Clustered Subset Selection and its Applications on IT Service Metrics

Christos Boutsidis
Computer Science Dept, RPI
Troy, NY
boutsc@cs.rpi.edu

Jimeng Sun
IBM T.J. Watson Lab
Hawthorne, NY
jimeng@us.ibm.com

Nikos Anerousis
IBM T.J. Watson Lab
Hawthorne, NY
nikos@us.ibm.com

ABSTRACT

Motivated by the enormous amounts of data collected in a large IT service provider organization, this paper presents a method for quickly and automatically summarizing and extracting meaningful insights from the data. Termed Clustered Subset Selection (CSS), our method enables program-guided data explorations of high-dimensional data matrices. CSS combines clustering and subset selection into a coherent and intuitive method for data analysis. In addition to a general framework, we introduce a family of CSS algorithms with different clustering components such as k-means and Close-to-Rank-One (CRO) clustering, and Subset Selection components such as best rank-one approximation and Rank-Revealing QR (RRQR) decomposition.

From an empirical perspective, we illustrate that CSS is achieving significant improvements over existing Subset Selection methods in terms of approximation errors. Compared to existing Subset Selection techniques, CSS is also able to provide additional insight about clusters and cluster representatives. Finally, we present a case-study of program-guided data explorations using CSS on a large amount of IT service delivery data collection.

Categories and Subject Descriptors

G.1.3 [**Mathematics of Computing**]: Numerical Analysis—*Numerical Linear Algebra*; H.4.0 [**Information Systems**]: Information Systems Applications—*General*

General Terms

Algorithms, Management

Keywords

subset selection, clustering, low rank matrix approximation, service delivery, service provider

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

Today, the vast majority of IT-enabled business processes generate large amounts of data that is stored and analyzed to provide useful insights and decision support. A typical scenario is to represent these data sets as high-dimensional data points in data matrices. Linear algebra and matrix techniques become necessary to process and analyze these data collections. In this paper, we present a novel matrix algorithm that combines clustering and subset selection into a coherent method for low rank matrix approximations. We utilize this novel method to analyze matrices that arise from a real IT service provider application.

In the modern statistical data analysis and data mining literature, there have been several efforts to analyze data using linear algebra and matrix techniques. **Streams and co-evolving sequences** can be envisioned as matrices; each data source (sensor) corresponds to a row of the matrix, and each time-tick to a column. We can perform multivariate analysis or Singular Value Decomposition [38], "sketches" and random projections [30] to find patterns and outliers. **Information retrieval** systems, where the data can be turned into document-term matrices, routinely utilize SVD or Latent Semantic Analysis (LSI) [19, 37]. In **web** applications we have matrices where both rows and columns represent web-pages; we can apply the HITS [34] or the pageRank [6] algorithm to find hubs, authorities and influential nodes. Finally, while analyzing **social networks**, and in fact, *any* graph (with un-labeled edges), we have matrices where people are rows and columns, and edges correspond to the non-zero entries in the adjacency matrix of the graph. The network value of a customer [29] has close ties to the first eigenvector of this matrix. In a similar setting, graph partitioning [32] is often done through matrix algebra (e.g. spectral clustering [31]).

In the process of developing techniques to help IT program executives to analyze their data, we came up with two basic challenges. The first was the development of simple, user-friendly methods. The standard techniques described above require advanced mathematical skills such as Singular Value Decomposition (SVD), Eigen-Decomposition, or Principal Components Analysis (PCA), which prohibits many domain experts, such as IT program executives, to employ them for data explorations. The second was to find techniques that analyze the data using concrete concepts that are familiar to the users, not some abstract notions of latent variables or principal components. For example, since users' expertise are from the original data, the analysis should also be represented directly in terms of the original data.

1.1 Motivation

Consider an information technology (IT) services environment where the IT service provider is hosting and managing the IT infrastructure for a number of clients. The services include a diverse set of offerings, e.g., data center operations, application, server, storage and network management, and end user support. By outsourcing these IT services the client reduces its operational cost while maintaining a predictable level of service quality through contractual service level agreements (SLAs).

The IT service provider continuously monitors a large set of metrics spanning applications, servers, networks, and the effectiveness (quality) of its service management operation. One aspect of service management is the problem and change management processes. Service requests arrive in the form of tickets, which are then routed to specific service matter experts (SMEs) for resolution. The service quality provided to a client depends on how efficiently these tickets are handled. There are a number of metrics to track quality (e.g., the time to resolve a ticket) and service level agreements are typically drafted on top of these metrics. The challenge for the service provider is to manage its resources effectively in order to meet all the committed SLAs.

Typical metrics for problem and change management include workload metrics such as number of opened or closed problems or changes and performance metrics such as the Mean Time To Resolve (MTTR). Each metric is further associated with the context from which it was obtained. This would include the timestamp, the business process, the business unit, and the client, to name a few.

The total number of accounts for a large service provider can be in the order of thousands. For example, the data used in the experiments presented in this paper, come by monitoring a set of ten metrics over a set of more than 1000 accounts. There is an enormous wealth of insights and business optimization decisions that can be inferred from the data. However, given the scale and complexity, it is difficult for common users without the right tools to mine this hidden value and discover latent relations. Of particular interest is the discovery of relationships between the time series between IT metrics and the clustering according to similar levels of performance. The discovery of "similarity" relationships in the data leads to particularly useful insights with respect to how quality can be maintained and improved.

To address these issues from an engineering standpoint, this paper answers the following questions: given n service accounts over time (modeled as an m-by-n matrix), how to cluster the accounts into groups, how to identify the key accounts in each group, and how to represent the others using the key accounts?

1.2 Contributions

In this paper, we leverage techniques from the numerical linear algebra to develop a powerful framework for program-guided data explorations. More specifically, we present a novel Subset Selection technique, named Clustered Subset Selection (CSS). CSS provides an algorithmic framework with the following properties: (*i*) **rigorous algorithm**: it is a novel deterministic Subset Selection method on matrices with consistent improvement over existing techniques, (*ii*) **comprehensive solution**: it provides a comprehensive process for clustering data dimensions, selecting key dimensions, and synthesizing the rest dimensions, and (*iii*) **intuitive mining result**: the results of CSS are in the form of actual dimensions of the data matrix, instead of abstract notion of latent variables or principal components. The actual dimensions can provide intuitive explanation to the users. Finally, in addition to this algorithmic framework, a case study shows how the results of CSS can be used for data exploration on real IT service metrics.

Organization: The rest of the paper is organized as follows. In section 2, we review related work and basic facts from linear algebra. Section 3 presents existing (deterministic and randomized) Subset Selection algorithms. In section 4, we present a novel Subset Selection method while in section 5 we experimentally evaluate the the new method. Finally, in section 6 we discuss how CSS can be leveraged to quickly explore and extract insights of real IT service data.

2. BACKGROUND AND RELATED WORK

We start by briefly presenting some basic definitions and facts from linear algebra; more details can be found in [24]. We then review two important stepping stones of our Clustered Subset Selection (CSS) algorithm, i.e. two low rank matrix approximation schemes that proceed with a clustering preprocessing step of the input matrix. More related work is presented in section 3, where we discuss existing Subset Selection algorithms for matrices. Our CSS framework combines ideas of the above domains to offer a novel way of quickly exploring, summarizing, and understanding large and/or complex datasets.

2.1 Notation and linear algebra basics

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For any matrix $A \in \mathbb{R}^{m\times n}$, let $A_{(i)}, i \in [m]$ denote the i-th row of A as a row vector, and let $A^{(j)}, j \in [n]$ denote the j-th column of A as a column vector. In addition, let $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$ denote the square of its Frobenius norm, and let $\|A\|_2 = \sup_{x\in\mathbb{R}^n,\ x\neq 0} |Ax|_2 / |x|_2$ denote its spectral norm. Π will denote an $n \times n$ permutation matrix. Also for an $m \times n$ matrix A, $A_{i,j}$ arises if we interchange the i-th and j-th columns of A. We occasionally use Matlab notation, like $A(:, 1 : k)$, to denote the submatrix of A with its k first columns.

Singular Value Decomposition (SVD). Let A be an $m\times n$ matrix. The Singular Value Decomposition (SVD) of A can be written as

$$A = U_A \Sigma_A V_A^T = \begin{pmatrix} U_k & U_{\rho-k} \end{pmatrix} \begin{pmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \Sigma_{\rho-k} \end{pmatrix} \begin{pmatrix} V_k^T \\ V_{\rho-k}^T \end{pmatrix}.$$

In this expression, $\rho \leq \min\{m, n\}$ denotes the rank of A, $U_A \in \mathbb{R}^{m\times\rho}$ is an orthonormal matrix, Σ_A is a $\rho\times\rho$ diagonal matrix, and $V_A \in \mathbb{R}^{n\times\rho}$ is an orthonormal matrix. Also, Σ_k denotes the $k \times k$ diagonal matrix containing the top k singular values of A, $\Sigma_{\rho-k}$ denotes the $(\rho - k) \times (\rho - k)$ matrix containing the bottom $\rho - k$ singular values of A, V_k denotes the $n \times k$ matrix whose columns are the top k right singular vectors of A, and $V_{\rho-k}$ denotes the $n \times (\rho - k)$ matrix whose columns are the bottom $\rho - k$ right singular vectors of A, etc. The $m \times k$ orthogonal matrix U_k consisting of the top k left singular vectors of A is the "best" k-dimensional basis for the column space of A since $A_k = U_k\Sigma_k V_k^T = U_k U_k^T A$ is the "best" rank-k approximation to A. In particular, A_k minimizes $\|A - A'\|_\xi$, for both $\xi = 2$ and F, over all $m \times n$ matrices A' whose rank is at most

k. We will use the notation $\|\cdot\|_\xi$ when writing an expression that holds for both the spectral and the Frobenius norm. We will subscript the norm by 2 and F when writing expressions that hold for one norm or the other.

QR decomposition. Another matrix decomposition that is useful in our discussion is the QR decomposition. Let A be an $m \times n$ matrix. The QR decomposition of A can be written as

$$A = QR,$$

where $Q \in R^{m \times n}$ is an orthonormal matrix and R is an $n \times n$ upper triangular matrix. The QR decomposition of A can be computed for example by applying the Gram-Schmidt orthogonalization process on the columns of A.

Pseudoinverse. The Moore-Penrose generalized inverse, or pseudoinverse, of A, denoted by A^+, may be expressed for example in terms of the SVD as $A^+ = V_A \Sigma_A^{-1} U_A^T$. For matrices $A \in R^{m \times n}$ and $C \in R^{m \times k}$:

$$C^+ A = \arg\min_{X \in R^{k \times n}} \|CX - A\|_\xi,$$

i.e. the matrix $X = C^+ A$ is the solution to the generalized regression problem (least squares problem with multiple right hand sides) with inputs C and A.

2.2 Low rank matrix approximations through clustering

Given an $m \times n$ matrix A, an important problem in linear algebra and scientific computing is to find a "good" low rank matrix approximation to A, that can be expressed as the product of an $m \times k$ matrix D and an $k \times n$ matrix G, as $A \approx DG$, for some $k \ll \{m, n\}$. As we discussed above, SVD provides the optimal solution to the appropriate optimization problem. Other suboptimal solutions to this problem are also of great theoretical and practical interest. In this section we present two suboptimal techniques that find the matrix D of the above factorization by first clustering the columns of A into k clusters and then computing a vector (column of D) from each cluster. Our method, that is presented in section 4, was motivated by these two algorithms in the sense that it also proceeds in a similar way for computing the matrix D.

Given a matrix $A \in R^{m \times n}$ and an integer k, Modha and Dhillon ([15]) present a low rank matrix approximation scheme where one (i) clusters the input matrix A into k clusters using the Spherical k-means algorithm, (ii) selects the centroid vector (concept vector) of each cluster to be a column of an $m \times k$ matrix D, and (iii) approximates A as the rank-k matrix $A \approx DD^+A$. Empirical results establish the efficiency of the above technique (also called Concept Decomposition), in the sense that the Frobenius norm of the residual error $A - DD^+A$ is close to the Frobenius norm of the residual error from the best rank-k approximation of A.

Along the same lines, Zeimpekis and Gallopoulos ([23]) present a low rank matrix approximation scheme where one (i) clusters the input matrix A into k clusters using the Spherical k-means algorithm, (ii) selects the leading left singular vector of each cluster to be a column of an $m \times k$ matrix D, and (iii) approximates A as the rank-k matrix $A \approx DD^+A$. Empirical results establish the efficiency of this technique (also called CLSI, an abbreviation of Clustered Latent Semantic Indexing) in a similar sense to the one described above. The authors also propose a variant of the above technique that first forms ℓ clusters of the columns of A, for some $\ell < k$, and then selects the leading k_i left singular vectors from each cluster for $1 : \ell$, for some positive integers k_1, k_2, ..., k_ℓ such that $\sum_{i=1}^{\ell} k_i = k$. Finally, note that Zeimpekis and Gallopoulos discuss similar ideas in [42].

The clustering aspect of both works is similar to the algorithm presented in this paper. However, our focus is on selecting actual columns from each cluster as representatives, not abstract notions of centroids or singular vectors.

3. SUBSET SELECTION

In this section we discuss Subset Selection algorithms for matrices. We do emphasize here that the term "Subset Selection", has appeared in many different settings in the literature (see example [39, 11, 12]). Our study focus on the family of algorithms from the numerical linear algebra literature, introduced in the seminal work of Gene Golub in [25]. To the best of our knowledge, these techniques besides their "good" empirical behavior, are equipped with a sound theoretical framework for their performance. Other subset/feature selection techniques that are commonly used by practitioners for similar purposes still remain pure heuristical choices from a theoretical perspective.

Given an $m \times n$ matrix A, the Subset Selection algorithms discussed in this paper select a small subset of k ($k \ll n$) columns of A, to minimize the distance $||A - CC^+A||_\xi$. Here C is an $m \times k$ matrix that contains the selected columns, and $C^+A = \arg\{\min_{X \in R^{k \times n}} ||A - CX||_\xi\}$. Intuitively, A is summarized to its column submatrix C, which contains only a small subset of the columns of A; A can then be reconstructed as $A \approx CC^+A$. Formally, a Subset Selection algorithm attempts to (approximately) solve the following combinatorial optimization problem:

PROBLEM 1. SUBSET SELECTION PROBLEM: *Given a matrix $A \in \mathbb{R}^{m \times n}$ and a positive integer k, pick k columns of A forming a matrix $C \in \mathbb{R}^{m \times k}$ such that the residual*

$$\left\|A - CC^+A\right\|_\xi$$

is minimized over all possible $\binom{n}{k}$ choices for the matrix C. Here $\xi = 2$ or F denotes the spectral norm or Frobenius norm.

Intuitively, a Subset Selection algorithm identifies the "most" linearly independent columns of the matrix A. Although other qualitative measures might be suitable for evaluating the performance of a Subset Selection Algorithm (e.g see section 4 in [8] and the discussion in [26]) we focus on the spectral norm or Frobenius norm of the residual error $A - CC^+A$.

Algorithms for constructing such a matrix C go back to the seminal work of Gene Golub in [25] on pivoted QR factorizations. Since then, much work has be done on Subset Selection algorithms. Current research spans two different fields of computer science and mathematics. In the numerical linear algebra literature, one can find deterministic Subset Selection algorithms, while in the theoretical computer science literature the technique of randomization has lead to efficient randomized Subset Selection techniques.

3.1 Deterministic subset selection

Within the numerical linear algebra community, Subset Selection algorithms leverage the so-called Rank Revealing QR (RRQR) factorization.

DEFINITION 3.1. ***(The RRQR factorization)*** *Given a matrix $A \in R^{m\times n}$ ($m \geq n$) and an integer k ($k \leq n$), assume partial QR factorizations of the form:*

$$A\Pi = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

where $Q \in R^{m\times n}$ is an orthonormal matrix, $R \in R^{n\times n}$ is upper triangular, $R_{11} \in R^{k\times k}$, $R_{12} \in R^{k\times(n-k)}$, $R_{22} \in R^{(n-k)\times(n-k)}$, and $\Pi \in R^{n\times n}$ is a permutation matrix. The above factorization is called a RRQR factorization if it satisfies

$$\begin{aligned} \frac{\sigma_k(A)}{p_1(k,n)} \leq &\quad \sigma_{min}(R_{11}) \quad \leq \sigma_k(A) \\ \sigma_{k+1}(A) \leq &\quad \sigma_{max}(R_{22}) \quad \leq p_2(k,n)\sigma_{k+1}(A), \end{aligned}$$

where $p_1(k,n)$ and $p_2(k,n)$ are functions bounded by low degree polynomials in k and n.

Subset Selection Algorithms as described above are linked to another matrix factorization, where the columns of the input matrix A are (approximately) expressed as linear combinations of actual columns of A.

DEFINITION 3.2. ***(The CX factorization)*** *Given a matrix $A \in R^{m\times n}$ and an integer k ($k \leq n$), assume an (approximate) matrix factorization of the form*

$$A \approx CX,$$

where C and X are $m \times k$ and $k \times n$ matrices, respectively. The above is a column-based or a CX factorization of the matrix A if the matrix C consists of exactly k columns of A, and X is any $k \times n$ matrix.

We now draw the connection between a $RRQR$ and a CX factorization of a matrix. We do emphasize here that the result of Lemma 1 is not a new theoretical contribution. Although most of the papers discussing $RRQR$ techniques did not explicitly state the aforementioned connection, results presented in this Lemma are essentially implied in most manuscripts. Since, to the best of our knowledge, a technical proof of this connection has not appeared in any paper discussing $RRQR$ methods, we include a proof of Lemma 1 for completeness of our discussion and for helping readers better understand how $RRQR$ algorithms lie in the core of the Subset Selection methods discussed in this paper.

LEMMA 1. *Given a matrix $A \in R^{m\times n}$ ($m \geq n$) and an integer k ($k \leq n$), **for any** $RRQR$ factorization of A,*

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \tag{1}$$

*with $Q \in R^{m\times n}$, $\Pi \in R^{n\times n}$, $R_{11} \in R^{k\times k}$, $R_{12} \in R^{k\times(n-k)}$, $R_{22} \in R^{(n-k)\times(n-k)}$, **there exists** a CX factorization of the form*

$$A \approx CX, \tag{2}$$

with $C = A\Pi I_{n\times k} \in R^{m\times k}$ ($I_{n\times k}$ is the $n \times k$ identity matrix), and $X = (A\Pi I_{n\times k})^+ A \in R^{k\times n}$, such that

$$||A - CX||_2 = ||R_{22}||_2 = \sigma_{max}(R_{22}), \tag{3}$$

and

$$||A - CX||_F = ||R_{22}||_F. \tag{4}$$

Here C consists of the k leading columns of $A\Pi$.

Proof: Let $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ for $Q_1 \in R^{m\times k}$ and $Q_2 \in R^{m\times(n-k)}$, and $R = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}$ for $R_1 \in R^{k\times n}$ and $R_2 \in R^{(n-k)\times n}$. We will first prove that

$$||R_{22}||_\xi = ||A\Pi - Q_1R_1||_\xi.$$

The QR factorization of $A\Pi$ can be expressed as

$$\begin{aligned} A\Pi &= \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} \\ &= Q_1R_1 + Q_2R_2 \\ &= Q_1R_1 + Q_2 \begin{pmatrix} Z & R_{22} \end{pmatrix}, \end{aligned} \tag{5}$$

with $Z \in R^{(n-k)\times k}$ an all-zeros entries matrix. Then

$$\begin{aligned} ||A\Pi - Q_1R_1||_\xi &= ||Q_2R_2||_\xi = ||Q_2 \begin{pmatrix} Z & R_{22} \end{pmatrix}||_\xi \\ &= ||\begin{pmatrix} Z & R_{22} \end{pmatrix}||_\xi \\ &= ||R_{22}||_\xi, \end{aligned} \tag{6}$$

since Q_2 is an orthonormal matrix, and $||UX||_\xi = ||X||_\xi$ for any orthonormal matrix U and any unitarian invariant norm ξ, such as the Frobenius and 2 norm. Also note that standard perturbation theory for matrices in [41] implies that padding a matrix A with all-zeros columns does not affect the Frobenius norm or the 2 norm of A.

The proof concludes by proving that $||A - CC^+A||_\xi = ||A\Pi - Q_1R_1||_\xi$.

$$\begin{aligned} ||A\Pi - Q_1R_1||_\xi &= ||A\Pi - Q_1Q_1^TA\Pi||_\xi \\ &= ||(A - Q_1Q_1^TA)\Pi||_\xi \\ &= ||A - Q_1Q_1^TA||_\xi \\ &= ||A - Q_1R_{11}(R_{11})^{-1}Q_1^TA||_\xi \\ &= ||A - Q_1R_{11}(Q_1R_{11})^+A||_\xi \\ &= ||A - CC^+A||_\xi \end{aligned}$$

In the above, we used that $R_1 = Q_1^TA\Pi$, $C = Q_1R_{11}$, and that Π is an orthonormal matrix. Also note that R_{11} is a square matrix with linear independent columns (thanks to the $RRQR$ of A); thus is invertible.

◇

Clearly, any algorithm that constructs a $RRQR$ factorization of the matrix A with provable guarantees for the leading singular value of the factor R_{22}, also provides a CX factorization of A with provable performance for the spectral norm of the residual error $A - CC^+A$:

$$\|A - CX\|_2 \leq p_2(k,n)\sigma_{k+1}(A) = p_2(k,n)\,\|A - A_k\|_2\,.$$

C and X can be selected as indicated in Lemma 1.

Table 1 summarizes existing deterministic Subset Selection methods ($RRQR$ factorization algorithms). Algorithms are listed in a chronological order. You can evaluate these algorithms in terms of their approximation error theoretical behavior (column 2), and their asymptotical running time bound (column 3).

Method	$p_2(k,n)$	Time
Pivoted QR [25]	$\sqrt{(n-k)2^k}$	$O(mnk)$
High RRQR [20]	$\sqrt{n(n-k)2^{n-k}}$	$O(mn^2)$
High RRQR [7]	$\sqrt{n(n-k)2^{n-k}}$	$O(mn^2)$
RRQR [28]	$\sqrt{k(n-k)+k}$	$O(n^k)$
Low RRQR [9]	$\sqrt{(k+1)n2^{k+1}}$	$O(mn^2)$
Hybrid-I RRQR [10]	$\sqrt{(k+1)(n-k)}$	$O(n^k)$
Hybrid-II RRQR [10]	$\sqrt{(k+1)(n-k)}$	$O(n^k)$
Hybrid-III RRQR [10]	$\sqrt{(k+1)(n-k)}$	$O(n^k)$
Strong RRQR [27]	$\sqrt{k(n-k)+1}$	$O(n^k)$
Strong RRQR [27]	$O(\sqrt{k(n-k)+1})$	$O(mn^2)$
DGEQPY [4]	$O(\sqrt{(k+1)^2(n-k)})$	-
DGEQPX [4]	$O(\sqrt{(k+1)(n-k)})$	$O(n^k)$
SPQR [40]	-	-
PT Algorithm 1 [36]	$O(\sqrt{(k+1)(n-k)})$	-
PT Algorithm 2 [36]	$O(\sqrt{(k+1)^2(n-k)})$	-
PT Algorithm 3 [36]	$O(\sqrt{(k+1)^2(n-k)})$	-
Pan Algorithm 2 [35]	$O(\sqrt{k(n-k)+1})$	$O(mn^2)$

Table 1: Deterministic Subset Selection Algorithms. A dash means that the algorithm either runs in $O(n^k)$ time, or the authors do not provide a running time bound.

3.2 Randomized subset selection

Within the theoretical computer science community, Subset Selection Algorithms introduced from Frieze, Kannan, and Vempala in [22] and popularized by Despande et al. in [13, 14], and Drineas et al. in [16, 17, 18, 5]. The algorithms from this community are randomized, which means that they come with a failure probability, and focus mainly on bounding the Frobenius norm of $A - CC^+A$. The best such algorithm ([5]) needs $O(mn^2)$ time to find k columns of A such that with high probability

$$\left\|A - CC^+A\right\|_F \leq O(k\sqrt{\log k}) \left\|A - A_k\right\|_F .$$

Randomized vs Deterministic techniques. For data analysis purposes, deterministic algorithms are often preferred by practitioners. Deterministic techniques can guarantee that the same result will be obtained on the same data, a fundamental property on real data mining applications. On the other hand, randomized algorithms typically have better theoretical properties and are simpler to be implemented and used. This paper focus on deterministic Subset Selection techniques since our proposed framework is motivated by a real data application.

4. CLUSTERED SUBSET SELECTION (CSS)

We now present the Clustered Subset Selection (CSS) problem and an approximation algorithm (CSS Algorithm) for this problem, which are our main contributions of this paper. Clustered Subset Selection is a relaxation of the Subset Selection problem discussed above. We basically propose to find an approximate solution to the Subset Selection problem by finding an approximate solution of this relaxation.

Problem 2 (Clustered Subset Selection). *Given a matrix $A \in \mathbb{R}^{m\times n}$ and a positive integer k, (i) find a number $\ell \leq k$ to be the number of clusters of A, (ii) partition the columns of A into ℓ clusters, denoted by A_i, and (iii) select k_i representative columns $C_i \in \mathbb{R}^{m\times k_i}$ from each cluster A_i, for $1 \leq i \leq \ell$, such that 1) $\sum_{i=1}^{\ell} k_i = k$ and 2) the error*

$$\sum_{i=1}^{\ell} \left\|A_i - C_i C_i^+ A_i\right\|_F^2$$

is minimized.

In the sequel we present an algorithm, named Clustered Subset Selection (CSS), to find an approximate solution to the above problem. Our Algorithm has the following properties: (i) empirically it consistently outperforms existing deterministic subset selection methods on the datasets used in this paper, and (ii) it provides a novel powerful framework for quick data exploration through clusters and representatives.

4.1 CSS general algorithm

Due to the complexity of the problem (see discussion below), the optimal solution might be infeasible to obtain for practical applications. Therefore, we propose a top-down greedy approach to independently (approximately) solve the clustering and subset selection sub-problems of Problem 2. Our approach is presented in Algorithm 1.

Algorithm 1 CSS general

1: **in:** data matrix $A \in \mathbb{R}^{m\times n}$
2: **in:** number of clusters and representatives ℓ, k
3: cluster the columns of A into ℓ clusters.
4: determine the number of representatives k_i such that $\sum_{i=1}^{\ell} k_i = k$
5: **for** $n = 1, \ldots, \ell$ **do** {find representatives}
6: select k_i columns C_i from the cluster A_i
7: **end for**
8: **return** cluster assignments and representatives $[C_i]_{i=1}^{\ell}$

Design choices. Depending on the clustering and subset selection algorithms, many different instances of algorithms can be designed. Moreover, the relative size of ℓ and k also affects the designs. When $\ell = k$, CSS will select one representative column per cluster. More generally, if $\ell \leq k$, first it has to determine the number of representatives from each cluster and then to select a number of representatives independently from each cluster. A simple heuristic for determining the number of representatives from each cluster is to make them proportional to the size of the cluster. Table 2 illustrates the design choices of the CSS meta-algorithm.

Component	Choices
clustering	k-means, hierarchical CRO [33]
$\ell = k$	Best Column Selection
$\ell < k$	RRQR Subset Selection [25]

Table 2: CSS design choices

Having discussed the general framework, we now present specific design options for clustering and subset selection.

4.2 Clustering components

Virtually all clustering algorithms can be applied for the clustering step in Algorithm 1. Here we present two options that are used in our experiments: 1) deterministic k-means and 2) Close-to-Rank-One clustering (CRO) [33].

Kmeans clustering. : k-means is a fairly standard method for clustering. The only customization that is needed in order to have a deterministic Clustered Subset Selection method is to start with a deterministic initialization of the cluster centers. For example, we can pick the first ℓ columns of A as the initial cluster centers. The assumption/objective of k-means is to put points close to each other into the same cluster. For completeness to our discussion we present the standard Loyd's algorithm as Algorithm 2 in this paper.

CRO clustering. : Another option is to partition the matrix A into clusters of close to rank-one matrices; then we can easily select one or a few columns to approximate the rest well. To the best of our knowledge, the only algorithm available in the literature for satisfying the above objective is the technique described in [33]. Algorithm 3 gives a high level description of this technique. We refer to the original paper for more details. Note that the CRO value of a matrix X is defined as $\text{CRO}(A) = \frac{\sigma^2_{max}(X)}{||X||^2_F}$. CRO (Closeness to rank one) value of a matrix X essentially measures how close to a rank-1 matrix X is. The larger the CRO the closest to a rank-1 matrix X is.

Algorithm 2 k-means Clustering

in: data matrix $A \in \mathbb{R}^{m\times n}$
in: number of clusters $\ell \leq n$
Initialize vectors c_1, c_2, ..., c_ℓ, with the first ℓ columns of A.
repeat
  For $i = 1 : \ell$, set the cluster A_i to be the columns of A that are closer to c_i than they are to c_j for all $j \neq i$.
  For $i = 1 : \ell$, $c_i = \frac{1}{|A_i|}\sum_{x\in A_i} x$.
until no changes occur to the clusters A_i
Return final clusters A_1, A_2, ..., A_ℓ.

Algorithm 3 Close-to-rank-one Clustering

in: data matrix $A \in \mathbb{R}^{m\times n}$
in: number of clusters $\ell \leq n$
Calculate CRO for every pair of clusters. Here a pair of clusters is a $m \times 2$ matrix.
Start with every column of A as its own cluster.
repeat
  Merge the pair of clusters with the largest CRO.
  Compute CRO between the new cluster and the remainings.
until ℓ clusters remain
Return final clusters.

4.3 Subset selection components

Many subset selection algorithms have been proposed in the literature (see Section 2 for a survey). In this paper, we focus on deterministic algorithms which are often preferred in the exploratory data analysis. More specifically, we present two fast deterministic algorithms for $\ell = k$ (one representative per cluster) and $\ell < k$ (more than one representative per cluster), respectively.

Algorithm 4 Best column selection

in: Data matrix $A \in \mathbb{R}^{m\times n}$
for $i = 1 : n$ **do** {all columns}
  Let $x_i = A^i$
  Let $e_i = ||A - \frac{x_i x_i^T}{x_i^T x_i}A||_2$
end for
Return x_i such that $i = \arg\min_{1\leq i\leq n} e_i$

Algorithm 5 RRQR Subset Selection

1: **in:** Data matrix $A \in \mathbb{R}^{m\times n}$,
2: **in:** number of representatives k
3: Initialize $R = A$, $\Pi = I_n$
4: **for** $n = 1, \ldots, k$ **do** {greedy selection}
5: $\begin{pmatrix} Q & R \end{pmatrix} = qr(R)$
6: Let R_{22} denote the $(n-i+1)\times(n-i+1)$ trailing submatrix of R.
7: $j = \arg\max_{1\leq j\leq(n-k)} \|R^j_{22}\|_2$
8: $R = R\Pi_{i,i+j-1}$
9: $\Pi = \Pi\Pi_{i,i+j-1}$
10: **end for**
11: **return** $C = A\Pi(:, 1:k)$

One representative per cluster. : When $\ell = k$, we need to select exactly one column from each cluster that best represents all the columns. This can be formalized as the following combinatorial problem: Given a cluster A_i, select a column $x_{opt} \in A_i$ such that:

$$x_{opt} = \arg\min_{x\in A_i} \|A_i - x(x)^+ A_i\|.$$

Note here that from basic linear algebra the pseudo-inverse of a vector x is $x^+ = \frac{x^T}{x^T x}$. In terms of algorithm, one could select the best column x_{opt} from the cluster A_i by enumerating through all columns in A_i (see running time analysis below for a justification of the efficiency of this approach). The pseudocode is presented in Algorithm 4.

More than one representative per cluster. : When $\ell < k$, we potentially need to select more than one column from each cluster. In particular, we need to answer two questions. How many representatives does each cluster have? How to select those representatives from each cluster? To answer the first question, we select the number of representatives based on the size of the cluster. More specifically, the size can be proportional to the number of columns of A_i. To answer the second question, we adopt a fast deterministic Subset Selection algorithm from Table 1. The method of [25] is presented as the Algorithm 5 of this paper.

4.4 CSS: Complexity analysis

We start be briefly commenting on the complexity of the Clustered Subset Selection problem defined above. The Clustered Subset Selection problem is a hard combinatorial optimization problem. Since clustering and subset selection are hard by themselves (e.g. k-means is provably NP-hard), we do believe that the Clustered Subset Selection prob-

lem is NP-hard. Though, we didn't attempt to prove the NP-hardness of the Clustered Subset Selection problem and leave this issue as an open question.

We now discuss the running time of our approximation Clustered Subset Selection Algorithm. Let's assume that A is an $m \times n$ matrix and that it is clustered into ℓ clusters A_i, for $1 \leq i \leq \ell$. The cluster A_i has dimensions $m \times n_i$, such that $\sum_{i=1}^{\ell} n_i = n$. The first step of the algorithm takes time proportional to the cost of the Clustering Algorithm. We denote this cost as $O(clustering)$. This cost obviously depends on the clustering technique selected in this step of our Clustered Subset Selection meta-algorithm. In our experiments we basically run twenty iterations of the aforementioned described iterative k-means algorithm; thus this step takes polynomial time on the size of the input matrix. Note that since we don't really care to find the optimal clustering assignments, twenty iterations suffice for our purposes. For the second step, when $\ell = k$, for each cluster A_i we need $O(mn_i^2)$ time to find the best column x_i via exhaustive enumeration. This might be of the order $O(mn^2)$, because $n_i = O(n)$. This implies that the second step of the Clustered Subset Selection algorithm needs $O(mn^2)$ time, and the overall process costs $O(clustering + mn^2)$ time. When $\ell < k$, we asymptotically need $O(clustering + mnk)$, mainly because the pivoted QR technique (Algorithm 5 of this paper) needs $O(mn_ik_i)$ time to select k_i columns from A_i, and $k_i = O(k)$, $n_i = O(n)$. Overall, if we carefully select the clustering algorithm of the Clustered Subset Selection meta-algorithm, the running time of our algorithm is polynomial on the size of the input matrix. Thus our algorithm can handle large scale datasets in reasonable time.

5. EXPERIMENTAL EVALUATION OF CSS

In this section we experimentally evaluate our CSS method with the existing deterministic Subset Selection techniques described in section 3. Note that our CSS algorithm is typically slower than the classic deterministic algorithms due to the clustering overhead. Despite that overhead, CSS methods can finish in seconds to minutes. The complexity of our method was analyzed in the previous section. Therefore, we do not report the detailed running time experiments; our comparison focus on the errors incurred by the related low rank matrix approximations.

5.1 Description of the Datasets

We use two well-known synthetic matrices from the numerical analysis literature and two matrices that arise from real world data applications (Finance and Image Processing). See below for a detailed description of our datasets.

Name	Size	default (k, l)
KAHAN	384-by-384	(30,6)
GKS	384-by-384	(30,12)
SANDP	1153-by-500	(40,8)
DIGITS	320-by-390	(10,3)

Table 3: Dataset summary

1. KAHAN: This is an $n \times n$ matrix which is constructed as a product of two matrices, i.e $A = SK$. S is a diagonal $n \times n$ matrix with diagonal elements $1, \zeta, \zeta^2, \ldots, \zeta^{n-1}$. K is an $n \times n$ upper triangular matrix. The diagonal elements of K are equal with 1, while all the other non-zero elements of K, i.e the upper triangular part, are equal with $-\phi$. Also $\phi = 0.285$ and $\phi^2 + \zeta^2 = 1$. This matrix was also utilized for experiments in [27]. We follow [27] by choosing $n = 384$ for our numerical experiments.

2. GKS: This is an $n \times n$ upper triangular matrix whose j-th diagonal element is $1/\sqrt{j}$ and whose (i, j)-th element is $-1/\sqrt{j}$, for $j > i$. This matrix was also utilized for experiments in [27]. We follow [27] by choosing $n = 384$ for our experiments.

3. SANDP: Yahoo! provides historical stock prices for the S&P 500 Index [1]. We collected historical prices for the 500 stocks of the index from Jan 2, 2003 to August 1, 2007, a total of 1153 days. Thus we constructed an 1153×500 date-by-stock matrix. The (i, j)-th element of this matrix represents the value of the j-th stock at the i-th day. We discarded 19 stocks that had missing values for a large number of dates, and we were left with a 1153×481 matrix. This matrix was also utilized for experiments in [5]

4. DIGITS: This is a 320×390 pixel-by-*digit* matrix. In more details, having a set of 390, 20×16, binary images, we vectorize each image and stuck it as a column to our matrix. Each image corresponds to a decimal digit from the set $\{0, 1, \ldots, 9\}$. There are exactly 39 examples of each digit.

5.2 Methods and software

We now describe the Subset Selection methods that are used in our experiments. We also provide pointers to publicly available software implementing these methods. Again, our study focus on deterministic Subset Selection techniques.

1. **Pivoted QR:** We employed MATLAB's `qr` function. This function implements the algorithm described by Golub in [25]. This method was also presented as Algorithm 4 of this paper.

2. **SPQR:** The Semi Pivoted QR (SPQR) method described by Stewart in [40]. A MatLab implementation is available from [2].

3. **qrxp:** qrxp is the algorithm implemented as the LAPACK routine DGEQPX in ACM Algorithm 782 [4, 3]. We will use the MatLab implementation of the Fortran routine DGEQPX from [21].

4. **qryp:** qryp is the algorithm implemented as the LAPACK routine DGEQPY in ACM Algorithm 782 [4, 3]. We will use the MATLAB implementation of the Fortran routine DGEQPY from [21].

5. **CSS1:** CCS1 stands for an instance of our Clustered Subset Selection algorithm. In CCS1 the k-means algorithm is leveraged for the clustering phase. The number of clusters is chosen to be equal with the number of representative columns we seek to select ($\ell = k$).

6. **CSS2:** CCS2 stands for an instance of our Clustered Subset Selection algorithm. In CCS2 the clustering phase leverages the k-means algorithm while $\ell < k$.

7. **CSS3:** CCS3 stands for an instance of our Clustered Subset Selection algorithm. In CCS3 the clustering phase leverages the CLO clustering algorithm while $\ell = k$.

8. **CSS4:** CCS4 stands for an instance of our Clustered Subset Selection algorithm. In CCS4 the clustering phase leverages the CLO clustering algorithm while $\ell < k$.

5.3 Numerical results and discussion

Our experimental results are depicted in Table 4. Each column of this table corresponds to a Subset Selection algorithm and each row to a dataset. The column and row ids of this table coincide with the above description. The (i, j)-th entry of Table 4, represents the value $||A - CC^+A||_F/||A - A_k||_F$. We use the same k and ℓ for all methods and the value of k and ℓ are presented in Table 3.

The CSS family algorithms are very competitive compared to the other deterministic methods. In particular, CSS1 consistently outperforms the other methods (up to 4X) in all cases. The results confirm that besides providing more information (clusters plus representatives), CSS family algorithms opens a new direction for the classical Subset Selection problem.

Finally, we report a numerical experiment for a comparison of our method with the CLSI method of [23] and the Concept decomposition method of [15]. We ran all three methods, CSS1, CLSI, and Concept Decomposition on the SANDP dataset with $k = 12$. The results (in the form of relative error as in Table 4) are 1.31, 1.12, and 1,13 for CSS1, CLSI, and Concept Decomposition methods, respectively.

6. ANALYSIS OF SERVICE DATA

Having shown the superiority of CSS methods on general datasets, now we present more fine-grained analysis and case study on IT service data.

6.1 Application and data description

As we motivated in the introduction, IT service providers provide diverse service lines for various companies (accounts) across the globe. The combination of geography, account and service line forms a monitoring basis called WID.

It is of key importance for them to monitor and correlate all the service metrics and to identify potential problems quickly.

Example metrics are presented in the following categories:

- **Work characteristics**: Volume of service requests, service severity distribution, service time distribution;
- **Service Quality**: Service resolution rate, Customer satisfaction score, critical service level violations;
- **Productivity metrics**: Service/FTE, Requests/FTE, Repeated Services;
- **Financial metrics**: Cost/FTE, Account size, Cost per service line.

As a service provider, the total number of metrics across accounts and service lines can be gigatic. It is very hard for the program exectutives to monitor all the metrics for all accounts and service lines. The goal of applying CSS on service metrics is to summarize the service data into manageable clusters based on their behaviors and to highlight the representatives from each cluster. Finally, the representatives can be easily utilized to predict other metrics.

The data used in the paper is obtained from the web-based data mining tool for IT service delivery monitoring, called "Prism Insight". It monitors problem and change data of 11 quality metrics for over 1,000 WIDs. Among which, we focus on two service quality metrics and two work characteristics metrics:

- **Service quality**: Mean Time To Resolved (MTTR) and Percentage of Service Level Agreement (SLA) violations.
- **Work characteristics**: The number of Open changes and the number of closed changes.

These metrics are computed on a daily basis. For each of the four metrics, we construct a timestamp-by-WID matrix A. Since the measurements across WIDs are not directly comparable[1], we normalize each WID (a column in the matrix) by its norm, i.e. $A^i = \frac{A^i}{\|A^i\|}$. Next we remove the all zero columns and rows. As shown in Table 5, we obtain four matrices of size 240×749 (MTTR), 240×671 (SLA), 240×679 (Opened changes) and 240×669 (Closed changes).

Name	size
MTTR	240×749
SLA violations	240×671
Number of opened changes	240×679
Number of opened changes	240×669

Table 5: IT service data

Next we first present a quantitative comparison as the one presented in the previous section, and then discuss the mining results of the application of CSS on the service data.

6.2 Quantatitive evaluation

Recall that the two parameters in the CSS methods are the number of clusters ℓ and the number of representatives k. Since CSS1-2 always outperform CSS3-4, we only include the results for CSS1-2. First, we vary ℓ by fixing $k = 100$. Results are presented in Table 6. Notice that ℓ do not affect the error much when k is fixed. Second, we vary k and fix

ℓ	2	10	20	25	50
MTTR	1.27	1.28	1.30	1.25	1.27
SLA violations	1.30	1.29	1.30	1.29	1.30
Closed Changes	1.36	1.35	1.35	1.37	1.36
Opened Changes	1.30	1.29	1.31	1.32	1.30

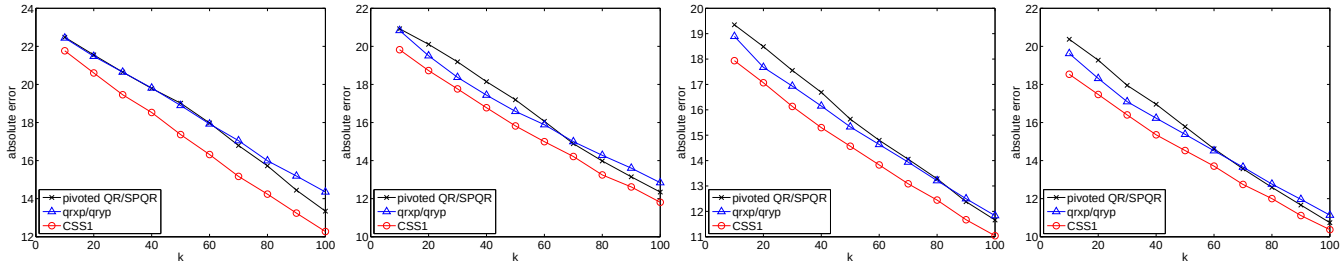
Table 6: Relative residual error vs. the number of clusters ℓ for $k = 100$

$\ell = k$ as shown in Figure 1. For this experiment, we present the absolute errors instead of relative errors. As expected, the error monotonically decreases smoothly as k increases for all methods. Note that our CSS1 method is significantly better than the others.

[1]The metric values are highly dependent on the particular service line and account size.

	Pivoted QR	qrxp	qryp	SPQR	CSS1	CSS2	CSS3	CSS4
KAHAN	1.89	4.87	4.87	1.89	**1.09**	3.37	3.63	2.58
GKS	1.63	2.69	2.69	1.63	**1.35**	1.48	3.09	1.94
SANDP	1.21	1.23	1.23	1.21	**1.15**	1.22	1.23	1.23
DIGITS	1.59	1.54	1.54	1.59	**1.42**	1.54	1.63	1.55

Table 4: Comparison on relative residual error to the best rank-k approximation, i.e., $||A-CC^+A||_F/||A-A_k||_F$. Given a dataset, the same k and ℓ are used for all methods. Note that CSS1 consistently outperforms the rest of methods.



(a) MTTR (b) SLA violation (c) Closed changes (d) Opened Changes

Figure 1: The absolute reconstruction error vs. the number of representatives k: All methods have linear drops as k increases. Our CSS1 has low reconstruction error compared to the other deterministic schemes.

6.3 Mining case study

The immediate application of the CSS method is to cluster WIDs and then to identify representatives within each cluster. We will show some example clusters and its representatives by using CSS1 on the MTTR matrix with $k = \ell = 100$.

First, Figure 2 shows the distribution of the cluster sizes. Note that, most of clusters are fairly small but with two exceptions. In particular, the largest cluster has 180 members. We manually investigate that cluster and find out that all the members have very few nonzero entries, which means they are the "healthy" WIDs[2].

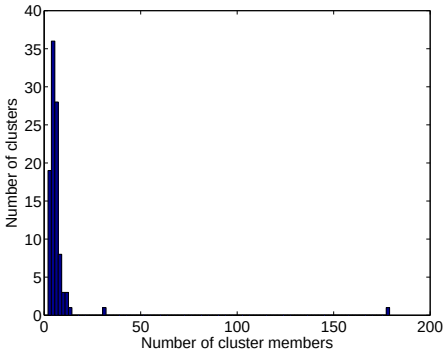


Figure 2: Histogram of cluster sizes:

Second, for the small clusters, we further investigate each one of them. Most of the WIDs either belong to the same service line or come from the same accounts. The representative WID from each cluster are the one that can best approximate the rest ones in the group. As shown in Table 7, some small clusters include service line such as Mainframe, DBDC and Storage and work streams (sub-service line) such as MVS, IMS and Dist Storage SAN. We are unable to show the account names due to the privacy complaint, but CSS does successfully group several relevant accounts into the same cluster.

[2]Healthy WIDs have very few problems or the problems got resolved quickly with small MTTR.

Representative WID	Cluster members
Mainframe - MVS	Mainframe -MVS DBM - Oracle DBM - MF DBA Mainframe - MVS
DBDC-IMS	DBDC-CICS DBDC-IMS Storage-Mainframe storage
Storage-Dist Storage SAN	Storage-Dist Storage B/R DBDC-CICS Storage-Dist Storage SAN

Table 7: Example clusters and their representatives: Each WID consists of service line and work stream. Accounts are not presented due to privacy constraints.

7. CONCLUSIONS

Modern applications generate complex data that are typically represented as high-dimensional points in data matrices. How to identify automatic summary and insight of such matrices? How to help users quickly gain intuitive understanding of complex data? To address the above questions, we present a powerful framework for program-guided data explorations named Clustered Subset Selection (CSS).

The Clustered Subset Selection approach provides a coherent method to quickly summarize a data matrix using the clusters and the representative columns. Various versions of the CSS general technique have been discussed in

this paper. The proposed methods are highly competitive against several other subset selection algorithms. In particular, one instance of our general CSS meta-algorithm, the CSS1 method, consistently outperforms others on all datasets. Finally, the Clustered Subset Selection approach demonstrates a great potential for data mining on summarizing IT service data. Future work includes developing time-evolving models for the CSS method as well as generalizing it to high-order data. Another interesting issue is to perform a theoretical analysis of our algorithm to derive bounds similar with those of Table 1.

8. REFERENCES

[1] http://finance.yahoo.com/q/hp?s=%5egspc.

[2] M. Berry, S. Pulatova, and G. Stewart. Computing sparse reduced-rank approximations to sparse matrices. Technical Report UMIACS TR-2004-32 CMSC TR-4589, University of Maryland, College Park, MD, 2004.

[3] C. H. Bischof and G. Qintana-Ortí. Algorithm 782: Codes for rank-revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24:254–257, 1998.

[4] C. H. Bischof and G. Quintana-Ortí. Computing rank-revealing QR factorizations of dense matrices. *ACM Trans. Math. Softw*, 24(2):226–253, 1998.

[5] C. Boutsidis, M. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual (web) search engine. In *WWW*, pages 107–117, 1998.

[7] T. F. Chan. Rank revealing QR factorizations. *Linear Algebra and Its Applications*, 88/89:67–82, 1987.

[8] T. F. Chan and P. C. Hansen. Some applications of the rank revealing QR factorization. *SIAM Journal on Scientific and Statistical Computing*, 13:727–741, 1992.

[9] T. F. Chan and P. C. Hansen. Low-rank revealing QR factorizations. *Numerical Linear Algebra with Applications*, 1:33–44, 1994.

[10] S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorizations. *SIAM J. Matrix Anal. Appl.*, 15:592–622, 1994.

[11] S. J. Cho and M. A. Hermsmeier. Genetic algorithm guided selection: Variable selection and subset selection. *Journal of Chemical Information and Computer Sciences*, 42(4):927–936, 2002.

[12] Couvreur and Bresler. On the optimality of the backward greedy algorithm for the subset selection problem. *SIJMAA: SIAM Journal on Matrix Analysis and Applications*, 21, 2000.

[13] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1117–1126, 2006.

[14] A. Deshpande and S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *RANDOM - APPROX*, 2006.

[15] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143, 2001.

[16] P. Drineas. *Randomized Algorithms for Matrix Operations*. PhD thesis, Yale University, 2002.

[17] P. Drineas and R. Kannan. Pass efficient algorithms for approximating large matrices. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 2003.

[18] P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal of Computing*, 36(1):158–183, 2006.

[19] P. W. Foltz and S. T. Dumais. Personalized information delivery: An analysis of information filtering methods. *Comm. of ACM (CACM)*, 35(12), 1992.

[20] L. V. Foster. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra Appl.*, 74:47–71, 1986.

[21] L. V. Foster and X. Liu. Comparison of rank revealing algorithms applied to matrices with well defined numerical ranks. *submitted*, 2006.

[22] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 370–378, 1998.

[23] E. Gallopoulos and D. Zeimpekis. CLSI: A flexible approximation scheme from clustered term-document matrices. In *SDM*, 2005.

[24] G. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989.

[25] G. H. Golub. Numerical methods for solving linear least squares problems. *Numer. Math.*, 7:206–216, 1965.

[26] G. H. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical Report TR-456, Computer Science, University of Maryland, College Park, MD, USA, 1976.

[27] M. Gu and S. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing*, 17:848–869, 1996.

[28] Y. P. Hong and C. T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Mathematics of Computation*, 58:213–232, 1992.

[29] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *KDD*, 2001.

[30] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *VLDB*, 2000.

[31] R. Kannan, S. Vempala, and A. Vetta. On clusterings – good, bad and spectral. In *FOCS*, 2000.

[32] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1), 1998.

[33] Y.-D. Kim and S. Choi. A method of initialization for nonnegative matrix factorization. In *Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[34] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *SODA*, 1998.

[35] C. T. Pan. On the existence and computation of rank-revealing LU factorizations. *Linear Algebra Appl.*, 316:199–222, 2000.

[36] C. T. Pan and P. T. P. Tang. Bounds on singular values revealed by QR factorizations. *BIT Numerical Mathematics*, 39:740–756, 1999.

[37] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *PODS*, 1998.

[38] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[39] Pruhs and Woeginger. Approximation schemes for a class of subset selection problems. *TCS: Theoretical Computer Science*, 382, 2007.

[40] G. Stewart. Four algorithms for the efficient computation of truncated QR approximations to a sparse matrix. *Numerische Mathematik*, 83:313–323, 1999.

[41] G. Stewart and J. Sun. *Matrix Perturbation Theory*. Academic Press, New York, 1990.

[42] D. Zeimpekis and E. Gallopoulos. Linear and non-linear dimensional reduction via class representatives for text classification. In *ICDM*, pages 1172–1177. IEEE Computer Society, 2006.