasets from the UCI[1] database.

- Iris[2]: low dimensional dataset
- Ionosphere[3]: high dimensional large dataset.
- Wine[4]: middle class dataset.
- Wisconsin Breast Cancer (WDBC)[5]: high dimensional large dataset.
- Soybean small[6]: high dimensional dataset with few data and many classes.
- Balance-scale[7]: low dimensional large dataset.

The details of these datasets can be seen in Tab. 1. Note that the final size of the similarity dataset is $\frac{N(N-1)}{2}$ where N is the number of points in the first one.

Table 1: Datasets attributes

DATASET	DIMENSION	SIZE	NUMBER OF CLASSES
Iris	4	150	3
Ionosphere	34	351	2
Wine	13	178	3
WDBC	32	569	2
Soybean small	35	47	4
Balance-scale	4	625	3

To remain as close as possible to music recommendation, we chose to use unlabeled datasets (even if this means removing labels on our own). In this study, only similarity is given.

3.2 Algorithms

We chose to compare the distance matrices generated from four algorithms based on our constraints: unlabeled data and similarity sets. The algorithms are:

- no algorithm: the identity matrix or Euclidean distance
- Xing's algorithm [7]: an iterative algorithm
- Information-Theoretic Metric Learning [3]: an iterative algorithm
- Coding similarity [6]

We chose them because of their different ways to formulate the similarity problem. This would give us an overview of what can be done today. We also studied other algorithms but decided not to incude them in this paper (see Appendix).

[1]http://archive.ics.uci.edu/ml/
[2]http://archive.ics.uci.edu/ml/datasets/Iris
[3]http://archive.ics.uci.edu/ml/datasets/Ionosphere
[4]http://archive.ics.uci.edu/ml/datasets/Wine
[5]http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
[6]http://archive.ics.uci.edu/ml/datasets/Soybean+(Small)
[7]http://archive.ics.uci.edu/ml/datasets/Balance+Scale

3.2.1 Xing's algorithm

This algorithm is the simplest someone can think about to solve the problem. The general idea is to minimize the distance between similar points and dissimilar ones. For that, we consider a distance matrix $A \in \mathcal{S}_d^+(\mathbb{R})$ and the following optimization problem.

$$\begin{aligned} \min_A & \sum_{(x_i,x_j)\in\mathcal{D}} \|x_i - x_j\|_A \\ \text{subject to } & \sum_{(x_i,x_j)\in\mathcal{S}} \|x_i - x_j\|_A^2 \leqslant 1 \\ & A \succeq 0 \end{aligned}$$

This formulation allows to put any condition we want on A. For example we can enforce A to be diagonal. This way, we can prevent overfitting, but perhaps decrease accuracy at the same time.

The optimization problem used to learn a full matrix is slighty different, but it is the same idea: move closer similar points and separate dissimilar ones.

$$\begin{aligned} \max_A \ g(A) = & \sum_{(x_i,x_j)\in\mathcal{S}} \|x_i - x_j\|_A^2 \\ \text{subject to } f(A) = & \sum_{(x_i,x_j)\in\mathcal{D}} \|x_i - x_j\|_A \geqslant 1 \\ & A \succeq 0 \end{aligned}$$

The details of the algorithm are described in the Appendix.

The main drawback is that the convergence is not sure. Sometimes, depending on the dataset or the initial conditions, it may not be able to gives a good result and get stuck in a loop where each iteration step gives a new matrix far from the previous one. On some datasets, it may find $A = 0$, which is not wanted.

In this paper, the algorithm was only run to learn full matrices.

3.2.2 Information Theoretic Metric Learning

This paper is one of the last published on this subject. It contains several new methods. The problem has evolved since its first publication. The last one, which is the one we used, is:

$$\begin{aligned} & \min_{A\succeq 0} D_{ld}(A, A_0) + \gamma \cdot D_{ld}(\text{diag}(\xi), \text{diag}(\xi_0)) \\ \text{subject to } & \|x_i - x_j\|_A \leqslant \xi_{i,j} \text{ if } (x_i, x_j) \in \mathcal{S} \\ & \|x_i - x_j\|_A \geqslant \xi_{i,j} \text{ if } (x_i, x_j) \in \mathcal{D} \end{aligned}$$

The Kullback-Leibler distance (or KL divergence) is a statistical distance between two distributions. The formula is:

$$D_{KL}(P\|Q) = \text{KL}(p_P(x)|p_Q(x)) = \int_\Omega P(x) \log\left(\frac{P(x)}{Q(x)}\right) dx$$
$$\text{and KL}(p(x;A_0)\|p(x;A)) = \frac{1}{2} D_{ld}(A, A_0)$$

ξ_0 is the set of thresholds defining the bound between similarity and dissimilarity and $\xi_{i,j}$ is the threshold for the pair (i, j), which can be in $\mathcal{S}$ or $\mathcal{D}$. γ controls the tradeoff between learning a matrix close to an arbitrary matrix A_0 and modifying the pre-computed threshold. This problem

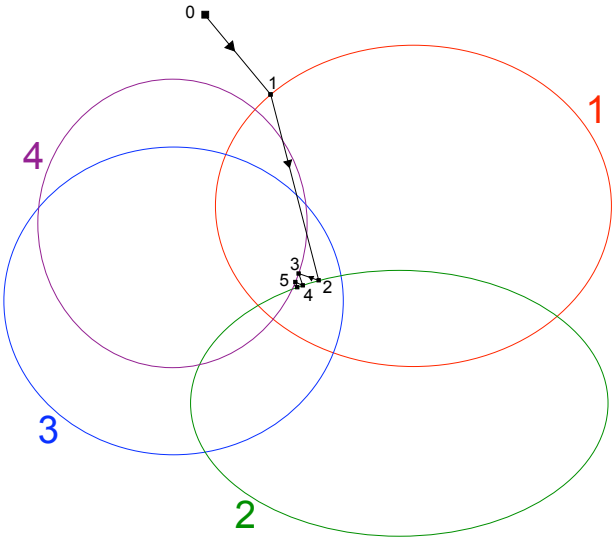


Figure 1: Projections on convex subspaces and convergence to a common intersection point (in the limit case)

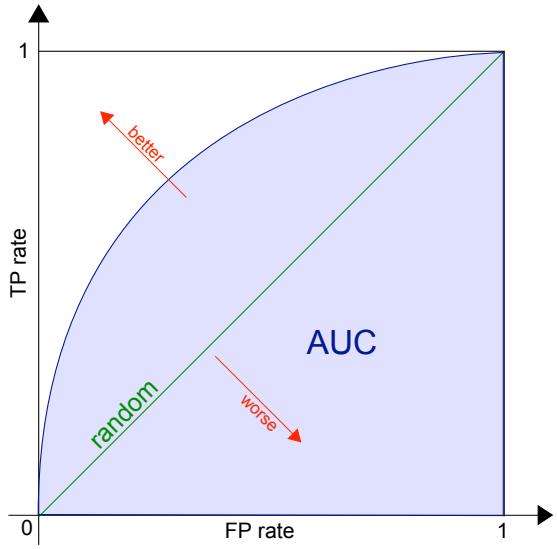


Figure 2: ROC curve

has many parameters which can be used to compute better results. However, it needs to decide at the beginning of the algorithm the values of A_0, ξ and γ. It is very difficult to guess them *a priori*.

The algorithm is an iterative method on each pair of $\mathcal{S}$ and $\mathcal{D}$. It performs successive projections on subspaces $\mathcal{S}_d^{(i,j)} = \left\{M \in \mathcal{S}_d^+ \,|\, \|x_i - x_j\|_M^2 \leqslant u\right\}$ (Fig. 1)[8] . The convergence is proven thanks to Bregman's research[2] in this field[9].

There is also an online version of this algorithm. We didn't report the result here since the accuracy was worse than the offline one. We can also enforce some constraints on A but for the same reasons, we kept with full matrices.

There are some problems such as the one in Xing's algorithm. However, they occur less frequently, almost never in fact.

3.2.3 *Coding Similarity*

This algorithm originally does not intend to learn a distance matrix. Finding a distance Matrix is just a consequence of it. The goal is to find a function that will estimate the similarity between two data. Coding similarity is defined by the amount of information a data contains about another: the similarity $\text{codsim}(x, x')$ is the information that x conveys about x'. This is a distribution based method. This amount of information is evaluated by the difference of coding length between two distribution: one where the *real* relation between x and x' is respected, and one there it is not. Note that this algorithm only use similar pairs. Coding length is $\text{cl}(x)$ is $-\log(p(x))$. The final definition is:

$$\begin{aligned}\text{codsim}(x, x') &= \text{cl}(x) - \text{cl}(x|x', H_1) \\ &= \log(p(x|x', H_1)) - \log(p(x))\end{aligned}$$

[8]for an animated applet, see http://www.lix.polytechnique.fr/~nielsen/BregmanProjection/

[9]for details, please see Stephen Boyd and Jon Dattorro course at Stanford University http://www.stanford.edu/class/ee392o/alt_proj.pdf

This algorithm can also performs dimension reduction to avoid overfitting.

Although it doesn't require iteration, the computation can be expensive (matrix inversion). Futhermore, since it requires symmetric matrix inversion, there can be some issues in this step. To avoid this, zero eigen values were set to a small amount.

In our tests, no dimension reduction was computed.

3.3 Tests and evaluation method

From each dataset, given the class of each data, we label each possible pair by similar or dissimilar pairs. These are the input of the algorithms. We performed a two fold cross-validation based on it. We also need a threshold to evaluate the similarity. However, the algorithms we use do not give one. So in the evaluation process, we want to remove the choice of one threshold. Given the distance matrix and several thresholds, we compute several confusion matrices. The threshold are chosen so that we describe the entire spectrum of interesting values: from the one that maps everything to dissimilar to the one that maps everything to similar. Then the Receiver Operating Characteristic (ROC) curve is computed and the accuracy of the model is given by the area under the ROC curve (AUC) (Figure 2).

For ITML, we need to initialize ξ_0. We set them as suggested the authors although it is a totally arbitrary choice: similarity at 5% of all pairs' distance distribution, dissimilarity at 95%.

We also wanted to study the effect of incoherent data on the overall result. The sets give perfect similarity whereas human-built ones may not pretend to this property. So we chose to insert some contradictory data by "flipping" the similarity of some pairs. The method does not add new pairs but modify the existing ones, so that the dataset does not have contradictory pairs but contains similarity evaluation errors.

Since we generated the inconsistencies at random, we chose to exclude them from the test set. In a real dataset, human-

built one, we cannot have this choice. However, this only reduce the results and does not help to compare the algorithms.

3.4 Software description

In addition to the AUC computation for both train and test sets, we created an application able to performs many tests (see Figure 3). It displays the ROC curve, the Precision and Recall curve and confusion matrices. The second curve gives an interesting threshold to separate the data. It can performs:

- Analysis of variance
- student's t-test: it compares the result of two algorithms and determines if there is a significant difference between them
- Tukey's test: same as student's t-test but compares several algorithms at the same time
- Spearman's rank correlation: it estimates if the data is correctly sorted with the computed distance.
- p-value

4. ALGORITHMS' DECRIPTION

4.1 Xing's algorithm

4.1.1 Algorithm for full matrix

Here we present the algorithm we derived to compute the full matrix.

Algorithm 1 Xing's algorithm for full matrix

repeat
  repeat
    $A := \text{argmin}_{A'} \left\{ \|A' - A\|_F : A' \in \mathcal{C}_1 \right\}$
    $A := \text{argmin}_{A'} \left\{ \|A' - A\|_F : A' \in \mathcal{C}_2 \right\}$
  until A converges
  $A := A + \alpha(\nabla_A g(A))_{\perp \nabla_A f}$
until convergence

where:

$$\mathcal{C}_1 = \left\{ A | \sum_{(x_i,x_j)\in\mathcal{S}} \|x_i - x_j\|_A^2 \leqslant 1 \right\}$$
$$\mathcal{C}_2 = \mathcal{S}_d^+(\mathbb{R})$$

Projection on $\mathcal{C}_1$.

$$\begin{aligned}
\delta &= \sum_{(x_i,x_j)\in\mathcal{S}} \|x_i - x_j\|_{A'}^2 \\
&= \sum_{(x_i,x_j)\in\mathcal{S}} \sum_{k=0}^{d} (x_{ik} - x_{jk}) \left[\sum_{p=0}^{d} a'_{kp}(x_{ip} - x_{jp}) \right] \\
&= \sum_{k,p} a'_{kp} \left[\sum_{(x_i,x_j)\in\mathcal{S}} (x_{ik} - x_{jk})(x_{ip} - x_{jp}) \right] \\
&= \sum_{k,p} a'_{kp} \beta_{kp}^{\mathcal{S}} \\
\text{where } \beta_{kp}^{\mathcal{S}} &= \sum_{(x_i,x_j)\in\mathcal{S}} (x_{ik} - x_{jk})(x_{ip} - x_{jp})
\end{aligned}$$

Now set up the Lagrangian:

$$\begin{aligned}
\mathcal{L}(\lambda) &= \left\|A - A'\right\|_F^2 + \lambda \left[\sum_{(x_i,x_j)\in\mathcal{S}} \|x_i - x_j\|_{A'}^2 - 1 \right] \\
&= \sum_{k,p} \left(a'_{kp} - a_{kp}\right)^2 + \lambda \left[\sum_{k,p} a'_{kp} \beta_{kp}^{\mathcal{S}} - 1 \right]
\end{aligned}$$

calculate the derivatives and solve the system $\mathfrak{S}$:

$$\begin{aligned}
\mathfrak{S} &\Leftrightarrow \begin{cases} \frac{\partial \mathcal{L}}{\partial a'_{kp}} &= 2\left(a'_{kp} - a_{kp}\right) + \lambda \beta_{kp}^{\mathcal{S}} = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= \left[\sum_{k,p} a'_{kp} \beta_{kp}^{\mathcal{S}} \right] - 1 = 0 \text{ or } \lambda = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} a'_{kp} = a_{kp} - \frac{\lambda \beta_{kp}^{\mathcal{S}}}{2} \\ \sum_{k,p} a'_{kp} \beta_{kp}^{\mathcal{S}} = 1 \text{ or } \lambda = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} a'_{kp} = a_{kp} - \frac{\lambda \beta_{kp}^{\mathcal{S}}}{2} \\ \sum_{k,p} \left[a_{kp} - \frac{\lambda \beta_{kp}^{\mathcal{S}}}{2} \right] \beta_{kp}^{\mathcal{S}} = 1 \text{ or } \lambda = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} a'_{kp} = a_{kp} - \frac{\lambda \beta_{kp}^{\mathcal{S}}}{2} \\ \sum_{k,p} -\frac{\lambda \beta_{kp}^{\mathcal{S}2}}{2} = 1 - \sum_{k,p} a_{kp} \beta_{kp}^{\mathcal{S}} \text{ or } \lambda = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} a'_{kp} &= a_{kp} - \frac{\lambda \beta_{kp}^{\mathcal{S}}}{2} \\ \lambda &= 2 \frac{\left(\sum_{k,p} a_{kp} \beta_{kp}^{\mathcal{S}} \right) - 1}{\sum_{k,p} \beta_{kp}^{\mathcal{S}2}} \text{ or } \lambda = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} a'_{kp} &= a_{kp} - \beta_{kp}^{\mathcal{S}} \frac{\left(\sum_{k,p} a_{kp} \beta_{kp}^{\mathcal{S}} \right) - 1}{\sum_{k,p} \beta_{kp}^{\mathcal{S}2}} \\ \lambda &= 2 \frac{\left(\sum_{k,p} a_{kp} \beta_{kp}^{\mathcal{S}} \right) - 1}{\sum_{k,p} \beta_{kp}^{\mathcal{S}2}} \text{ or } \lambda = 0 \end{cases}
\end{aligned}$$

The update is:

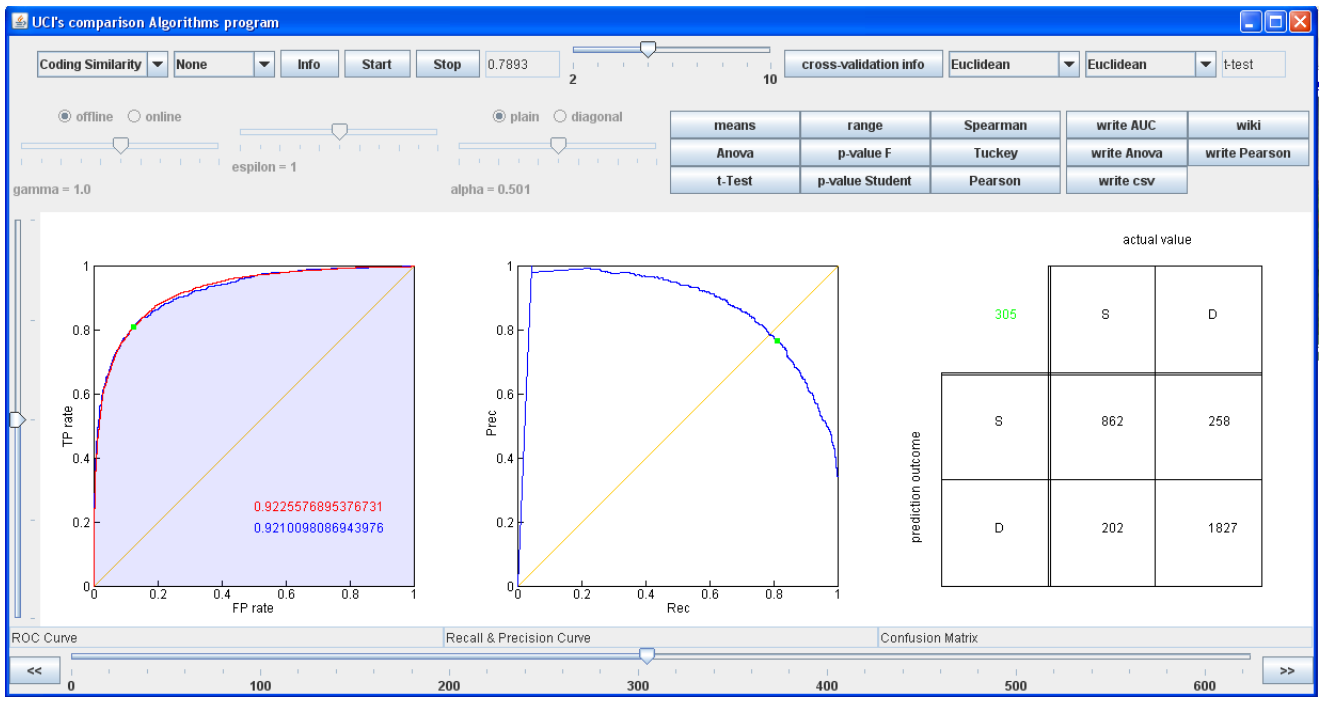


Figure 3: Snapshot of the software for benchmarking metric learning methods

$$a'_{kp} = a_{kp} + \beta^{\mathcal{S}}_{kp} \frac{1 - \sum_{k,p} a_{kp}\beta^{\mathcal{S}}_{kp}}{\sum_{k,p} \beta^{\mathcal{S}2}_{kp}} \text{ if } f(A) > 1$$

$$\text{where } \beta^{\mathcal{S}}_{kp} = \sum_{(x_i,x_j)\in\mathcal{S}} (x_{ik} - x_{jk})(x_{ip} - x_{jp}) = \beta^{\mathcal{S}}_{pk}$$

We can find an interesting formulation since:

$$\text{if } \beta^{\mathcal{S}} = \left(\beta^{\mathcal{S}}_{kp}\right)_{k,p} = \sum_{(x_i,x_j)\in\mathcal{S}} (x_i - x_j)(x_i - x_j)^T$$

$$\sum_{k,p} \beta^{\mathcal{S}2}_{kp} = \left\|\beta^{\mathcal{S}}\right\|_F^2$$

$$\sum_{k,p} a_{kp}\beta^{\mathcal{S}}_{kp} = \sum_{k,p} a_{kp}\beta^{\mathcal{S}}_{pk} = \text{trace}\left(A\beta^{\mathcal{S}}\right)$$

We get the final formulation:

$$\boxed{A' = A + \beta^{\mathcal{S}} \frac{1 - \text{trace}\left(A\beta^{\mathcal{S}}\right)}{\|\beta^{\mathcal{S}}\|_F^2}} \text{ if } f(A) > 1 \text{ , else } \boxed{A' = A}$$

Projection on $\mathcal{C}_2$.

set the negative eigen values to 0:

$$\boxed{\begin{array}{l} A = X\Lambda X^T \text{ where } \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d) \\ \Lambda' = \text{diag}\left(\max(\lambda_1, 0), \max(\lambda_2, 0), \ldots, \max(\lambda_d, 0)\right) \\ \text{set } A = A' = X\Lambda' X^T \end{array}}$$

Here it can be difficult to avoid $A = 0$. This was a common issue in this algorithm.

Gradient ascent.

$$\overrightarrow{\nabla}_A g(A) = \begin{pmatrix} \frac{\partial g}{\partial a_{1,1}} & \cdots & \frac{\partial g}{\partial a_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g}{\partial a_{n,1}} & \cdots & \frac{\partial g}{\partial a_{n,n}} \end{pmatrix} (A)$$

$$\text{where } \frac{\partial g}{\partial a_{kp}} = \frac{\partial}{\partial a_{kp}} \left[\sum_{\mathcal{D}} \sqrt{\sum_{u,v} a_{uv}(x_{iu} - x_{ju})(x_{iv} - x_{jv})}\right]$$

$$= \sum_{\mathcal{D}} \frac{(x_{iu} - x_{ju})(x_{iv} - x_{jv})}{2\sqrt{\sum_{u,v} a_{u,v}(x_{iu} - x_{ju})(x_{iv} - x_{jv})}}$$

$$\text{so } \overrightarrow{\nabla}_A g(A) = \sum_{(x_i,x_j)\in\mathcal{D}} \frac{(x_i - x_j)(x_i - x_j)^T}{2\|x_i - x_j\|_A}$$

$$\text{where } \frac{\partial f}{\partial a_{kp}} = \frac{\partial}{\partial a_{kp}} \left[\sum_{\mathcal{S}} \sum_{u,v} a_{uv}(x_{iu} - x_{ju})(x_{iv} - x_{jv})\right]$$

$$= \sum_{\mathcal{S}} (x_{iu} - x_{ju})(x_{iv} - x_{jv}) = \beta^{\mathcal{S}}_{uv}$$

$$\text{so } \overrightarrow{\nabla}_A f(A) = \beta^{\mathcal{S}}$$

We have the final formulation:

$$\left[\overrightarrow{\nabla}_A g(A)\right]_{\perp \overrightarrow{\nabla}_A f} = \overrightarrow{\nabla}_A g(A) - \left[\frac{\overrightarrow{\nabla}_A g \cdot \overrightarrow{\nabla}_A f}{\left\|\overrightarrow{\nabla}_A f\right\|^2} \overrightarrow{\nabla}_A f\right] (A)$$

which can be written:

$$\left[\overrightarrow{\nabla}_A g(A)\right]_{\perp \overrightarrow{\nabla}_A f} = \overrightarrow{\nabla}_A g(A) - \frac{\overrightarrow{\nabla}_A g(A) \cdot \beta^{\mathcal{S}}}{\|\beta^{\mathcal{S}}\|_F^2} \beta^{\mathcal{S}}$$

4.2 Algorithm for diagonal matrix

If we want to learn a diagonal matrix $A = \text{diag}\,(a_{1,1} \dots a_{n,n})$, we just look for minimizing the function $h(A)$:

$$h(A) = h\,(a_{1,1} \dots a_{n,n})$$

$$= \sum_{(x_i,x_j)\in\mathcal{S}} \|x_i - x_j\|_A^2 - \log\left(\sum_{(x_i,x_j)\in\mathcal{D}} \|x_i - x_j\|_A\right)$$

We use a Raphson-Newton method to find the minimum of h. However, because of the log function, we cannot have $\exists i, a_{i,i} < 0$ or $\forall i, a_{i,i} = 0$. So monitoring the Newton-Raphson algorithm prevents this case (Fig. 4).

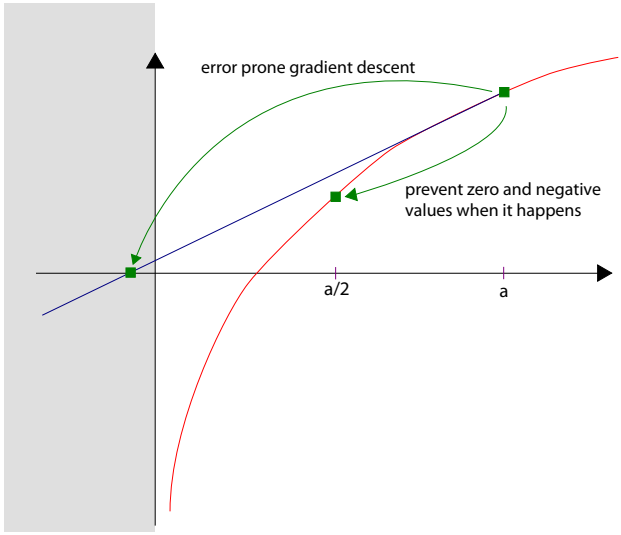


Figure 4: monitoring the Newton-Raphson algorithm

5. CODING SIMILARITY

The main advantage of this method is that there is no iteration. It is very fast in low dimension. However, due to the matrices inversions, high dimension problems can be very slow. Also notice that due to precision errors, in the program I had to force the distance matrix to be perfectly symmetric thanks to the update: $A = \frac{A+A^T}{2}$.

We also modified this algorithm in order to learn a diagonal matrix. We finally removed that option, the results were not significantly different.

6. RAW RESULTS

The tables below gives the AUC for each algorithm and for several percentages of errors in the dataset. We only put these results here since the ones from the other tests weren't as relevant as these ones. The noise rate corresponds to the percentage of input pairs whose similarity label was "flipped".

Algorithm 2 Coding Similarity algorithm

1: let $L = \#\mathcal{S}$
2: $Z = \frac{1}{2L}\sum_{\mathcal{S}}(x_i - x_j)$
3: remove Z to each data
4: $\Sigma_x = \frac{1}{2L}\sum_{\mathcal{S}}\left[x_i x_i^T + x_j x_j^T\right]$
5: $\Sigma_{xx'} = \frac{1}{2L}\sum_{\mathcal{S}}\left[x_i x_j^T + x_j x_i^T\right]$
6: $\Sigma_\Delta = \frac{\Sigma_x - \Sigma_{xx'}}{2}$
7: Distance matrix A is $(4\Sigma_\Delta)^{-1} = [2\,(\Sigma_x - \Sigma_{xx'})]^{-1}$

6.1 Ionosphere

Table 2: Ionosphere results

noise rate	Euclidean	Xing	ITML	CodeSim
0	0.645	**0.8499**	0.7913	0.7787
5	0.645	**0.8322**	0.5322	0.7752
10	0.645	0.7208	0.5106	**0.7728**
20	0.645	0.6273	0.4988	**0.7683**
25	0.645	0.7175	-	**0.7662**
30	0.645	0. 6792	-	**0.7639**

The results are written in table 2. The Euclidean norm gives poor results, so learning a distance is interesting. Xing algorithm performs well but shows low robustness when incoherent data occur, whereas Coding Similarity almost gives constant reliability.

6.2 Iris

Table 3: Iris results

noise rate	Euclidean	Xing	ITML	CodeSim
0	0.9395	0.9646	**0.9837**	0.9652
5	0.9395	0.8233	**0.9825**	0.9013
10	0.9395	0.7173	**0.9672**	0.86
20	**0.9395**	0.6014	0.5214	0.7977
25	**0.9395**	0.5964	0.5964	0.7775
30	**0.9395**	0.5775	0.5514	0.7637

The results are written in table 3. The size of the small Iris dataset explains the abrupt decrease in similarity accuracy. Given the result of the Euclidean norm, using a learnt distance can be dangerous.

6.3 Wine

Table 4: Wine results

noise rate	Euclidean	Xing	ITML	CodeSim
0	0.7624	0.7694	0.8104	**0.9219**
5	0.7624	0.7697	0.6777	**0.8369**
10	0.7624	0.7682	0.7436	**0.7837**
20	0.7624	**0.775**	0.5753	0.711
25	0.7624	**0.779**	0.6698	0.6845
30	0.7624	**0.7789**	0.6075	0.6737

The results are written in table 4. Coding simlarity performs well with correct data. Surprisingly, Xing's algorithm gives the best results with really bad data. However, the accuracy is too close to the one from the Euclidean distance.

6.4 WDBC

Table 5: WDBC results

noise rate	Euclidean	Xing	ITML	CodeSim
0	0.8204	0.8358	**0.8567**	0.7231
5	0.8204	**0.8291**	0.7381	0.7012
10	**0.8204**	0.8199	0.634	0.6864
20	**0.8204**	0.795	0.7425	0.6659
25	**0.8204**	0.7852	0.6408	0.6594
30	**0.8204**	0.7766	0.6764	0.6527

The results are written in table 5. This dataset seems diffucult because neither the euclidean or the learnt distances give good results. In this case, using the Euclidean distance is the safest way to evaluate similarity.

6.5 Soybean-small

Table 6: Soybean-small results

noise rate	Euclidean	Xing	ITML	CodeSim
0	0.9417	0.9999	**1**	**1**
5	0.9417	0.9187	**1**	0.9539
10	0.9417	0.9146	**0.999**	0.8388
20	0.9417	0.8008	**0.9836**	0.7527
25	**0.9417**	0.7271	0.9274	0.7468
30	**0.9417**	0.6487	0.7849	0.7183

The results are written in table 6. The Euclidean distance gives good results enough to consider using another norm, which can behave pretty badly.

6.6 Balance-scale

Table 7: Balance-scale results

noise rate	Euclidean	Xing	ITML	CodeSim
0	0.6583	0.4146	**0.7771**	0.7476
5	0.6583	0.3946	0. 5135	**0.7356**
10	0.6583	-	0.5126	**0.7225**
20	0.6583	-	0.5269	**0.7012**
25	0.6583	-	-	**0.6914**
30	0.6583	-	-	**0.6848**

The results are written in table 7. With many wrong data, Xing's algorithm is unable to converge (at least in a decent time or number of iterations), such as ITML. This can be a result of the size of the dataset.

7. ANALYSIS OF RESULTS

First of all, with good data, the distance from the chosen algorithms almost always gives better results than the Euclidean distance. This confirms (if needed) the results published by their authors.

On small datasets (Iris, Soybean small), ITML performs well. However, the gain compared to the Euclidean distance is low. The tradeoff between a good distance and a safe one matters. In these case, perhaps the safest distance is the Euclidean one.

If we don't know anything about the set, the Euclidean norm can give random results. The Coding Similarity algorithm performs well in most cases and shows reasonable reliability. It has also the advantage to compute the distance without iterations. The process is really fast and cannot be caught in an infinite optimization loop.

8. CONCLUSIONS

The main result is this study that data drive the accuracy of the algorithms. No algorithm tends to dominate the other ones. Futhermore, results on well-defined sets may not represent the behavior on human-built ones. Who should define the similarity other than users when no class exists?

Because of possible errors in real datasets, one would choose an algorithm which shows a good robustness to the data's inconsistencies. However and once again, the data seem to decide what is the best algorithm.

The method to inject incoherent data can be discussed. We thought it was a fast and good way to simulate partially bad datasets. How bad can be the result on a similarity set which was created from by human being? This is really difficult to evaluate. The similarity may not be understand as a binary evaluation (similar / not similar). It can also not be seen as a quadratic function. The similarity evaluation errors can follow a pattern and not be totally distributed at random. Also these "choices" may be personal. If the training set was created by a unique user, the distance matrix reflects his definition of the similarity. Data can be close or far in a continuous way. Futhermore, this study is limited to Mahanalobis distances. Maybe a good "distance" is one which is not quadratic.

The result of some algorithms should not discourage to use them. If coding similarity performs well, it can be used to learn a good distance function and one can adjust A with new data captured on-the-fly with for example the online version of ITML.

So the difficulty to learn a good distance should not prevent from trying to use it. There is also a room left for further experiments with human-built datasets and non-quadratic distances. It may have many application in future recommendation processes, especially on-the-fly ones.

9. ACKNOWLEDGMENTS

We thank Sony Japan for this research carried out during an internship.
Financially supported by ANR-07-BLAN-0328-01 GAIA (Computational Information Geometry and Applications) and DIGITEO GAS 2008-16D (Geometric Algorithms & Statistics)

10. REFERENCES

[1] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *J. Mach. Learn. Res.*, 6:937–965, 2005.

[2] L. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. In *USSR Computational Math. and Math. Physics*, 1967.

[3] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 209–216, New York, NY, USA, 2007. ACM.

[4] A. Globerson and S. Roweis. Metric learning by collapsing classes, 2005.

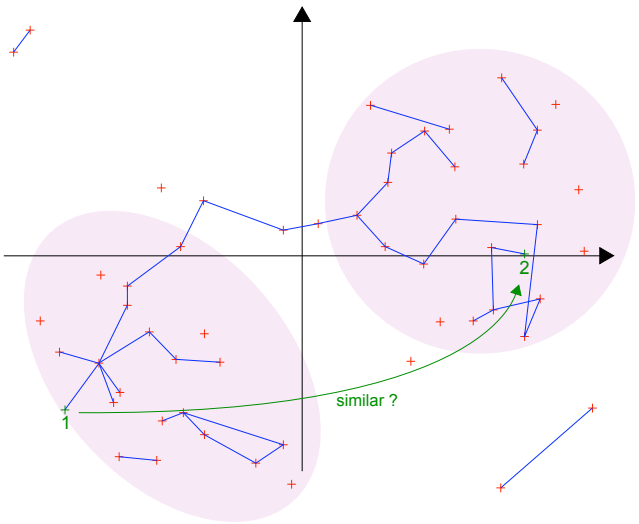


Figure 5: similarity closure

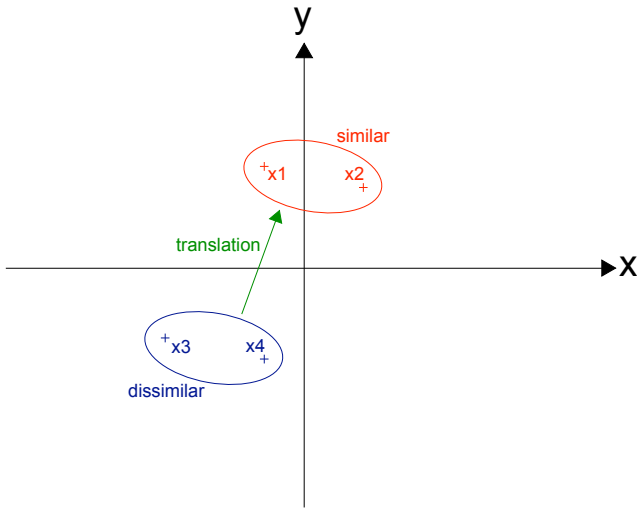


Figure 6: The distance between the points of each pair are the same, even if their similarity is not

[5] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2004.

[6] A. B. Hillel and D. Weinshall. Learning distance function by coding similarity. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 65–72, New York, NY, USA, 2007. ACM.

[7] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

[8] L. Yang and R. Jin. An efficient algorithm for local distance metric learning. In *Proceedings of AAAI*, 2006.

APPENDIX

A. OTHER ALGORITHMS STUDIED

Many algorithms focus on looking for a distance. However, many of them can't be applied in our scenario. Here are some of the interesting ones we studied but at the end didn't use.

A.1 Learning a Mahalanobis Metric from Equivalence Constraints

This simple algorithm[1] does not require a large amount of equivalence constraints since it creates them. However, they are created from initial small groups of points (called *chunklets*) and these groups are extended thanks to the transitive closure of each. However, if there is no label on the points, the groups are not well defined. The transitive closure can reach the entire set(Fig. 5)... In our datasets, these labels exists but in music space, they are too fuzzy.

In fact, this algorithm is strongly related to *Coding Similarity* algorithm which is only an extension.

A.2 Neighbourhood Component Analysis

The problem formulation is very interesting here[5]. The goal is not to move closer similar points and separate dissimilar ones, but the goal is directly one application of learning a distance: maximize the k-Nearest Neighbour cross-validation accuracy. This is close to the traditional formulation of distance learning but this is not identical. However, labeled data are required, which does not fit our model.

A.3 Metric Learning by Collapsing Classes

This algorithm's goal[4] is to find the closest distance matrix to an ideal distance d_0 which perfectly separates the points. With a chosen Mahanalobis matrix A, we can define the distance $d^A_{i,j} = d_A(x_i, x_j) = \|x_i - x_j\|^2_A$ and the distribution $p^A(j|i) = \frac{e^{d^A_{i,j}}}{\sum_{k \neq i} e^{d^A_{i,k}}}$.

$$\min_{A \succeq 0} \sum_i \mathrm{KL}\left[p_0(j|i) \| p^A(j|i)\right]$$

$$\text{where } p_0(j|i) = \begin{cases} 0 & \text{if } (x_i, x_j) \in \mathcal{S} \Leftrightarrow d_0(x_i, x_j) = 0 \\ 1 & \text{if } (x_i, x_j) \in \mathcal{D} \Leftrightarrow d_0(x_i, x_j) = \infty \end{cases}$$

This algorithm is supposed to used labeled data, however simple similarity constraints are enough. However, several precision errors[10] makes it difficult to use.

A.4 An Efficient Algorithm for Local Distance Metric Learning

This paper[8] is perhaps one of the most interesting we came across, but it is at the same time one of the most difficult. The purpose is to locally learn a distance to prevent unsolvable cases such as translated points (Fig. 6). In this last figure, each pair has the same distance between its points. If one is similar, the other is dissimilar, it becomes impossible to solve the learning problem. This algorithm wasn't used because of lack of time but could give interesting results.

[10] $\sum_{k \neq i} e^{d^A_{i,k}}$ often exceeds `double` precision

A Spectral-based Clustering Algorithm for Categorical Data Using Data Summaries

Eman Abdu
Computer Science Department
The Graduate Center
The City University of New York
eabdu@gc.cuny.edu

Douglas Salane
Mathematics & Computer Science
John Jay College of Criminal Justice
The City University of New York
(212) 237-8836
dsalane@jjay.cuny.edu

ABSTRACT
We present a novel spectral-based algorithm for clustering categorical data that combines attribute relationship and dimension reduction techniques found in Principal Component Analysis (PCA) and Latent Semantic Indexing (LSI). The new algorithm uses data summaries that consist of attribute occurrence and co-occurrence frequencies to create a set of vectors each of which represents a cluster. We refer to these vectors as "candidate cluster representatives." The algorithm also uses spectral decomposition of the data summaries matrix to project and cluster the data objects in a reduced space. We refer to the algorithm as SCCADDS (Spectral-based Clustering algorithm for CAtegorical Data using Data Summaries). SCCADDS differs from other spectral clustering algorithms in several key respects. First, the algorithm uses the attribute categories similarity matrix instead of the data object similarity matrix (as is the case with most spectral algorithms that find the normalized cut of a graph of nodes of data objects). SCCADDS scales well for large datasets since in most categorical clustering applications the number of attribute categories is small relative to the number of data objects. Second, non-recursive spectral-based clustering algorithms typically require K-means or some other iterative clustering method after the data objects have been projected into a reduced space. SCCADDS clusters the data objects directly by comparing them to candidate cluster representatives without the need for an iterative clustering method. Third, unlike standard spectral-based algorithms, the complexity of SCCADDS is linear in terms of the number of data objects. Results on datasets widely used to test categorical clustering algorithms show that SCCADDS produces clusters that are consistent with those produced by existing algorithms, while avoiding the computation of the spectra of large matrices and problems inherent in methods that employ the K-means type algorithms.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Clustering.*

General Terms
Algorithms

Keywords
Spectral Algorithms, Categorical Data

1. INTRODUCTION
Clustering categorical data, i.e., data in which attribute domains consist of discrete values that are not ordered, is a fundamental problem in data analysis. Despite many advances and the vast literature in the clustering of data objects with numerical domains, clustering categorical data, where there is neither a natural distance metric nor geometric interpretation for clusters, remains a significant challenge. In addition, categorical clustering presents many of the same difficulties found in clustering numerical data, e.g., high dimensionality, large data sets and the complexity of the discrete clustering problem. Moreover, to be effective most algorithms for clustering categorical data often require the careful choice of parameter values, which makes these algorithms difficult to use by those not thoroughly familiar with the method.

Recent algorithms specifically designed to cluster categorical data include the K-modes algorithm [16], STIRR [13], CACTUS [12], ROCK [15], COOLCAT [3], LIMBO [1], CLICKS [26], and others[1]. The K-modes algorithm is an extension of the simple and widely used K-means clustering algorithm and naturally suffers from the same problems as K-means, namely convergence to a local minimum, and reliance on initial cluster seeds as well as record order. On the other hand, CACTUS and CLICKS use attribute value co-occurrence frequencies to discover dense regions – regions with support greater than the expected support[2]. STIRR is an algorithm based on a non-linear dynamical system that first clusters attribute values and then clusters data objects in

[1] Clustering algorithms are usually classified by the type of cluster structure they produce: hierarchical or partitional. Hierarchical algorithms (agglomerative or divisive) form a tree-like cluster structure where each child node is a sub-cluster of its parent node, examples include single link and complete link algorithms. By contrast, partitional methods produce flat unrelated clusters, for example, the K-means algorithm. ROCK and LIMBO are examples of hierarchical algorithms designed for categorical data.

[2] Support is defined as the number of data objects that contain a given set of attributes. Expected support is calculated based on the assumed probability distribution for the data set.

a post-processing step. COOLCAT and LIMBO are based on information entropy. LIMBO is a hierarchical algorithm that builds on the Information Bottleneck [24] to detect the clustering structure in a dataset. Each of the aforementioned clustering methods offers a different methodology to cluster categorical data with varying degrees of success in terms of cluster quality, time/space complexity, difficulty in parameter setting, and additional clustering features. ROCK is quadratic in terms of the number of data objects which makes the algorithm unsuitable for large datasets. In a comparative analysis between LIMBO and COOLCAT, Andritsos et al. [1] showed that LIMBO outperforms COOLCAT in terms of parameter stability and clustering quality. Moreover, COOLCAT relies on sampling which may result in lower quality clustering if the sample mostly contains outliers. Zaki et al. [26] present a comparative study of CLICKS, CACTUS, ROCK and STIRR where they showed that CLICKS outperformed all of these algorithms in terms of clustering quality and scalability on synthetic datasets. As for additional clustering features, CLICKS is distinguished by its ability to mine subspace clusters.

In this paper, we present a different methodology for clustering categorical data. We combine the use of data summaries and spectral techniques to design a clustering algorithm that is scalable to large data sets. Spectral techniques offer an approximate solution to the discrete clustering problem (NP-hard) by relaxing some of the constraints for the partitioning optimization problem. Spectral clustering has been shown to outperform other traditional algorithms [6]. The algorithm presented here has two main features. First, it uses data summaries that consist of attribute occurrence and co-occurrence frequencies to create a set of vectors each of which represents a cluster. We refer to these vectors as "candidate cluster representatives." Second, it uses spectral decomposition of the data summaries matrix to project and cluster the data objects in a reduced space. Unlike most non-recursive spectral-based clustering algorithms, which typically require K-means or some other iterative clustering method after the data objects have been projected into a reduced space, our algorithm clusters the data objects directly by comparing them to the candidate cluster representatives and does not require an iterative clustering method. We use the acronym SCCADDS (Spectral-based Clustering algorithm for CAtegorical Data using Data Summaries) to refer to the algorithm.

The following is an outline of the paper. Section 2 presents an overview of spectral clustering and provides motivation for the new algorithm. Section 3 gives the mathematical preliminaries needed to describe SCCADDS. In Section 4, we present the new algorithm. Section 5 examines the accuracy of SCCADDS clusterings for standard test data sets and evaluates the performance of SCCADDS on synthetically generated test data sets. Section 6 compares the quality of clusters produced by SCCADDS and several well-known algorithms for clustering categorical data.

2. SPECTRAL CLUSTERING

Spectral clustering is a term used to characterize clustering methods that employ a few eigenvectors or singular values of a matrix. Spectral clustering for numerical data has been an intensive area of research during the past decade with major contributions coming from various communities including the genetics, image processing, information retrieval and the parallel computing communities. Researchers often show that their clustering method finds a minimum cut in a weighted undirected graph where the nodes are data objects and the weights associated with the edges reflect the similarity between data objects. Much work focuses on the use of spectral techniques to find a relaxed solution to the K-means clustering problem. The relaxed solution then serves as a starting solution for an iterative procedure that determines the final clusters. Another approach uses a single eigenvalue to partition a similarity graph and compute two clusters. Such methods then apply this procedure recursively to produce finer clusters. This approach allows sparsity of the similarity matrix to be exploited and iterative techniques to be used to compute the eigenvector if the similarity graph is large (See, for example, [6]). A key issue in spectral clustering, and one that differentiates many of the methods, is the construction of the similarity graph [25].

The key feature of spectral clustering is that it finds a global optimum and as such clustering methods based on spectral techniques are not prone to the problem of convergence to local minima, which often occurs in coordinate descent methods such as the K-means. There are two approaches for deriving a relaxed spectral solution for the discrete clustering problem. The first approach minimizes the cut (normalized cut or ratio cut [25]) of a graph where the data objects are the graph nodes and the edges represent the association between the data objects. The second approach minimizes the sum of the squares of the error (SSE) between the data objects. Zha et al.[27] show that the minimization of SSE can be formulated as a trace maximization of the Gram matrix (data objects similarity matrix) whose solution is the k eigenvectors associated with k largest eigenvalues. Furthermore, Ding and He [9] show that the principal components (PCA) are the continuous cluster membership indicators. Shi and Malik [22] present a spectral clustering algorithm for image segmentation that minimizes the normalized cut[3] of a graph-based normalized similarity matrix (Laplacian matrix) of the data objects. They show that the eigenvector associated with the second smallest eigenvalue contains the continuous solution for the cluster membership indicator vector. The algorithm recursively bi-partitions the graph based on the second smallest eigenvector. [9], [20], and [27] present k-way clustering algorithms that use the k eigenvectors (associated with the k largest eigenvalues) of the data objects similarity matrix; these algorithms rely on the K-means to cluster the data objects in the reduced space. Ng et al. [20] recommend choosing k equal to the expected number of clusters whereas Ding et. al.[9] recommend k to be 1 less than the number of expected clusters.

The approach presented here uses spectral methods to cluster attribute values based on the frequency of their co-occurrence in data objects. Co-occurrence frequencies provide a natural mechanism for defining an attribute value similarity matrix. Our algorithm relies on reduced rank SVD subspace projections to build clusters of similar attribute values. The ability of SVD subspace projection to amalgamate similar terms and separate dissimilar terms has been exploited in various clustering applications (see, for example [4], [5], [10] and [18]), but not for general categorical data clustering. The approach here is motivated both by the LSI literature and previous work on categorical clustering that attempts to build clusters of data objects by first forming clusters of attribute values. Our

[3] A normalized cut is the ratio of the cut cost to the total edge connections between the nodes in the graph partition and all other nodes in the graph.

experiments on widely used categorical test datasets show the algorithm is capable of determining high quality clusters in a reduced space without resorting to a K-means or other iterative methods. The algorithm is linear in the number of data objects and we demonstrate its scalability on synthetic data sets. In addition, we believe the resulting algorithm is particularly well suited to criminal justice data applications where the number of attributes often is small relative to the number of data objects.

3. PRELIMINARIES

In this section, we present the mathematical preliminaries needed to describe the method. In handling categorical data, we follow a common approach that converts categorical data objects into binary vectors [21]. Each indicator corresponds to one value in the domain of each attribute[4]. A dataset of n data objects in d dimensions determines an m x n matrix A where each column corresponds to a data object. The value m is equal to the number of dimensions after binary expansion. (We note that matrix A need not be formed in memory or on disk; it only is introduced in this paper to describe the method. Please see the appendix)

The Singular Value Decomposition (SVD) of matrix A [14] is written as

$$A = USV^t$$

where U is an m x m matrix, S is an m x n diagonal matrix and V is an n x n matrix. The columns of U are called the left singular vectors of A and are the eigenvectors of AA^t whereas the columns of V are called the right singular vectors of A and are the eigenvectors of A^tA. Both U and V are orthogonal matrices (i.e. $U^tU = I$ and $V^tV = I$, where I is the identity matrix.) The diagonal elements of S, the singular values, are arranged in a descending order.

For a positive integer k, an approximation to A, matrix A_k can be constructed as follows:

$$A_k = U_kS_kV^t_k$$

where k is a positive integer, S_k is the k x k matrix of the k largest singular values and U_k and V_k have the columns that correspond to these singular values. U_k is an m x k matrix and V_k is an n x k matrix. The Eckart-Young Theorem [11] states that A_k is the best rank-k matrix approximation of A in the Frobenius-norm (matrix 2-norm). In other words, the truncated matrix A_k provides the best rank-k least squares approximation to A.
We note that

$$U_k^tA = S_kV_k^t \quad \text{or} \quad A^tU_k = V_kS_k$$

and

$$AV_k = U_kS_k$$

The rank-k matrix U_kS_k is the projection of the attribute vectors into a lower dimensional space (k-dimensional space) using axes defined in matrix V_k Similarly, the rank-k matrix V_kS_k is the projection of the data object vectors into a k-dimensional space defined by matrix U_k. From a factor-analysis point of view, matrices V_k and U_k contain the principal components of the data objects and principal directions, respectively, provided that the zero-centered vectors are used instead of raw binary vectors.

[4] Most practical datasets in data mining contain categorical attributes with few values. For attributes with large domains, low-frequency attributes often can be eliminated and not impact the cluster analysis.

4. ALGORITHM AND DISCUSSION

The following description of SCCADDS assumes that we have a binary vector representation of the data matrix A as defined in section 3. We note that matrix A does not have to be formed. Since our purpose here is simply to demonstrate the efficacy of the method, we describe the algorithm and present details of an optimal implementation elsewhere. The following is an overview of the SCCADDS algorithm.

SCCADDS Algorithm

1. *Normalize and zero-center each row in matrix A.*
2. *Compute k eigenvectors of the attribute categories covariance matrix that are associated with the k largest eigenvalues. Matrix AA^t is the attribute categories covariance matrix. Compute matrix $U_kS_k^2$ whose rows contain the projected rows of the attribute categories covariance matrix.*
3. *Project each data object into the new space using the k eigenvectors.*
4. *Compute the similarity between each projected data object and each projected row of the attribute categories covariance matrix. The similarity is defined as the dot product of two vectors. Each row of the attribute categories covariance matrix represents a cluster. Assign each data object to the cluster with the maximum similarity.*
5. *Merge the clusters. Create a new cluster centroid for each group created in step 4 and use it to determine the final clusters. (This step is optional and is included just to reduce the number of clusters, if desired.)*

For the remainder of the paper, we assume that matrix A contains the zero-centered attribute value rows. Using the definitions in section 3, the algorithm computes the top k left singular vectors of matrix A (matrix U_k). Also, we will use the term "attributes" to refer to attribute categories. Note that the attribute covariance matrix can be computed using attribute occurrence and co-occurrence frequencies (please see appendix).

SCCADDS is motivated by the observation presented in [8] regarding the "Duality of word & document clustering." The observation simply brings to light the premise that documents clusters are induced by term clusters and vice versa. In our view, the setting of categorical data clustering is identical to that of documents clustering where attribute categories are represented by terms and data objects are represented by documents. As such, following Dhillon's observations, we can use attribute clusters to cluster data objects in categorical data sets. A similar concept is used in the methodology in CACTUS and CLICKS. The early phases of CLICKS and CACTUS mainly find those sets of attributes that are strongly connected based on their occurrence and co-occurrences frequencies in the data objects. These sets of attributes are then used to form "candidate" clusters' representatives which are later validated by scanning the data objects for the presence of these attributes.

To find attributes clusters, we form the attributes covariance matrix where each cell in the matrix reflects the covariance between two attributes. As shown in [27], the first k

eigenvectors associated with the k largest singular values of the data objects similarity matrix form the optimal solution for the non-discrete clustering problem (clusters membership indicators matrix). Since the goal is to cluster the attributes, the solution is U_kQ where Q is an arbitrary orthogonal matrix. If Q is the identity matrix I, then U_k is a solution to the problem. We then can project the attribute categories into the space defined by the cluster membership indicators vectors ($AA^tU_k = U_kS_k^2$). Each row of matrix $U_kS_k^2$ corresponds to an attribute category and as such it contains the coordinates for each attribute category in the space defined by the columns of U_k. Each column of U_k defines a cluster membership indicator vector and each value in that column defines the intensity of the membership of each attribute category in that cluster. It follows that similar attribute categories would have similar coordinates with respect to each column of U_k. Each row of $U_kS_k^2$ is considered a cluster representative since it contains the coordinates for the attribute categories in the new space (each row of the attribute categories covariance matrix represents a cluster on that dimension where all data objects in the cluster have the same value for that attribute). Since it is not possible to directly compare the data objects to these clusters' representatives (columns of U_k), SCCADDS takes the approach of projecting the data objects into the same space (A^tU_k) as the attributes. The algorithm clusters the data objects by comparing the data objects in the new reduced space to the attributes in the new space ($U_kS_k^2$).

Our work has been motivated by the success of Latent Semantic Indexing (LSI) [7], a spectral-based algorithm that discovers latent information (relationships) among attributes and documents in Information Retrieval (IR). LSI is based on Singular Value Decomposition (SVD) and has been used successfully in document retrieval [4, 7] and to cluster data sets that arise in law enforcement applications [5, 23]. In LSI, documents are selected based on their similarity to a query vector in the reduced space.[5] In the case of document retrieval, the main goal of LSI is the detection of hidden relationships (indirect relationships) between terms, documents, or documents and terms. These hidden or high order relationships are captured in the reduced term and documents matrices [19]. Unlike LSI, we do not use raw occurrence/co-occurrence frequencies. In LSI, the matrix U_kS_k contains the term vectors and the matrix V_kS_k contains the document vectors. Each row of U_kS_k represents a term and each row of V_kS_k represents a document. Similarly, in SCCADDS, we calculate matrix $U_kS_k^2$ which is similar to matrix U_kS_k (term matrix) but differing only by a scaling factor of S_k. In LSI, the columns of U_k represent some artificial concepts that are present in a data set. In clustering terminology, these artificial concepts represent clusters representatives. Since $U_kS_k^2 = (AA^t)U_k$, it follows that the rows of $U_kS_k^2$ represent the rows of AA^t projected into a space defined by the orthogonal axes defined by the columns of U_k (clusters representatives). The rows of AA^t correspond to the attributes; the rows of $U_kS_k^2$ represent the attributes in the new reduced space. Note that in LSI, queries are considered pseudo-documents and as such are projected into the new space prior to being compared to the data objects. Similarly, SCCADDS projects the attributes (rows of AA^t) into the new space and then compares them to the data objects.

In SCCADDS, we find the scores for each data object using the principle components for the data set ($A^tU_k=V_kS_k$). Principal Component Analysis(PCA)[17] allows us to represent each data object using a small number of dimensions that capture the most variations in the data set. Ding and He (2004) showed that the first k eigenvectors that are associated with the k largest eigenvalues of the Gram Matrix (A^tA) form the continuous solution for the cluster membership indicators problem. Using SVD decomposition of $A=USV^t$, the relaxed solution would be the first k eigenvectors of matrix V. By SVD decomposition of A, matrix $A^tU_k=V_kS_k$. Therefore, in SCCADDS, the rows of the matrix A^tU_k have two interpretations: 1) (PCA interpretation) they contain the scores for each data object using the first k principal components; 2) (continuous cluster indicators) they are coordinates for the data objects in a space defined by the attributes clusters membership indicators.

The last step in SCCADDS, which is optional, is to merge the clusters. The merge method begins by re-computing a cluster centroid for each of the newly defined clusters. A cluster centroid is defined as the mean of all data objects vectors in a cluster and is computed using the reduced (projected) data objects vectors. The input to the algorithm is a matrix of the cosine similarities of each pair of cluster centroids and a merge threshold. The algorithm merges clusters that have a cosine similarity that is above the threshold. A threshold of one, for example, would merge only identical clusters.

5. EVALUATION

We have developed a MATLAB® (version 2008a, the MathWorks Inc., Natick, MA) implementation of SCCADDS. Tests with the implementation on several UCI Machine Learning Repository [2] datasets that are used extensively to evaluate the quality of categorical clustering algorithms show SCCADDS produces highly accurate clusters. For the Soybean, Congressional Votes, and the Mushroom datasets, SCCADDS produces clusters that either are more accurate than or comparable to published methods. The Soybean dataset contains 47 data objects with 21 categorical attributes that describe 4 categories of soybean diseases. Three of the clusters contain 10 data objects each; the fourth cluster contains 17 data objects. The Congressional Votes dataset contains 435 data objects with 16 categorical attributes. The dataset contains the 1984 United States congressional voting records. Each data object is classified as either democrat (267) or Republican (168). Each attribute can have a value of "yes" or "no". The data set has some missing data (undecided votes). One data object did not contain any values (all values were missing) and as such the data object was excluded from the analysis. The Mushroom dataset contains 8,124 data objects classified as edible or poisonous mushrooms. The dataset has 22 categorical attributes with multiple values in the domain. There are 4,208 data objects in the edible class and 3,916 in the poisonous class. In the data set, 2,480 attribute values are missing. SCCADDS treats missing values as additional domain values.

Following [16], we calculate clustering accuracy (recall) as

$$\frac{1}{n}\sum_{i=1}^{h} a_i$$

where a_i is the number of data objects in cluster i that are properly classified, n is the number of data objects in the dataset and h is number of clusters. To calculate the accuracy level, each cluster is assigned to one of the clusters known a priori. We will refer to these pre-defined clusters as classes. A data object is classified properly if it belongs to the class assigned to its cluster.

[5] A query vector is a weighted or binary vector of search terms.

For the Soybean dataset, SCCADDS achieves an accuracy level of 100% using the top 2 eigenvectors. As for the Congressional Votes and Mushroom datasets, the method attains accuracy levels of 88% and 89%, respectively, using the top eigenvector. For the Soybean dataset, the clusters created using the top 2 eigenvectors are perfect sub-clusters of the clusters created using only the top eigenvector. Note that Table 1 presents the results for the case where the number of clusters constructed is equal to the number of pre-defined classes. Higher accuracy levels are attainable if the number of clusters is not restricted to the number of predefined classes (see Figures 1 and 2). In general, for all three standard datasets, as the number of eigenvectors (k) used in SCCADDS increases, the clustering granularity increases (number of clusters) as well as clustering accuracy. For the Mushroom dataset, the clustering accuracy is 99% or above for all k values greater than 17 with a merge threshold set at 50%. Figure 1 and Figure 2 show the results of multiple executions of SCCADDS using different number of eigenvectors. From these plots, it can be seen that the clustering accuracy increases as more eigenvectors are used for the Congressional Votes and Mushroom datasets, respectively. As for the Soybean dataset, clustering accuracy remains at 100% for k greater than or equal to 2 as shown in Figure 3.

Table 1. SCCADDS on Soybean, Congressional, and Mushroom data sets

Dataset Name	Number of records	Number of eigenvectors	Number of clusters	Accuracy
Soybean	47	2	4	100%
Congressional Votes	434	1	2	88%
Mushroom	8,124	1	2	89%; 99% with 19 clusters using 17 eigenvectors.

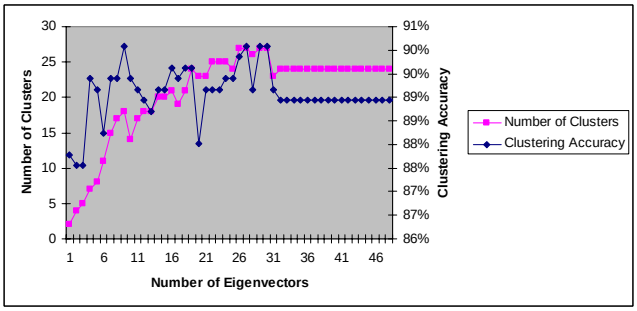


Figure 1. SCCADDS on the Congressional Votes data set with different values of k (number of eigenvectors)

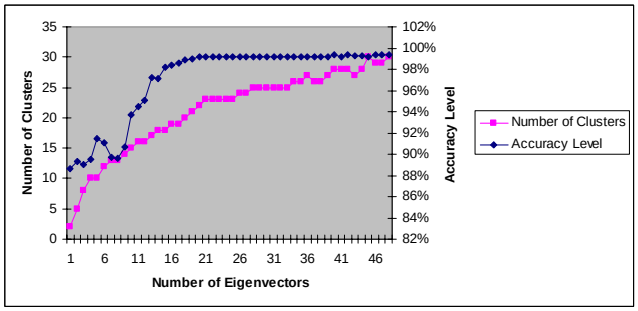


Figure 2. SCCADDS on the Mushroom data set with different values of k (number of eigenvectors)

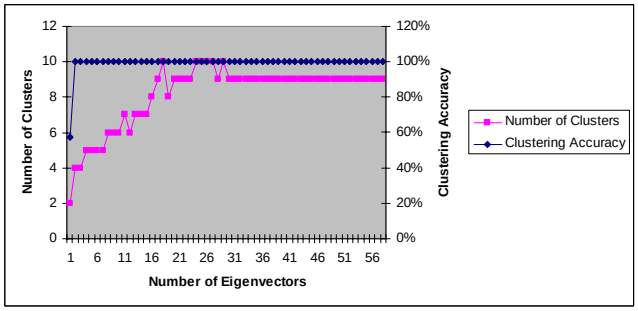


Figure 3. SCCADDS on the Soybean data set with different values of k (number of eigenvectors)

5.1 SCCADDS on Synthetic Data Sets

We use synthetic datasets[6] to further test the clustering quality and scalability of SCCADDS. We created five synthetic data sets with different characteristics to test the clustering quality. The datasets are described in table 2.

Table 2. Synthetic data sets used in quality testing

Data Set Name	Domain size of attributes	Number of Attributes	Number of attributes that participate in a rule	Number of Clusters	Dataset Size	Noise Ratio
DS1	10-20	10	4	5	1000	0
DS2	10-20	10	4	10	5000	0
DS3	10-20	10	5	10	5000	2%
DS4	10-20	10	5	10	5000	10%
DS5	10-20	20	10	10	5000	10%

We compared the clustering quality of SCCADDS and Eigencluster (a spectral-based clustering algorithm).[7] On all datasets, SCCADDS outperformed Eigencluster as shown in Table 3. For SCCADDS, the table shows the minimum number of eigenvectors needed to achieve the number of clusters and clustering accuracy shown in the table. For example, for

[6] We use the synthetic data generator available at http://www.datgen.com.

[7] Eigencluster is a spectral-based algorithm that uses the divide-and-merge methodology presented in [6]. Eigencluster is based on the spectral algorithm presented in [18]. The engine for Eigencluster is available at http://arc2.cc.gatech.edu/cluster.html.

synthetic dataset DS1, SCCADDS found five clusters with an accuracy level of 100% using 4 or more eigenvectors.

To test the scalability of SCCADDS, we created several datasets with different sizes in terms of the number of attributes and data objects. First, we tested the effect of a change in the number of attributes on the performance of SCCADDS. We created five synthetic data sets with 5000 data objects in each data set. Each data set has a different number of attributes. The domain size for each attribute is 10. The description of the datasets and the elapsed time of the execution of SCCADDS on these datasets are shown in Table 4. The last two columns of Table 4 show the percent of change in the number of attribute categories and the corresponding percent of change in elapse time. The change in elapse time is almost linear with the change in the number of attribute categories that is because the number of attribute categories is small relative to the number of data objects. In other words, most of the elapsed time is spent in comparing the data objects to candidate cluster representatives.

Table 3. SCCADDS vs. Eigencluster on synthetic data sets

	SCCADDS			Eigencluster		
Data Set Name	Number of eigen vectors	Number of clusters found	Accuracy	Alpha	Number of clusters	Accuracy
DS1	4	5	100%	0.4	5	93%
DS2	9	10	99%	0.4	10	91%
DS3	9	10	100%	0.4	10	91%
DS4	9	10	99%	0.4	10	85%
DS5	7 9	10 10	99% 100%	0.2 .35	9 11	82% 95%

Table 4. Scalability testing – Number of attributes

				SCCADDS			
Data Set Name	Number of Data Objects	Number of Attributes	Domain	Number of Attributes Categories	Elapsed Execution Time (sec.)	% change in Number of attributes	% of Change in Elapsed Execution Time
DS1A	5000	5	10	38	4.2	0	0
DS2A	5000	10	10	94	7	147.37%	66.67%
DS3A	5000	15	10	150	10.9	59.57%	55.71%
DS4A	5000	20	10	200	14.1	33.33%	29.36%
DS5A	5000	30	10	300	22.5	50.00%	59.57%

Next, we created 10 data sets that range in size from 6,500 to 65,000 data objects to test the scalability of SCCADDS in terms of data set size. Figure 4 illustrates that the change in elapse time is linear with the change in data set size (number of data objects).

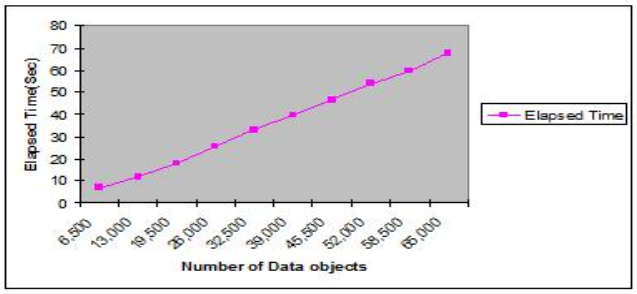


Figure 4. Scalability testing – Number of data objects

5.2 Cluster Compactness and Separation

We created two metric to evaluate the similarity within and between clusters (cluster compactness and separation). We use the cosine similarity metric to measure the intra-cluster and inter-cluster similarity. For each cluster, we calculate the intra-cluster similarity by taking the average cosine similarity between the data objects and the associated cluster center. The cluster center is simply the average of the values of the data objects vectors. For inter-cluster similarity, we calculate the weighted average of the inter-cluster similarity taking into account the size of each cluster. For a pair of clusters, the inter-cluster similarity is the cosine similarity between their cluster centers. These metrics illustrate how compact and separated the clusters are. The inter-cluster similarity metric can have a value between 0 and 1 with a value closer to 0 indicating that the clusters are well separated. Similarly, the intra-cluster similarity metric can have a value between 0 and 1 with a value closer to 1 indicating high similarity between the data objects in the cluster. In general, the objective is to have high intra-cluster similarity and low inter-cluster similarity. We evaluated SCCADDS using these metrics. On the Soybean, Congressional Votes, and Mushroom datasets, we noticed that a decrease in inter-cluster similarity and an increase in intra-cluster similarity as we ran SCCADDS using more eigenvectors. The ratio between intra-cluster to inter-cluster similarity peaked at the level where we achieved the highest accuracy level and optimal number of clusters (1 eigenvector for the congressional data set and 2 eigenvectors of the Soybean data set). Table 5 shows the results for these metrics for 5 runs of SCCADDS using 1 to 5 eigenvectors on the Soybean data set.

Table 5. Intra-cluster and inter-cluster similarity of clusters produced by SCCADDS - Soybean data set

Number of Eigenvectors	Number of Clusters	Accuracy	intra-cluster similarity	Inter-cluster similarity	Ratio of intra/inter similarity
1	2	57%	79%	58%	136%
2	4	100%	87%	52%	169%
3	4	100%	87%	52%	169%
4	5	100%	88%	55%	159%
5	5	100%	88%	56%	157%

Table 6. Intra-cluster and inter-cluster similarity of clusters produced by SCCADDS – Synthetic data set DS5

Number of Eigenvectors	Number of Clusters	Accuracy	intra-cluster similarity	Inter-cluster similarity	Ratio of intra/inter similarity
1	2	21%	38%	41%	92%
2	7	40%	47%	36%	131%
3	10	46%	50%	29%	169%
4	10	53%	53%	25%	214%
5	10	63%	56%	18%	310%
6	10	71%	59%	16%	375%
7	10	99%	66%	15%	439%
8	10	99%	66%	15%	433%
9	10	100%	67%	15%	451%
10	10	100%	67%	15%	451%
11	10	100%	67%	15%	451%
12	10	100%	67%	15%	451%

Table 6 shows the results of these metrics on the synthetic data set DS5. DS5 has a noise ratio of 10%. Note that the intra-cluster to inter-cluster similarity also peaks at the highest accuracy level and optimum number of clusters (10) for this data set. Based on our experiments, we believe that these metrics can be good indicators of the number of eigenvectors that may be used to obtain a clustering that increase cluster separation and similarity within clusters.

6. COMPARATIVE RESULTS

We compared the clustering quality of SCCADDS with the clustering quality of several well-known algorithms for clustering categorical data. On standard datasets SCCADDS produces clusters that are either comparable to or more accurate than these algorithms. Tables 7 and 8 present clustering accuracy and entropy[8] of SCCADDS and comparable algorithms such as COOLCAT, CLICKS, LIMBO, ROCK, and Eigencluster [6].[9] Tables 7 and 8 show the results for the Congressional Votes and Mushroom datasets. Note that the results shown in these tables are for the case where the number of desired clusters is restricted to the number of pre-defined clusters. (Please see the remarks column in tables 7 and 8 for accuracy levels when the number of clusters is not restricted to the number pre-defined classes.) For the Congressional Votes dataset, SCCADDS achieves the highest accuracy level and lowest entropy (88% and .45, respectively) of all the algorithms presented. With the exception of LIMBO, SCCADDS performs better than all other algorithms presented in the table for the Mushroom dataset and achieves an accuracy level of 89%.

Table 7. Clustering accuracy results for the Congressional Votes data set (2 clusters)

	Accuracy	Entropy	Remarks
SCCADDS	88%	0.452	This is the result for 2 clusters. Higher accuracy levels may be achieved with a higher number of clusters
Traditional Hierarchical Clustering Algorithm	86%		157 and 215 records were properly classified [15].
ROCK	79%	0.499	144 and 201 records were properly classified. 62 records were excluded as they were considered outliers and were not classified [15]. We use a dataset size of 434 when we calculated the accuracy level for ROCK to provide comparability with other algorithms presented. Entropy results were obtained from [6].
LIMBO	87%		Recall results obtained from [1].
COOLCAT	85%	0.498	Recall results are from [1]. Entropy results are from [6].
CLICKS	Not available for 2 clusters	Not available for 2 clusters	An accuracy level of 96% is achieved with 13 clusters. One of the 13 groups is designated for outliers [26].
Eigencluster		0.48	Entropy results were obtained from [6].

[8] Clustering entropy is computed as defined in [6].

[9] The clustering accuracy and entropy results for these algorithms were obtained from several publications. The sources are noted in the table.

7. CONCLUSION

We presented a spectral-based algorithm for clustering categorical data that uses data summaries (SCCADDS). The algorithm applies spectral techniques to the attribute categories covariance matrix and is practical for large datasets. Results on standard test datasets show the algorithm produces highly accurate clusters and is very competitive with other algorithms for clustering categorical data. SCCADDS provides an approach to clustering data objects through applying spectral analysis of the attribute-based proximity matrix instead of a data objects-based similarity matrix. Constructing a data objects-based similarity matrix can be computationally expensive and impractical for large data sets. SCCADDS is linear in terms of the number of data set objects but quadratic in terms of the number of attribute categories. We believe that SCCADDS offers an approach that is viable for large datasets with a moderate number of attributes and domain size. Additionally, we are researching the applicability of SCCADDS in a fuzzy clustering setting where we can define the intensity with which a data object belongs to a cluster.

Table 8. Clustering accuracy results for the Mushroom data set (2 clusters)

	Accuracy	Remarks
SCCADDS	89%	This is the accuracy level for 2 clusters. Accuracy level is 99% with 19 clusters using 17 eigenvectors.
ROCK	57%	This is the clustering recall result with 2 clusters [1]. Guha et al. [15] implemented Rock that partitioned the dataset into 21 clusters and could not further combine the clusters to form the two pre-defined classes; only 32 records were not clustered properly.
LIMBO	89%	Clustering recall for 2 clusters obtained from [1].
COOLCAT	73%	Clustering recall for 2 clusters is from [1].
CLICKS	Not available for 2 clusters	An accuracy level of 97% is achieved with 19 clusters [26].
Eigencluster	81%	Precision and recall [6].

8. ACKNOWLEDGEMENTS

This work is supported in part by National Science Foundation grant 0430444 and the Center for Cybercrime Studies at John Jay College.

9. REFERENCES

[1] Andritsos, P., Tsaparas, P., Miller, R., and Sevcik, K. 2004. LIMBO: Scalable Clustering of Categorical Data. The Ninth International Conference on Extending Database Technology (EDBT), pages 123-146.

[2] Asuncion, and Newman, D.J. 2007. UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. DOI=http://www.ics.uci.edu/~mlearn/MLRepository.html

[3] Barbara, D., Li, Y., and Couto, J. 2006. COOLCAT: an Entropy-based Algorithm for Categorical Clustering. In

Proceedings of the eleventh International Conference on Information and Knowledge Management, pages 582-589.

[4] Berry, M., Dumais, S., and O'brien, G. 1995. Using Linear Algebra for Intelligent Information Retrieval. SIAM Review, Vol. 37, No. 4, pages 573-595.

[5] Bradford, R.B. 2006. Relationship Discovery in Large Text Collections Using Latent Semantic Indexing. SIAM conference on Data Mining, workshop on Link Analysis, Counterterrorism and Security. DOI=http://www.siam.org/meetings/sdm06/workproceed/Link%20Analysis/15.pdf

[6] Cheng, D., Kannan, R., Vempal, S., and Wang, G. 2006. A Divide-and-Merge Methodology for Clustering. ACM Transactions on Database Systems, Vol. 31, No. 4, pages 1499-1525.

[7] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. 1990. Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science. Vol. 41, pages 391-407.

[8] Dhillon, I. 2001. Co-clustering Documents and Words Using Bipartite Spectral Graph Partitioning. In Proceedings of Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, California, USA, page 269-274.

[9] Ding, C., and He, X. 2004. K-means Clustering via Principal Component Analysis. In proceedings of the 21st International Conference on Machine Learning, Canada, pages 29-36.

[10] Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. 2004. Clustering Large Graphs via the Singular Value Decomposition. Machine Learning, Vol. 56, pages 9-33.

[11] Eckart, C., and Young, G. 1936. The approximation of one matrix by another of lower rank. Psychometrika, Vol.1, pages 183-187.

[12] Ganti, V., Gehrke, J., and Ramakrishnan, R. 1999. CACTUS Clustering Categorical Data Using Summaries. In proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 73-83.

[13] Gibson, D., Kleinberg, J., and Raghavan, P. 1998. Clustering Categorical Data: An Apporach Based on Dynamical Systems. In proceedings of the 24rd International Conference on Very Large Data Bases, pages 311-322.

[14] Golub, G. H., and Van Loan, C. F. 1996. Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore.

[15] Guha, S., Rastogi, R., and Shim, K. 1999. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia.

[16] Huang, Z. 1998. Extensions to the K-means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery, Vol. 2, pages 283-304.

[17] Jolliffe, I.T. 2002. Principal Component Analysis, 2nd edition, Springer Series in Statistics, Springer , New York, USA.

[18] Kannan, R., Vempala, S., and Vetta, A. 2004. On Clusterings: Good, Bad, and Spectral. Journal of the ACM Vol. 51, No. 3, pages 497-515.

[19] Kontostathis, A., and Pottenger, W. 2006. A Framework for Understanding Latent Semantic Indexing (LSI) Performance. Information Processing and Management, Volume 42, Issue 1, pages 56-73.

[20] Ng, A., Michael, J., and Weiss, Y. 2002. On Spectral Clustering: Analysis and an Algorithm. In Advances in Neural Information Processing Systems (NIPS) 14, MIT Press, page 849-856.

[21] Ralambondrainy, H. 1995. A conceptual version of the K-means algorithm. Pattern Recognition Letters, Vol. 16, pages 1147–1157.

[22] Shi, J., and Malik, J. 2000. Normalized Cuts and Image Segmentation. IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 22, No. 8, pages 888-905.

[23] Skillicorn, D. 2004. Applying Matrix Decomposition to counterterrorism. External Technical Report, ISSN-0836-0227-2004-484 Queen's University, Kingston, Canada.

[24] Tishby, N., Pereira, F. C., and Bialek, W. 1999. The Information Bottleneck Method. In 37th Annual Allerton Conference on Communication, Control and Computing, Urban-Champaign, IL.

[25] Von Luxburg, U. 2006. A Tutorial on Spectral Clustering. Technical Report No. TR-149, Max-Planck-Instiut für biologische Kybernetik, Tübingen, Germany.

[26] Zaki, M., Peters, M., Assent, I., and Seidl, T. 2007. Clicks: An effective Algorithm for Mining Subspace Clusters in Categorical Datasets. Data & Knowledge Engineering, Vol. 60, pages 51-70.

[27] Zha, H., Ding, C., Gu, M., He, X., and Simon, H. 2001. Spectral relaxation for K-means clustering. Advances in Neural Information Processing Systems. 14 (NIPS'01), pages 1057–1064.

APPENDIX

The computation of the attribute covariance matrix of a dataset requires one scan of the dataset to calculate the frequency of each attribute and the co-occurrence frequency of each pair of attributes. SCCADDS initializes a counter to zero for each attribute value and another counter for each pair of attribute values. For each data object in the dataset it increments the appropriate counters. To illustrate the computation of the covariance, we define matrices B and D, and vector F. Matrix B is an m x m matrix that contains attribute value co-occurrence frequencies where m is the number of attribute values. The m by m diagonal matrix D and the m vector F contain the attribute value frequencies. The value n is the number of data objects in the data set. The attribute covariance matrix is calculated as follows:

$$\left(D^{-1/2}BD^{-1/2}\right)-\left(\frac{F^{1/2}(F^{1/2})^{T}}{n}\right)$$

Sequential Latent Semantic Indexing

Mikhail Krivenko
The Institute of Informatics Problems of the Russian Academy of Sciences, Moscow
mkrivenko@ipiran.ru

Vitaly Vasilyev
The Institute of Informatics Problems of the Russian Academy of Sciences, Moscow
vvg_2000@mail.ru

ABSTRACT
This paper presents a new sequential version of the latent semantic indexing method based on the notion of the relative error of approximation of the matrix of observations. The paper gives theoretical and experimental justification of the effectiveness of the proposed method.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.6 [**Artificial Intelligence**]: Learning;

General Terms
Algorithms, Performance, Design, Experimentation, Theory.

Keywords
Latent Semantic Indexing, Clustering, Matrix Approximation.

1. INTRODUCTION
Latent semantic indexing (LSI) was developed in the field of information retrieval to solve problems of synonymy and polysemy of words of natural language [2]. It is based on representation of texts with a small number of generalized concepts (factors), which are founded by calculating the singular value decomposition of term-document matrix. Formally this method is defined as follows.

Let A matrix of size $d \times n$ which elements describe the occurrences of terms in documents, d - number of unique terms in all documents, n - number of documents in the collection. Then the presentation of matrix A in the space of m factors is:

$$X = U_m^T A,\ m << \mathrm{r},$$

where U_m - matrix of left singular vectors for the m largest singular values of matrix A, r - rank of matrix A.

DMMT'09, June 28, 2009, Paris.

Main advantages of LSI are determined by the following points:

1. $\tilde{A} = U_m U_m^T A$ is the best rank m approximation of the matrix A [4];

2. It provides much higher degree of dimension reduction of the feature space in comparison to the feature selection methods;

3. It improves the quality of classification and cluster analysis of texts;

4. It's linear character of transformation and the high dimensionality of the texts collection leads to the formation of the final distribution of data similar to a mixture of normal distributions.

However, during the implementation of the iterative learning procedure for the statistical classificators, based on the use of this method, we are faced with a number of algorithmic and computational difficulties.

1. The use of the fixed dimension of the factor space does not allow to take into account the emergence of new themes in the texts collection. It leads to a decrease the quality of classification.

2. The time needed for dimension reduction dramatically increases with the growing of the size of the texts collection.

3. Updating texts collection requires re-executing of procedure of dimension reduction for the entire collection.

Thus, the LSI method has some remarkable characteristics, but for its effective use in practice it is required to develop the effective factor space updating procedures.

2. EXISTING LSI UPDATING METHODS
In the modern literature many different algorithms for SVD updating are discussed. The most popular are the following algorithms:

- mapping into the existing factor space (FOLDIN) [1];

- SVD updating (SVDUPD) [8];

– modified SVD updating (MSVDUPD) [8, 10, 11];

Let B - arbitrary matrix of size $l \times s$. The symbol $\Lambda_m(B)$ will define diagonal matrix of size $m \times m$ containing on the diagonal m first largest singular values of matrix B, sorted in descending order. The symbols $U_m(B)$ and $V_m(B)$ will define matrices of size $l \times m$ and $s \times m$ representing the left and right singular vectors of B, matching the singular values in $\Lambda_m(B)$. Symbol $G(B)$ will define a matrix of factors with orthonormal columns

corresponding to the matrix B and the symbol $X(B)$ will define the representation of matrix B in the factor space $G(B)$.

Note that in standard LSI method, based on the computation of singular value decomposition, $G(B)=U_m(B)$ and $X(B)=U_m^T(B)B$, but in described further sequential algorithms matrices $G(B)$ and $X(B)$ will be considered as approximations of the matrices $U_m(B)$ and $U_m^T(B)B$, respectively.

Let A_1 - matrix of the «old» texts of size $d\times n_1$, A_2 - matrix of the «new» texts of size $d\times n_2$ and $A=(A_1,A_2)$ - concatenated array of observations. Then the task of correcting the factor space is to find the matrices $G(A)$ and $X(A)$ using known matrices $G(A_1)=U_m(A_1)$ and $X(A_1)$.

Mapping into the existing factor space [2]. In this case, the matrices $G(A)$ and $X(A)$ are calculated as follows:

$$G(A)=U_m(A_1),$$

$$X(A)=U_m^T(A_1)(A_1,A_2),$$

where m - the dimension of existing factor space. The advantage of this approach is the low computational complexity, and the obvious weakness is the inability to take into account the emergence of new relationships in the data.

SVD updating [8]. The new text first mapped into the existing factor space, as well as in the previous algorithm, but then performed a special procedure of factors correction. The final formulas for calculating the matrices $G(A)$ and $X(A)$ are as follows:

$$G(A)=U_m(A_1)U_m(F),$$

$$X(A)=\Lambda_m(F)V_m(F)^T\begin{pmatrix}V_m(A_1) & 0\\ 0 & I_{n_2}\end{pmatrix},$$

where $F=\left[\Lambda_m(A_1),U_m^T(A_1)A_2\right]$, m – the dimension of factor space.

Note that in this case new factors are simply a linear combination of old factors, which leads to the conclusion of the practical equivalence of this algorithm to FOLDIN.

Modified SVD updating [8, 10, 11] (in [10, 11] this method is called Partitioned R-SVD). In this case new texts are mapped into the existing factor space and into its orthogonal complement to the entire feature space. After that the factor space is corrected with a special procedure. The final formula for calculating the matrices $G(A)$ and $X(A)$ are as follows:

$$G(A)=[U_m(A_1),P_m]U_m(F),$$

$$X(A)=\Lambda_m(F)V_m(F)^T\begin{pmatrix}\Lambda_m^T(A_1) & 0\\ 0 & I_{n_2}\end{pmatrix},$$

where $F=\begin{pmatrix}\Lambda_m(A_1) & U_m^T(A_1)A_2\\ 0 & R\end{pmatrix}$, $P_m R$ is a QR decomposition of the matrix $(I-U_m(A_1)U_m^T(A_1))A_2$, P_m - orthonormal matrix and R - an upper triangular matrix.

It should be noted that in this case the resulting factor space is not equal to the initial factor space. This property gives the opportunity to taking into account the appearance of a new topics on the collection of texts. However, the experiments conducted with the different collections show it achieved at the excessive increase of computational complexity. For example, reducing the dimension of the «Reuters 21578» collection using this algorithm requires two orders of magnitude more time than using the SVDUPD algorithm. For this reason, this algorithm will not be considered in this paper.

In some papers there have been proposed other decompositions, such as ULV [3] or SDD [6]. However, analysis of algorithms, proposed for updating those decompositions, showed that they have the same restrictions and limitations as algorithms used for updating SVD decomposition.

Thus, the main property of the described methods of SVD updating is that the number of factors is fixed. This property may lead to the gradual decline of the classification quality during the growth of the size of the collection and the appearance of a new themes in it.

3. SEQUENTIAL LSI METHOD

The sequential method proposed in this paper, is based on the adaptive correction of the factor space dimension, depending on the value of relative approximation error of the data matrix A.

The relative approximation error of matrix B with the help of the matrix $\tilde{B}$ is defined as follows:

$$\left\|B-\tilde{B}\right\|_F^2/\|B\|_F^2.$$

Note that in the case of using LSI method $\tilde{B}=QQ^T$, where Q - the matrix of factors such that $Q^TQ=I$.

In this case the dimension of the factor space depends on the properties of the texts collection and it is automatically adjusted while collection is changing.

Before turning into the description of a sequential dimension reduction algorithm, consider some properties of the relative approximation error.

Lemma 1. Let B – arbitrary non zero matrix of size $l\times s$, $\tilde{B}=U_m(B)U_m(B)^TB$, m – dimension of factor space, then

$$\min_{rank(D)=m}\frac{\|B-D\|_F^2}{\|B\|_F^2}=\frac{\left\|B-\tilde{B}\right\|_F^2}{\|B\|_F^2},$$

where D – an arbitrary matrix of rank m.

Proof. The lemma directly follows from the theorem about approximation of arbitrary matrix using a matrix of smaller rank, given in [4]. ■

Thus, the relative approximation error is the lowest in the case when matrix of factors is $U_m(B)$.

Lemma 2. Let B – arbitrary non zero matrix of size $l \times s$, $\tilde{B} = QQ^T B$ – approximation of matrix B in factor space Q, Q – arbitrary matrix with orthonormal columns of size $l \times m$, m - dimension of factor space, then

$$0 \le \frac{\left\| B - \tilde{B} \right\|_F^2}{\left\| B \right\|_F^2} \le 1$$

and the following simplified formulas for calculating the relative approximation error are true:

$$\frac{\left\| B - \tilde{B} \right\|_F^2}{\left\| B \right\|_F^2} = 1 - \frac{\left\| \tilde{B} \right\|_F^2}{\left\| B \right\|_F^2} \text{ and } \quad \frac{\left\| B - \tilde{B} \right\|_F^2}{\left\| B \right\|_F^2} = 1 - \frac{\left\| Q^T B \right\|_F^2}{\left\| B \right\|_F^2}.$$

Proof. Based on the using of the properties of orthonormal matrices and the representation of Frobenious norm using matrix trace function. ■

Thus, in the case of using SVD relative approximation error is in the interval from 0 to 1, and for its calculation, there are simple formulas that can be used for efficiently find errors values without direct access to the original matrix B.

Now, let's look at the statement, which connects the relative approximation error of arbitrary matrix with relative approximation errors of it sub matrices.

Lemma 3. Let us suppose there are non-zero matrix B_1 of size $l \times s_1$ and B_2 of size $l \times s_2$, as well as the matrices $\tilde{B}_1$ and $\tilde{B}_2$ such, that $\left\| B_1 - \tilde{B}_1 \right\|_F^2 / \left\| B_1 \right\|_F^2 \le \varepsilon$ and $\left\| B_2 - \tilde{B}_2 \right\|_F^2 / \left\| B_2 \right\|_F^2 \le \varepsilon$, then the following inequality is true:

$$\left\| \left(B_1 - \tilde{B}_1, B_2 - \tilde{B}_2 \right) \right\|_F^2 / \left\| \left(B_1, B_2 \right) \right\|_F^2 \le \varepsilon$$

Proof. The proof is based on the properties of the Frobenious norm. ■

Let now the symbol $U_\varepsilon(B)$ will define the matrix, consisting of the so minimum possible number of the first columns of the matrix $U(B)$ as the relative approximation error of matrix B of size $l \times s$ and rank $r > 0$ with the help of a matrix $\tilde{B} = U_\varepsilon(B) U_\varepsilon^T(B) B$ was no more ε, i.e.

$$U_\varepsilon(B) = U_k(B)$$

where $k \in \{1,..., r\}$ and $1 - \frac{\left\| U_k^T(B) B \right\|_F^2}{\left\| B \right\|_F^2} \le \varepsilon$, $1 - \frac{\left\| U_{k-1}^T(B) B \right\|_F^2}{\left\| B \right\|_F^2} > \varepsilon$, ε - constant, specifying the allowable level of relative approximation error, $U_0(B) \equiv \varnothing$ and $U_0^T(B) B = 0$.

Accuracy of this definition can be resulted from the monotonicity of the values of expression $\left\| U_k^T(B) B \right\|_F^2 / \left\| B \right\|_F^2$ under $k = 1,..., r$.

Consider now the procedure of correction of factor space which is the basis of the sequential algorithm of dimension reduction.

Let A_1 the matrix of «old» texts of size $d \times n_1$, the relative approximation error of the matrix A_1 in the space of $G(A_1)$ no more ε, A_2 the matrix of «new» texts of size $d \times n_2$, $A = (A_1, A_2)$ - full array of observations, $0 < \varepsilon < 1$ - allowed relative approximation error of a matrix of observations.

In this case the procedure of correction of factor space will be the following.

Algorithm of correction of factor space

1. Find projection $Y = G^T(A_1) A_2$ of matrix A_2 on «old» factor space $G(A_1)$.

2. .Check condition $1 - \left\| Y \right\|_F^2 / \left\| A_2 \right\|_F^2 \le \varepsilon$. If it is true, then put $G(A) = G(A_1)$, otherwise put $G(A) = (G(A_1), U_\delta(C))$, where $\delta = \varepsilon \left\| A_2 \right\|_F^2 / \left\| C \right\|_F^2$, $C = A_2 - G(A_1) Y$.

3. Finish algorithm. ■

An important property of this algorithm is that it does not appeal to the matrix of already processed texts A_1. This property can substantially reduce the requirements for the amount of memory. However, the rejection of the use of the matrix A_1 makes impossible to calculate the relative approximation error for the full observation matrix A. For this reason in the presented algorithm the number of added factors is chosen in such way as to ensure a given level of relative approximation only for the matrix A_2.

Let's show that the columns of the resulting matrix $G(A)$ are orthonormal, the relative approximation error of the matrix A with the matrix $G(A) X(A)$ is no more ε and a final dimension of the factor space is the minimum possible to achieve a given level of relative approximation error of matrix A_2, where $X(A) = G^T(A) A$ - a presentation of the matrix A in the factor space $G(A)$.

Theorem 1. Under the described above conditions the following is true:

$G^T(A) G(A) = I$,

$\left\| A - G(A) X(A) \right\|_F^2 / \left\| A \right\|_F^2 \le \varepsilon$,

$G(A)$ has the lowest possible rank among all types of matrices $(G(A_1), Z)$ such as $(G(A_1), Z)^T (G(A_1), Z) = I$ and $\left\| A_2 - (G(A_1), Z)(G(A_1), Z)^T A_2 \right\|_F^2 / \left\| A_2 \right\|_F^2 \le \varepsilon$.

Proof. When $G(A)=G(A_1)$ correctness of this theorem directly follows from the initial conditions. For this reason, we will believe that $G(A)=(G(A_1),U_\delta(C))$.

Let's show that $G(A)^T G(A)=I$.

According to the construction $1-\left\|G(A_1)^T A_2\right\|_F^2/\|A_2\|_F^2>\varepsilon$. Hence, using Lemma 2 and the definition of the matrix *C*, we will find that

$$\|C\|_F^2=\left\|A_2-G(A_1)G(A_1)^T A_2\right\|_F^2=\|A_2\|_F^2-\left\|G(A_1)G(A_1)^T A_2\right\|_F^2>$$

$$>\varepsilon\|A_2\|_F^2>0.$$

From the definition of the Frobenious norm we have that $C\neq 0$ and from definition of matrix C we have that $G(A_1)^T C=0$. Hence we get that $G(A_1)^T C=\left(G(A_1)^T U(C)\right)\left(\Lambda(C)V(C)\right)=0$. Since $C\neq 0$, from the last equality we get that $G(A_1)^T U(C)=0$. Given now that the matrix $U_\delta(C)$ is composed of columns of the matrix $U(C)$, we get that $G(A_1)^T U_\delta(C)=0$. Now, by direct verification it is easy to make sure that $G(A)^T G(A)=I$

Let show now that $\left\|A-G(A)X(A)\right\|_F^2/\|A\|_F^2\leq\varepsilon$.

The definition of the matrix $G(A)$ and Lemma 2 implies that

$$\left\|A_2-G(A)G(A)^T A_2\right\|_F^2/\|A_2\|_F^2=$$

$$1-\left(\left\|G(A_1)^T A_2\right\|_F^2+\left\|U_\delta^T(C)A_2\right\|_F^2/\|A_2\|_F^2\right).$$

At the same time from the definition of the matrix *C* we have that $\left\|U_\delta^T(C)C\right\|_F^2=\left\|U_\delta^T(C)A_2\right\|_F^2$. Hence we find that

$$\left\|A_2-G(A)G(A)^T A_2\right\|_F^2/\|A_2\|_F^2=$$

$$1-\left\|G(A_1)^T A_2\right\|_F^2/\|A_2\|_F^2-\left(\left\|U_\delta^T(C)C\right\|_F^2/\|C\|_F^2\right)\left(\|C\|_F^2/\|A_2\|_F^2\right)\leq$$

$$\leq 1-\left\|G(A_1)^T A_2\right\|_F^2/\|A_2\|_F^2-(1-\delta)\left(\|C\|_F^2/\|A_2\|_F^2\right)=$$

$$=\left(\|A_2\|_F^2-\left\|G(A_1)^T A_2\right\|_F^2-(1-\delta)\|C\|_F^2\right)/\|A_2\|_F^2.$$

Since $\|A_2\|_F^2-\left\|G(A_1)G(A_1)^T A_2\right\|_F^2-\|C\|_F^2=0$, we get that

$$\left\|A_2-G(A)G(A)^T A_2\right\|_F^2/\|A_2\|_F^2\leq\varepsilon.$$

From lemma 2 also we get that:

$$0\leq\left\|A_1-G(A)G(A)^T A_1\right\|_F^2/\|A_1\|_F^2=$$

$$1-\left\|(G(A_1),U_\delta(C))^T A_1\right\|_F^2/\|A_1\|_F^2=$$

$$=1-\left\|G(A_1)^T A_1\right\|_F^2/\|A_1\|_F^2-\left\|U_\delta^T(C)A_1\right\|_F^2/\|A_1\|_F^2\leq$$

$$\varepsilon-\left\|U_\delta^T(C)A_1\right\|_F^2/\|A_1\|_F^2\leq\varepsilon.$$

Now from lemma 3 we get that

$$\left\|A-G(A)X(A)\right\|_F^2/\|A\|_F^2=$$

$$\left\|(A_1,A_2)-G(A)G(A)^T(A_1,A_2)\right\|_F^2/\|(A_1,A_2)\|_F^2=$$

$$\left\|\left(A_1-G(A)G(A)^T A_1,A_2-G(A)G(A)^T A_2\right)\right\|_F^2/\|(A_1,A_2)\|_F^2\leq\varepsilon.$$

Now, let's show that the matrix $G(A)$ has the minimum possible number of columns among all matrices $(G(A_1),Z)$ such that $(G(A_1),Z)^T(G(A_1),Z)=I$ and $\left\|A_2-(G(A_1),Z)(G(A_1),Z)^T A_2\right\|_F^2/\|A_2\|_F^2\leq\varepsilon$.

From the condition $(G(A_1),Z)^T(G(A_1),Z)=I$ it follows that $Z^T G(A_1)=0$. Hence, we find that $ZZ^T C=ZZ^T(A_2-G(A_1)G(A_1)A_2)=ZZ^T A_2$. Consequently,

$$\left\|A_2-(G(A_1),Z)(G(A_1),Z)^T A_2\right\|_F^2/\|A_2\|_F^2=$$

$$\left\|A_2-G(A_1)G(A_1)^T A_2-ZZ^T A_2\right\|_F^2/\|A_2\|_F^2\leq\varepsilon.$$

Hence, we find that the condition $\left\|A_2-(G(A_1),Z)(G(A_1),Z)^T A_2\right\|_F^2/\|A_2\|_F^2\leq\varepsilon$ is equivalent to the condition $\left\|C-ZZ^T C\right\|_F^2/\|C\|_F^2\leq\varepsilon\|A_2\|_F^2/\|C\|_F^2$. Using Lemma 1, it is easy to show that the matrix Z has the lowest possible rank when $Z=U_\delta(C)$, where $\delta=\varepsilon\|A_2\|_F^2/\|C\|_F^2$. Hence we get the required approval of the theorem. ∎

Thus, the relative approximation error of the matrix A in the resulting factor space $G(A)$ is no greater than ε, and its dimension is the minimum possible under the imposed restrictions on the type of the matrix of factors and subject to recourse only to the matrix A_2. However, in the general case in the absence of these restrictions the final dimension of the factor space $G(A)$ may be redundant because of the fact that the "new" factors were also presented in the "old" data.

For this reason, after the correction of factor space it's reasonable to carry out an additional dimension reduction of the renewed factor space. Since the dimension of the features space in this case is not very large (several tens of features), then this operation will be much easier as compared with reducing the dimension of the original matrix of observations A. Moreover when data are processed sequentially additional dimension reduction can be executed only once after the processing of entire array has been finished.

Consider now entire sequential latent semantic indexing algorithm (SLSI), based on the algorithm for correcting the space of factors. Let the observation matrix A of size $d\times n$ is broken into *b* sub

matrices, i.e. $A=(A_1,...,A_b)$, $0<\varepsilon<1$ is constant representing the relative approximation error.

SLSI Algorithm

1. Put $Q_1=U_\varepsilon(A_1)$ and $t=2$.

2. Find projection $Y_t=Q_{t-1}^T A_t$ of matrix A_t on the factor space Q_{t-1}.

3. If $1-\|Y_t\|_F^2/\|A_t\|_F^2>\varepsilon$ then put $Q_t=(Q_{t-1},U_\delta(C_t))$ otherwise put $Q_t=Q_{t-1}$, where $\delta=\varepsilon\|A_t\|_F^2/\|C_t\|_F^2$ and $C_t=A_t-Q_{t-1}Y_t$.

4. If $t=b$, then go to step 5, otherwise put $t=t+1$ and go to step 2.

5. Find projection $Y=(Q_b^T A_1,...,Q_b^T A_b)$ of matrix A on factor space Q_b through a sequential processing of $A_1,...,A_b$.

6. Check the condition $\eta=1-\|Y\|_F^2/\sum_{i=1}^{b}\|A_i\|_F^2<\varepsilon$. If it is true, then put $G(A)=Q_bU_\tau(Y)$ and $X(A)=U_\tau(Y)^T Y$ otherwise put $G(A)=Q_b$ and $X(A)=Y$, where $\tau=1-(1-\varepsilon)\sum_{i=1}^{b}\|A_i\|_F^2/\|Y\|_F^2$.

7. Finish algorithm. ■

Thus, the algorithm consists of two stages. In the first stage (steps 1-4) the described above algorithm of correction of factor space is sequentially applied. On the second stage (steps 5 and 6) mapping of matrix *A* into the constructed factor space is sequentially carried out and, if necessary, additional dimension reduction is performed.

It should also be noted that at each stage a permanent presence in the memory only of one block of matrix of observations is required, which allows the efficient handling of arrays that will not fit in memory.

Now let us show that after finishing the SLSI algorithm matrix of factors $G(A)$ is orthogonal and the relative approximation error of matrix A using matrix $G(A)X(A)$ does not exceed ε.

Theorem 2. In an algorithm SLSI just true the following ratios:

$$G(A)^T G(A)=I,$$

$$\|A-G(A)X(A)\|_F^2/\|A\|_F^2\le\varepsilon.$$

Proof. From theorem 1 it follows that $Q_b^T Q_b=I$ and definition of matrix $U_\tau(Y)$ implies that true equality $U_\tau^T(Y)U_\tau(Y)=I$. Hence, by direct inspection we find that $G(A)^T G(A)=I$.

Now show that the inequality $\|A-G(A)X(A)\|_F^2/\|A\|_F^2\le\varepsilon$ is true. If $\eta=\varepsilon$ then a justice of the inequality immediately follows from Theorem 1. Let now $\eta<\varepsilon$. Using Lemma 2 we get that

$$\|A-G(A)X(A)\|_F^2/\|A\|_F^2=\left\|A-Q_bU_\tau(Y)U_\tau^T(Y)Q_b^T A\right\|_F^2/\|A\|_F^2=$$

$$1-\left\|Q_bU_\tau(Y)U_\tau^T(Y)Q_b^T A\right\|_F^2/\|A\|_F^2=1-\frac{\left\|U_\tau(Y)U_\tau^T(Y)Y\right\|_F^2}{\|Y\|_F^2}\frac{\|Y\|_F^2}{\|A\|_F^2}\le$$

$$\le 1-(1-\tau)\|Y\|_F^2/\|A\|_F^2=\varepsilon.\ \blacksquare$$

4. REALIZATION

It should be noted that for the practical use of the sequential LSI method it is required to solve a number of additional issues related to the organization of the efficient storage of data in memory, effective calculating of singular value decomposition and the choice of a relative approximation error.

Data storing in memory. The matrix $A=(A_1,...,A_b)$ is strongly sparse (it contains less than 1% of non-zero elements), so it is appropriate to store in memory only the nonzero elements. Experiments with different collections show that the matrices Q_t and $G(A)$ also contains a large number of elements that are close to zero. By setting these elements equal to zero one can achieve significant saving of memory without noticeable loss of orthogonality (in this work there set equal to zero such a number of elements in order not to lead to the change of the matrix by more than 0.1%). For example, in the performing of this procedure for the "Reuters-21578-6" collection amount of memory needed to store the matrix is reduced by more than 5 times.

Computation of singular value decomposition. For finding $U_\varepsilon(A_1)$, $U_\delta(C_t)$, $U_\tau(Y)$ it is required to compute the singular vectors corresponding to the largest singular values of matrices A_1, C, Y.

The matrices A_1 and $C=A_t-Q_{t-1}Y_t$, $t=2,...,b$, are usually highly sparse and contains a large number of rows and columns. For this reason, to find their singular vectors and values, it was decided to use the Lanczos algorithm which is oriented to work with sparse matrices. With it use it is not required direct access to the data. It is only need to create functions for calculating values of the expressions $A_1^T A_1 x$ and $C^T Cx$ for an arbitrary vector x.

The matrix Y is dense and the number of its rows is several times less than the number of its columns. For this reason, it was decided to carry out the calculation of its singular value decomposition by finding the eigenvalues and vectors of the matrix YY^T (for this task we used the standard algorithm from the MATLAB package).

Now let's compare the computational complexity of the basic procedure of the correction of factor space used in the SLSI algorithm, with the computational complexities of the other algorithms for updating SVD decomposition. In evaluating the computational complexity it was suggested to use the Lanczos algorithm [5] in all methods for finding the singular vectors and values.

The table 1 gives the asymptotic evaluation of computational complexity, where ρ is the average number of terms in the text,

$n = n_1 + n_2$ is the total number of texts in the collection, m is the final dimension of the feature space, $n_b = n_2$ is the number of new texts, d is the dimension of input space, m_b is the number of new factors, μ is the average number of nontrivial elements in the columns of the matrix U_δ ($\mu < d$).

Table 1. Complexity of the SVD updating methods

Method	Flops
SVD (Lanczos)	$O(\rho mn + dm^2)$
FOLDIN	$O(\rho mn_b)$
SVDUPD	$O(\rho mn_b + dm^2 + nm^2)$
MSVDUPD	$O(\rho mn_b + dn_b^2 + dm(m+n_b) + (m+n_b)^3)$
SLSI	$O(\rho mn_b + \mu m_b m + n_b m_b m)$

It should be noted that in the presented formula for SLSI algorithm the complexity of dimension reduction in step 6 (which is performed only after processing the entire array) is not taking into account. It is possible to show that its computational complexity is $O(m^3 + nm^2)$ flops.

The SVD updating methods described above have been realized using MATLAB package. Realization of the SLSI algorithm on the MATLAB programming language can be downloaded from Mathworks MATLAB Central (http://www.mathworks.com/matlabcentral/fileexchange/22795). Realization of Lanczos algorithm is taken from MATLAB Version of the PROPACK package (http://soi.stanford.edu/~rmunk/PROPACK/).

5. EXPERIMENTS

The focus of experiments has been given to evaluating the effect of sequential procedures on the quality of the classification of the texts, as well as evaluating their computational complexity.

Experiments were conducted on the «Reuters-21578» collection, as well as datasets OHSCAL, CRANMED, which are available in the preprocessed form in the CLUTO site (http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download). The characteristics of these arrays are given in the Table 2.

Table 2. Datasets

Dataset	Documents	Features	Rubrics
Reuters-21578	14027	32016	142
OHSCAL	11162	11465	10
SPORTS	8580	126355	7

To classify the texts k-nearest neighbors algorithm was used. This algorithm has two main parameters: the measure of similarity and the number of nearest neighbors. In this work cosine similarity measure and the number of nearest neighbors equal to 5 were used. Also it should be noted that for simplify calculations each text was related only to one class.

For measuring the quality of classification we used error E, which is defined as the proportion of wrong classified texts in relation to the total number of texts (it should be noted that in the case of non overlapping classes, this indicator is equivalent to F-measure).

Processing collections of texts was carried out as follows:

1. Random partitioning of the collection into training and testing sets in a ratio 4 to 1.

2. Processing the training set (computation of TF-IDF weights of words and dimension reduction by the LSI method).

3. Training of k-nearest neighbors classifier using training set.

4. Processing the testing set (computation of TF-IDF weights of words and dimension reduction by LSI method using estimates of IDF weights and estimates of factors obtained for training set).

5. Classification texts from the testing set and estimation of the error.

It should be noted that in all experiments computations have been organized so that initially all the texts from the first class are under processing, then all the texts from the second class, etc.

Lets consider the results of experiments now.

Figures 1, 2, 3 show the dependence of the error on the size of the collection for the "Reuters", "OHSCAL" and "SPORTS" collections. In this case, the number of factors $m = 10,20,50$. It can be noticed that when the number of factors is fixed, the classification error increase with the increase of the number of classes and the number of documents. But the error can be reduced by increasing the number of factors.

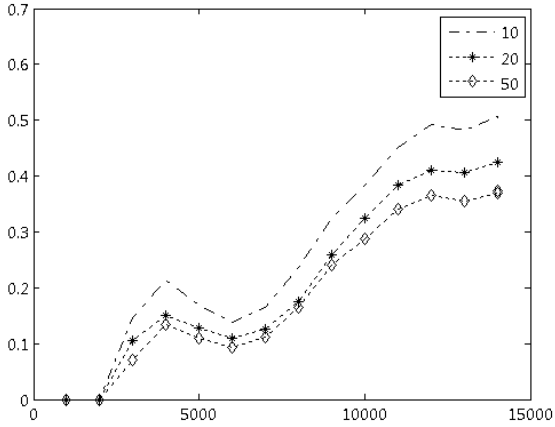


Figure 1. The dependence of the error on the size of the collection for the "Reuters-21578" collection.

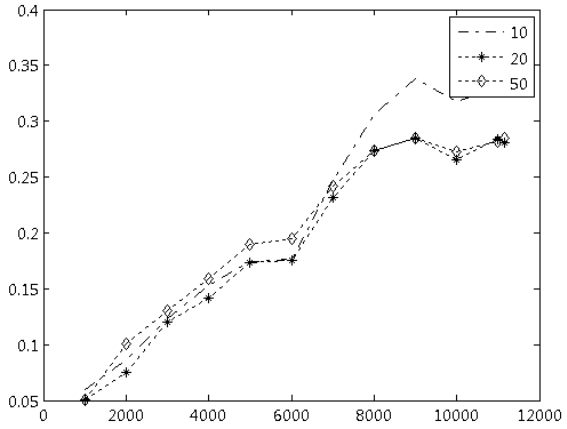


Figure 2. The dependence of the error on the size of the collection for the "OHSCAL" dataset.

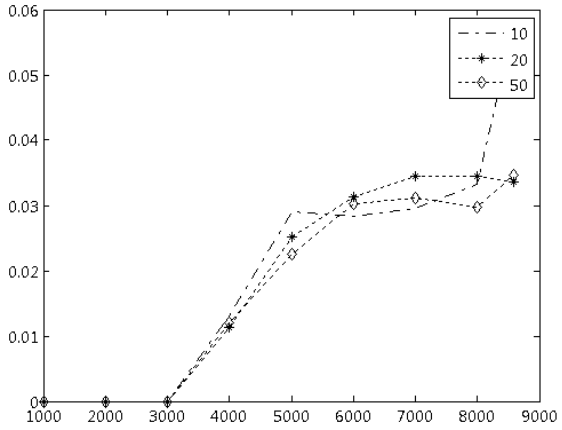


Figure 3. The dependence of the error on the size of the collection for the "SPORTS" dataset.

Figures 4, 5, 6 show the dependence of the factor space dimension on the size of the collection in the situation when the value of relative error of approximation is fixed (for the array "Reuters-21578" $\varepsilon = 0.5$, and for arrays "OHSCAL" and "SPORTS" $\varepsilon = 0.85$). One can notice that in this case, the number of factors is automatically changed when the collection enlarged.

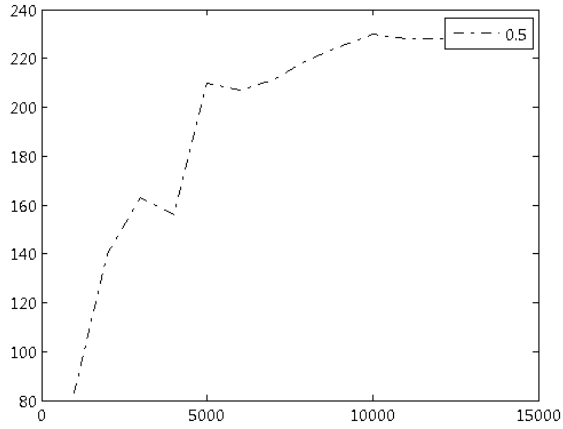


Figure 4. The dependence of factor space dimension on the size of the collection for the "Reuters-21578" collection.

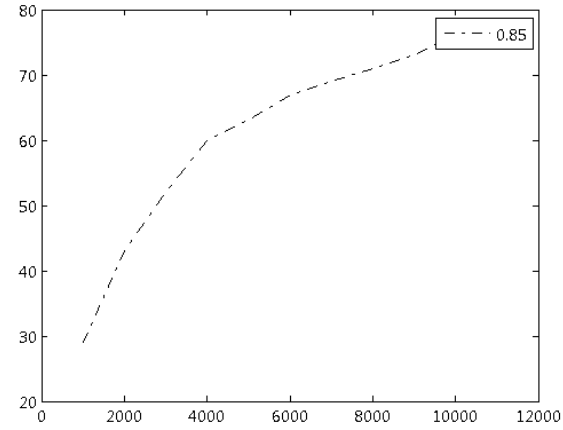


Figure 5. The dependence of the factor space dimension on the size of the collection for the "OHSCAL" dataset.

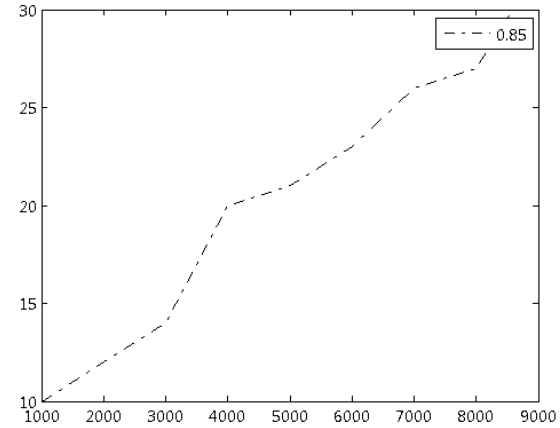


Figure 6. The dependence of factor space dimension on the size of the collection for the "SPORTS" dataset.

Figures 7, 8 and 9 show the dependence of the working time on the size of the collection for SVD, SVDUPD and SLSI methods for "Reuters-21578", "OHSCAL" and "SPORTS" collections. In this experiments block size is 2500 and the relative approximation error is equal to 85%. One can notice that the working time of the SLSI is about the same as of the SVDUPD, which is simpler.

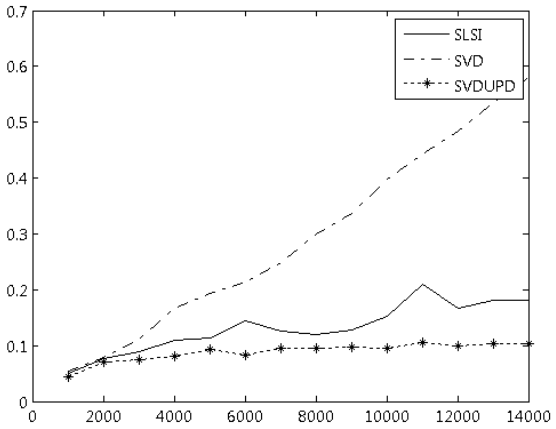


Figure 7. Working time of the dimension reduction algorithms for the "Reuters-21578" collection.

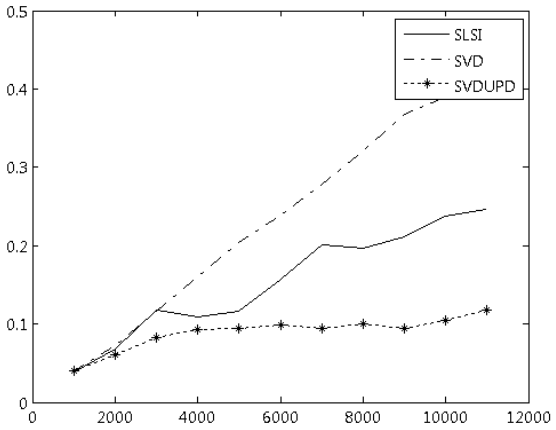


Figure 8. Working time of the dimension reduction algorithms for the "OHSCAL" dataset.

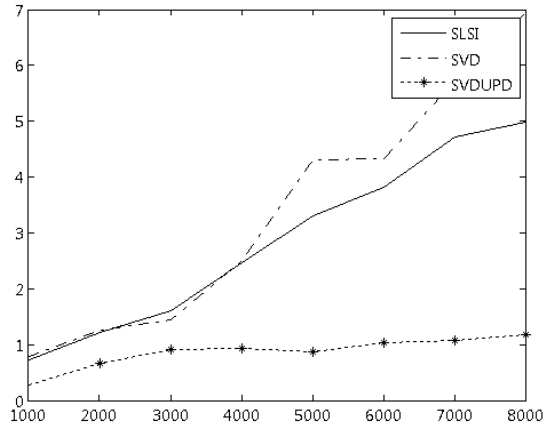


Figure 9. Working time of the dimension reduction algorithms for the "SPORTS" dataset.

Figures 10, 11 and 12 show the dependence of the classification error on the size of the collection for SVD, SVDUPD and SLSI methods for "Reuters-21578", "OHSCAL" and "SPORTS" collections. In this experiments block size is 2500 and the relative approximation error is equal to 85%.

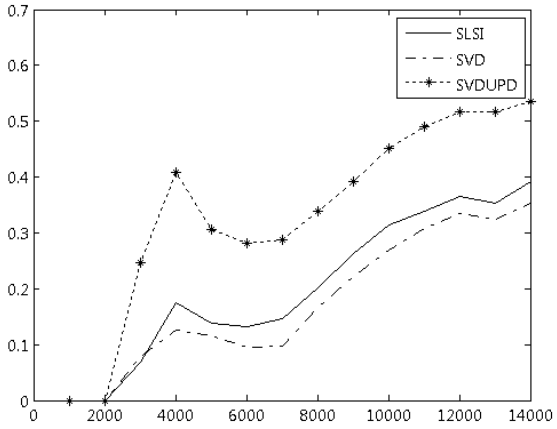


Figure 10. The dependence of the classification error on the number of texts for the "Reuters-21578" collection.

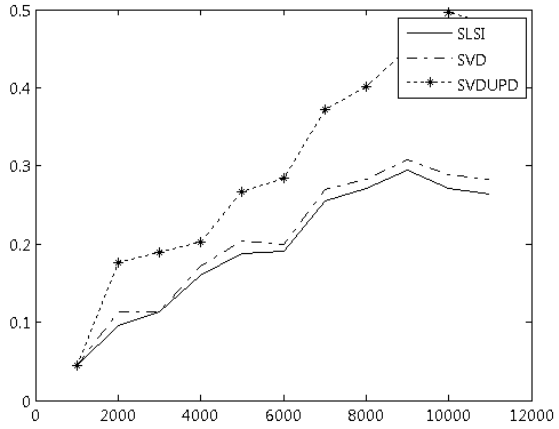


Figure 11. The dependence of classification error on the number of texts for the "OHSCAL" dataset.

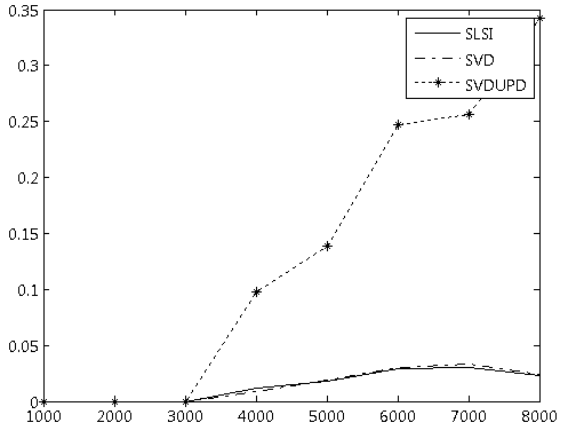


Figure 12. The dependence of the classification error on the number of texts for the "SPORTS" dataset.

Thus, these experiments show that the SLSI algorithm is a tradeoff between the algorithm of full SVD decomposition and the SVDUPD algorithm in terms of the classification error and computational complexity.

6. CONCLUSIONS

Thus, this work offers a new sequential version of the LSI algorithm (SLSI). The main difference of the SLSI from the existing sequential algorithms is that the dimension of space is not fixed and dynamically changes to ensure a given level of relative approximation error of a matrix of observations. The experiments with the different collections of texts demonstrated that the SLSI algorithm could be seen as a tradeoff solution, which have a lower computational complexity and memory requirements compared to the standard LSI method and did not lead to a decrease of the quality of classification in contrast to other sequential algorithms. This work is supported by the grant of the President of the Russian Federation MK-12.2008.10.

7. REFERENCES

[1] Berry M. W., Drmac Z., and Jessup E. R. Matrices, Vector Spaces, and Information Retrieval // SIAM Review, 1999. Vol. 41. No. 2, 335–362.

[2] Berry M. W., Dumais S. T., O'Brien G. W. Using linear algebra for intelligent information retrieval // SIAM Review, 1995. Vol. 37. No. 4, 573–595.

[3] Berry M. W., Fierro R. D. Low-Rank Orthogonal Decompositions for Information Retrieval Applications // Numerical Linear Algebra with Applications, 1996. Vol. 1. No. 1, 1–27.

[4] Eckart C., Young G. The approximation of one matrix by another of lower rank // Psychometrika, 1936. Vol. 1, 211–218.

[5] Golub G.H., Van Loan C. F. Matrix Computations (3rd Ed.). Johns Hopkins University Press, 1996. – 694 p.

[6] Kolda T., O'Leary D. A semi-discrete matrix decomposition for latent semantic indexing in information retrieval // ACM Trans. Inform. Systems, 1998. Vol. 16, 322–346.

[7] Sebastiani F. Machine learning in automated text categorization // ACM Computing Surveys, 2002. Vol. 34. No. 1, 1–47.

[8] Simon H., Zha H. On Upadating Problems in Latent Semantic Indexing // Tech. Rep. CSE-97-011. – The Pennsylvania State University, University Park, 1997, 9 p.

[9] Vasilev V.G. and Krivenko M.P. Methods of automated text processing. – Moscow. IPI RAS, 2008. – 304 p. (in russian).

[10] Levy A., Lindenbaum M. Sequential Karhunen-Loeve Basis Extraction and Its Application to Images // IEEE Trans. on Image Processing, Vol. 9, 2000.

[11] Lim J., Ross D., Lin R., Yang M. Incremental Learning for Visual Tracking // NIPS, 2004.

Multi-Way Set Enumeration in Real-Valued Tensors

Elisabeth Georgii
MPI for Biological Cybernetics/
Friedrich Miescher Laboratory
of the Max Planck Society
Tübingen, Germany
georgii@tuebingen.mpg.de

Koji Tsuda
National Institute of Advanced
Industrial Science and
Technology (AIST)
Tokyo, Japan
koji.tsuda@aist.go.jp

Bernhard Schölkopf
MPI for Biological Cybernetics
Tübingen, Germany
bs@tuebingen.mpg.de

ABSTRACT

The analysis of n-ary relations receives attention in many different fields, for instance biology, web mining, and social studies. In the basic setting, there are n sets of instances, and each observation associates n instances, one from each set. A common approach to explore these n-way data is the search for n-set patterns. An n-set pattern consists of specific subsets of the n instance sets such that all possible n-ary associations between the corresponding instances are observed. This provides a higher-level view of the data, revealing associative relationships between groups of instances. Here, we generalize this approach in two respects. First, we tolerate missing observations to a certain degree, that means we are also interested in n-sets where most (although not all) of the possible combinations have been recorded in the data. Second, we take association weights into account. More precisely, we propose a method to enumerate all n-sets that satisfy a minimum threshold with respect to the average association weight. Non-observed associations obtain by default a weight of zero. Technically, we solve the enumeration task using a reverse search strategy, which allows for effective pruning of the search space. In addition, our algorithm provides a ranking of the solutions and can consider further constraints. We show experimental results on artificial and real-world data sets from different domains.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms

1. INTRODUCTION

Tensors are multi-dimensional arrays representing associations between instances. The analysis of such multi-way data becomes increasingly popular in the data mining community. Tensor decomposition is one of the most prominent topics (see [15] for a review). The factorized representation of the tensor can serve as a basis for applications like clustering of instances or anomaly detection [16]. Furthermore, there exist approaches to analyze dynamic changes in tensors [20]. Another approach to investigate n-way data comes from relational data mining. This line of research focuses on the development of methods for the detection of n-sets [6, 11, 12]. An n-set consists of specific subsets of instances for each of the n dimensions such that all possible n-ary associations are observed in the given data. In the tensor framework, the relational setting would correspond to the binary-value case, where 1-entries represent observed associations and all other entries are set to 0; an n-set corresponds to a subtensor that contains only 1-entries.

In this paper, we propose a more general definition of n-set as well as an enumerative mining algorithm for this kind of pattern. Our criterion is the average value of the entries within a subtensor; subtensors where this value exceeds a given threshold are considered as solutions. Here, the tensor is not restricted to binary values, but may contain arbitrary real-valued weights. Furthermore, a solution pattern might also contain some 0-entries, so missing observations are tolerated to a certain degree. This can be advantageous in applications where data are sparse (i.e. where it is likely that some observations are missing) and where the associations should be weighted differently (e.g. because the reliability or the significance of the observations is subject to variation).

In the following, these subtensor patterns are called *(multi-way) clusters*, and their average within-association value is called the *cluster density*; a cluster that satisfies the minimum density threshold is called a *dense cluster*. In the two-way case, a tensor corresponds to a matrix, and a bi-cluster is defined by a subset of rows and a subset of columns. There exist many approaches to detect bi-clusters (also called bi-sets) in data matrices, and various criteria have been used in the literature to assess their coherence [18]. For instance, some methods focus on bi-clusters with relatively homogeneous values [4]. Zhao and Zaki used a similar criterion for mining tri-clusters in biological three-way data [24]. In contrast, our density-based approach searches for clusters with a significantly large average value. To our knowledge, we present the first method to enumerate all dense clusters in the general matrix or tensor setting. It extends a dense cluster enumeration approach for *symmetric* matrices [21, 8]. From a graph-theoretic point of view, an n-way tensor corresponds to a weighted multi-partite hypergraph, where

DMMT'09, June 28, 2009, Paris.

each hyperedge connects n nodes which represent instances of the n different dimensions. So our approach can also be seen as extracting all densely connected subgraphs from such a hypergraph.

The paper is organized as follows. In section 2, we review reverse search and the cluster enumeration technique for symmetric matrices. In section 3, we describe our algorithmic approach for dense cluster enumeration in n-way tensors. Section 4 provides some additional extensions, and in section 5, we present our experimental results. The appendix contains the proof for a central lemma.

2. REVERSE SEARCH

This section reviews the reverse search paradigm [2]. We illustrate the basic idea for the application of enumerating dense sets (clusters) in a symmetric matrix [8, 21], which we will extend to n-way cluster enumeration in the following section.

2.1 Search Space

Let us consider the symmetric association matrix shown in Figure 1a. We denote by w_{ij} the matrix entry representing the association weight between instance i and instance j. Furthermore, we define the density of an instance subset U as

$$\rho(U) = \sum_{i,j \in U, i<j} w_{ij} \Big/ \frac{|U|(|U|-1)}{2}.$$

We would like to enumerate all subsets U with $\rho(U) \geq \theta$, where θ is a pre-specified constant.

All possible subsets form a natural graph-shaped search space, where one can move downwards or upwards by adding or removing an instance, respectively (Figure 1b); the root node of the search space corresponds to the empty set. However, for an efficient traversal of the search space, a spanning tree is more suitable than a graph because it avoids duplicate traversals of subspaces. A common approach is to construct a search tree that respects a lexicographical ordering of the instances (Figure 1c). In contrast to the frequency criterion used in itemset mining [9], the density is *not anti-monotonic* in this tree, i.e. the density does not decrease monotonically when traversing the tree from the root to a leaf. Therefore, it is not possible to use the pruning strategy applied in itemset mining [9]. However, there exists a search tree which shows the anti-monotonicity property with respect to the density (Figure 1d). We explain in the next subsection how it can be constructed.

2.2 Reduction Map

In reverse search, the search tree is specified by defining a *reduction map* $f(U)$ which transforms a child to its parent. In our case, the parent is obtained by removing the instance with minimum degree from the child. Here, the *degree* of an instance is defined as the sum of its association weights to the other instances in U.[1] If there are multiple instances with minimum degree, the one with the smallest index is removed. In [8], it is proven that the density of a parent is at least as large as the maximum density among the children. This ensures the *anti-monotonicity* of the search tree that is induced by the reduction map. In addition, a valid reduction map must satisfy the following *reachability* condition: starting from any node of the search space, we can reach a root node after applying the reduction map a finite number of times. This condition ensures that each possible solution can be reached from the root node, i.e. the induced search tree is indeed spanning. For the reduction map stated above, this is trivial to show, because any cluster is reduced to the empty set by removing instances repeatedly.

[1] We use the term *degree* because the symmetric association matrix can be seen as the adjacency matrix of a weighted graph.

2.3 Search Strategy

To enumerate all clusters with density $\geq \theta$, one has to traverse this implicitly defined search tree in a depth-first or breadth-first manner. During the traversal, children are generated *on demand* if the parent satisfies the density threshold. As the reduction map defines how to get from children to parents and not vice versa, we cannot directly derive the children from a given parent. Instead, to generate the children of a cluster U, we have to consider all candidates $U \cup \{i\}$, $i \notin U$, and apply the reduction map to every candidate (*reverse search principle*). Qualified candidates with $f(U \cup \{i\}) = U$ are then taken as children. A naive implementation of this child generation process can make the algorithm slow. Thus, it is important to engineer this process. Due to the anti-monotonicity, one can prune the tree whenever the density goes below θ. To conclude, reverse search provides a systematic way of obtaining anti-monotonic trees based on reduction maps. This is helpful for score functions where the anti-monotonicity property is not given in general, but can be achieved by a well-defined reduction map, which applies in the case of cluster density.

3. MULTI-WAY CLUSTER ENUMERATION

This section presents a novel n-way cluster enumeration method based on reverse search. In order to develop a reverse search algorithm, we have to 1) define a graph-shaped search space, 2) design a reduction map, and 3) engineer the child generation process for efficiency. After defining the problem, we proceed in this order.

3.1 Problem Definition

Our goal is to detect all dense clusters from a multi-way real-valued data array (tensor). To formalize the problem, we first introduce some notation. Let $n > 0$ be the number of *dimensions* in the input array (also called *ways* or *modes*). We write the given n-dimensional data array in the following form,

$$W = (w_{k_1,\ldots,k_n})_{k_i \in V_i \ \forall i=1,\ldots,n}\,. \tag{1}$$

The index k_i is used to access the i-th dimension and takes values from a finite index set $V_i = \{1, \ldots, |V_i|\}$ (the "range"). V_i is also called the instance set for the i-th dimension. The elements (entries) of W are real numbers indicating the association of n instances stemming from different sets. Negative values are allowed. For convenience, we normalize the array such that $w_{k_1,\ldots,k_n} \leq 1$ for all $k_i \in V_i$, $i = 1, \ldots, n$. An n-way cluster U is defined by specifying for each dimension a non-empty subset of the corresponding index set,

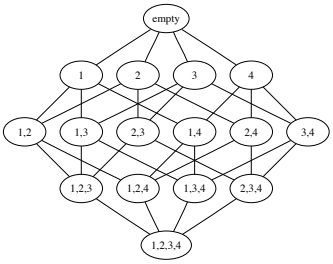
$$U = (U_1, \ldots, U_n), \quad U_i \subset V_i, \tag{2}$$

where $|U_i| \geq 1$ for all $i = 1, \ldots, n$. Let us define the *cardinality* of a cluster as the sum of the cardinalities of the

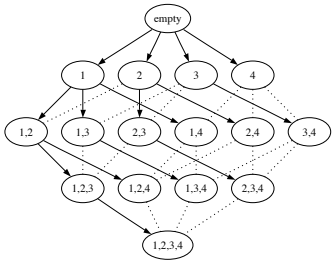
(a) Example matrix

	1	2	3	4
1	0	0.1	1.0	0.9
2	0.1	0	0.5	0
3	1.0	0.5	0	0.9
4	0.9	0	0.9	0

(b) Graph-shaped search space



(c) Tree by lexicographical order



(d) Tree by reverse search

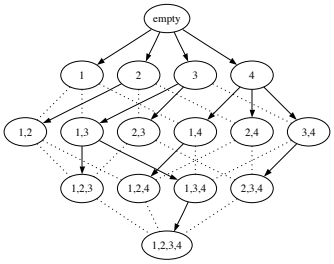


Figure 1: Illustration of reverse search.

index subsets in all n dimensions, i.e. the total number of instances involved in the cluster:

$$\mathrm{card}(U) = \sum_{i=1}^{n} |U_i|.$$

This is not to be confused with the *cluster size*, which corresponds to the number of elements in the induced subarray,

$$\mathrm{size}(U) = \prod_{i=1}^{n} |U_i|.$$

The *density* of a cluster U is defined as the average value of its elements,

$$\rho_W(U) = \sum_{k_i \in U_i} w_{k_1,\ldots,k_n} \Bigg/ \prod_{i=1}^{n} |U_i| \ . \qquad (3)$$

Due to our normalization of the data array W, the largest possible cluster density is 1. Using the above definitions, we state the problem of dense cluster enumeration as follows.

DEFINITION 1 (DENSE CLUSTER ENUMERATION (DCE)). *Given a data array W and a minimum density threshold θ with $0 < \theta \leq 1$, find all clusters U such that $\rho_W(U) \geq \theta$.*

3.2 Global Index Set

As defined in (1), an n-way array is represented using multiple index sets $V_1, \ldots, V_n$. To identify the v-th element of set V_i in a concise way, we map it to a global index as follows:

$$\mathcal{C}(v,i) = \begin{cases} v & \text{if } v \in V_i,\ i = 1 \\ v + \sum_{j=1}^{i-1} |V_j| : & \text{if } v \in V_i,\ i \geq 2 \end{cases}$$

The global index set is given by

$$\mathcal{V} = \{1, \ldots, \sum_{i=1}^{n} |V_i|\}\,.$$

A cluster U can also be represented as a subset of $\mathcal{V}$,

$$\mathcal{U} = \bigcup_{i=1}^{n} \bigcup_{u \in U_i} \{\mathcal{C}(u,i)\}.$$

Note that U and $\mathcal{U}$ are alternative representations of a uniquely determined cluster and can easily be transformed into each other. In the following, we will use the representation which is more convenient in the particular context.

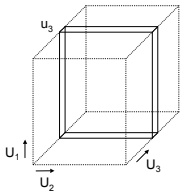


Figure 2: Visualization of a three-way cluster (dotted lines). The degree of a fixed index (here: u_3) corresponds to the sum of all array elements that fall within the slice shown by solid lines.

3.3 Search Space

Our search space can be represented as an undirected graph where each node corresponds to a cluster U. Two clusters are connected by an edge if they can be transformed into each other by adding or removing exactly one instance in one of the sets $U_1, \ldots, U_n$, that means if $\mathcal{U}$ shrinks or increases by exactly one element. Furthermore, we have to determine the root nodes, from which the search is started. In our case, each *trivial cluster* constitutes a root node, where a cluster U is defined to be trivial if it consists of exactly one tensor element, i.e. $\mathrm{size}(U) = 1$. Consequently, $|U_i| = 1$ for $i = 1, \ldots, n$ and $\mathrm{card}(U) = n$. Next, we define a reduction map for the search space.

3.4 Reduction Map

A reduction map specifies an anti-monotonic search scheme. The reduction map in multi-way cluster enumeration is similar to the special case described in section 2. Essentially, the parent of a cluster is obtained by removing the instance with minimum degree. However, in contrast to section 2, there exist multiple root nodes, so we define a spanning forest for the search space rather than a spanning tree. In the following, we give the precise definition of our reduction map. First of all, the degree in an n-way tensor is defined as follows.

DEFINITION 2 (DEGREE). *Given a cluster U, the degree of $u_j \in U_j$ with respect to U is defined as*

$$\deg_U(u_j) = \sum_{k_i \in U_i, i \neq j} w_{k_1,\ldots,k_{j-1},u_j,k_{j+1},\ldots,k_n}\,. \qquad (4)$$

W is not included as an explicit parameter of deg because our method deals with just one given data array at a time;

likewise, we often write ρ instead of ρ_W. In Figure 2, we illustrate the notion of degree in the three-way case. Fixing the $j-$th index to a specific value $u_j \in U_j$ defines a slice of the cluster. The degree corresponds to the sum of all array elements that fall within this slice. In the rest of the paper, we will use the term *slice* also for the general n-way case. Using the global index representation, the reduction map is defined as follows.

DEFINITION 3 (REDUCTION MAP). *Let $\mathcal{U}$ be a cluster and $v \in \mathcal{V} \setminus \mathcal{U}$. The reduction map is defined as $f(\mathcal{U}) = \mathcal{U} \setminus \{v\}$ where v is the minimum degree element in $\mathcal{U}$. If there are multiple minimum degree elements, the smallest one is chosen.*

This induces the following parent-child relationship between clusters.

DEFINITION 4 (PARENT-CHILD RELATIONSHIP). *For $v \in \mathcal{V} \setminus \mathcal{U}$, $\mathcal{U}^* = \mathcal{U} \cup \{v\}$ is a child of $\mathcal{U}$ if and only if*

$$(\deg_{\mathcal{U}^*}(v) < \deg_{\mathcal{U}^*}(u)) \vee ((\deg_{\mathcal{U}^*}(v) = \deg_{\mathcal{U}^*}(u)) \wedge (v < u))$$

holds for $\forall u \in \mathcal{U}$.

It remains to show that this parent-child relationship guarantees anti-monotonicity and reachability during the search.

LEMMA 1 (REDUCTION). *Let $\mathcal{U}$ be a non-trivial cluster and $\rho(\mathcal{U}) > 0$. Then for all $v \in \mathcal{U}$ with minimum degree, i.e. $v \in \operatorname{argmin}_{u \in \mathcal{U}} \deg_{\mathcal{U}}(u)$, the following properties hold:*

1. $\text{size}(\mathcal{U} \setminus \{v\}) \geq 1$.
2. $\rho(\mathcal{U} \setminus \{v\}) \geq \rho(\mathcal{U})$.

PROOF. See appendix. □

The first property refers to the reachability condition. It ensures that, when applying the reduction map to an arbitrary cluster $U = (U_1, \ldots, U_n)$, all sets U_i stay non-empty. This implies that any cluster is eventually reduced to a trivial cluster, one of our root nodes. The second property implies anti-motonicity, ensuring that a parent cluster is at least as dense as any child cluster. These properties directly suggest the following algorithm.

3.5 The DCE Algorithm

To enumerate all clusters in an n-dimensional data array W that satisfy a minimum density threshold θ, we perform the procedure shown in Algorithm 1. The first step consists in finding trivial clusters that serve as roots for further expansion. These are simply the elements $\geq \theta$ of the data array. For each of them, we perform a depth-first traversal of the search tree, generating descendants of increasing cardinality according to the parent-child relationship specified above. Note that this involves the production of child candidates, among which the actual children are selected. As soon as the density threshold is violated, we can prune the search tree. The correctness follows from lemma 1.

3.6 Implementation Details

To be able to deal with arbitrary dimensionality, our implementation expects as input a matrix where the first n columns contain the index values for the n dimensions respectively, and column $n+1$ contains real values; thus, each row represents a (non-zero) element of the multi-dimensional

Algorithm 1 Pseudocode of the DCE algorithm. W is the given n-dimensional data array with global index set $\mathcal{V}$ (and corresponding mapping $\mathcal{C}$), and θ denotes the minimum density threshold.

```
1: DCE (V, C, W, θ) :
2:   for each (k_1, ..., k_n) with w_{k_1,...,k_n} ≥ θ do
3:     DCE_Rec(V, W, θ, ⋃_{i=1}^{n} {C(k_i), i})
4:   end for

1: DCE_Rec (V, W, θ, U) :
2:   for each v ∈ V \ U do
3:     if ρ_W(U ∪ {v}) ≥ θ then
4:       if U ∪ {v} is child of U then
5:         DCE_Rec (V, W, θ, U ∪ {v})
6:       end if
7:     end if
8:   end for
9:   output U
```

array. These elements are then stored in a sparse format to keep the memory requirements low. For each element, we create a data object containing the n-dimensional index vector (converted to global indices $v \in \mathcal{V}$) and the corresponding value. To access these elements, we generate for each $v \in \mathcal{V}$ a list of pointers to the objects containing v (also called *adjacency list* of v). To search for root elements, it is sufficient to traverse the first $|V_1|$ adjacency lists: by this, we visit all data objects because each of them has to contain an index between 1 and $|V_1|$ in the first dimension.

Further, to allow for an efficient search, we maintain an array of length $|\mathcal{V}|$ which contains degree values for the current cluster $\mathcal{U}$. More specifically, for $v \in \mathcal{U}$ it contains the value $\deg_{\mathcal{U}}(v)$, and for $v \in \mathcal{V} \setminus \mathcal{U}$ the value $\deg_{\mathcal{U} \cup \{v\}}(v)$. To initialize this array for a trivial cluster $\mathcal{U}$, we fill in the value of its data element for each $v \in \mathcal{U}$, whereas for each $v \in \mathcal{V} \setminus \mathcal{U}$ we fill in the value of the data element agreeing in all indices except v with $\mathcal{U}$. After the addition of a new instance v, we traverse the adjacency list of v, filtering for data elements which contain at most one instance which is not member of $\mathcal{U} \cup \{v\}$ and updating the array accordingly. This requires at most $O(l_v n)$ operations, where l_v denotes the length of the adjacency list of v.

Using this degree array, we can check in constant time whether an extension of the current cluster by a specific instance v would satisfy the density criterion. Furthermore, it facilitates the check of the child conditions considerably. In some cases it is possible to decide by few comparisons whether $\mathcal{U} \cup \{v\}$ is a child of $\mathcal{U}$ or not, without updating the degree values of the instances in $\mathcal{U}$. The following lemma states two simple rules.

LEMMA 2 (SIMPLE PARENT CHECK). *Let v^* be the previously added instance in cluster $\mathcal{U}$, i.e.*

$$v^* = \min_{\substack{u^* \in \operatorname{argmin}_{u \in \mathcal{U}} \deg_{\mathcal{U}}(u) \\ u \in \mathcal{U}}} u^*.$$

1. *If W contains only non-negative elements and one of the following conditions holds,*
   - $\deg_{\mathcal{U} \cup \{v\}}(v) < \deg_{\mathcal{U}}(v^*)$
   - $\deg_{\mathcal{U} \cup \{v\}}(v) = \deg_{\mathcal{U}}(v^*) \wedge v < v^*$,

then $\mathcal{U} \cup \{v\}$ is a child of $\mathcal{U}$.

2. *Let a be the number of elements that the v^*-slice of $\mathcal{U}$ gains by adding v to the cluster. More precisely,*
$$a = \begin{cases} 0 & \text{if } d(v) = d(v^*) \\ \frac{\text{size}(U)}{|U_{d(v)}| \cdot |U_{d(v^*)}|} & \text{otherwise,} \end{cases}$$
where U is the subarray representation corresponding to $\mathcal{U}$ and $d(v)$ is the dimension to which v belongs. Then, $\mathcal{U} \cup \{v\}$ is **not** *a child of $\mathcal{U}$ if one of the following conditions hold:*

- $\deg_{\mathcal{U}\cup\{v\}}(v) > \deg_{\mathcal{U}}(v^*) + a$
- $\deg_{\mathcal{U}\cup\{v\}}(v) = \deg_{\mathcal{U}}(v^*) + a \wedge v > v^*$

PROOF. 1. If W contains only non-negative entries, by the addition of v the degree entries either increase or stay equal for all $u \in \mathcal{U}$.

2. Assuming a maximum weight of 1 in W, a corresponds to the maximum increase of the degree of v^* after addition of v. So, if the degree of v exceeds the maximum possible degree of v^*, or equals it, but $v > v^*$, we are sure that v cannot satisfy the conditions of the child relation. □

To check these special cases, we keep track of $\min_{u\in\mathcal{U}} \deg_{\mathcal{U}}(u)$ and the smallest minimum degree element v^*, which corresponds to the previously added instance. In all other cases, we traverse the adjacency list of v to determine the new values $\deg_{\mathcal{U}\cup\{v\}}(u)$ for all $u \in \mathcal{U}$. Then we subsequently check the conditions given in Definition 4. If they are violated by one $u \in \mathcal{U}$, we know that $\mathcal{U} \cup \{v\}$ is not a child, so we can stop.

3.7 Complexity

As we exactly solve the dense cluster enumeration problem, the algorithm might produce an exponential number of solutions, depending on the choice of the density parameter θ. Clearly, already the number of bicliques in a bipartite graph can be exponential in the number of nodes [18], which is a special case of dense cluster enumeration for a binary two-dimensional data array with density threshold 1. Therefore, in analyzing such methods, the crucial factor is the maximum computation time needed to reach the subsequent solution, also called *delay* [21]. In our implementation, each iteration of DCE_Rec needs in the worst case $O(|\mathcal{V}| + l_{v^*}n + |\mathcal{V}\setminus\mathcal{U}| \cdot (ln + |\mathcal{U}|))$ operations, where l_{v^*} is the length of the adjacency list of the previously added instance and l is the average length of an adjacency list (which is $O(|\mathcal{V}|^{n-1})$ for full tensors[2], but much lower under the assumption of sparsity). The worst case is realized if we go for each $v \in \mathcal{V}\setminus\mathcal{U}$ through the adjacency list, checking the whole n-dimensional index vector for each element, determine the updated degree values for the members of $\mathcal{U}$ and then check all conditions in Definition 4.

In practice, the number of operations is much smaller due to the following reasons. First, the density check fails for many candidate instances v, so the child conditions do not have to be checked. Second, applying the rules from the previous section, the check of the child conditions takes only constant time in many cases. Third, when traversing the adjacency list, not each object is fully investigated because data elements with too many mismatches in the index vector can be skipped.

[2]More precisely, it is $(\prod_{i=1,\, i\neq j}^{n} |V_i|)/|V_j|$ for dimension j.

Furthermore, note that constant-time access to specific tensor elements, which can be for instance achieved via hashing or by working with the full tensor representation, can significantly reduce the time complexity of one iteration. More precisely, the complexity is given by the following term, which reflects the costs for updating the degree array after adding an instance and checking the child condition for all possible extensions (by updating the degree values for the members of $\mathcal{U}$):

$$O(|\mathcal{V}|\,|\mathcal{U}|^{n-2}n + |\mathcal{V}\setminus\mathcal{U}|\,|\mathcal{U}|^{n-1}n)$$

For simplicity, the implementation we use in the experiments works with the sparse representation via adjacency lists described above. If the data is sufficiently sparse, there is not too much computational overhead compared to constant-access representations.

To show that the complexity of the delay corresponds to the complexity of DCE_Rec, we can apply the odd-even trick [21]. Returning the current cluster before the recursive calls for odd recursion depths, and after the recursive calls for even recursion depths, we obtain at least one solution during three executions of DCE_Rec. Consequently, DCE is a polynomial-delay algorithm with respect to $|\mathcal{V}|$, whereas brute-force enumeration strategies are exponential in $|\mathcal{V}|$ [21]. The memory requirements of the algorithm are quite low. In addition to the input, we basically have to store for each recursive step $\mathcal{U}$ and the degree array of length $|\mathcal{V}|$. The maximal recursion depth corresponds to the maximal cardinality $|\mathcal{U}|$ among the solution clusters $\mathcal{U}$ (minus $(n-1)$ because we start with clusters of cardinality n).

Finally, the DCE approach is compatible with distributed computation because different search trees or different branches of the same search tree can be investigated independently.

4. EXTENSIONS

4.1 Output Filtering

Usually, the user is not interested in clusters that are subsets of other cluster solutions. A straightforward approach to tackle this problem would be to go for each newly detected cluster through all previous solutions, checking for inclusions. However, the structure of our reverse search algorithm allows us to reduce the number of solutions in the output in a meaningful way without any additional costs. We can set a flag which indicates whether there exists a direct supercluster (with one additional instance) which also satisfies the minimum density threshold. If that is the case, we do not output the current cluster, otherwise we do. Trivial clusters (i.e. single-element clusters) are excluded by default. This yields us the set of all *locally maximal* clusters, i.e. all clusters $\mathcal{U}$ such that $\forall v \in \mathcal{V}\setminus\mathcal{U} : \rho_W(\mathcal{U}\cup\{v\}) < \theta$ and $\text{size}(\mathcal{U}) > 1$.

Furthermore, it might be the case that some clusters contain instances that are not at all associated with the rest of the cluster, that means the minimum degree of the cluster is ≤ 0. While such clusters with isolated instances must be allowed during the execution of our algorithm to ensure correctness for arbitrary density thresholds (see Figure 3), they are not desirable as outputs. Therefore we filter them out, so our output consists of all clusters U such that $\min_{u\in\mathcal{U}} \deg_{\mathcal{U}}(u) > 0$, $\text{size}(\mathcal{U}) > 1$, and $\forall v \in \mathcal{V}\setminus\mathcal{U} : \rho_W(\mathcal{U}\cup\{v\}) < \theta \vee \min_{u\in\mathcal{U}\cup\{v\}} \deg_{\mathcal{U}\cup\{v\}}(u) \leq 0$. Of course,

(a) Example matrix

	b_1	b_2
a_1	1.0	0
a_2	0	0.9
a_3	0.4	0.4

(b) Reduction of a cluster

$$\{a_1, a_2, a_3, b_1, b_2\}$$
$$\downarrow$$
$$\{a_1, a_2, b_1, b_2\}$$
$$\downarrow$$
$$\{a_1, b_1, b_2\}$$

[order: $a_1 < a_2 < a_3 < b_1 < b_2$]

Figure 3: Example reduction of a two-way cluster, yielding an ancestor with an isolated instance.

it is also possible to set the minimum degree threshold to values greater than zero.

4.2 Cluster Ranking

Depending on the chosen density threshold, the result set might get large. Consequently, it is important to provide a meaningful ranking criterion for clusters. Due to the exhaustiveness of our approach, it is possible to calculate for each cluster an *exact p-value*, i.e. the probability that random selection produces a cluster with at least the same density [3]. For a cluster $U = (U_1, \ldots, U_n)$ from a data array $W = (w_{k_1,\ldots,k_n})_{k_i \in V_i \ \forall i=1,\ldots,n}$, the exact p-value is given by

$$\frac{|\{U' : U'_i \subset V_i \wedge |U'_i| = |U_i| \wedge \rho_W(U') \geq \rho_W(U)\}|}{\prod_{i=1}^{n} \left(\begin{array}{c} |V_i| \\ |U_i| \end{array} \right)}. \quad (5)$$

To obtain the numerator, we have to keep track of the densities and the dimension-specific cluster sizes for all the solutions we pass during execution of the algorithm (not only the locally maximal ones). This can be done by storing for each occuring size combination an individual list of density values, either in files or in appropriate data structures. Then we group the output clusters according to their dimension-specific sizes. For each size group, we sort the corresponding density list and traverse it once to obtain the p-values for all output clusters. Finally, we sort the whole set of output clusters according to their p-values, where the lowest p-values are most interesting. By this, we can measure the statistical significance of our results without having to rely on simplified data models [17] or to perform multiple randomization tests.

4.3 Symmetry Adaptations

Often, multi-way data contain inherent symmetries. One example would be a three-way tensor that represents a set of weighted undirected networks. An entry w_{ijk} represents the weight of the edge between the i-th and the j-th node in the k-th network. Consequently, the tensor contains the same set of instances in the first two dimensions, and the entries w_{ijk} and w_{jik} are equivalent, so we have to store only one of them. Furthermore, diagonal entries w_{iik} are irrelevant and therefore should not be taken into account. In such cases, it is usually desirable that the clusters are symmetric as well, that means the clusters contain the same set of instances in the symmetric dimensions. In the above example, we would require $U_1 = U_2$. This scenario is a further generalization of the cluster detection in symmetric matrices discussed in section 2. It leads to a slightly different definition of *density*:

$$\rho(U_1, \ldots, U_n) = \frac{\sum_{k_i \in U_i, k_1 < \ldots < k_j} w_{k_1 \ldots k_n}}{\left(\begin{array}{c} |U_1| \\ j \end{array} \right) \prod_{i=j+1}^{n} |U_i|}$$

Here, we assume that the tensor is symmetric for the dimensions 1 to j, and $U_1 = \ldots = U_j$. The definitions of cluster size and degree are adapted accordingly. Lemma 1 still holds true for this generalization; the proof is analog to the basic version. It is also conceivable that several symmetry groups exist, e.g. for a four-way tensor, the first and the second dimension can be symmetric as well as the third and the fourth.

5. EXPERIMENTAL RESULTS

We implemented our DCE algorithm in C++[3]. In the following, we describe our experiments on synthetic and real-world data sets.

5.1 Artificial Data Analysis

First we tested the performance of our method on artificial data. For that purpose, we generated sparse tensors with hidden clusters. For simplicity, we used binary values, i.e. each tensor element is either 0 or 1. Let n be the number of dimensions, and m the number of clusters. Furthermore, we assumed a hypercubic model where each dimension had the same index set size d and each hidden cluster contained exactly s instances in each dimension. The clusters were allowed to overlap without any restriction. In addition, we imposed different levels of noise onto the data. Here, the noise corresponds to random 0-1 flips. Initially, all tensor elements within clusters were set to 1. Given a noise level p, we randomly selected p% of all 1-elements and the same number of 0-elements. Then the selected elements were flipped, i.e. the 1-elements were set to 0 and vice versa.

Given this model for data generation, we investigated the scalability of DCE with respect to the different model parameters d, n, m, and s. Our basic setting was $d = 100$, $n = 3$, $m = 20$, and $s = 3$. Starting from that, we did four series of experiments, varying one of the parameters while keeping the others fixed. For each configuration, we generated ten random data sets and considered noise levels from 0% to 30%. The density threshold for the DCE algorithm was chosen in dependence of the noise level, $\theta = (100 - p)$%. Figure 4 shows the resulting DCE runtime curves for each parameter. In addition to the total runtime, we report the delay, i.e. the runtime per solution. The values are averages across the ten random data sets. All measurements were performed on a 3 GHz machine.

DCE scales favorably with d, the number of instances per dimension (Figure 4a). For noise levels from 0% to 20%, the runtime remains approximately constant with increasing set size, for 30% noise it increases linearly. In the 30% noise case, the noise elements cover more instances and, due to the lowered density threshold, there exist more cluster solutions; consequently, the 30% noise curve depends much stronger on d than the other curves. The delay shows a linear increase in all cases. Regarding the number of hidden clusters, there is a linear relationship with the total runtime

[3]implementation available at `http://www.kyb.tuebingen.mpg.de/~georgii/dce.html`

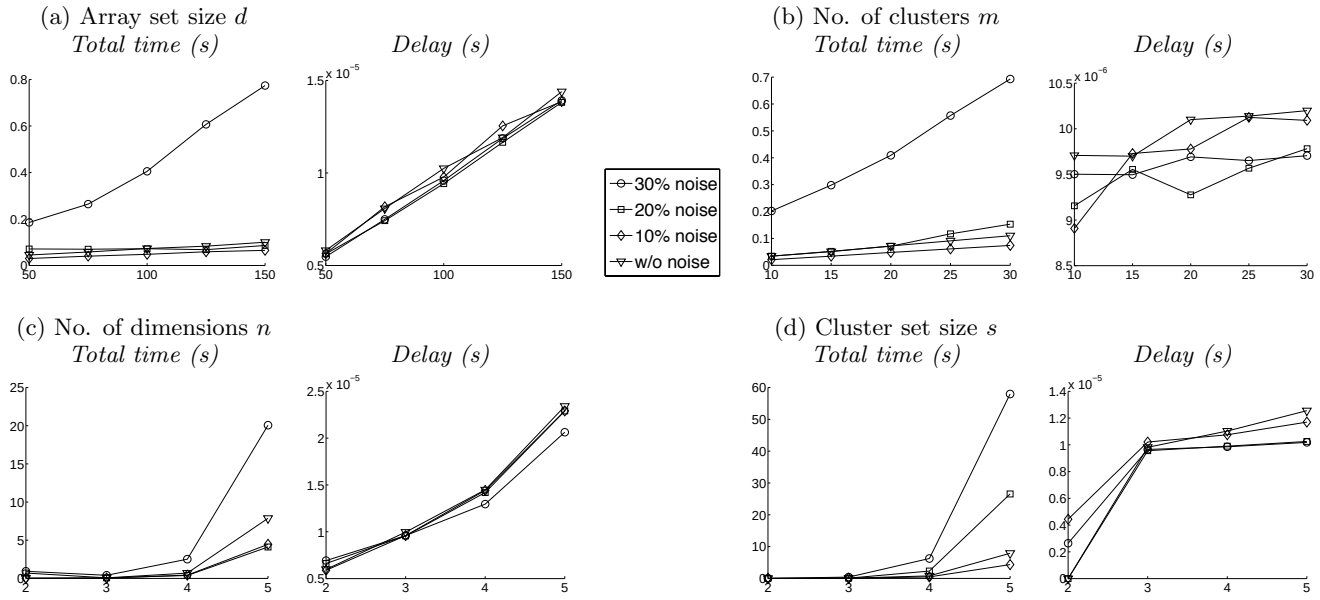


Figure 4: DCE runtime measurements for artifical data in dependence of different parameters. For each experiment we report the total runtime and the delay. The x-axis contains the values of the varied parameter, the y-axis contains the time in seconds.

at all noise levels (Figure 4b). The delay increases only very slightly; in fact, the higher number of clusters makes cluster overlaps more likely, so some instances may have longer adjacency lists, which leads in some cases to an increased computational effort per solution. Actually, the 0% curve is on top because the cluster instances have longer adjacency lists than in the noisy cases. The number of subcluster solutions per hidden cluster increases exponentially with the number of dimensions n, which is reflected by the total runtime (Figure 4c). In contrast, the increase of the delay is much more moderate. Likewise, when increasing the number of instances in the hidden clusters, the delay grows very slowly, whereas the total runtime of DCE increases significantly (Figure 4d). Again, this can be explained by the increased number of subcluster solutions. Also, the effect is much stronger for higher noise levels.

To conclude, DCE is appropriate for finding dense clusters in sparse settings; however, the number of solutions may grow considerably with the dimensionality and the size of the clusters. Remarkably, the computational effort per solution scales well, although we use the adjacency list representation of the tensor and therefore do not have constant-time access to elements. This indicates that our rules to speed up the search process are effective. Furthermore, it encourages to combine the DCE search with heuristics that restrict the generation of overlapping clusters.

5.2 Email Traffic Analysis

We applied DCE to a subset of the ENRON email data set [14] taken from [5]. It records information about the sender, the recipients, and the time stamp of emails. From this, we generated a three-way tensor as follows. We mapped each time stamp to the corresponding week and then determined the number of emails a certain sender sent to a certain recipient in a certain week. This yielded a $120 \times 143 \times 123$ tensor with 6995 non-zero elements. We were interested in groups of persons that regularly exchange a large number of emails. The individual frequency values ranged from 1 to 202, however 81% of them were lower or equal to 10, therefore we set elements with values greater than 10 to 10, in order to avoid results which are dominated by one or very few maximum elements and consequently are not so interesting. Then all tensor elements were divided by 10 to obtain normalized density values. After the preprocessing, we ran DCE with a density threshold of 0.8. That means, for a cluster solution, each sender sends in each week on average at least 8 emails to each recipient.

Restricting the maximum number of instances per dimension for each cluster to 4, we obtained approximately $3.5 \cdot 10^7$ clusters in total. This seems to be a large number of clusters, but it is reasonably small compared to the number of cluster candidates for the tensor at hand, which is $2.0 \cdot 10^{22}$ for the given maximum size constraint. Focusing on locally maximal patterns with at least two instances in each dimension, the size of the result set shrinks to 240675, and among them, there are only 142 clusters with at least three instances in each dimension. The top-ranking cluster (density: 0.82, p-value: $4.7 \cdot 10^{-20}$) is shown in Figure 5. It contains three senders, four recipients and four weeks. Senders and recipients are given by personal identifiers. Two persons appear as senders and recipients, one person appears only as sender, and two persons only as recipients. The only zero elements of the cluster are due to the absence of self-emails for the two persons in the overlap. This cluster remains the top-ranking cluster even if we drop the maximum size constraints for senders and recipients, which means that there do not exist such dense clusters involving more people.

5.3 Coexpression Analysis

As a further application of DCE, we analyzed coexpres-

	Senders:	155, 162, 169			
	Recipients:	114, 155, 165, 169			
	Weeks:	103, 108, 118, 120			
Sender	Recipient	No. of emails in week			
		103	108	118	120
155	114	≥10	≥10	≥10	≥10
155	155	0	0	0	0
155	165	≥10	≥10	≥10	8
155	169	≥10	≥10	≥10	8
162	114	≥10	≥10	≥10	≥10
162	155	≥10	≥10	≥10	≥10
162	165	≥10	≥10	≥10	≥10
162	169	≥10	≥10	≥10	≥10
169	114	≥10	≥10	≥10	≥10
169	155	≥10	≥10	≥10	≥10
169	165	8	≥10	≥10	8
169	169	0	0	0	0

Figure 5: Top-ranking cluster for email traffic data.

sion networks from multiple gene expression experiments in yeast. For that purpose, we used the gene expression data set from [7] and preprocessed it similarly as described in [10]: first, we selected the experiments with at least 6 individual measurements; then we calculated for each experiment the Pearson correlation coefficients between the expression profiles of all genes; if the correlation was positive and significant at a level of 10^{-5}, we connected the corresponding genes by an edge (weight 1). This resulted in 17 different co-expression networks on the same set of genes, each of which contained 9237 edges on average. They can be represented as a three-dimensional tensor with the genes in the first two dimensions and the identifier of the experiment (i.e. the network type) in the third dimension. As the networks are undirected, the tensor is symmetric with respect to the first two dimensions. Thus, we used the extended version of DCE that deals with symmetry.

Our task was to analyze a set of networks for the cooccurrence of dense substructures. For this special case of multi-way data analysis, there exist several competitive approaches. The Cocain method (COC) [23] detects all frequent closed γ-quasi-cliques. A quasi-clique is a set of nodes U such that each of them has edges to at least $\lceil\gamma(|U|-1)\rceil$ other nodes in U. A set of nodes U is a frequent γ-quasi-clique if it is a γ-quasi-clique in at least minsup networks, where minsup is a natural number. This criterion is stricter than our density criterion because each frequent γ-quasi-clique corresponds to a subtensor with density at least γ, but not vice versa. The closeness property of frequent quasi-cliques is similar to our local maximality criterion: it excludes subsets of quasi-cliques that have the same or lower support.

Another approach to analyze such data is the Codense algorithm (COD) [10]. It aims at detecting dense subnetworks where the edges have similar occurrence profiles across the whole set of networks. That means, in contrast to DCE and COC, it does not only require a (local) cooccurrence of dense subnetworks in a subset of the given networks, but a global correlation. For this, it first compiles edges with frequency $\geq$ minsup into a summary network, from which dense subgraphs are extracted; these subgraphs are then further analyzed with respect to the correlation of edges across all given networks. The dense cluster detection is based on a non-enumerative network partitioning strategy. The density criterion is exactly the same as for DCE, but it is applied on the summary network. However, note that a dense subtensor consisting of frequent edges will have a corresponding dense cluster in the summary network. Finally, relational data mining approaches [12] are equivalent to DCE with density threshold 100%, so we do not have to consider them separately in our evaluation.

We compared the different approaches on the coexpression networks described above. The COC code was obtained from the authors [23], and COD was downloaded from `http://zhoulab.usc.edu/CODENSE/`. The minimum edge frequency threshold in COD was set to 3. This yielded a summary network with 1444 edges involving 411 nodes. For comparison purposes, we restricted the 17 individual networks considered in DCE and COC to the same set of edges, and set the minimum network support of clusters (i.e. the minimum number of instances in the third dimension) to 3. Furthermore, COD requires a p-value threshold for the similarity of occurrence profiles, which was set to 0.01. The minimum number of genes per cluster was set to 6 in all approaches. Table 1 summarizes the results of DCE, COC, and COD for different density thresholds.

To evaluate clusters of genes, we performed a functional enrichment analysis with respect to the Gene Ontology annotation. The Gene Ontology [1] provides a framework for functional categorization of genes; it is organized in a hierarchical way. We used the Expander tool [19] to detect functional categories that are significantly overrepresented in the predicted clusters; for this, we applied the default parameters with a corrected p-value threshold of 0.05. In addition to the number of functionally enriched clusters, we report the average gene-wise reliability, the average pair-wise reliability, as well as the overall precision and recall. These measures were determined as follows. Given a cluster with one or several significantly enriched functional categories, genes that belong to the same enriched category are called homogeneous. Let $\mathrm{hg_i}$ be the size of the largest group of homogeneous genes in cluster i, and let g_i be the total number of genes in the cluster (here also called size of the cluster). Then, the gene-wise reliability of cluster i is given by $\frac{\mathrm{hg_i}}{g_i}$. The average across all clusters is given by $\frac{\sum_i \mathrm{hg_i}}{\sum_i g_i}$. Further, let $\mathrm{hgp_i}$ be the number of homogeneous gene pairs in cluster i and $\mathrm{gp_i}$ the total number of gene pairs. The pair-wise cluster reliability is defined as $\frac{\mathrm{hgp_i}}{\mathrm{gp_i}}$. Compared to the gene-wise reliability, this measure takes into account all different enriched categories of a cluster. Finally, the precision and recall measures refer to unique (homogeneous) gene pairs across all clusters. That means, each gene pair is only counted once even if it occurs in more than one predicted cluster. Note that all methods predict overlapping clusters. The table shows the number of homogeneous pairs relative to the number of all within-cluster pairs (precision) as well as the absolute number of recalled homogeneous pairs.

For 100% density, DCE, and COC are equivalent to the relational data mining setting and therefore yield the same results. However, for lower density values DCE is more flexible than the quasi-clique approach used by COC, so it achieves much higher recall, while precision and reliability remain in a comparable range. Interestingly, both for DCE and COC, the average cluster reliability with density threshold 85% is larger than with density threshold 100%. This can be explained by the fact that larger clusters are more likely to be biologically significant (note that the average cluster size is

Table 1: Comparative evaluation on coexpression data. See text for details.

	Density (%)	No. clusters	No. enriched clusters	Maximum size	Average size	Gene-wise reliability (%)	Pair-wise reliability (%)	Precision (%)	No. recalled pairs	Time (s)
DCE	100	53	52	9	6.7	95.2	92.6	84.3	215	1.7
	95	239	238	11	7.8	95.9	93.1	84.2	388	3.2
	90	1057	1048	13	8.6	95.6	92.9	81.7	642	15.9
	85	3269	3240	16	10.7	96.3	94.1	82.6	1041	114.8
COC	100	53	52	9	6.7	95.2	92.6	84.3	215	1.4
	95	53	52	9	6.7	95.2	92.6	84.3	215	1.4
	90	53	52	9	6.7	95.2	92.6	84.3	215	2.4
	85	109	108	10	8.2	97.2	95.4	85.0	260	7.6
COD	100	0	-	-	-	-	-	-	-	0.2
	95	3	3	11	9.7	100.0	100.0	100.0	80	1.4
	90	10	9	11	7.5	90.7	91.7	83.6	107	1.3
	85	9	8	10	7.9	84.5	81.3	76.1	140	1.3

increased). On the other hand, a lower density threshold allows to include genes with less connections, which might be functionally unrelated, therefore the overall pair-wise precision of DCE was slightly reduced when going from 100% to 85% density. In contrast, COC kept the precision level, as it applies the more rigid quasi-clique criterion. The cluster criterion required by COD is more restrictive than that of the other approaches, so the recall is much lower. In spite of that, while the precision and reliability values are perfect for a density threshold of 95%, at 85% density they are significantly worse than for the other approaches.

To conclude, COD scaled best with decreasing density threshold, but it recovered a smaller number of homogeneous pairs than the other approaches, and in some cases it even had lower reliability and precision. COC achieved better recall and always yielded reliable predictions. DCE is the least restrictive with respect to the cluster criterion and hence obtained the best recall values. Nevertheless, the predicted clusters were still biologically meaningful: reliability and precision were in a similar range as for COC. Besides that, DCE is applicable to more general settings, namely arbitrary tensors and non-binary weights; the pruning techniques were not specifically adapted to this setting.

6. CONCLUSION

We described an enumerative approach called DCE to identify dense clusters in tensors of arbitrary dimensionality. The density criterion is exploited in an effective way using a reverse search algorithm. In addition, our method ranks the results by exact p-values and can deal with symmetry constraints. Compared to methods that coanalyze multiple networks [10, 13, 22, 23], DCE is more general and flexible, allowing to analyze tensor data with an arbitrary number of dimensions, real or binary values, including symmetries or not. Due to its completeness guarantee, the solution set might get very large. This can be controled by tuning the density threshold and the weight distribution or sparsity of the tensor. For large-scale applications, the efficiency of DCE can be improved by using distributed computation or by exploiting more sophisticated data structures. It is also possible to integrate constraints on external data sources in our framework. For example, constraints that behave antimonotonically with the cluster cardinality can be used directly as additional pruning criteria [8]. In future work, we will introduce additional criteria like balanced weight distribution within a cluster and overlap restrictions.

Finally, multi-dimensional arrays occur in many fields, for instance neuroscience, computational biology, social studies, and web mining. Our dense cluster enumeration provides a general framework to analyze such data.

7. ACKNOWLEDGMENTS

We thank K. Borgwardt for providing the email data set and Z. Zeng and J. Wang as well as H. Hu and X. Zhou for making their implementations available. We are very grateful to G. Rätsch for his encouragement and support.

8. REFERENCES

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein *et al.*. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, 2000.

[2] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65:21–46, 1996.

[3] G. Bejerano, N. Friedman, and N. Tishby. Efficient exact p-value computation for small sample, sparse, and surprising categorical data. *J Comput Biol*, 11(5):867–86, 2004.

[4] J. Besson, C. Robardet, L. D. Raedt, and J.-F. Boulicaut. Mining bi-sets in numerical data. In S. Dzeroski and J. Struyf, editors, *KDID*, volume 4747 of *Lecture Notes in Computer Science*, pages 11–23. Springer, 2006.

[5] K. M. Borgwardt, H.-P. Kriegel, and P. Wackersreuther. Pattern mining in frequent dynamic subgraphs. In *ICDM 2006*, pages 818–822, Washington, DC, USA, 2006. IEEE Computer Society.

[6] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Data peeler: Contraint-based closed pattern mining in n-ary relations. In *SDM*, pages 37–48, 2008.

[7] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes. *Mol. Biol. Cell*, 11(12):4241–4257, 2000.

[8] E. Georgii, S. Dietmann, T. Uno, P. Pagel, and K. Tsuda. Enumeration of condition-dependent dense modules in protein interaction networks. *Bioinformatics*, 25(7):933–940, 2009.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data

Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition, 2006.

[10] H. Hu, X. Yan, Y. Huang, J. Han, and X. J. Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21(suppl_1):i213–221, 2005.

[11] R. Jaschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. Trias–an algorithm for mining iceberg tri-lattices. In *ICDM 2006*, pages 907–911, Washington, DC, USA, 2006. IEEE Computer Society.

[12] L. Ji, K.-L. Tan, and A. K. H. Tung. Mining frequent closed cubes in 3d datasets. In *VLDB '06*, pages 811–822. VLDB Endowment, 2006.

[13] D. Jiang and J. Pei. Mining frequent cross-graph quasi-cliques. *ACM Trans. Knowl. Discov. Data*, 2(4):1–42, 2009.

[14] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004*, pages 217–226. Springer, 2004.

[15] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. Technical Report SAND2007-6702, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2007.

[16] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM* 2008, pages 363–372, 2008.

[17] M. Koyuturk, W. Szpankowski, and A. Grama. Assessing significance of connectivity and conservation in protein interaction networks. *J Comput Biol*, 14(6):747–64, 2007.

[18] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45, 2004.

[19] R. Shamir, A. Maron-Katz, A. Tanay, C. Linhart *et al.*. Expander - an integrative program suite for microarray data analysis. *BMC Bioinformatics*, 6(1):232, 2005.

[20] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, New York, NY, USA, 2006. ACM.

[21] T. Uno. An efficient algorithm for enumerating pseudo cliques. In *Proceedings of ISAAC 2007*, pages 402–414, 2007.

[22] X. Yan, X. J. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 324–333, New York, NY, USA, 2005. ACM.

[23] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 797–802, New York, NY, USA, 2006. ACM.

[24] L. Zhao and M. J. Zaki. Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data. In *SIGMOD '05*, pages 694–705, New York, NY, USA, 2005. ACM.

APPENDIX

A. PROOF OF LEMMA 1

Without loss of generality, consider the case where $v \in U_j$, i.e. $d(v) = j$. First we show that $(U_1, \ldots, U_{j-1}, U_j \setminus \{v\}, U_{j+1}, \ldots, U_n)$ is a valid cluster where all index sets are non-empty. For that purpose, let us assume that $|U_j| = 1$. Then, $U_j = \{v\}$ and $deg_U(v) = \sum_{k_i \in U_i} w_{k_1,\ldots,k_n} =: T$, that means the degree of v is equal to the sum of all elements in the subtensor induced by U. Furthermore, T is positive because $\rho(U) > 0$. As U is non-trivial, there exists a $j' \in \{1, \ldots, n\}$ such that $|U_{j'}| > 1$. Let u' be an instance of minimum degree in $U_{j'}$, i.e. $u' = \operatorname{argmin}_{u \in U_{j'}} \deg_U(u)$. Then, $T > \deg_U(u')$ (for $\deg_U(u') \geq 0$: $T = \sum_{u \in U_{j'}} \deg_U(u) \geq |U_{j'}| \cdot \deg_U(u') > \deg_U(u')$; for $\deg_U(u') < 0$: obvious because $T > 0$). So we have found a $u' \in \mathcal{U}, u' \neq v$ with $deg_U(u') < deg_U(v)$, which contradicts the assumption of the lemma. Consequently, $|U_j| > 1$ and therefore $|U_j \setminus \{v\}| > 0$.

Further,

$$
\begin{aligned}
&\rho(U_1, \ldots, U_{j-1}, U_j \setminus \{v\}, U_{j+1}, \ldots, U_n) - \rho(U_1, \ldots, U_n) \\
&= \frac{\sum_{u_j \in U_j \setminus \{v\}, k_i \in U_i, i \neq j} w_{k_1,\ldots,k_{j-1},u_j,k_{j+1},\ldots,k_n}}{(|U_j| - 1) \cdot \prod_{i=1, i\neq j}^{n} |U_i|} - \frac{\sum_{k_i \in U_i} w_{k_1,\ldots,k_n}}{\prod_{i=1}^{n} |U_i|} \\
&= \frac{\sum_{u_j \in U_j \setminus \{v\}} \deg_U(u_j)}{(|U_j| - 1) \cdot \prod_{i=1, i\neq j}^{n} |U_i|} - \frac{\sum_{u_j \in U_j} \deg_U(u_j)}{\prod_{i=1}^{n} |U_i|} \\
&= \frac{\left(\sum_{u_j \in U_j} \deg_U(u_j) - \deg_U(v)\right) - \left(\sum_{u_j \in U_j} \deg_U(u_j)\right) \frac{|U_j|-1}{|U_j|}}{(|U_j| - 1) \cdot \prod_{i=1, i\neq j}^{n} |U_i|} \\
&= \frac{\sum_{u_j \in U_j} \deg_U(u_j) \left(1 - \frac{|U_j|-1}{|U_j|}\right) - \deg_U(v)}{(|U_j| - 1) \cdot \prod_{i=1, i\neq j}^{n} |U_i|} \\
&= \frac{\frac{1}{|U_j|} \cdot \sum_{u_j \in U_j} \deg_U(u_j) - \deg_U(v)}{(|U_j| - 1) \cdot \prod_{i=1, i\neq j}^{n} |U_i|} \\
&\geq \frac{\frac{1}{|U_j|} \cdot |U_j| \cdot \deg_U(v) - \deg_U(v)}{(|U_j| - 1) \cdot \prod_{i=1, i\neq j}^{n} |U_i|} \\
&= 0
\end{aligned}
$$

The inequality holds due to the choice of v. Hence, $\rho(U_1, \ldots, U_{j-1}, U_j \setminus \{v\}, U_{j+1}, \ldots, U_n) \geq \rho(U_1, \ldots, U_n)$.

Efficient computation of PCA with SVD in SQL

Mario Navas
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

Carlos Ordonez
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

ABSTRACT

PCA is one of the most common dimensionality reduction techniques with broad applications in data mining, statistics and signal processing. In this work we study how to leverage a DBMS computing capabilities to solve PCA. We propose a solution that combines a summarization of the data set with the correlation or covariance matrix and then solve PCA with Singular Value Decomposition (SVD). Deriving the summary matrices allow analyzing large data sets since they can be computed in a single pass. Solving SVD without external libraries proves to be a challenge to compute in SQL. We introduce two solutions: one based in SQL queries and a second one based on User-Defined Functions. Experimental evaluation shows our method can solve larger problems in less time than external statistical packages.

Categories and Subject Descriptors

G.1.3 [**Mathematics of Computing**]: Numerical Linear Algebra—*Singular value decomposition*; G.3 [**Probability and Statistics**]: Multivariate statistics; H.2.4 [**Database Management**]: Systems—*Relational databases*; H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms

Algorithms, Performance, Theory

Keywords

PCA, SVD, SQL, DBMS

1. INTRODUCTION

Principal component analysis or PCA is a popular technique in statistics and data mining for dimensionality reduction. It is a valuable tool to reveal hidden patterns, compress and extract relevant information from complex data sets. Several applications for image processing [6], data compression [3], pattern recognition [23], clustering [5], classification [9] and time series prediction [22] have developed over time around PCA. By successfully applying linear algebra, PCA finds the optimal linear scheme, in sense of least square errors, to reduce dimensions of a data set. Traditionally PCA is computed by exporting data as flat files compatible with statistical tool specifications. Since the complete data set is uploaded into main memory, handling large amount of information is a major concern for analysis. Few proposals have taken into account the integration of statistical methods into the DBMS, due to its limitations to perform complex matrix and vector operations. Constructing statistical models efficiently inside the DBMS is a key factor to add data mining capabilities into the DBMS [16]. Previous research has used user defined functions (UDFs) to enrich the DBMS with PCA [20]. We have extended past work by successfully implementing all steps participating in PCA with SQL statements. Compared with UDFs, SQL is portable among DBMS providers. Also the query optimizer manipulates disc and memory I/O operations to run without data size limitations. Additionally this article includes a comprehensive analysis of implementation alternatives to optimize execution time without affecting correctness of the results. Our focus is to deal with large amount of data and overcome limitations of statistical applications.

The article is organized as follows. Section 2 presents definitions. Section 3 presents our contributions, how PCA is solved inside the DBMS with SQL statements, along with some implementation choices. Section 4 presents experiments comparing different methods of correlation analysis, SVD implementation inside and outside the DBMS, SVD optimizations to reduce execution time. Section 5 includes related work. In Section 6 we present conlusions and future work.

DMMT'09, June 28, 2009, Paris.

2. PRELIMINARIES

In this section we explain PCA, the relation with SVD and the algorithm to solve the eigen-problem. The initial step is to compute the correlation matrix used as input for PCA. Next, we explain the fundamentals of dimensionality reduction, together with the connection to the eigen-problem. We show how the QR algorithm performs the Householder tridiagonalization of the correlation matrix and iteratively the QR decomposition, returning the principal components.

2.1 Definitions

Let $X = \{x_1, ..., x_n\}$ be the input data with n points, where each point has d dimensions. X is a $d \times n$ matrix, where the point x_i is represented by a column vector (equiv-

alent to a $d \times 1$ matrix). We use the subscripts i and j to indicate position of a value in a matrix, thus x_{ij} is the value at the row i and the column j of X. The subscripts k and s are used to identify the number of the current execution step. Matrix transposition is denoted by T, likewise norm of a vector as $||x_i||$. The matrix Q, from the summary matrices L and Q, is not to be confused with Q from the QR decomposition. The storage of the data set X and table definitions in a DBMS, can be found in Section 3.1.

2.2 Data Preprocessing

In order to perform PCA, the first step is either to compute the covariance or the correlation matrix of the input data set with data centered on the mean (substracting μ). The covariance matrix of X is a $d \times d$ matrix with measures of how the dimensions change together and a main diagonal of variances. It is defined as follows:

$$C_X = \frac{1}{n-1} X X^T \tag{1}$$

Correlation coefficients are more desirable because they indicate the strength and direction of the linear relationship between two variables in the range $[-1, 1]$. Therefore, a correlation matrix is normalized, symmetric $d \times d$ and populated with the degree of correlation among dimensions. Such matrix can be calculated using the summary matrices: L and Q [16]. Let L (see Equation 2) be a column vector $d \times 1$ with the linear sum of points with elements of the form L_i where $i = 1, 2, .., d$. At the same time, let Q (see Equation 3) be the quadratic sum of points, in the sense that the $d \times d$ matrix with values Q_{ij} is the sum of the cross-products of each point with its transpose. These summary matrices allow us to compute several statistical techniques efficiently, due to its small size compared to X when $d << n$. Thus, elements in the correlation matrix are given by Equation 4.

$$L = \sum x_i \tag{2}$$

$$Q = \sum x_i {x_j}^T \tag{3}$$

$$\rho_{ij} = \frac{n Q_{ij} - L_i L_j}{\sqrt{n Q_{ii} - L_i^2}\sqrt{n Q_{jj} - L_j^2}} \tag{4}$$

2.3 PCA Overview

In PCA the objective is to find a change of basis with fewer attributes while retaining as much of the variations present in the original data set as possible. The transformation U^T that maps the original data X into a new dimensionally reduced space $U^T X$ is called the principal components of X. Such projection of the sample space is defined by a linear basis where the variance of each direction is maximized and the covariance between directions is minimized. Therefore, each linear transformation or column vector of U is associated to the variance of its projection. After sorting the linear transformations accordingly to variance, we can pick the k^{th} most representative vectors to be the principal components [11]. Moreover, PCA can be used to identify the most representative features in the data set [1] by solving the column subset selection problem.

The singular value decomposition (SVD) is used to solve PCA. The duality of change of basis establishes the relationship between the right, the left eigenvalue decompositions and the input data set. Given a matrix X the equation for the SVD is as follows,

$$X = UEV^T \tag{5}$$

Solving the SVD of X produces two orthogonal basis, one defined by the left singular vectors U, and the second given by the right singular vectors V. Since U^T is the linear transformation of PCA, we need to solve the left eigenvalue decomposition problem as shown in Equation (6). The equation is obtained by substituting X by its SVD (5), where E^2 is the diagonal matrix of eigenvalues.

$$XX^T = UEV^T VEU^T = UE^2U^T \tag{6}$$

Let us consider the covariance matrix of the projection $U^T X$ given by (7). As $U^T = U^{-1}$ (U is orthogonal), we can use the eigenvalue decomposition (6), such that $U^T XX^T U = U^T UE^2U^T U = (U^{-1}U)E^2(U^{-1}U) = E^2$. Consequently, the variance of the dimensions in the projection $U^T X$ are proportional to the eigenvalues in E^2 and for the algebraic solution of the decomposition, the covariance measures between dimensions are zero. Such characteristics explain the selection of the eigenvectors in U^T as the principal component scores of X.

$$C_{U^T X} = \frac{1}{n-1} U^T X (U^T X)^T = \frac{1}{n-1} E^2 \tag{7}$$

Solving PCA is equivalent to solve the eigen-problem for the covariance or the correlation matrix. In our experiments we use correlation as it is already normalized and it requires minimum preprocessing. The eigenvalue decomposition method is performed by using Householder tridiagonialization and QR factorization. Such algorithm is explained next.

2.4 QR algorithm to solve PCA

Here we present in detail the steps of a QR based algorithm to solve the eigen-problem. Given either the correlation or the covariance matrix as an approximation of XX^T, the eigenvectors U and E^2 are computed. In Section 3, we will match every step in the algorithm with optimized operations in the DBMS, revealing guidelines to implement efficiently QR based algorithms for the eigenvalue decomposition problem. To solve SVD and PCA the approach applied in our system (see Figure 1) uses Householder tridiagonalization followed by the QR algorithm [21].

1	Compute summarization matrices
2	Calculate correlation matrix
3	Perform Householder tridiagonalization
4	Execute QR algorithm

Figure 1: Solving PCA with SVD overiew.

The summary of steps is as follows: (1) Start by computing a correlation matrix of X, $A_0 = XX^T$, the correlation matrix on the d space. (2) Apply Householder method by reducing the correlation matrix to an equivalent tridiagonal matrix, with a linear transformation T. $B_0 = T^T A_0 T$, where $T^{-1} = T^T$. (3) Find the eigenvalue decomposition of the tridiagonal matrix $B_0 = C_s B_s C_s^T$ by applying the QR factorization method, where the diagonal of B_s is the

matrix of eigenvalues, C_s is the orthogonal matrix and s is the number of iterations taken to satisfy the error criteria. (4) Finally, we can combine both decompositions to get $XX^T = (TC_s)B_s(TC_s)^T$. As a result, the diagonal matrix of eigenvalues E^2 is constructed by the diagonal elements of B_s and columns of $U = TC_s$ are the eigenvectors of XX^T.
As shown in Section 2.3, the computation of SVD can be done by solving the eigenvalue decomposition of the correlational matrix. Given $A_0 = XX^T$, the Householder tridiagonalization is done in $d-2$ steps. A rotation P_k is generatated to obtain $A_k = P_k A_{k-1} P_k$, where $k = 1, 2, .., d-2$. Consequently, we have that $T = P_1 P_2 ... P_{k-2}$. The name of QR algorithm is due to the iterative use of the QR decomposition. Thus it finds values for the matrices Q_k and R_k, given a matrix B_{k-1}, such that $B_{k-1} = Q_k R_k$. Starting from $B_0 = A_{d-2}$, in every iteration a value $B_k = R_k Q_k$ is computed; where $k = 1, 2, .., s$ and s is the iteration at which the error criteria is overcome. Once the algorithm has converged, the eigenvectors of B_0 are the rows of the matrix $C_s = Q_1 Q_2 ... Q_s$. Finally, the matrices E^2 and U are computed as $E^2 = diag(B_s)$ and $U = TC_s = P_1 P_2 ... P_{d-2} Q_1 Q_2 ... Q_s$.
The implementation receives XX^T as a parameter and computes E^2 together with U. The matrix E^2 is calculated by performing transformations over the original input correlation matrix getting a tridiagonal matrix and later the diagonal matrix of eigenvalues, this matrix will be called A. Meanwhile, the matrix U is computed at the same time, starting from an identity matrix. These two tables will keep their names during the execution of the algorithm, and they will use the subindex k to specify the step counter.

1	For k=1 to d-2
2	$\alpha = -sign(a_{k+1,k})(\sum_{j=k+1}^{d}(a_{jk})^2)^{\frac{1}{2}}$
3	$r = (\frac{1}{2}\alpha^2 - \frac{1}{2}\alpha a_{k+1,k})^{\frac{1}{2}}$
4	$w_1 = w_2 = ... = w_k = 0$
5	$w_{k+1} = \frac{a_{k+1,k} - \alpha}{2r}$
6	$w_j = \frac{a_{jk}}{2r}$ for $j = k+2, k+3, ..., d$
7	$P_k = I - 2ww^T$
8	$A_k = P_k A_{k-1} P_k$
9	$U_k = U_{k-1} P_k$
10	End

Figure 2: Householder Implementation.

The linear transformation defined by Householder [21] is composed by $d-2$ rotations P_k derived with the scalars α and r, and the vector w of n cardinality. Initially $A_0 = XX^T$ and $U_0 = I$ where I is the identity matrix. The implementation is shown in Figure 2.

1	Do
2	Find Q_k and R_k for $A_{k-1} = Q_k R_k$
3	$U_k = U_{k-1} Q_k$
4	$A_k = R_k Q_k$
5	While($error > \epsilon$)

Figure 3: QR Algorithm Implementation.

For the QR algorithm (see Figure 3), the number of steps required to reach a solution depends on an error criteria ϵ, which is computed out of the diagonal elements of A_k. The loop starts with $k = 0$ and at each step $A_{k-1} = Q_k R_k$ is calculated using the QR Factorization (see Figure 4), it becomes a nested loop of the iterative QR algorithm [21]; initially $Q_k = A_{k-1} = \{v_0, v_1, ..., v_d\}$, and the values of R_k are given by $R_k = \{r_{i,j}\}$. Notice that Householder is finished before the QR algorithm starts, therefore their step counters k are not the same.

1	For $i = 1$ to d
2	$r_{ii} = \|\|v_i\|\|$
3	$v_i = v_i / r_{ii}$
4	For $j = i+1$ to d
5	$r_{ij} = v_i \cdot v_j$
6	$v_j = v_j - r_{ij} v_i$
7	End
8	End

Figure 4: QR Factorization Implementation.

After completion of the main steps the eigenvalues need to be positive. Thus for every negative value $a_{i,i}$ in the diagonal of A, $a_{i,i} = -a_{i,i}$ and $u_{j,i} = -u_{j,i}$; where $j = 1, 2, ..., d$. Finally, $E^2 = \{a_{i,i}\}$ and $E^{-1} = \{\frac{1}{\sqrt{a_{i,i}}}\}$. In the next sections the detailed implementation of the main steps of SVD with SQL statements as well as the technical overview of the coding inside the database is exposed.

3. SVD IN SQL

In this section we explain the techniques for matrix multiplication, matrix transposition, along with the fundamental operations used in our SQL implementation. The detailed explanation of the steps of the algorithm can be found in Section 2.4, our SQL implements such definition. Further analysis of space, I/O and complexity of our SQL queries is performed for each step of the algorithm.

3.1 Table Definitions

In order to represent matrices as tables in a DBMS, we can define the columns as dimensions and rows as records. For the input matrix X the storage table, `tX`, contains the attributes $\{X_1, X_2, ..., X_d\}$. This matrix is used to compute the input correlation matrix of the algorithm in order to find its decomposition (see Section 2.4). Matrix operations, such as multiplication, can be performed with this representation by pivoting the left or the right operator to pair rows of one matrix with columns in the other and obtain the resulting table with an AGGREGATE statement.
Another way to store matrices is vertically [14]. Vertical layout matrices $\{i, j, val\}$ are implemented to perform multiplications and other matrix operations needed to solve the eigen-problem. There is no need to pivot multiplying matrices and it is a very convenient method to implement operations in the algorithm steps. For instance, the multiplication of a pair of matrices A and B, which are stored with vertical layout, could be done in one SQL statement with one JOIN condition and an AGGREGATE function. Moreover trivial values are not required to be stored, such as, zero values of an sparse matrix or values which can be computed from or are equal to other elements in the same matrix. We also exploit the fact that for a multiplication result, the element $\{i, j, val\}$ can only exist if there exists at least one element $\{i, k, val\}\}$ for the left operator and at least one match $\{k, j, val\}$ for the right operator. This is

used to avoid unnecessary pairing and to reduce the number of values involved in the AGGREGATE. Notice that the computation of a value during multiplication of two $d \times d$ matrices will have the cost of joining (pairing), a product operation and an aggregation of d values. Pursuing to alleviate the computation time of JOINs, which are the most expensive computation in SQL, our code uses constraints that cut down the unnecessary records during operations.
For PCA (see Section 2.4), the resulting principal components and their associated variances will be stored on tables `tU` and `tA` respectively. Table 1 has the summary of tables used for storage and their scope. In the QR factorization the tables `tQ` and `tR` have an extenal scope in the sence that they exist outside the loop, then the QR algorithm uses them to compute new values for the tables `tA` and `tU`. Therefore, tables `tQ` and `tR` are not needed beyond the scope of an iteration of the QR algorithm. Thus, even though names overlap for local table definitions, the only common tables between steps are `tU` and `tA`.

Table 1: Summary of tables in SQL.

Step	Global	Local
Correlation	tX, tA	tL, tQ
Householder	tA, tU	tW, tP, tH, tT
QR algorithm	tA, tU	tQ , tR, tT

Further optimization is done to take advantage of SQL when performing Householder and QR factorization. The details about how SQL statements and tables map the steps of the algorithm will be shown in the next sections.

3.2 Correlation Computation

The computation of n, L, Q are sufficient statistics to compute the correlation matrix (see Section 2.2). When the data set is in vertical layout, the table stores one correlation value per row as $\{i, j, \rho_{ij}\}$. Since the matrix is symmetric, it is sufficient compute a triangular matrix with the number of rows of the resulting matrix equal to the total number of correlation pairs ($d(d+1)/2$). The number of INSERT statements to compute n is 1, L is d and Q is $\frac{d(d+1)}{2}$. Retrieving every singular value with an AGGREGATE function involving a scan of `tX`, can be avoided with a more conservative approach [15, 17]. It is a single SELECT with $1+d+\frac{d(d+1)}{2}$ AGGREGATE statements using `tX`.

```
SELECT sum(1.0) AS n
,sum(X_1),sum(X_2)...,sum(X_d) /*L*/
,sum(X_1 * X_1) /*Q*/
,sum(X_2 * X_1),sum(X_2 * X_2)
⋮
,sum(X_n * X_1),sum(X_n * X_2),...,sum(X_n * X_n)
FROM tX;
```

The resulting query is stored on a temporary table and then inserted with vertical layout in the tables `tL` and `tQ`. Finally, the values of the correlation matrix are computed from the summary matrices without performing more scans on the original data set. The final table storing the correlation coefficients, `tA`, has 3 columns and $(d(d+1)/2)$ records.

3.3 Householder Transformation

Our implementation of Householder in SQL uses vertical matrices to exploit the storage of sparse matrices, and also other optimizations of operations. The resulting tables `tA` storing A_{k-2} and `tV` storing V_{d-2} are obtained using only INSERT statements. We use the auxiliary table `tH` to store the part of the matrix A_k that will be rotated during following iterations and the table `tT` to store the sub matrix of V_k that will continue to be involved in further operations. Such implementation is possible as the matrix P_k is composed of a diagonal identity matrix of k elements and a square matrix starting at the position $(k+1, k+1)$. Consequently, the values of P_k that are not trivial have the form $\{[k+1, d], [k+1, d], val\}$ and stored on the table `tP`. During each iteration instances of the tables `tH`, `tT`, `tP` and `tW` are computed, along with the variables `alpha` and `r`. Meanwhile the values inserted in `tA` and `tV` are never updated.
First `tH` is instantiated with the input correlation matrix. Values for the variables `alpha` and `r` are computed with SELECT and AGGREGATE statements over the table `tH`. The table `tW`, storing the vector w, has the form $\{[k+1, d], val\}$, not including zero values. Since `tW` has $d-k-1$ values and the table `tP` is symetric, only the upper or bottom diagonal matrix is stored. `tP` has size $((d-k+1)(d-k+2)/2)+1$, which is calculated with two statements: one to insert the record $\{k, k, 1\}$, necessary to include the column and the row k in our operation results, and other to compute the difference between I and the cross product of the values in `tW`. As shown in Figure 5, during iteration k only the sub

$$\begin{bmatrix}
a & a & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
a & a & a & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & a & a & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & a_{k,k} & a_{k,k+1} & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & a_{k+1,k} & h & h & \dots & h \\
0 & 0 & 0 & \dots & 0 & h & h & \dots & h \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & 0 & h & h & \dots & h
\end{bmatrix}$$

Figure 5: Storage of matrix A at step k, in `tA`:$\{i, j, a_{ij}\}$ and `tH`:$\{i, j, h_{ij}\}$.

matrix $\{[k, d], [k, d], value)\}$ changes from A_{k-1} to A_k. The rotation `tP` of `tH` from the previous iteration with the form $\{[k, d], [k, d], val\}$ has the elements $a_{k,k}$, $a_{k+1,k}$ and $a_{k,k+1}$ of the final matrix A_{n-2} to be stored on `tA`, also the sub matrix $\{[k+1, d], [k+1, d], val\}$ to be the new instance of `tH`. The matrices stored on `tH`, `tP`, `tA` are symetric, therefore only the triangular upper or lower matrix is stored, moreover computations are cut down only for such values. Nevertheless, it is important to include the values not stored at the moment of doing the aggregations of matrix multiplications.

Since P_k is applied as a transformation at the right side of U_{k-1}, it only alters values for the columns of the range $[k+1, d]$. At each iteration a new instance for the table `tT` is generated, which stores a sub matrix of U_k (see Figure 6) with tuples of the form $\{[2, d], [k+2, d], val\}$ obtained from the multiplication between the instance of `tT` at it-

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & u & u & \dots & u_{2,k+1} & t & t & \dots & t \\ 0 & u & u & \dots & u_{3,k+1} & t & t & \dots & t \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & u & u & \dots & u_{k+1,k+1} & t & t & \dots & t \\ 0 & u & u & \dots & u_{k+2,k+1} & t & t & \dots & t \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & u & u & \dots & u_{d,k+1} & t & t & \dots & t \end{bmatrix}$$

Figure 6: Storage of matrix U at step k, in `tU`:$\{i, j, u_{ij}\}$ and `tT`:$\{i, j, t_{ij}\}$.

eration $k-1$ and the values in tP. The remaining tuples $\{[2, d], k+1, val\}$ are to be inserted in `tU`.

3.4 QR Factorization

As described in Section 2.4, the QR factorization is iteratively called by the QR algorithm. In each step, a row of R_k is computed and the columns v_j of Q_k where $j \geq i$ are modified. As shown in Figure 7, Q_k is stored on `tT` and `tQ`. The table `tT` is initiated with a copy of the input matrix A_{k-1}. For the step i, v_i and $r_{i,i}$ are computed, both to be inserted in `tQ` and `tR` respectively. Vertical matrix representation allows us to compute the internal loop with three SQL statements. One for the column of R_k $\{[i], [i, d], val\}$, with the AGGREGATE of the JOIN between v_i and `tT` on the row value. The second to compute the new instance of `tT`, with a join between the previous `tT`, v_i and `tR`. Notice that this matching will result on records of the form $\{[1, i+1], [i+1, d], val\}$, consequently the remaining $\{[i+2, d], [i+1, d], val\}$ from the old `tT` is inserted with a third statement. Given $@i = i$ and $@r = r_{i,i}$, the SQL is as follows,

```
/*New values for tQ*/
INSERT INTO tQ
SELECT tT.i, tT.j, tT.val / @r
FROM tT
WHERE tT.j = @i;

/*New values for tR*/
INSERT INTO tR
SELECT @i, @i, @r
UNION ALL
SELECT tQ.j, tT.j, sum(tQ.val * tT.val)
FROM tQ,tT
WHERE tQ.j=@i AND tT.j>@i AND tQ.i=tT.i
GROUP BY tQ.j, tT.j;

/*New instance of tT*/
INSERT INTO new_tT
SELECT tT.i, tT.j, tT.val - tQ.val * tR.val
FROM tQ, tT, tR
WHERE tT.j>@i AND tQ.j=@i AND tR.i=@i AND tR.j=tT.j
      AND tQ.i=tT.i
UNION ALL
SELECT i, j, val
FROM tT
WHERE tT.j>@i and tT.i>@i+1
```

Finally, we have that no UPDATE statement is required to obtain Q_k and R_k. Which is very close from the database point of view. And the only table that needs to be recomputed per iteration is `tT`.

$$\begin{bmatrix} q & q & q & \dots & q_{1,i} & t & t & t & \dots & t \\ q & q & q & \dots & q_{2,i} & t & t & t & \dots & t \\ 0 & q & q & \dots & q_{3,i} & t & t & t & \dots & t \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & q_{i,i} & t & t & t & \dots & t \\ 0 & 0 & 0 & \dots & q_{i+1} & t & t & t & \dots & t \\ 0 & 0 & 0 & \dots & 0 & t & t & t & \dots & t \\ 0 & 0 & 0 & \dots & 0 & 0 & t & t & \dots & t \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & t \end{bmatrix}$$

Figure 7: Storage of matrix Q_k at step i, in `tQ`:$\{i, j, q_{ij}\}$ and `tT`:$\{i, j, t_{ij}\}$.

3.5 QR Algorithm

During each iteration of the QR algorithm, there is a QR factorization, two matrix multiplications and an *error* computation. Since the convergence iteration is not known beforehand, new solution tables `tU` and `tA` are computed each stage. The SQL for QR factorization was explained in Section 3.4. As for the multiplications, only optimization of the computation A_k is feasible. Since A_k is a tridiagonal symetric matrix, the SQL is constrained to calculate the main and one of side diagonals of the multiplication between the matrices in the tables `tR` and `tQ`, result to be stored on `tA`. The table `tU` with the value of U_k is computed with a query of the previous instance of `tU` and `tQ`. A value for the variable *error* is computed with a SELECT of the maximum difference of the diagonal elements between the tables storing A_{k-1} and A_k. Finally, the loop will conclude when the value of the *error* is less or equal the parameter ϵ.

3.6 Implementation Alternatives

Data access and manipulation in a DBMS could be done using different tools, hence, PCA is not restricted to SQL statements. Here we explain how external libraries, UDFs and Database Connectivity APIs are used to implement the main steps of PCA: correlation computation and eigenvalue decomposition. Even though external libraries are specialized pieces of software that have been developed over time to optimize execution time of their algorithms, most of them are not designed to overcome memory heap constraints. Such limitation is evident when attempting to apply large data sets to these external libraries for statistical and data mining analysis [7]. APIs, like JDBC or ODBC, allow the user to access and manipule data from external programs. This collection of libraries grant data mining applications connectivity to data sources from different DBMS providers and they can be used to minimize memory usage. Database connectivity components are extremely portable, for most of them few properties need to be changed along with the driver to switch the DBMS. Since the execution speed depends on the communication channel and for security reasons, it is desirable to minimize the amount of data transmitted through the network. Other exploitable feature

of a DBMS are user defined functions or UDFs, which are compiled to be embedded into the DBMS and can be called using SQL statements. Therefore, a UDF executes inside the DBMS without delay transmitting information. On the other hand, UDFs depend on the DBMS specification, making them not portable among providers and sometimes not feasible due to access constraints. Our focus is to optimize performance of PCA computation for large data sets by minimizing memory allocation and execution time, without altering correctness of the results.
So far we have explained how to perform each step of SVD with SQL statements(see Section 3). To avoid redundancy, the structural components of the other implementation alternatives besides SQL are generalized with pseudo-code. The correlation matrix can be computed with one pass over the original data set [16]. Reading one record in the table at a time, the summary matrices are computed as an aggregation. Thus, a data structure is used to store n, L and Q, where the values obtained by combining attributes of the current record are accumulated. After finished over the whole or part (sampling) of the table, the correlation matrix is generated as specified in Section 2.2. Finally, the steps for eigenvalue decomposition, given the correlation matrix as an array of two dimensions, are mapped for most programming languages [19, 4, 10]. For our experiments we use the algorithm specification in Section 2.4.

3.7 Time and Space complexity

For the computation of the summary matrices L and Q, along with the covariance matrix, the complexity of operations in the algorithm (Section 3.2) are $O(dn)$ and $O(d^2)$ respectively, the same complexity is shared by the SQL, Java and the UDF implementations. In the other hand, for the SVD (Section 2.4) all implementations maintain the same steps for the algorithm. However, for SQL operations, we need to take into account the complexity of cross product and joins. This problem is alleviated with reduction of records participating in computations by constraints and by not physically represent elements with predictable values. Complexity of the SVD [19] is analyzed by operation and iteration. Solving Householder is $O(d^3)$, with $n-2$ steps of $O(d^2)$ each. Table 2 shows the number of operations for Java and the UDF, along with the records involved in operations for SQL. Even though each step of the QR factorization is $O(d^2)$ (see table 3), after completion of d steps the overall complexity is $O(d^3)$. Once again the representation in SQL incorporates equal or less records for operations. The QR algorithm includes matrix multiplications besides the QR factorization, preserving complexity of $O(d^3)$ per step. However, with the number of iterations for convergence s, the QR algorithm is $O(sd^3)$. In table 3 we observe that eventhough the records for U are not reduced in SQL, only the elements on the three diagonals of A are computed.

As for the implementation alternatives to SQL, space requirements are static. Operations of matrices are done by changing values already allocated in memory. In SQL the space requirements are closely related to the number of records used in operations or query statements. Therefore, tables in Householder and the QR factorization reduce on size when the step counter increases. Reducing the numerosity of records taking part in the join statements, improves the execution speed of this approach when compared to the counter scenario were every single value is computed for each step of the algorithm.

Table 2: Householder I/O per step.

Matrix	UDF & Java	SQL
A	$\frac{(d+1)(d)}{2}$	$\frac{(d-k+2)(d-k+1)}{2}$
U	d^2	$(d-k+1)^2$

Table 3: QR algorithm I/O per iteration.

Matrix	UDF & Java	SQL
Q_k	$\sum_{i=1}^{d} d(d-i)+d$	$\sum_{i=1}^{d} i(d-i)+d$
A	$d \times d$	$2d-1$
U	$d \times d$	$d \times d$

4. EXPERIMENTAL EVALUATION

In this section we present the experimental evaluation on SQL Server DBMS. The server had an Intel Dual Core CPU, 2.6GHz, 4GB of memory and 1TB on disk. The DBMS ran under the Windows XP operating system and has implementations of the UDFs compiled. We also used a workstation with a 1.6 GHz CPU, 256 MB of main memory and 40GB on disk with JDBC connection, a SQL script generator, an implementation of SVD, JAMA and the R application for comparison purposes. We implemented three UDFs: one to compute the correlation matrix, a second to compute the SVD given a matrix and a third that computes the correlation matrix together with its SVD decomposition to return the principal components of the given table. Our optimizations combine UDFs and SQL to improve execution time. Likewise, the optimizations are also implemented with JDBC by executing some operations inside the DBMS with SQL statements and others in the workstation (see Section 3.6). For R, all the data sets were transformed into flat files, the time of exportation was not taken into account during speed measurements. JAMA has a JDBC connection to access data in the DBMS, therefore measurements include time taken to import data.
We used three real data sets from the UCI Machine Learning Repository. This information comes from different backgrounds which require dimensionality reduction. The first data set contains general information on Internet users with a combination of categorical and numerical data(n=10000, d=72). A second data set is of cartographic variables for forest cover type prediction of 30×30 meter cells (n=100000, d=54). The last data set is of US Census Data with numeric and categorical attributes (n=100000, d=62).
We varied the number of dimensions d for time performance comparisons and to test scalability. Since the results are numerical approximations, we use the maximum expected relative error δ of the most representative eigenvalue as measure of precision. Notice that the only parameter for execution is the convergence criteria ϵ which can be modified to warranty any desired expected maximum error. The external Java package JAMA (A Java Matrix Package) and R do not

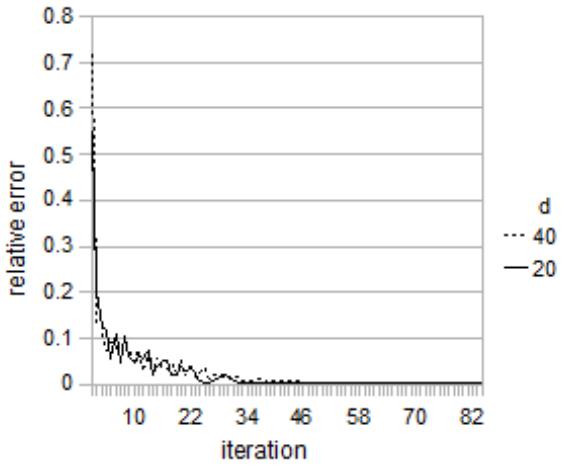


Figure 8: PCA with SQL relative error (δ).

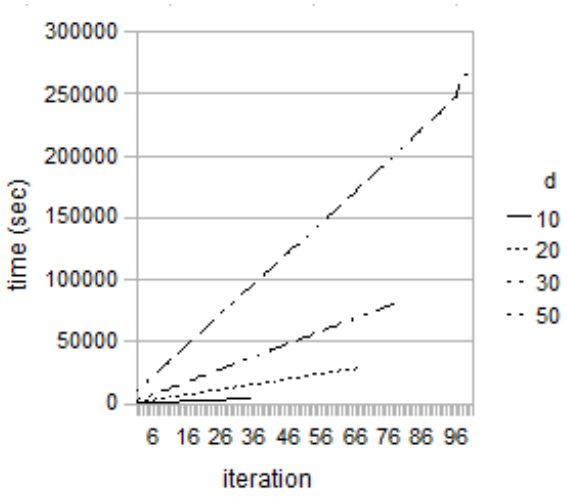


Figure 9: PCA with SQL time.

require to set any parameter to compute the eigenvalue decomposition. Considering that all the implementations converge to the same direction, yet all solutions are numerical approximation, we report correctness based on the maximum error expectation. Given an input table of $d \times n$, the resulting principal components or eigenvectors matrix will have size $d \times d$. Finally, we have that each experiment is repeated 5 times and the average is reported.

4.1 Result Correctness

Our goal is to minimize the execution time without affecting the correctness of solutions. In Section 3, we showed the implementation of our QR algorithm (see Section 2.4) in SQL, without modifying any step or main operation. Optimizations are required not only to minimize the storage requirements of the matrices, but also to reduce accumulated error; since trivial or known zero values are cut off the operations. As expected, roundoff errors can swamp the process from the true solution. Also in SQL we are bound to work with the floating point operation precision of the DBMS. However, results show that the optimizations efficiently help to prevent accumulated roundoff and floating point operation errors.

Convergence of the SQL implementation solving SVD over the US Census data set ($n = 1000$) is presented in Figure 8. The plot shows how the relative error of the eigenvalue diagonal rapidly decreases during the first steps of the algorithm eventually converging bellow the desired error criteria. As presented in Figure 9 the execution time of the SQL implementation quickly increases with the number of dimensions d. Since SVD operates over a $d \times d$ matrix, the eigenvalue decomposition is not affected by any change on the number of instances in the data set. We explained in Section 3.1 how the implementation reduces the number of values to be computed during each iteration to the minimum. Also how we take advantage of SQL and our matrix representation to acomodate matrix operations in such a way that only sub matrices interact during each step. Moreover, a value computed will only be used no more than two times before the table where it is contained is replaced by a new instance. However, the volatile nature of the matrices and the number of cross joins performed during every execution are not to be avoided. The remarkable advantages of an SQL implementation is to be linearly scalable with n and not to have a memory heap limitation. Therefore, this approach can be used to compute PCA for a table of any size. Finally, more analysis needs to be done comparing the results against different external tools. From our experiments we have that our implementations and external libraries converge to the same direction, yet all the values vary on decimal places. Such analysis would focus on the impact of roundoff errors in the SQL floating point operations and how different algorithms to solve SVD perform better for certain datasets.

4.2 Implementation comparison

Based on the algorithm, the first step is to compute the correlation matrix. Table 4 shows a computation time of the correlation matrix in SQL and the time required by an external library to generate the same matrix in main memory. The JAMA package outperforms SQL for all the cases of $n = 10$. Since L requires d and Q a number $d \times (d+1)/2$ INSERT queries, it is not beneficial to use SQL when the number of records is relatively small. However, for all the cases with $n \geq 100$, the advantage of executing operations inside the database is evident. The database not only overcame the memory issue, but also surpassed Java execution time. When correlation is computed vertically, the number of scans done to the original data set increases with d. For convenience we improved our vertical correlation matrix computation using one horizontal aggregation query with all the values for n, L and Q (see Section 3.2). Therefore, no more than one scan of the data is needed. The time of inserting the resulting attributes of the query, one at a time in the vertical result table, is insignificant when compared to the multiple scans of a table with a large number of rows. SQL Server limits the number of columns based on the total size of the row, which is capped at 8KB. As a result, when only using floating point columns, the maximum number of dimensions that can be computed with one scan is 42. A larger number of dimensions would require additional scans on the original table because we need to split the aggregate in order to fit the maximum row size allowed by the DBMS.

This explains why the horizontal aggregate is outperformed by the vertical aggregate when $n = 10$ and $d \geq 50$. The UDF implementation (see Section 3.6) looks promising in comparison to the SQL when $n \leq 100$, yet not enough to have faster results than JAMA if the data set is relatively small $n = 10$. Finally, we have that the horizontal aggregate dominates for $n \geq 100$, due to DBMS optimizations to perform aggregates fast on large tables.

Table 4: Correlation computation time in seconds with JAMA using JDBC (*out of memory).

$n \times 1000$	d	Vertical aggregate	Horizontal aggretate	UDF	JAMA
10	30	7	2	3	5
10	50	20	21	16	11
10	70	40	59	34	12
100	10	3	1	2	13
100	20	10	7	7	25
100	30	23	16	16	32
100	50	57	53	47	*
1000	20	>1K	33	72	*
1000	40	>1K	154	261	*

Table 5 compares the execution time of SVD for our SQL and UDF implementations. They use the correlation matrix from the previous step to generate a numeric SVD decomposition. The relative error δ with respect to the most representative eigenvalue is set the same to compare speed. Since both implementations overcome the error criteria after certain number of steps s, there is no precision lost when executing operation in SQL. However, the price to pay is on the number of iterations executed. Considering that all the operations are done in main memory, it was expected for the UDF implementation to surpass SQL in speed. Notice that the time to compute the correlation matrix matrix was not taken in account during measurement. In order to optimize execution speed of PCA, we can combine the different implementations, this comparison is analyzed bellow.

Table 5: SVD computation time in seconds with s iterations for convergence.

$n \times 1000$	d	δ	SQL	s	UDF	s
10	30	3.76^{-3}	92	78	1	80
10	50	3.65^{-3}	481	146	3	115
10	70	2.92^{-3}	978	172	5	122
100	10	3.49^{-5}	6	34	<1	34
100	20	2.71^{-4}	30	66	2	66
100	30	2.52^{-3}	82	77	4	76
100	50	2.44^{-3}	506	174	10	126
1000	20	2.07^{-4}	39	84	<1	84
1000	40	1.15^{-3}	113	64	2	62

4.3 Optimizations

In this section we present strategies to improve the computation of PCA. For comparison purposes we used the statistic package R and the Java library JAMA. Since the execution of R is independent from the DBMS and data files are used as input, there is no importation cost or connection delay with the database. R is dominant as long as memory limitations are not reached. It has the reading cost from the hard drive and I/O operations at memory level which increases linearly with the data size $n \times d$. Our results show that we can get similar execution speed with minimal memory requirements and without compromising solutions. In the other hand, JAMA is used with a JDBC connection to import the data into main memory before execution. It has I/O operation cost of the JDBC connection, plus execution time in main memory. Such implementation point up the main limitation of data mining applications, to find the appropriate way to manipulate data inside the DBMS. Like R, JAMA has memory heap allocation constraints. It is important to analyze the strength of each implementation since an optimal configuration depends on environment constraints and data set size. Table 6 presents total execution time of our implementations. The first of them has every step of PCA with SQL statements. We also combined the SQL correlation computation with SVD as a UDF in the DBMS, and outside with JDBC and Java. Furthermore, PCA is implemented as a UDF which receives an input table and returns its principal components as tables in the DBMS.

Table 6: PCA execution time comparison in seconds against R without flat file creation time cost, and JAMA with JDBC (*out of memory)(*/* for hybrid methods).

$n\times$ 1000	d	SQL	SQL/ UDF	SQL/ Java	UDF	R	JAMA
10	30	96	3	6	3	3	8
10	50	501	22	23	17	4	11
10	70	1020	47	50	34	6	13
100	10	7	2	4	2	5	14
100	20	37	9	9	8	8	27
100	30	98	20	20	17	13	34
100	50	559	62	67	47	21	*
1000	20	72	34	39	72	*	*
1000	40	267	156	158	262	*	*

The implementation that uses JDBC and JAMA outperformed our approaches when the data set size is relatively small $n = 10$. However, for larger data sets, the external libraries suffer because the memory limitation prevents them from analyzing these data sets. Likewise, time difference with R is more evident for the same data set when $d = 50$ and $d = 70$. The reason for this results is that we focused on improving execution time for large data, with the flaw of not performing so fast for small data sets.

The main weakness of our SQL implementation is data set dimensionality. To solve this issue, we also implemented a version of the program with DBMS connection. It uses JDBC to execute queries that compute the correlation matrix, extracts it from the database and solves the eigenvalue decomposition. The combination of this two techniques yields a desirable implementation for a data mining implementation which performs remarkably for large data sets. Results show growth that is similar to the SQL/UDF implementation, which uses a similar procedure to calculate PCA with correlation computation inside the DBMS and the UDF to compute SVD. The difference between both implementations is due to the transmission velocity of the data containing the correlation matrix. SQL/Java requires the transmission of data through the communication network. Unlike our dedicated server for experiments, transmition

time can increase on a concurrent network. The two implementations assume the correlation matrix can be stored on main memory. Therefore, computations of SVD, the $O(sd^3)$ numerical iterative process, execute extremely fast. PCA implemented as UDF has better result for data with moderate size. With the data set of $n = 100$ its results are similar to our SQL implementation. Having the UDF to perform only one scan of the data set is efficient and has fewer speed impact when the number of dimensions increases. However, optimizations inside the DBMS to execute aggregate queries are the more efficient approach for large data sets. Our results show performance, along with a guideline to select the best combination, for speed and correctness, depending on the data set characteristics. Since we present the impact of the data set size n and dimensionality d on the options available to perform PCA over information stored on a DBMS. The current work gives a precise perspective of the weaknesses and strengths of each implementation alternative to solve SVD and PCA.

Table 7: Execution time percentage (%) of PCA computation (*/* for hybrid methods).

		Correlation			SVD		
$n\times$			SQL/	SQL/		SQL/	SQL/
1000	d	SQL	UDF	Java	SQL	UDF	Java
10	30	2	67	33	98	33	67
10	50	3	95	91	97	5	9
10	70	3	85	80	97	15	20
100	10	14	50	25	86	50	75
100	20	19	78	78	81	22	22
100	30	43	56	56	57	44	44
100	50	9	85	79	81	15	21
1000	20	45	97	85	55	3	15
1000	40	58	99	97	42	1	3

5. RELATED WORK

There is a wide range of sucessful applications based on PCA for image processing [6] and pattern recognition [23]. Data mining has also exploited dimensionality reduction for data compression [3], clustering [5] and classification [9]. Although there has been considerable amount of work in machine learning and data mining to develop efficient and accurate techniques, most data mining work has concentrated on proposing efficient algorithms assuming the data set is a flat file outside the DBMS. Statistics and machine learning have paid little attention to large data sets, whereas that has been the primary focus of data mining.

Most research work has focused on modifying existing algorithms for sparse data, and to optimize methodologies to solve SVD [4]. One of the recent research extensions of PCA is feature selection [1], to define a subset of the original set of attributes that represents characteristics of the data with minimal information lost. Considering that nowadays most of the information is captured using a DBMS, this methodologies lack tools to appropiatetly do the preprocessing required to execute its algorithms.

There are some research in databases involving visualization [12] and text mining using PCA. SVD has been used in database research to perform Collaborative Filtering (CF) [18] and information retrieval [2]. PCA [8] and correlation analysis are two common techniques to analyze data. Investigation has been done to get statistical models of the data inside the DBMS fast [16, 13, 17]. In order to add PCA capabilities to a DBMS, a recent approach was to use UDFs to implement the complex matrix operations involved in SVD for microarray data analysis [20]. We have gone beyond integrating such methodologies with UDFs in a non declarative programming language such as C++ or Basic. Our approach is to implement efficiently every operation in PCA with SQL statements, providing guidelines of time performance, correctness and scalability of the alternatives to incorporate PCA into the DBMS.

6. CONCLUSIONS

Our proposal is about integrating PCA computation capabilities with a DBMS using SQL queries and UDFs. We focus on solving PCA with singular value decomposition (SVD). We explored techniques in order to improve execution time, get correct results and achieve scalability. Since memory size is a limitation of statistical packages, our methods overcome memory limitation without compromising accuracy and improving performance. To build a summarization of the data set the number of scans was reduced down to one, and the resulting summarization is used to compute the correlation matrix. Computing a decomposition of the correlation matrix to find the principal components is a complex methodology that requires several matrix operations. We studied how efficiently compute SVD with different alternatives, including SQL statements, UDFs and external libraries. We proposed some optimizations for each alternative to improve time performance, we tested several combinations to overcome limitations of our implementation alternatives. The results show that our scheme can execute fast for large data sets compared to statistical packages and to find solutions when they have reached memory limitations. In order to test the data mining client experience, our experiments are done from a workstation with a database connectivity API to execute operations outside the DBMS. We experimentally compared all the alternatives, we found that computing the correlation matrix with SQL statements is faster for large data sets, due to sufficient summarization matrices. UDFs are faster to compute SVD, since the algorithm used is iterative. Consecuently, the hybrid method of computing the correlation matrix with SQL and SVD with a UDF, has the best time performance for large data sets. The scalability is linear in data set size and cubical in dimensionality. Our solution can work efficiently completely inside the DBMS.

These are several issues for future research. Statistical and data mining techniques can benefit from our approach to compute PCA inside the DBMS. Deep analysis of how to best implement numerical methods with minimal I/O against the DBMS. The column subset selection problem is to find a subset of the most important attributes of a data set, this can be done exploiting our scheme to compute PCA, more analysis of SQL and UDFs to compute the NP-complete procedure is still required. Moreover, we have only considered the problem when the input data set has more records than dimensions, and focused on optimizing performance for large data sets. Future work must include number of dimensions greater than records, for which our implementations are not efficient in execution time.

7. REFERENCES

[1] C. Boutsidis, W.M. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–69, New York, NY, USA, 2008. ACM.

[2] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, 2002.

[3] S. Chitroub, A. Houacine, and B. Sansal. A new pca-based method for data compression and enhancement of multi-frequency polarimetric sar imagery. *Intell. Data Anal.*, 6(2):187–207, 2002.

[4] A. d'Aspremont, F. Bach, and L. Ghaoui. Optimal solutions for sparse principal component analysis. *J. Mach. Learn. Res.*, 9:1269–1294, 2008.

[5] C. Ding and X. He. K-means clustering via principal component analysis. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 29, New York, NY, USA, 2004. ACM.

[6] J.J. Gerbrands. On the relationships between svd, klt and pca. *Pattern Recognition*, 14(1-6):375–381, 1981.

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, September 2000.

[8] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*. Springer, New York, 1st edition, 2001.

[9] M. Hubert and S. Engelen. Robust pca and classification in biosciences. *Bioinformatics*, 20(11):1728–1736, 2004.

[10] N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 6:1783–1816, 2005.

[11] S. Mosci, L. Rosasco, and A. Verri. Dimensionality reduction and generalization. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 657–664, New York, NY, USA, 2007. ACM.

[12] V. Nadimpally and M.J. Zaki. A novel approach to determine normal variation in gene expression data. *SIGKDD Explor. Newsl.*, 5(2):6–15, 2003.

[13] C. Ordonez. Horizontal aggregations for building tabular data sets. In *DMKD '04: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 35–42, New York, NY, USA, 2004. ACM.

[14] C. Ordonez. Vertical and horizontal percentage aggregations. In *SIGMOD Conference*, pages 866–871, 2004.

[15] C. Ordonez. Optimizing recursive queries in sql. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 834–839, New York, NY, USA, 2005. ACM.

[16] C. Ordonez. Building statistical models and scoring with udfs. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1005–1016, New York, NY, USA, 2007. ACM.

[17] C. Ordonez and J. García-García. Vector and matrix operations programmed with udfs in a relational dbms. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 503–512, New York, NY, USA, 2006. ACM.

[18] H. Polat and W. Du. Svd-based collaborative filtering with privacy. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 791–795, New York, NY, USA, 2005. ACM.

[19] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 2007.

[20] W. Rinsurongkawong and C. Ordonez. Microarray data analysis with pca in a dbms. In *DTMBIO '08: Proceeding of the 2nd international workshop on Data and text mining in bioinformatics*, pages 13–20, New York, NY, USA, 2008. ACM.

[21] S. Salleh, A.Y Zomaya, and A.B. Sakhinah. *Computing for numerical methods using visual C++*. Wiley, Hoboken, NJ, 2008.

[22] Polyxeni Zacharouli, Michalis Titsias, and Michalis Vazirgiannis. Web page rank prediction with pca and em clustering. In *WAW '09: Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph*, pages 104–115, Berlin, Heidelberg, 2009. Springer-Verlag.

[23] X.S. Zhuang and D.Q. Dai. Improved discriminate analysis for high-dimensional data and its application to face recognition. *Pattern Recogn.*, 40(5):1570–1578, 2007.

Data Mining using Matrices and Tensors (DMMT'09)

ISBN 978-1-60558-673-1