Updating the Centroid Decomposition with Applications in LSI

Jason R. Blevins and Moody T. Chu
Department of Mathematics, N.C. State University

May 14, 2004

Abstract

The centroid decomposition (CD) is an approximate singular value decomposition (SVD) with applications in factor analysis and latent semantic indexing (LSI). This paper presents updating methods for the centroid decomposition based on recent work in SVD updating methods. A general rank-1 updating framework is developed first and then more specific updates used in LSI are examined in this framework.

1 Introduction

The centroid decomposition (CD) [9, 13, 18, 6] is an approximation to the singular value decomposition (SVD) that is widely used in the applied statistics/psychometrics (AS/P) community for factor analysis research [12, 14]. A recent paper by Chu and Funderlic [9] unifies notions of the CD from the AS/P and numerical linear algebra (NLA) communities in the factor analysis setting. They also develop a generalized centroid decomposition with applications in latent semantic indexing (LSI) [10, 1, 2, 16] and present it as a member of a larger class of truncated SVD approximations. Noting the fundamental similarities between the centroid and singular value decompositions, this paper develops a framework for a class of CD updating methods by building on methods from SVD updating literature [20, 19, 8, 11, 7, 5] and specifically based on Brand's recent work on fast incremental SVD updating [3, 4].

Updating methods are particularly important for LSI applications, where the truncated SVD, and the truncated centroid decomposition, can be used to find documents relevant to given term queries. In real-time, on-line applications, where the term-document (indexing) matrix is frequently modified, it becomes costly to keep the corresponding centroid decomposition up to date.

This paper is organized as follows. Section 2 provides a brief review of the ideas and methods that provide a basis for our work: the centroid decomposition, LSI, and SVD updating. Section 3 describes the notation and basic framework used for the general rank-1 updating methods developed in section 4. Finally, section 5 describes the specific types of matrix updates used in LSI.

2 Background

2.1 SVD Updating

The singular value decomposition is one of the most powerful tools of linear algebra and provides many forms of insight about the data contained in a matrix. A SVD of an $m \times l$ matrix H is any factorization of the form $H = U\Sigma V^T$ where U is an $m \times m$ orthogonal matrix, Σ is an $m \times l$ diagonal matrix containing the singular values of H, $\sigma_1, \ldots, \sigma_l$, and V is an $l \times l$ orthogonal matrix. The columns of U and V are called the left and right singular vectors. We assume that the singular values are ordered so that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_l \geq 0$. In this case, the closest rank-k approximation to H is, conveniently, the rank-k truncated SVD of H which can be found by setting all but the first k singular values equal to zero, so that $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$. This is equivalent to taking only the first k columns of U, the $k \times k$ principal submatrix of Σ, and the first k columns of V. The latter method is useful in computational applications where storage is important.

Computing the SVD of a large matrix is computationally intensive and if the SVD must be kept up to date when the matrix changes (as in LSI), recalculating it each time can become overly burdensome in on-line applications. SVD updating methods concern the following problem: Given a matrix and its SVD, if the matrix is modified, how can we determine the new SVD without explicitly recalculating it? The first methods appeared in the 1970's and such methods generally fall into two categories: iterative updates based on Lanczos or Ritz-Raleigh methods and subspace methods based on the relationship between the full SVD and SVDs in a subspace.

2.2 The Centroid Decomposition

Among the AS/P community, Thurstone is generally credited with originating the centroid method [18] although it was used as early as 1917 by Burt [6] and Horst, a student of Thurstone, also provides an insightful treatment [13]. There are several variations of the method but this paper follows the generalized centroid method and the corresponding LSI model introduced by Chu and Funderlic [9] (Algorithm 8.1 and Section 7 respectively). They have revisited the centroid method in order to provide a more mathematical treatment of its stochastic properties, explored its similarities with the SVD, and shown that it can be generalized and applied to other applications such as LSI.

The centroid decomposition of an $m \times l$ matrix A is a factorization of the form $A = BV^T$ where $B \in \mathbb{R}^{mtimesl}$ is called the loading matrix and $V \in \mathbb{R}^{l \times l}$ is called the factor matrix. It can be formed by iteratively finding the centroid factors and their corresponding factor loadings. Each step proceeds by finding the most significant centroid factor and using it to perform a rank reduction on the original matrix. The centroid factor at each step is found by choosing the sign vector, a vector containing only the values 1 and -1, which maximizes the "weight" of the data in its direction.

The algorithm proceeds as follows: Define $A_1 \equiv A$ and $A_i = A_{i-1} - \mathbf{b}_i \mathbf{v}_i^T$ for $i = 1 \ldots l$, where

$$\mathbf{z}_i = \arg \max_{|\mathbf{z}|=\mathbf{e}} \|A_i^T \mathbf{z}\|, \tag{2.1}$$

$$\mathbf{w}_i = A_i^T \mathbf{z}_i, \tag{2.2}$$

$$\mathbf{v}_i = \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|}, \tag{2.3}$$

CD: $A = BV^T$	SVD: $A = U\Sigma\hat{V}^T$
$\mathbf{z}_1 = \arg\max_{\|\mathbf{z}\|=\mathbf{e}} \mathbf{z}^T AA^T \mathbf{z}$ First sign vector	$\mathbf{u}_1 = \arg\max_{\|\mathbf{x}\|=1} \mathbf{x}^T AA^T \mathbf{x}$ First Left Singular Vector
$\mu_1 = \frac{1}{n} \max_{\|z\|=\mathbf{e}} \mathbf{z}^T AA^T \mathbf{z}$ First Centroid Value	$\lambda_1 = \max_{\|\mathbf{x}\|=1} \mathbf{x}^T AA^T \mathbf{x}$ First Eigenvalue
$\mathbf{v}_1 = \frac{A^T \mathbf{z}_1}{\sqrt{n\mu_1}}$ First Centroid Factor	$\hat{\mathbf{v}}_1 = \frac{A^T \mathbf{u}_1}{\sqrt{\lambda_1}}$ First Right Singular Vector

Table 1: Similarities between the centroid and singular value decompositions.

and

$$\mathbf{b}_i = A_i^T \mathbf{v}_i. \tag{2.4}$$

Then the vector $\mathbf{z}_i$ is the sign vector corresponding with the centroid factor $\mathbf{v}_i$ and loading vector $\mathbf{b}_i$. Choosing the sign vector is essentially an $O(n)$ hypercube traversal problem.

The decomposition is only unique up to the orientation of the sign vectors. Any sign vector and its inverse (negative) will extract essentially the same centroid factor with opposite signs. Thus, depending on the starting sign vector and the order in which successive sign vectors are chosen, the decomposition could be different. However, if an initial sign vector, such as $\mathbf{e} = (1, \ldots, 1)^T$, and a traversal path is chosen, the decomposition is effectively unique for our purposes and it is safe to speak of "the" centroid decomposition.

The centroid decomposition and SVD have many similarities, most notably in the function of the sign vectors of the centroid method and the left singular vectors of the SVD. The variational forms for the first sign vector and first left singular vector of a matrix A are very similar (See Table 1). The sign vector $\mathbf{z}$ is constrained to the vertices of an m-dimensional hypercube while the singular vector $\mathbf{x}$ is allowed to vary unconstrained in $\mathbb{R}^m$. The flexibility of the singular vector allows it to assume the direction which maximizes its significance with respect to the data in A but it is expensive to compute. The sign vector can be found using the centroid method, a fast $O(n)$ iterative method, but it is only a close approximation to the singular vector. Furthermore, there is a close correspondence between the centroid values and the eigenvalues and between the centroid factors and right singular vectors. See Section 6 of [9] for more on these geometric and statistical similarities and numerical results.

2.3 LSI

Latent Semantic Indexing (LSI) is a method for automatic document indexing and retrieval which takes advantage of the semantic structure between terms and the documents containing them in order to improve detection of relevant documents for a given query. The main problems of previous lexical-matching indexing methods that LSI seeks to overcome are *synonymy*, multiple words associated with a single concept, and *polysemy*, a single word with multiple meanings [10]. LSI represents the documents in a term-document matrix and uses the singular value decomposition (SVD) to model this relationship in a reduced-dimension representation, using the closest rank-k approximation to the full SVD.

LSI represents the documents in a term-document matrix and uses the singular value decomposition (SVD) to model this relationship in a reduced-dimension representation, using the closest rank-k approximation to the full SVD. Let $H = [h_{ij}] \in \mathbb{R}^{m \times l}$ be the term-document matrix, where h_{ij} is the weight of term i in document j and let $H = U\Sigma\hat{V}^T$ be the SVD of H. Then $H_k \equiv U_k \Sigma_k \hat{V}_k^T$ is the closest rank-k approximation to H where U_k and $\hat{V}_k$ are formed by the first k columns of U and $\hat{V}$ respectively and Σ_k is the k-th leading principal submatrix of Σ.

Chu and Funderlic have suggested using the centroid decomposition in LSI because of its similarity to the SVD and the low computational cost. We represent a user query by a vector $\mathbf{q}_i^T = (q_{i1}, \ldots, q_{im}) \in \mathbb{R}^m$ where the q_{ik} are term weights in the query. To determine how well the documents match the query, calculate the matrix vector product

$$\mathbf{s}_i^T = \mathbf{q}_i^T H, \tag{2.5}$$

where the entry s_{ik} in $\mathbf{s}_i^T$ indicates the relevance of document k to the query. This can be extended for multiple queries $\mathbf{q}_i^T$ for $i = 1, \ldots, n$. As with the SVD version of LSI, there is a need store the term-document matrix in a more compact form without losing accuracy. Thus instead of the original indexing matrix H, we can score the query using the rank-k truncated CD approximation $H_k = B_k V_k^T$ where B_k and V_k consist of the first k columns of B and V respectively. In choosing an appropriate k for the approximation, there is a trade-off between parsimony and accuracy. There is no "right" choice of k, it depends heavily on the nature of the documents and the application.

3 Framework

3.1 Notation

Throughout the paper $\mathbf{e}_i$ will be used to denote the i^{th} standard basis vector, where the i^{th} component is 1 and all other components are 0. $\mathbf{e}$ is the vector of all ones. All vectors are column vectors and are typeset in bold. We will sometimes denote the dimensions of a matrix with subscripts for quick reference. For example, $H_{m\times l}$ indicates that H is an element of $\mathbb{R}^{m\times l}$.

We will make frequent use of the fundamental matrix subspaces. Suppose H is an $m \times l$ matrix where $H = (\mathbf{h}_1, \ldots, \mathbf{h}_l)$, and $\mathbf{h}_i$ are column vectors in $\mathbb{R}^m$. Then,

$$\mathcal{R}(H) = \operatorname{span}\{\mathbf{h}_1, \ldots, \mathbf{h}_l\} \subseteq \mathbb{R}^m \tag{3.1}$$

denotes the range, or column space, of H and

$$\mathcal{N}(H) = \{\mathbf{v} \mid H\mathbf{v} = \mathbf{0}\} \subseteq \mathbb{R}^l \tag{3.2}$$

denotes the nullspace of H. The rowspace of H is the span of the rows of H or $\mathcal{R}(H^T) \subseteq \mathbb{R}^l$, the range of H^T. The $\perp$ symbol will denote the orthogonal compliment of a vector space. Chapter 4 of [17] provides a more complete treatment of these concepts.

The relationship between the SVD and CD and the fundamental matrix subspaces is central to the workings of these updating methods. Let $H = U\Sigma\hat{V}^T$ be the SVD of $H_{m\times l}$ and $r = \operatorname{rank}(H)$. Then $\Sigma \in \mathbb{R}^{m\times l}$ with diagonal $(\sigma_1, \ldots, \sigma_l)$ where $\sigma_1, \ldots, \sigma_r > 0$ and $\sigma_{r+1}, \ldots, \sigma_l = 0$, U is $m \times m$, and $\hat{V}$ is $l \times l$. If we partition the columns of $U = (\mathbf{u}_i)$ and $\hat{V} = (\mathbf{v}_i)$ such that $U = [\, U_1 \; U_0 \,]$ and $\hat{V} = [\, \hat{V}_1 \; \hat{V}_0 \,]$ where $U_1 = (\mathbf{u}_1, \ldots, \mathbf{u}_r)$, $U_0 = (\mathbf{u}_{r+1}, \ldots, \mathbf{u}_m)$, $\hat{V}_1 = (\mathbf{v}_1, \ldots, \mathbf{v}_r)$, and $\hat{V}_0 = (\mathbf{v}_{r+1}, \ldots, \mathbf{v}_l)$, then:

1. $\hat{V}_0$ is an orthonormal basis for $\mathcal{N}(H)$

2. $\hat{V}_1$ is an orthonormal basis for $\mathcal{N}(H)^\perp$

3. U_1 is an orthonormal basis for $\mathcal{R}(H)$

4. U_0 is an orthonormal basis for $\mathcal{R}(H)^\perp$

See [15] for a more in-depth treatment of the relationship between these subspaces and the SVD.

These subspaces are central to subspace based updating methods such as Brand's [3, 4]. These methods project the updated data onto the fundamental subspaces which allows the updated SVD to be rewritten in terms of the SVD of a simpler matrix. The properties of the simple matrix can be leveraged to calculate its SVD quickly which makes finding the updated SVD much easier.

Because the columns of the factor matrix $V = (\mathbf{v}_i)$ from the centroid decomposition are orthogonal and are close approximations to the right singular vectors, we have developed similar subspace based updating methods for the centroid decomposition. One drawback is the absence of any vectors emulating the left singular vectors. Thus, we are limited to working in $\text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_l\} = \mathcal{R}(H^T)$, the rowspace of H.

By projecting the updated data onto $\mathcal{R}(H^T)$ and its orthogonal complement, we can rewrite the updated CD in terms of the CD of a simpler matrix. The task then becomes finding a fast way to find the CD of the simple matrix, which looks promising because the simple matrix is usually formed from B and we show in section 4.2 that B is a stationary point of the CD algorithm.

3.2 Matrix Updates

A matrix "update" usually involves modifying the existing elements, rows or columns of a matrix or augmenting it with additional rows or columns which correspond to new or incoming data. Other operations such as removing rows or columns are sometimes called "downdates." Instead of distinguishing between the many types of matrix modifications we develop a single framework to handle all such operations using four basic matrix operations: permutation, augmentation, truncation, and additive rank-1 updates. This framework allows us to avoid defining algorithms for many special cases and focus only on rank-1 updates.

Rank-1 updates are of the form $H_1 = H_0 + \mathbf{a}\mathbf{b}^T$ where H_0 is an $m \times l$ matrix and $\mathbf{a}$ and $\mathbf{b}$ are column vectors in $\mathbb{R}^m$ and $\mathbb{R}^l$ respectively. They are performed in the following context: Suppose that the original data matrix H_0 and its centroid decomposition, $H_0 = B_0 V_0^T$, are given, where $H_0 \in \mathbb{R}^{m \times l}$, $B_0 \in \mathbb{R}^{m \times k}$, and $V_0 \in \mathbb{R}^{l \times k}$. Any updated matrix H_1 can now be formed by performing a sequence of the four basic operations on H_0. Using this method, the dimensions of the updated data matrix H_1 need not be the same as H_0.

The basic method proceeds as follows:

1. Suppose $H_0 = B_0 V_0^T$, $\mathbf{a}$, and $\mathbf{b}$ are given.

2. Permute and pad H_0 to the correct dimension with zero vectors to form $\tilde{H}_0$ if necessary.

3. Perform the rank-1 update $\tilde{H}_1 = \tilde{H}_0 + \mathbf{a}\mathbf{b}^T$ using an appropriate update rule.

4. Truncate $\tilde{H}_1$ to the correct dimension by removing zero vectors and permute if needed to form H_1.

For example, appending a column $\mathbf{c}$ to a matrix H_0 can be accomplished by appending a column of zeros to H_0 and performing a rank-1 update as follows: Let $\tilde{H}_0 = [\, H_0 \; \mathbf{0} \,]$, $\mathbf{a} = \mathbf{c}$ and $\mathbf{b} = \mathbf{e}_{l+1}$ then:

$$H_1 = \tilde{H}_0 + \mathbf{a}\mathbf{b}^T = \begin{bmatrix} H_0 & \mathbf{0} \end{bmatrix} + \mathbf{c}\mathbf{e}_{l+1}^T = \begin{bmatrix} H_0 & \mathbf{c} \end{bmatrix} \tag{3.3}$$

In general, the data matrix H_1 can be represented as a series of rank-1 updates of a matrix $\tilde{H}_0$ constructed from H_0 by permuting rows and columns and appending or truncating rows and columns of zeros. Permutations are needed for updates such as removing the i-th column of H_0. In many cases we will omit $\tilde{H}_0$ and $\tilde{H}_1$ for simplicity and assume that H_0 has already been padded to the correct dimension or that H_1 will be truncated later.

4 Rank-1 Updating Method

4.1 Central Identity

Suppose we are given the centroid decomposition of a matrix $H_0 = B_0 V_0^T$ and we want to perform a rank-1 update $H_1 = H_0 + \mathbf{a}\mathbf{b}^T$. How do we use the CD of H_0 to find the CD of the updated matrix H_1 without explicitly calculating it? The problem can be reduced to finding the CD of a simpler matrix S which serves as an intermediate estimation of the loading matrix B_0.

The columns of the factor matrix V_0 form an orthogonal basis for $\mathcal{R}(H_0^T)$, the rowspace of H_0. We can write $H_0^T = \left(\mathbf{h}_1^T, \ldots, \mathbf{h}_m^T\right)$, $B_0^T = \left(\mathbf{b}_1^T, \ldots, \mathbf{b}_l^T\right)$, and $V_0 = \left(\mathbf{v}_1, \ldots, \mathbf{v}_l\right)$ so that

$$H_0^T = V_0 B_0^T = (\mathbf{h}_1, \ldots, \mathbf{h}_m) = (\mathbf{v}_1, \ldots, \mathbf{v}_l)\,(\mathbf{b}_1, \ldots, \mathbf{b}_l)\,. \tag{4.1}$$

Each column $\mathbf{h}_j$ of H_0^T is a linear combination of columns of V_0:

$$\mathbf{h}_j = (\mathbf{v}_1, \ldots, \mathbf{v}_l)\,\mathbf{b}_j = V_0 \mathbf{b}_j \tag{4.2}$$

and the columns of V_0 are mutually orthogonal so they are a basis for $\mathcal{R}(H_0^T)$.

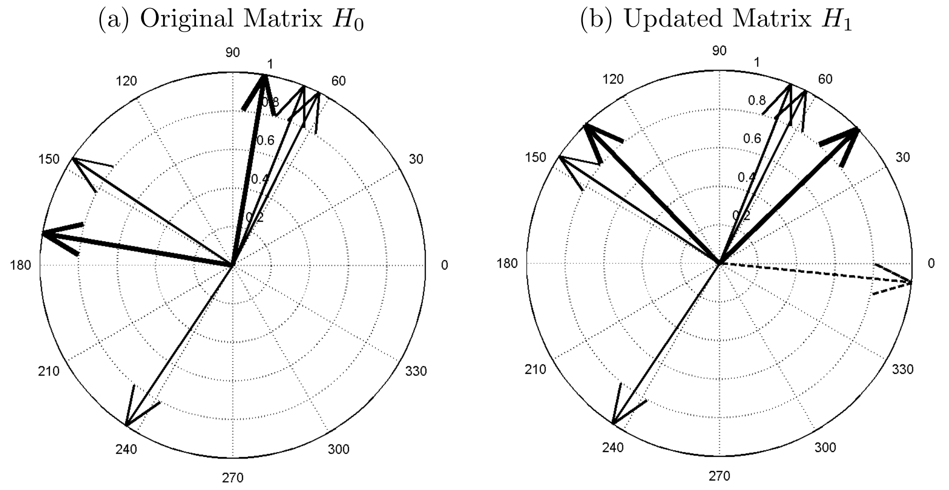
A rank-1 update can increase or decrease the rank of the matrix, and thus the dimension of its rowspace, by at most 1. So we have:

$$\dim \mathcal{R}(H_0^T) - 1 \leq \dim \mathcal{R}(H_1^T) \leq \dim \mathcal{R}(H_0^T) + 1. \tag{4.3}$$

If the update increases the dimension of the rowspace, we can choose a matrix Q such that $[\, V_0 \; Q \,]$ is an orthogonal basis for $\mathcal{R}(H_1^T)$. The vectors in Q are needed to expand the basis to include components of the update that aren't in the span of the original centroid factors. If the rank of the data matrix is not increased by the update, we can simply omit Q and use the original factors. In the rank-1 cases we consider, if we require Q it will be a single vector.

Although $[\, V_0 \; Q \,]$ is a basis for the updated rowspace $\mathcal{R}(H_1^T)$, it does not necessarily contain the true centroid factors because the update can modify the direction of the original centroid factors as well. Therefore the "temporary" factors $[\, V_0 \; Q \,]$ must be counter-rotated to align with the true ones (see Figure 1). In this way, we have identified two independent effects of the matrix update: changing the number of factors and modifying the data associated with those factors.

Consider the factorization $H_1 = S\,[\, V_0 \; Q \,]^T$ for some matrix S (defined below) using the old centroid factors in V_0 augmented with the temporary factors in Q. This is



A new data vector (dashed) is added which rotates the centroid factors (bold). Data vectors have been normalized to the unit circle.

Figure 1: Row Update

"almost" the centroid decomposition of H_1 because adding a single vector will usually not dramatically change the factors. We can find the new centroid factors V_1 by finding the centroid decomposition $S = B_S V_S^T$ and then rotating $[\, V_0 \; Q \,]$ by V_S:

$$\begin{aligned} H_1 &= H_0 + \mathbf{a}\mathbf{b}^T \\ &= S \begin{bmatrix} V_0 & Q \end{bmatrix}^T \\ &= B_S V_S^T \begin{bmatrix} V_0 & Q \end{bmatrix}^T \\ &= B_S \left(\begin{bmatrix} V_0 & Q \end{bmatrix} V_S \right)^T \\ &= B_1 V_1^T. \end{aligned} \tag{4.4}$$

Therefore, by finding the CD of S we can construct the CD of H_1 where

$$B_1 = B_S \tag{4.5}$$

and

$$V_1 = \begin{bmatrix} V_0 & Q \end{bmatrix} V_S. \tag{4.6}$$

We can rewrite S in terms of known matrices as:

$$\begin{aligned} S &= \left(H_0 + \mathbf{a}\mathbf{b}^T \right) \begin{bmatrix} V_0 & Q \end{bmatrix} \\ &= \begin{bmatrix} H_0 V_0 & H_0 Q \end{bmatrix} + \mathbf{a}\mathbf{b}^T \begin{bmatrix} V_0 & Q \end{bmatrix} \\ &= \begin{bmatrix} B_0 & \mathbf{0} \end{bmatrix} + \mathbf{a} \begin{bmatrix} \mathbf{b}^T V_0 & \mathbf{b}^T Q \end{bmatrix} \\ &= \begin{bmatrix} B_0 & \mathbf{0} \end{bmatrix} + \mathbf{a} \begin{bmatrix} V_0^T \mathbf{b} \\ Q^T \mathbf{b} \end{bmatrix}^T. \end{aligned} \tag{4.7}$$

Thus, S can be formed by a rank-1 update of $[\, B_0 \; \mathbf{0} \,]$. We show in Section 4.2 that B is a stationary point for the centroid method. If we can exploit this in calculating the CD of S then we can potentially significantly reduce the time required to compute the updated CD. In practice we would use $\tilde{H}_0 = \tilde{B}_0 \tilde{V}_0^T$ as the starting point but we assume here that the zero vectors have already been appended.

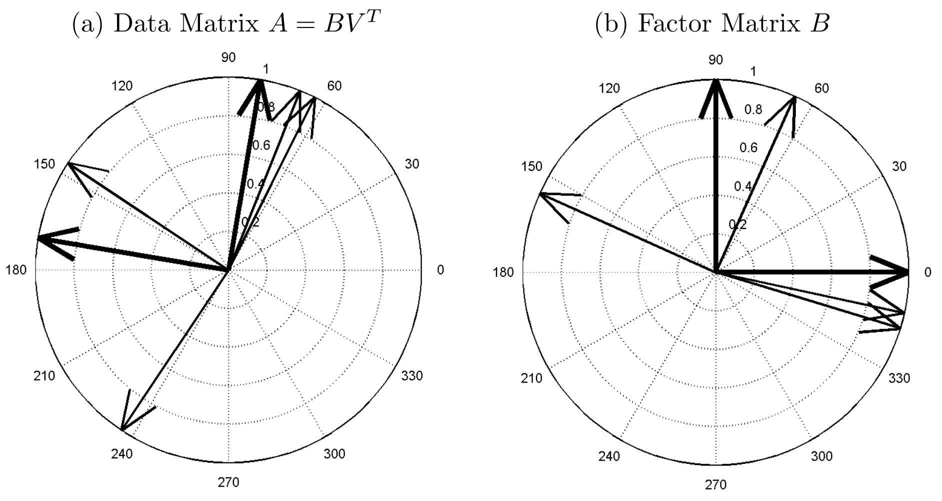


Figure 2: The data vectors and centroid factors (bold) of A and B compared.

4.2 A Stationary Point

We will show below that the loading matrix B is a stationary point for the centroid algorithm. This is important because we hope to take advantage of efficiencies in finding the CD of S in equation (4.7) and S is only a rank-1 update away from B. Figure 2 provides a geometric interpretation of this proposition. Essentially, the centroid factors have already been "factored out" of the data matrix leaving B with the primary factors aligned with the standard basis vectors.

Proposition 4.1. *If $A_{m\times l} = B_{m\times l}V_{l\times l}^T$ is the centroid decomposition of some matrix A, then $B = BI^T$ is the centroid decomposition of B.*

Proof. We apply the general centroid algorithm as described in Section 2.2 to A and B and denote the steps of the algorithm applied to B with primes. For example: $B_1 \equiv B$ and $B_i = B_{i-1} - \mathbf{b}'_i\mathbf{v}'^T_i$. Let $\mathbf{e}_i$ denote the i-th column of the identity matrix and $\mathbf{e}$ denote the vector of all ones. It is important to note that for any j, $\mathbf{v}_j^T V = \mathbf{e}_j^T$ since $VV^T = I$. The proof proceeds by induction over the iteration index by showing that at each step in the algorithm applied to B: $\mathbf{z}'_i = \mathbf{z}_i$, $\mathbf{v}'_i = \mathbf{e}_i$ and $\mathbf{b}'_i = \mathbf{b}_i$.

For the initial iteration, $i = 1$, we have $A = BV^T$, $A_1 \equiv A$, and $B_1 \equiv B = AV = A_1V$. It follows that

$$\mathbf{z}'_1 = \arg\max_{|\mathbf{z}|=\mathbf{e}} \|B_1^T\mathbf{z}\| = \arg\max_{|\mathbf{z}|=\mathbf{e}} \|V^T A_1^T\mathbf{z}\| = \arg\max_{|\mathbf{z}|=\mathbf{e}} \|A_1^T\mathbf{z}\| = \mathbf{z}_1, \tag{4.8}$$

$$\mathbf{v}'_1 = \frac{\mathbf{w}'_1}{\|\mathbf{w}'_1\|} = \frac{B_1^T\mathbf{z}'_1}{\|B_1^T\mathbf{z}'_1\|} = \frac{(A_1V)^T\mathbf{z}_1}{\|(A_1V)^T\mathbf{z}_1\|} = V^T\frac{A_1^T\mathbf{z}_1}{\|A_1^T\mathbf{z}_1\|} = V^T\mathbf{v}_1 = \mathbf{e}_1, \tag{4.9}$$

and

$$\mathbf{b}'_1 = B\mathbf{v}'_1 = B\mathbf{e}_1 = \mathbf{b}_1. \tag{4.10}$$

So the induction hypothesis is true for $i = 1$. Suppose it holds true for $i = j - 1$. That is, suppose $\mathbf{z}'_{j-1} = \mathbf{z}_{j-1}$, $\mathbf{v}'_{j-1} = \mathbf{e}_{j-1}$ and $\mathbf{b}'_{j-1} = \mathbf{b}_{j-1}$. Then,

$$\begin{aligned}B_j = B_{j-1} - \mathbf{b}'_{j-1}\mathbf{v}'^T_{j-1} = A_{j-1}V - \mathbf{b}'_{j-1}\mathbf{v}'^T_{j-1} = A_{j-1}V - \mathbf{b}_{j-1}\mathbf{e}^T_{j-1} \\ = A_{j-1}V - \mathbf{b}_{j-1}\mathbf{v}^T_{j-1}V = (A_{j-1} - \mathbf{b}_{j-1}\mathbf{v}^T_{j-1})V = A_jV,\end{aligned} \quad (4.11)$$

$$\mathbf{z}'_j = \arg\max_{|\mathbf{z}|=\mathbf{e}} \|B_j\mathbf{z}\| = \arg\max_{|\mathbf{z}|=\mathbf{e}} \|A_jV\mathbf{z}\| = \arg\max_{|\mathbf{z}|=\mathbf{e}} \|A_j\mathbf{z}\| = \mathbf{z}_j, \quad (4.12)$$

$$\mathbf{v}'_j = \frac{\mathbf{w}'_j}{\|\mathbf{w}'_j\|} = \frac{B_j^T\mathbf{z}'_j}{\|B_j^T\mathbf{z}'_j\|} = \frac{(A_jV)^T\mathbf{z}_j}{\|(A_jV)^T\mathbf{z}_j\|} = V^T\frac{A_j^T\mathbf{z}_j}{\|A_j^T\mathbf{z}_j\|} = V^T\mathbf{v}_j = \mathbf{e}_j, \quad (4.13)$$

and finally,

$$\mathbf{b}'_j = B\mathbf{v}'_j = B\mathbf{e}_j = \mathbf{b}_j. \quad (4.14)$$

So if $B = B'V'^T$ is the centroid decomposition of B, then $B' = (\mathbf{b}'_i) = (\mathbf{b}_i) = B$ and $V' = (\mathbf{v}'_i) = (\mathbf{e}_i) = I$.

□

5 LSI Operations

More specific types of matrix updates commonly used in LSI can now be examined in the context of the updating framework outlined in Section 4. The basic LSI database operations are outlined in Table 2 and two of them, term or row updates and document or column updates are explored in more detail. Using the rank-1 updating framework allows us to apply essentially the same updating algorithm to each situation, rather than defining separate updates for rows and columns or terms and documents. We look at each update in terms of equation (4.4) by first finding an appropriate Q to extend the original orthogonal basis V and then using information about the specific update, given in Table 2, to simplify the rule for S given by equation (4.7).

Let H_0 be an $m \times l$ term-document matrix with $m >> l$ let $B_0V_0^T = H_0$ be it's centroid decomposition. Suppose a rank-1 update is applied to the matrix to form a new matrix H_1 so that $H_1 = H_0 + \mathbf{ab}^T$ where $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^l$. Depending on the type of update, we can choose a and b accordingly.

5.1 Term Updates (Appending a Row)

A term update, adding a new term to the database, consists of appending a row to the term-document matrix. We want to find the CD of the matrix $H_1 = \begin{bmatrix} H_0 \\ \mathbf{r}^T \end{bmatrix} \in \mathbb{R}^{(m+1)\times l}$. In order to cast this problem into our rank-1 updating framework, we begin by appending a row of zeros to H_0 to form the intermediate matrix $\tilde{H_0} = \begin{bmatrix} H_0 \\ \mathbf{0} \end{bmatrix}$. $\tilde{H_0}$ has the same dimensions as the updated matrix H_1 and since the CD of H_0 is given, finding the CD of $\tilde{H_0}$ is straightforward:

$$\tilde{H_0} = \begin{bmatrix} H_0 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} B_0 \\ \mathbf{0} \end{bmatrix} V_0^T = \tilde{B_0}\tilde{V_0}^T. \quad (5.1)$$

Now, the row append can be expressed as the rank-1 update $H_1 = \tilde{H_0} + \mathbf{ab}^T$, where $\mathbf{a} = \mathbf{e}_{m+1}$ and $\mathbf{b} = \mathbf{r}$. Knowing $\mathbf{a}$ and $\mathbf{b}$ allows us to simplify equation (4.7) using a subspace argument. As before, we suppose H_0 has already been padded with zeros in order to drop the tilde.

Operation	Known	Desired	$\mathbf{a}$	$\mathbf{b}^T$
Add Term (Row, $\mathbf{r} \in \mathbb{R}^l$)	$\begin{bmatrix} B_0 \\ \mathbf{0} \end{bmatrix} V_0^T = \begin{bmatrix} H_0 \\ \mathbf{0} \end{bmatrix}$	$B_1 V_1^T = \begin{bmatrix} H_0 \\ \mathbf{r}^T \end{bmatrix}$	$\mathbf{e}_{m+1}$	$\mathbf{r}^T$
Add Document (Column, $\mathbf{c} \in \mathbb{R}^m$)	$B_0 \begin{bmatrix} V_0 \\ \mathbf{0} \end{bmatrix}^T = \begin{bmatrix} H_0 & \mathbf{0} \end{bmatrix}$	$B_1 V_1^T = \begin{bmatrix} H_0 & \mathbf{c} \end{bmatrix}$	$\mathbf{c}$	$\mathbf{e}_{l+1}^T$
Remove Term (Row, $\mathbf{r} \in \mathbb{R}^l$)	$B_0 V_0^T = \begin{bmatrix} H_1 \\ \mathbf{r}^T \end{bmatrix}$	$B_1 V_1^T = \begin{bmatrix} H_1 \\ \mathbf{0} \end{bmatrix}$	$\mathbf{e}_{m+1}$	$-\mathbf{r}^T$
Remove Document (Column, $\mathbf{c} \in \mathbb{R}^m$)	$B_0 V_0^T = \begin{bmatrix} H_1 & \mathbf{c} \end{bmatrix}$	$B_1 V_1^T = \begin{bmatrix} H_1 & \mathbf{0} \end{bmatrix}$	$-\mathbf{c}$	$\mathbf{e}_{l+1}^T$
Weight Adjustment (Element, $h_{ij} + w$)	$B_0 V_0^T = H_0$	$B_1 V_1^T = H_0 + w\mathbf{e}_i \mathbf{e}_j^T$	$\mathbf{e}_i$	$w\mathbf{e}_j^T$

Table 2: LSI operations expressed as rank-1 updates where $H_0 \in \mathbb{R}^{m \times l}$, $\mathbf{c} \in \mathbb{R}^m$, $\mathbf{r} \in \mathbb{R}^l$, and $w \in \mathbb{R}$.

First in order to satisfy equation (4.4), we must find a matrix Q so that $\begin{bmatrix} V_0 & Q \end{bmatrix}$ is a basis for $\mathcal{R}(H_1^T)$, the rowspace of the updated matrix. We would need at most one additional basis vector in the rank-1 update case so here Q is a single vector. It is possible that the update could decrease the rank of the matrix, but this case would best be dealt with when considering term removal. We consider the other two cases: either the update increases the rank of the matrix, and thus the dimension of the updated rowspace, or it doesn't. In the former case we will require Q and in the latter we can omit it.

Next equation (4.7) can be simplified by projecting $\mathbf{b}$ onto two orthogonal complementary subspaces of $\mathbb{R}^l$. $\mathcal{R}(H_0^T) = \mathcal{R}(V_0)$ is the rowspace of H_0 and let $\mathcal{R}(V_0)^\perp$ be its orthogonal complement in the rowspace of H_1. The vector Q, which we find later, will be the extra basis vector that spans this orthogonal complement. The component of $\mathbf{b}$ in $\mathcal{R}(V_0)$ is

$$\mathbf{n} \equiv V_0^T \mathbf{b} \tag{5.2}$$

and the projection of $\mathbf{b}$ onto the orthogonal complement is

$$\mathbf{q} \equiv (I - V_0 V_0^T)\mathbf{b} = \mathbf{b} - V_0 V_0^T \mathbf{b} = \mathbf{b} - V_0 \mathbf{n}. \tag{5.3}$$

So $\|\mathbf{q}\|$ indicates whether the update has increased the rank of the matrix. Now there are two cases.

Case 1: $\|\mathbf{q}\| \neq 0$**.** In this case, part of $\mathbf{b}$ lies in $\mathcal{R}(V_0)^\perp$ and there exists a basis vector Q such that

$$\mathbf{b} = QQ^T\mathbf{b} + VV^T\mathbf{b} = Q\|\mathbf{q}\| + V\mathbf{n}. \tag{5.4}$$

Then from equation (5.3) we find $Q = \frac{\mathbf{q}}{\|\mathbf{q}\|}$. Substituting Q into equation (4.7) gives

$$\begin{aligned} S &= \begin{bmatrix} \begin{bmatrix} B \\ \mathbf{0} \end{bmatrix} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{n} \\ \|\mathbf{q}\| \end{bmatrix}^T \\ &= \begin{bmatrix} B & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{n}^T & \|\mathbf{q}\| \end{bmatrix} \\ &= \begin{bmatrix} B & \mathbf{0} \\ \mathbf{n}^T & \|\mathbf{q}\| \end{bmatrix} \end{aligned} \tag{5.5}$$

Case 2: $\|\mathbf{q}\| = 0$. In this case, $\mathbf{b}$ lies completely within the rowspace of H_0 so an additional basis vector is not needed. Thus Q and $\|\mathbf{q}\|$ can be omitted from the equation to find

$$S = \begin{bmatrix} B \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \mathbf{n}^T = \begin{bmatrix} B \\ \mathbf{n}^T \end{bmatrix}. \tag{5.6}$$

After finding the CD of S, the updated CD of H_1 can be constructed using equations (4.5) and (4.6).

5.2 Document Updates (Appending a Column)

Suppose now that we want to append a column $\mathbf{c}$, representing a new document, to H_0. What is the updated CD $B_1 V_1^T = [\,H_0\ \mathbf{c}\,]$? First, by restating the problem as a rank-1 update we can simplify the problem. We find, as in Table 2, that $\mathbf{a} = \mathbf{c}$ and $\mathbf{b}^T = \mathbf{e}_{m+1}^T$. First we find $\mathbf{n}$, $\mathbf{q}$ and Q as above:

$$\mathbf{n} \equiv \tilde{V_0}^T \mathbf{b} = \begin{bmatrix} V_0^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} = 0, \tag{5.7}$$

$$\mathbf{q} \equiv (I - V_0 V_0^T)\mathbf{b} = \mathbf{b} - V_0 V_0^T \mathbf{b} = \mathbf{b} - V_0 \mathbf{n} = \mathbf{b} = \mathbf{e}_{l+1}, \tag{5.8}$$

and

$$Q = \frac{\mathbf{q}}{\|\mathbf{q}\|} = \mathbf{e}_{l+1} \tag{5.9}$$

Since the column update involves a column of the identity matrix, equation (4.7) can be simplified:

$$\begin{aligned} S &= \begin{bmatrix} \tilde{B_0} & \mathbf{0} \end{bmatrix} + \mathbf{c} \begin{bmatrix} \tilde{V_0}^T \mathbf{b} \\ Q^T \mathbf{b} \end{bmatrix}^T \\ &= \begin{bmatrix} B_0 & \mathbf{0} \end{bmatrix} + \mathbf{c} \begin{bmatrix} \begin{bmatrix} V_0 \\ \mathbf{0} \end{bmatrix}^T \mathbf{e}_{l+1}^T \\ Q^T \mathbf{e}_{l+1}^T \end{bmatrix}^T \\ &= \begin{bmatrix} B_0 & \mathbf{0} \end{bmatrix} + \mathbf{c} \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_{l+1}^T \mathbf{e}_{l+1} \end{bmatrix}^T \\ &= \begin{bmatrix} B_0 & \mathbf{0} \end{bmatrix} + \mathbf{c} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}^T \\ &= \begin{bmatrix} B_0 & \mathbf{c} \end{bmatrix} \end{aligned} \tag{5.10}$$

The updated centroid decomposition of H_1 can be determined by using the CD of S and equations (4.5) and (4.6).

6 Conclusion

We have developed a flexible rank-1 updating framework for the centroid decomposition that is built on ideas from subspace based SVD updating methods. Unfortunately the efficiencies of simplifying the SVD updating problem do not directly translate to the CD problem. We have shown that B is a stationary point of the algorithm and that two fundamental LSI updates can be expressed in terms of B. Thus if a relationship can be found to quickly calculate the CD of the simple S matrices based on B then finding the CD of the updated matrix will only require a few simple orthogonal rotations. This should be a major focus for future work along with investigating the implications of using the rank-k truncated CD as a starting point, analyzing the complexity and storage requirements of the algorithms, determining the resulting updated sign vectors, and implementing additional updates used in LSI as well as factor analysis.

References

[1] M. W. Berry. Large-scale sparse singular value computations. *The International Journal of Supercomputer Applications*, 6(1):13–49, Spring 1992.

[2] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, December 1995.

[3] M. E. Brand. Incremental singular value decomposition of uncertain data with missing values. *European Conference on Computer Vision (ECCV)*, 2350:707–720, May 2002.

[4] M. E. Brand. Fast online svd revisions for lightweight recommender systems. In *Proceedings, SIAM 3rd International Conference on Data Mining*, 2003.

[5] J. R. Bunch and C. P. Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31:111–129, 1978.

[6] C. Burt. *The Distribution and Relations of Educational Abilities.* P. S. King and Son, London, 1917.

[7] P. A. Businger. Updating a singular value decomposition. *BIT*, 10:376–397, 1970.

[8] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical models and image processing: GMIP*, 59(5):321–332, 1997.

[9] M. T. Chu and R. E. Funderlic. The centroid decomposition: Relationships between discrete variational decompositions and svds. *SIAM Journal on Matrix Analysis and Applications*, 23(4):1025–1044, 2002.

[10] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[11] M. Gu and S. C. Eisenstat. A stable and fast algorithm for updating the singular value decomposition. Technical Report YALEU/DCS/RR-966, Yale University, New Haven, CT, 1993.

[12] H. H. Harman. *Modern Factor Analysis.* University of Chicago Press, Chicago, 1967.

[13] P. Horst. *Factor Analysis of Data Matrices.* Holt, Rinehart and Winston, New York, 1965.

[14] W. Heiser L. Hubert, J. Meulman. Two purposes for matrix factorization: A historical appraisal. *SIAM Review*, 42:68–82, 2000.

[15] S. Leach. Singular value decomposition - a primer. Unpublished Manuscript, Department of Computer Science, Brown University, Providence, RI, USA, 1995.

[16] T. A. Letsche and M. W. Berry. Large-scale information retrieval with latent semantic indexing. *Information Sciences*, 100:105–137, 1997.

[17] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM, 2000.

[18] L. L. Thurstone. Multiple factor analysis. *Psychological Review*, 38:406–427, 1931.

[19] D. I. Witter and M. W. Berry. Downdating the latent semantic indexing model for conceptual information retrieval. *The Computer Journal*, 41(8):589–601, 1998.

[20] H. Zha and H. D. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.