Spectral Analysis for Information Retrieval and Data Mining

Jared Saia

June 29, 2000

1 Introduction

The growth of the Internet in recent years has led to a proliferation of digital data that is readily available for computational analysis. This data tends to have the following qualities: 1) it occurs in large(even overwhelming) quantities 2) it is noisy, imprecise and redundant 3) it contains underlying structure that can be described concisely and is useful for clustering, prediction, outlier detection and human analysis of the data. This paper discusses spectral methods for discovering the underlying structure in large data sets of this form. These techniques have had excellent empirical and theoretical success in finding useful structure in vast quantities of data.

1.1 Mathematical Preliminaries

This paper relies heavily on the mathematics of linear algebra. Most of the necessary background facts that will be used are given in the appendix in Section A. Proofs of all of theorems presented in the paper can also be found in the appendix. The following standard linear algebra notation is used in this paper: if A is a matrix then we use a_i to refer to its i^{th} column and $a_{i,j}$ to refer to the cell in the i^{th} row and j^{th} column.

The primary mathematical tool we will be using is the singular value decomposition(also known as the SVD or spectral decomposition) of a matrix - an example is given in Figure 1. It's the case that *any* $m \times n$ matrix A of rank r can be factored into three matrices U,D and V^T where U and V are orthonormal and D is diagonal and has r nonzero entries decreasing in magnitude from top to bottom. U is $m \times m$, D is $m \times n$ and V is $n \times n$. This factoring is called the SVD. The columns in the matrix U are called the left singular vectors of A and the columns in the matrix V are called the right singular vectors. The entries in D are the singular values of A (σ_i). We will generally use the phrase "i-th left(right) singular vector" to refer to $u_i(v_i)$ which are the singular vectors associated with σ_i for $i \leq min(m,n)$.

For any matrix A let $A = UDV^T$ be the SVD. If we let U_k be a $m \times k$ matrix whose first k columns are

Figure 1: The Singular Value Decomposition

$$A = \overbrace{\begin{bmatrix} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{bmatrix}}^{U:m\times m} \overbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}}^{D:m\times n} \overbrace{\begin{bmatrix} - & v_1 & - \\ & \vdots & \\ - & v_n & - \end{bmatrix}}^{V^T:n\times n}$$

the same as those of U and let $A_k = U_k U_k^T A$ (the projection of the columns of A onto the space spanned by the top k largest left singular vectors) then we know:

Fact 1 *The matrix A_k minimizes $|A - B|$ and $|A - B|_F$ over all matrices B that have rank $\leq k$ and $|A - A_k| = \sigma_{k+1}(A)$, $|A - A_k|_F^2 = \sum_{i=k+1}^{r} (\sigma_i(A))^2$*

Another fact important for this paper is that:

Fact 2 *For any matrix A, $maxarg_{|x|=1,|y|=1} x^T Ay = (u_1(A), v_1(A))$ and $max_{|x|=1,|y|=1} x^T Ay = \sigma_1(A)$.*

There is one remaining crucial fact about the SVD which is that it takes a long time to compute. Standard algorithms to find the SVD for a $m \times n$ matrix where $m \leq n$ run in time $O(m^2 n)$. In section 6 we'll discuss preliminary steps from [8] to try to find faster algorithms.

1.2 Applications of the SVD

In this paper, we discuss four applications of the SVD: link analysis [2], latent semantic indexing [13, 5, 6], Ratio Rules [9] and discrete clustering [8]. In each of these domains, we can translate the input data we're given into a matrix. Also in each of these applications the singular values and singular vectors in this matrix will give us crucial information about the input data.

In the Link Analysis problem of Section 2 the input data is a graph of the link structure of web pages which we wish to use to identify "important" pages. We translate this graph into a matrix A by letting $a_{i,j}$ be 1 if page i points to page j and 0 otherwise. In this application, the largest left singular vector and largest right singular vector of A give an assignment of (what are referred to as) "authority" and "hub" weights to the nodes which maximizes the sum over all edges in the graph of the "hub weight" of the edge's source times the "authority weight" of its sink. The largest components of the first singular vectors then give us the pages that are "important" according to the link structure. The largest singular value gives the score of the weight assignments.

In the Latent Semantic Indexing problem of Section 3, the input data is a collection of n documents which contain m unique terms and we want to measure the topical similarity of any two documents. We translate this input into a matrix A by letting $a_{i,j}$ be the frequency of term i in document j. The left singular vectors of A now intuitively correspond to "topics" of the documents and the ith *column* of DV^T gives for document i the degree to which it participates in each topic. The singular values give a measure of the importance of the topics their corresponding left singular vectors define.

In the Ratio Rules problem of Section 4, the input data is information for n customers giving how much money they spent on m items and we wish to find the ratios with which items are bought together(i.e. apples:oranges:cookies is 1:2:3). We translate this input into a matrix A by letting $a_{i,j}$ be a function of the amount customer i spent on item j. If we assume there are underlying ratios in which each item appears only once, then we'd expect the left singular vectors of A to correspond to the ratios with which items are bought and the singular values to give the "importance" of their corresponding left singular vector in describing the data.

In the Discrete Clustering Problem(DCP) of Section 5, the input data is a collection of n points in an m dimensional space over the Real numbers and we wish to partition these points into a constant number k clusters such that sum of the squared distances of each point to the centroid of its cluster is minimized(this is an NP-Hard problem). We translate this input into a $m \times n$ matrix A by letting the points be columns

Figure 2: Example Page Link Graph and Authority/Hub Weightings

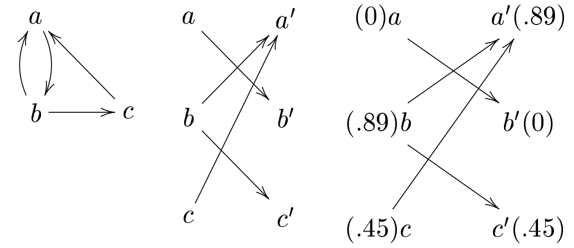

Table 1: AH-QUERY Algorithm

Algorithm AH-QUERY:

Input: A set of terms q, the link graph G

Constants: $t(=200)$

1) $R \leftarrow$ top t documents search engine returns for q

2) Grow R by all including all pages it points to and some pages pointing to it.

3) $G' \leftarrow$ subgraph induced by R' on G

4) Return $AH-GRAPH(G')$

Subroutine AH-GRAPH:

Input: Graph G (G has n nodes and adjacency matrix A)

Constants: $\epsilon > 0$, $t(=10)$

1) $\vec{x}_0 \leftarrow$ length n all 1's vector, $\vec{y}_0 \leftarrow \vec{0}$, $i \leftarrow 0$

2) Repeat

3) .....$i \leftarrow i+1$

4) .....$\vec{x}_i \leftarrow A^T\vec{y}_{i-1}$, $\vec{y}_i \leftarrow A\vec{x}_{i-1}$

5) .....$\vec{x}_i \leftarrow \vec{x}_i/|\vec{x}_i|$, $\vec{y}_i \leftarrow \vec{y}_i/|\vec{y}_i|$

6) Until $|\vec{x}_i - \vec{x}_{i-1}| \leq \epsilon$ and $|\vec{y}_i - \vec{y}_{i-1}| \leq \epsilon$

7) (Auth. nodes, Hub nodes) $\leftarrow$ top t weight nodes in $\vec{x}_i$,$\vec{y}_i$

of A. The largest k left singular vectors of A now span a subspace in which 1) it's fast to solve DCP since the dimension of the space is a constant 2) an optimal solution in this smaller dimensional space has cost provably no more than twice that of an optimal solution in the larger space.

For the link analysis, LSI and Ratio Rules problems, we'll present empirical evidence supporting the effectiveness of spectral techniques. For the LSI problem, we'll also present in Subsection 3.1 some theoretical analysis on when and why spectral techniques are effective. In Section 6, we'll present the empirical and theoretical results on approximation algorithms for computing the SVD and finally in Section 8, we'll present promising open problems for spectral methods.

2 Spectra for Link Analysis

Our first application of spectral methods is from [2] and involves finding important pages on the web using information about how web pages link to each other. We make the assumption that a link from one page to another means that the author of the first page found the second page useful. We are interested in using information about link structure between pages on a particular subject to find two types of useful pages 1) pages which are authorities on that subject or *authoritative* pages and 2) pages which point to many authoritative sites on the subject or *hub* pages.

Table 1 gives the algorithm developed in [2]. The high level idea is that AH-QUERY finds a collection of sites on a particular subject and AH-GRAPH finds the authoritative and hublike sites in that collection. At each step of AH-GRAPH, nodes which have high authority score confer a high "hub" weighting to all nodes that point to them while all nodes that have high "hub" score confer high authority weighting to the nodes that they point to. In [2], the weights at the end of this process are given as the *definition* of the authoritative and hub score of a page.

The first step of AH-QUERY gets a core set of pages, R from a search engine on the web using the query q. The second step grows this core set of pages by including all of the pages referenced by pages in R and a subset of all the pages that reference pages in R; it's necessary to take the subset because some pages on the web can have a too many referencing pages. The subset of pages referencing R included is computed by taking for every page r in R either a random sample of all the pages pointing to r if there are $> d$ such pages or all of the pages pointing to r if there are $\leq d$ such pages where d is a constant typically set to 50. AH-QUERY then feeds the subgraph containing this grown core to the subroutine AH-GRAPH.

The subroutine AH-GRAPH basically formalizes the notion of "hubness" and "authoritativeness". In AH-GRAPH, $\vec{x}_i$ corresponds to the best current guess of authority scores for each node and $\vec{y}_i$ corresponds to hub score estimates. The matrix multiplication gives the conference of hub weight to a page from authority weights of out neighbors and the conference of authority weight to a page from hub weights of in neighbors. The subroutine in practice converges in about 10 iterations.

We also present a second more mathematical definition of "good" authority and hub weights for a graph which turns out to be equivalent to the algorithmic definition given here. Fix a unique numbering of the n nodes of the graph and let a weight vector v be a vector of length n such that $|v| = 1$ and $v[i]$ is the weight associated with node number i. Define the *conductance* of two weight vectors x and y in a graph to be the sum over all edges e of the graph of $x[i] * y[j]$ where e is from node number i to node number j. the weight given to the source of the edge by x times the weight given to the target of the edge by y. Then we can define the authority and hub weights of a graph as given by the weight vectors x and y with maximum conductance. This corresponds to the vectors x and y which maximize $x^T Ay$ where A is the adjacency matrix for the graph. By fact 2, the best weight vectors are the left and right singular vectors.

Figure 2 gives an example of good authority and hub weight vectors for a small link graph. The original link graph is on the left, the middle graph contains two copies of each node x in the original graph and puts all edges out from x on the first copy and all edges into x on the second copy. Intuitively, the left copy gets the hub links of the node and the right copy the authority links. The rightmost graph in the figure gives a pair of weight vectors with good conductance and which are hence good authority and hub weights.

The following theorem in [2] shows that the maximum conductance based definition of authority and hub weights is the same as the algorithmic definition and proves that the algorithm converges. In either case, the good authority and hub pages are those that have high corresponding values in the *first* left and right singular vectors.

Theorem 1 *As $i \to \infty$ $x_i \to u_1$ and $y_i \to v_1$ where u_1, v_1 are the left and right singular vectors of A. if $\sigma_1(A) > \sigma_2(A)$.*

2.1 Empirical Analysis

According to [3], There are four empirical problems with Kleinberg's original algorithm: 1) the existence of high authority documents in the neighborhood graph which are not relevant to the search query 2) automatically generated links (e.g. pages that are automatically created from USENET news articles have automatic

Table 2: Spectral Web Clustering Results

Authorities	Precision	Recall
AH-QUERY:	.46	.27
improved:	.67	.41
Hubs	Precision	Recall
AH-QUERY:	.56	.29
improved:	.71	.4

links to the Hypernews web page) 3) mutually reinforcing hosts (e.g. two separate but closely affiliated sites which have many links between them) 4) cases where there are not many edges in the neighborhood graph.

Solutions to problems 1-3 proposed in [3] are 1)Calculate a relevance score between a document and the query based on words in the document, remove all documents in the query graph(G') below a certain similarity score 2) regulate the influence of a node based on its relevance to the query so that the authority and hub scores it confers is based on its relevance 3) Vary the weights of edges so there is not too much edge weight between two hosts.

Another extension of this basic algorithm is in [4] where a match between the query terms and text *nearby* the links determines the weight of the link.

2.2 Empirical Results

In [3], a set of 28 queries ranging from very popular queries("mutual funds") to somewhat rare ones("classical guitar") were fed into the algorithms and the set of the top 10 authoritative and hub-like documents from each algorithm were pooled together randomly and then independently rated for relevance by 3 volunteers. The ratings were combined and a final rating was decided by majority vote. The rating was essentially binary: relevant and authoritative(hub-like) or not.

For each of the algorithms, relative recall was computed as the number of truly relevant documents in the top 10 returned by the algorithm divided by the number of truly relevant documents in the entire pool. Precision was calculated as the number of truly relevant documents returned in the top 10 list by the algorithm on the query divided by 10.

Results on average precision and relative recall were returned across all queries. Table 2 gives results for AH-QUERY and for the algorithm which makes the corrections discussed above(improved).

Another positive empirical results for spectral based link analysis is that these methods are currently used by popular search engines such as "Google" and "Clever".

3 Spectral methods for Latent Semantic Indexing

For the remaining three application areas of LSI, Ratio Rules and DCP, the spectra of a matrix will be used to obtain a low dimensional orthonormal basis which is in some sense optimal for describing the data. If we have a $m \times n$ matrix and view the columns as points in a m dimensional space, then we will use SVD to find a k dimensional space (where k is some constant) which will retain much of the information about the points.

The first application of this form, Latent Semantic Indexing(LSI) (described in [13, 5, 6]) most commonly

Table 3: Example rank-2 SVD on four document vectors

$$
\begin{array}{r}
\text{computer} \\
\text{mouse} \\
\text{rodent}
\end{array}
\begin{bmatrix} 0 & 0 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 0 & 0 \end{bmatrix}
=
\begin{bmatrix} 1/\sqrt{6} & 1/\sqrt{2} \\ 2/\sqrt{6} & 0 \\ 1/\sqrt{6} & -1/\sqrt{2} \end{bmatrix}
\begin{bmatrix} \sqrt{24} & 0 \\ 0 & \sqrt{8} \end{bmatrix}
\begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ -1/2 & -1/2 & 1/2 & 1/2 \end{bmatrix}
$$

refers to a method for extracting high level semantic meaning from linguistic tokens. In the typical problem, we are given a $m \times n$ matrix, A, where the columns of A represent m documents and the rows of A represent words. The entry $a_{i,j}$ represents some function of the number of times word i appears in document j. This function can be the number of times the word appears, the percentage of the document the word makes up or simply a 1 if the word appears and 0 otherwise.

We can view the columns of the matrix A as vectors in a m dimensional space, commonly called "term space". The top k left singular vectors of A can be viewed as "topic" vectors and the space they span is called "topic space". Table 3 gives an example of rank-2 LSI on a collection of four documents(columns and rows that are all 0 are omitted for clarity). Intuitively, the first two documents are on the topic "rodents" and the second two documents are on the topic "computer peripherals". The topic vectors are $(1/\sqrt{6}, 2/\sqrt{6}, 1/\sqrt{6})$ and $(1/\sqrt{2}, 0, -1/\sqrt{2})$. The first topic vector is basically "rodent or computer peripheral" and the second topic vector is "computer peripheral". If we think of the topic vectors as coordinate axis, then the *columns* of the product of the two rightmost matrices gives the coordinates of each document on these two axis. For example, document 1 in the figure has coordinate$\sqrt{24}/2$ in direction of the first topic and $-\sqrt{8}/2$ in direction of the second topic.

We can think of these coordinates as the degree to which each of the four documents participate in each of these topic vectors. We note that each document participates equally in the first topic vector but the second topic vector nicely separates the documents on "rodents" from ones on "computer peripherals".

The key question asked in LSI is: How close are two documents in subject matter? In other words, given two vectors in term space, how close are their corresponding documents. For example if we wish to query the database with a set of terms, if we view the query as a vector in term space and if we have a good measure of "closeness", we can find the document vector which is closest to the query. One problem though is that "closeness" may not necessarily mean "containing the same words as". For example the query "mouse, computer" shares words in common with all the documents in Table 3 but its topic is close only to the third and fourth documents.

The most commonly used measure for determining the topical closeness of documents is the angle between the vectors representing the documents. Another possible measure, the Euclidean distance between vectors, does not work well because it overestimates closeness between two short documents and underestimates closeness between two long ones. The vector angle measurement of "closeness" does not suffer from this problem.

The fact that the word mouse in Table 3 can have two separate meanings is known as polysemy. We see that in the above simple case, LSI differentiated these two meanings nicely. A related linguistic phenomena, synonymy, is the fact that two words can have the same meaning. This can cause problems if we want the query “house” to return documents which also contain the word “home”. By looking at the underlying topic space of the document rather than terms, we can use LSI to partially solve this problem. We'll see that the angle between document vectors in the lower dimensional “LSI Space” is often a better measure of closeness of the documents than the angle between the vectors in “term space”

3.1 Theoretical Results

Here we give some theoretical results from [5, 6] on when “LSI Space” might be a better subspace to measure document closeness in than “Topic Space”. The basic crux of the results is that (given certain assumptions) if we fix two documents in a matrix and measure the angle between those documents before and after an error matrix is added to the document matrix then the angle will be perturbed less in “topic space” than in “term space”. If we assume a document generation model of the form where documents have an underlying topic representation and then noise is added, this result can explain the effectiveness of LSI.

In the following discussion, we will assume that we are given a perturbed matrix $\widehat{A}$ of the form $\widehat{A} = A+E$ where A is the original matrix and E is a “noise” matrix with small norm. We will show that under certain conditions, spectral techniques can determine whether two columns of $\widehat{A}$ were originally from the same or different topics in the original matrix A.

The basis for our results is a lemma in [5] which says that the singular vectors are perturbed by a “small” amount if a matrix is perturbed by a “small” amount. More formally, if we define $< X >$ to be the vector space spanned by the columns of the matrix X, then the lemma says that if $|E|$, is sufficiently small, there exists some matrix E_U such that $< \widehat{U}_k >=< U_k + E_U >$ where $|E_U|$ is a δ fraction of $|E|$.

This lemma can be used to show that angles between documents are approximately preserved even after adding the noise matrix E so long as E is not too large. We state this result and the conditions necessary for it to hold below. As per our usual notation, we will let a_i, $\widehat{a}_i$ be the i^{th} columns of A, $\widehat{A}$ respectively. We also let $a_i' = U_k^T a_i$ which we'll refer to as the projection of a_i into the “topic space” or “LSI Space” of A. Similarly, we let $\widehat{a}_i' = \widehat{U}_k^T \widehat{a}_i$. The following definition helps formalize our goal.

Definition 1 *Two columns $\widehat{a}_i$ and $\widehat{a}_j$ of $\widehat{A}$ are $O(\alpha)$-*skewed *(with respect to A) when it's true that 1)if $a_i' \cdot a_j' = 0$ then $\widehat{a}_i' \cdot \widehat{a}_j' \leq \alpha|\widehat{a}_i'||\widehat{a}_j'|$ and 2) if $a_i' \cdot a_j' = |a_i'||a_j'|$ then $\widehat{a}_i' \cdot \widehat{a}_j' \geq (1-\alpha)|\widehat{a}_i'||\widehat{a}_j'|$*

We will say the matrix $\widehat{A}$ is α-skewed with respect to A if each pair of columns of $\widehat{A}$ are α-skewed with respect to A. Generally, we'd like to α to be a number close to 0. Intuitively, if there are only k topics in the matrix A then if $a_i' \cdot a_j' = 0$, this corresponds to the two documents being on completely different topics and if $a_i' \cdot a_j' = |a_i'||a_j'|$, this corresponds to the documents being on the same linear combination of topics. So if a matrix $\widehat{A}$ is α-skewed with respect to A, we know that the k-rank SVD will allow us to determine if documents are on the same linear combination of topics or orthogonal combinations despite the error matrix.

The following Theorem is presented in [6].

Theorem 2 *Let A be a $m \times n$ matrix with $SVD(A) = U_k D_k V_k^T$ and let $\widehat{A} = A + E$, $\gamma = |E|$ and $\epsilon = \gamma/\delta$ where $\epsilon \leq 1$. If for two columns i, j, $1 \leq i, j \leq n$ the following statements are true where x stands for either i or j: 1) $|e_x| \leq |a_x|$ 2) $|a_x'| = \Theta(|a_x|)$ 3) $|\widehat{a}_x'| = \Theta(|\widehat{a}_x|)$ 4) $|U_k^T e_x| = O(\epsilon|\widehat{a}_x|)$ then $\widehat{a}_i'$ and $\widehat{a}_j'$ are $O(\epsilon)$-skewed with respect to A.*

In the above asymptotic notation, we are treating $|a_x|, \delta, \gamma$, etc. as functions which vary m and n the

Table 4: Angles in term space versus "LSI space"

Same Topic	average	standard deviation
Original Space:	1.09	.079
LSI Space:	.0177	.0374
Different Topics	average	standard deviation
Original Space:	1.57	.00791
LSI Space:	1.55	.153

dimensions of the array. For example, assumption 2) is true if $|a'_x| = \sqrt{n}$ and $|a_x| = c\sqrt{n}$ for any constant c. Although it's not proven in [5], it's not too much harder to show that even for documents i and j whose angle in topic space is not 0 or $\pi/2$, the topic space angle in the perturbed matrix is approximately the same.

3.2 Experimental results

In an effort to explain why LSI works well on documents, a probabilistic document generation procedure is proposed in [5] Assume we have a set of m *terms*, $\mathcal{T}$ and a set of *documents* where a document is defined to be a subset of $\mathcal{T}$. Define a *topic* to be a probability distribution over $\mathcal{T}$. Assume that there are exactly k topics and each document is a convex combination of these topics.

The document generation model in [5] is the following: For some $0 < \epsilon \leq 1$, 1) A document consists of only a single topic. 2) Each topic has a set of *core* terms that it assigns $1-\epsilon$ of its probability mass to. These core terms are mutually disjoint across topics. Intuitively this means that most of the probability mass of a topic is on unique terms. Such a corpus is defined to be ϵ separable.

A document is assumed to be generated by selecting a *single* topic, a length l uniformly at random over the positive integers and then selecting l terms from $\mathcal{T}$ according to the probability distribution of the given topic. We select m such documents and let A be a $m \times n$ matrix whose columns are these documents. In [5], it's proven using Theorem 2 that ϵ-separable corpi are $O(\epsilon)$-skewed.

To test these theoretical results, the following experiment was done. 1000 documents each 50 to 100 terms long were generated from a corpus model with 2000 terms and 20 topics. Each topic was assigned a disjoint set of 100 terms as its primary set and .95 of its probability distribution was distributed over these terms and the rest of the probability mass was distributed evenly over all the terms(so the corpus was .05-separable). Angles were measured between all pairs of documents in the original space and all pairs of documents in rank-20 LSI space. Table 3.2 reports these result - angle measurements are given in radians.

We note that we would expect the angle between documents to decrease somewhat moving from the original space to LSI space since we are moving to a space with lower dimension but in fact for documents on different topics, this angle does not decrease too much. The reduction in angle between documents on the same topic moving from the original space to LSI space is impressive.

3.3 Analysis of these Results

The theoretical results from [5, 6] give some insight into when and why LSI might work to preserve angles between documents in topic space in the presence of noise. The first assumption of the theoretical result is not particularly interesting. Certainly if $|e_x| > |a_x|$ for either $x = i$ or j, the error vector can perturb one of the documents so that we can't determine the original angle between a_i and a_j.

The second and third assumptions that $|a'_x| = \Theta(|a_x|)$ and $|\widehat{a}'_x| = \Theta(|\widehat{a}_x|)$ are more interesting. We can't guarantee that every column will meet the assumption that $|a'_x| = \Theta(|a_x|)$ however we can say something about the *average* column if we assume that σ_{k+1} is small as the following chain of inequalities shows:

$$\begin{aligned}
\textstyle\sum_{i=1}^{n}(|a_i| - |a'_i|)^2 &= \textstyle\sum_{i=1}^{n}(|a_i| - |U_k U_k^T a_i|)^2 \\
&\leq \textstyle\sum_{i=1}^{n}(|a_i - U_k U_k^T a_i|)^2 \\
&\leq |A - A_k|_F^2 \\
&= \textstyle\sum_{i=k+1}^{n}(\sigma_i(A))^2 \\
&\leq n\sigma_{k+1}(A)^2
\end{aligned}$$

where the first step follows since $|U_k^T x| = |U_k U_k^T x|$ for any x, the second step follows by the triangle inequality (Fact 7), and the third and fourth by Fact 1.

The above analysis implies that the *average* difference between a_x and a'_x is $\sigma_{k+1}(A)$. This suggests that if σ_{k+1} is smaller than all the column norms(or perhaps even most of them), the average column will meet the assumption. In other words, if σ_{k+1} is small with respect to the column norms and the column norms are reasonably distributed, we'd expect that the theorem would hold for most columns.

The fourth assumption that $|U_k^T e_x| = O(\epsilon|\widehat{a}_x|)$ is shown in [6] to be satisfied with high probability if the entries of the matrix E are independent, identically distributed random variables. The basic intuition behind this fact is that $|U_k^T e_x| = |U_k U_k^T e_x|$ where the right hand side is to the projection of e_x into a low dimensional space; it's unlikely that a random vector would have large norm in a low dimensional space. For example if we assume that the error vectors are generated by an adversary who has no knowledge of the space $U_k U_k^T$ projects onto, we'd expect that the norm of e_x would be much diminished by projecting into this space.

To conclude, the important intuitive requirements for LSI to eliminate the effects of an error matrix on document-document angles are 1) most of the matrix A is captured by the top k singular vector pairs (i.e. $\sigma_{k+1}(A)$ is small) 2) The error matrix is small compared to ($|E| \leq \delta$) 3) The error matrix is generated without information about the singular vectors ($|U_k^T e_x|$ is "small")

3.3.1 LSI in Practice

In practice, there are several ad hoc tricks used to improve the performance of LSI. The first of these is term weighting which involves going in by hand and changing the weights of different terms. Term weights can be changed globally through all documents - for example if we want to decrease the weight of frequently occurring terms like "the" and "and" or changed locally for only a single document.

Two more tricks used allow for quickly adding new document vectors into a SVD computation that's already been done. The first of these, "folding in", simply projects the new document vector into the "LSI Space" of the old matrix. "Folding in" is fast but we completely miss out on any possible enriching of the "topic space" by adding new documents since the topic space is guaranteed not to change. It also doesn't preserve orthogonality of the rows of the rightmost matrix. The second trick "SVD-Updating" does update the "topic space" when new documents are added and preserves orthogonality but takes longer. Details of this technique are given in [13]

Table 5: Example Ratio Rules Computation

$$\begin{array}{r} \textit{apples} \\ \textit{oranges} \\ \textit{milk} \\ \textit{cookies} \end{array} \begin{bmatrix} 2 & -1 & -1 \\ 2 & -1 & -1 \\ 0 & 2 & -2 \\ 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 \\ 0 & 2/\sqrt{3} \\ 0 & 1/\sqrt{3} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 \\ 0 & \sqrt{6} \end{bmatrix} \begin{bmatrix} 2/\sqrt{6} & -1/\sqrt{6} & -1/\sqrt{6} \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

There are a wide variety of areas to which LSI techniques have been applied. Two of those listed in [13] are discussed here - information retrieval and information filtering. Information retrieval is the problem of retrieving the most relevant documents for a given query discussed above - empirical results show that LSI is about 30% better on this problem then techniques that use standard term vector methods. It's also reported that across a wide variety of domains, the performance of LSI methods peak when the rank of the SVD (k) is around 70 to 100. Information filtering is the problem of determining over time for a particular user what sort of documents they might like. In this area, LSI techniques have a 12 to 23% increase in performance over term vector methods.

4 Ratio Rules

In this section, we describe a new application of SVD introduced in [9], which is very similar to LSI. Here we will use the top k left singular vectors of the data matrix to identify the *ratios* with which different row quantities occur. These ratios will help us 1) analyze the data 2) do outlier detection 3) predict values for new data.

A primary motivating problem for this section is "market basket analysis" where we have data on the dollar amount spent on each item by all customers that visited a store over a fixed period. The columns of the matrix correspond to customers and the rows to items. Intuitively, the ratio rules should give us the "ratios" with which items are bought. So for example, a ratio rule might be that the ratio of money spent on beer to money spent on peanuts is $2:1$ which if there are only two items in the database would correspond to a singular vector of the form $(2/\sqrt{5}, 1/\sqrt{5})$. If there are underlying ratios that govern buying habits of the customers then we'd expect these ratios to be the left singular vectors.

4.1 Computing the Ratio Rules

Let A be a $m \times n$ matrix with columns equal to customers, rows equal to items and cell $a_{i,j}$ equal to the amount of item i bought by customer j. Let $C(A)$ be the "centered" version of A which is computed by subtracting the average of each row of A from each cell in the row. We note that the rows of $C(A)$ all sum to 0. If we think of the columns of A as points in a multidimensional space, we can view $C(A)$ as an orthogonal translation of the axis of the space so that the origin becomes the centroid(vector sum divided by number of points) of all the points. This centering can also be thought of as a multiplication of A by a projection matrix which projects onto the space perpendicular to the all 1's vector. The purpose of the centering is to make it easier to find correlations that include cheap items as well as expensive ones. The ratio rules are just the top k left singular vectors of the matrix $C(A)$ for some fixed k.

Table 6: Using Ratio Rules for Prediction

$$customer = [1, ?, 4, ?]$$
$$prediction = [1, 1, 4, 2]$$

$$\begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 2/\sqrt{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{3} \end{bmatrix} \begin{bmatrix} x_1 \leftarrow \sqrt{2} \\ x_2 \leftarrow 2\sqrt{3} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

4.2 Using the Ratio Rules

Table 5 gives an example centered matrix for 3 customers and 4 items and its SVD with $k = 2$. The ratio rules are just the columns of the left matrix in the SVD. If we imagine that the items are apples, oranges, milk and cookies, then the first ratio tells us that apples and oranges are correlated 1:1 and the second rule tells us that milk and cookies are correlated 2:1.

Assume we are given a new customer vector for which some of the values are known and some are unknown. The ratio rules can be used to guess unknown values for new customers by 1)finding what linear combination of the ratio rules best best fits the known values 2) using this same linear combination of the ratio rules to predict the unknown values. Step 1) corresponds to finding the solution to a set of linear equations, a problem which can be exactly specified, under-specified or over-specified. There are well known techniques for the latter two cases which are described in more detail in [9].

Table 4.2 uses the Ratio Rules from Table 4.1 for predicting unknown values for a customer vector $c = [1, ?, 4, ?]$. Here the "centered"quantity of apples and milk is known but the quantity of oranges and cookies is unknown. The first calculation in the table gets values for x_1 and x_2 which give the coefficients for the linear combination of ratio rules best predicting the known values for c. This particular system of linear equations is exactly specified - the solution is $x_1 = \sqrt{2}$, $x_2 = 2\sqrt{3}$. In the next step, we plug in these values for x_1 and x_2 in a matrix computation where the rows are the unknown values of c and the columns are the ratio rules. We get our prediction that the "centered" amount of oranges and cookies bought is 1 and 2 respectively for this customer.

4.3 Empirical Evaluation of Ratio Rules

Uses for rule sets in data mining include 1) outlier detection 2) finding trends in the data and 3) prediction.

Three separate empirical results are presented in [9] for evaluating ratio rules. The first evaluates the run time of the algorithm, the second evaluates the ability of the rules to do outlier detection and finding trends and the third evaluates the ability of the rules to do prediction.

4.3.1 Run time

The run time of the algorithm for generating Ratio Rules was evaluated using synthetic data generated by the Quest Synthetic Data Generation Tool on matrices with 100 columns and a number of rows ranging from

0 to $100,000$. The expected run time is $100^2 m$ where m is the number of rows and in fact, the empirical results verify that the run time increases linearly with m. For $100,000$ rows, the run time is about 12 minutes. The time spent on the real data sets used in the experiments was much less than this.

4.3.2 Trends and Outlier Detection

The second empirical result is qualitative in nature and used the ratio rules and domain knowledge to find trends and do outlier detection on the set of basketball statistics. The first ratio rule in this domain gives a correlation between time played and points score and basically separates star players from bench warmers. The second ratio rule gives a negative correlation between points and both rebounds and time played. The argument is that this rule separates defensive from offensive players. Plotting the players along the two axis defined by these rules reveals Dennis Rodman (offensive) at one extreme and Michael Jordan(defensive) at the other. The results of applying ratio rules to this domain are interesting but suffer from the major weakness that they require a human with both significant domain knowledge *and* knowledge of linear algebra to obtain them.

4.3.3 Predictive Accuracy

Experiments for doing predictive accuracy were set up in the following way. A matrix A whose columns represent the points in "attribute space" of the database was split into two sub matrices B and C by randomly placing 90% of the columns of A in B and 10% in C. A matrix C_H was created from C by randomly hiding one value from each column of C. The ratio rules were computed for the B matrix and used to predict the hidden values of C_H. Given B and C_H, the prediction algorithm returned a matrix C' which had guesses for the hidden values. The *general error* of the predictive algorithm on C is defined to be $\frac{|C-C'|_F}{\sqrt{nm}}$ where m, n are the dimensions of C. The *relative error* is defined to be the general error of the Ratio Rules algorithm divided by the general error of a straw horse algorithm which always just guesses the row average of B for hidden values in C.

Relative error is reported on three data sets: 1)basketball statistics which is a 459×12 matrix 2) batting statistics (1574×17) 3) abalone measurements (4177×7). Relative Error is close to 20% for each of these 3 databases. The Relative Error is also fairly stable as the number of "holes" increase. This error is not compared to error of other types of machine learning algorithm so it's difficult to evaluate the ratio rules more generally.

4.4 Analysis of the Ratio Rules Technique

Ratio rules are well suited for predicting the values of new customers when data is time invariant and the correlation between items bought is linear because of the Fact 1. For example if flour and sugar occur with the ratio 2 : 1 in an often used recipe then the ratio rules will find this correlation in customer buying habits. However, if for example there are two items a and b and b is bought only when a certain threshold of a is bought then the ratio rules will not predict well. In general there's no reason to expect correlation between two items to be linear.

A further problem is that the rules really don't give "ratios" if there can be more than one rule with nonzero weights for the same item. For example, if one ratio rule weights milk and sugar 1 : 1 and another rule weights milk and cookies 1 : 1, customers who buy sugar, milk and cookies in the ratio 1 : 2 : 1 are linear combinations of these rules but obey neither of the two "ratios" given by the rules. It's important not to interpret the rules in the wrong way.

Three problems with using Ratio Rules for finding trends in the data are: 1) we would generally be interested in trends involving a constant number of items but the ratio rules would tend to include a number of items proportional to the total. 2) there may be important trends which will not be detected by the top k singular vectors 3) ratio rules do not lend themselves well to human comprehension (compared to say "association rules"). As an example of 2), if there is a strong correlation between two or three items that are bought by only a small fraction of the customers, the ratio rules would be unlikely to find this correlation.

5 Spectral Techniques for Discrete Clustering in Euclidean space

In this section, we introduce the problem of partitioning points in high dimensional Euclidean space into a constant number of clusters such that the sum of the squared distances of all the points to the centroid of their unique cluster is minimized - this is called the discrete clustering problem(DCP). A motivating application for this problem is clustering a collection of documents represented in term space into a fixed number of good clusters. In particular, this may be useful for a search engine which wished to partition the documents returned by a query into say 5 clusters. This is an alternative to an LSI-type method of clustering documents which would use distances between points in "topic space" rather than angles between vectors to do clustering.

We will give a polynomial time algorithm which gives a clustering where the sum of the distances squared of all points to their cluster centroids is no more than twice the optimal value. No assumptions at all are made about the points. Unfortunately the constants in the polynomial time algorithm given are quite large so the algorithm is at this point of more theoretical than practical interest.

Formally, DCP is the following problem: we are given a set $\mathcal{A}$ of n points in m dimensional space and a positive integer k and we want to find a set of k points, $\mathcal{B} = \{b_1, \ldots, b_k\}$ such that $f_{\mathcal{A}}(\mathcal{B}) = \sum_{i=1}^{n} [dist(a_i, \mathcal{B})^2]$ is minimized where $dist(v, \mathcal{B})$ is defined to be $min_{b \in \mathcal{B}} |v - b|$. This problem is NP-hard even for $k = 2$.

A key idea towards approximating the discrete clustering problem is that the cluster centers naturally define a partition of $\mathcal{A}$ into k sets if we put each point into set i iff it's closest to b_i. Another key idea is that if we are given a partition of the $\mathcal{A}$ points into k sets and we wish to choose a b_i for each partition so as to minimize the score of the DCP, then the best solution is to choose b_i equal to the centroid of the points in partition i.

This follows from the fact that if we have a set X of r points $X = \{x_1, \ldots x_r\}$ and any other point b then $\sum_{i=1}^{r} |x_i - b|^2 = \sum_{i=1}^{r} |x_i - \bar{x}|^2 + r|b - \bar{x}|^2$ where $\bar{x}$ is the centroid of the points in X. We can see the truth of this equality by translating the points so $\bar{x}$ is the origin and substituting dot products for squared norms on both sides of the equation. This fact implies that choosing cluster centers is equivalent to choosing a partition which minimizes square distances of each point to the centroid of points in its partition.

If we let the points be columns of a matrix A then since $|A|_F^2 = \sum_i |a_i|^2$, Fact 1 tells us that the space spanned by the columns of U_k is a k dimensional space which minimizes the sum of squared distances between columns of A and any such low dimension vector space.

We can use this fact to devise an algorithm of the following form: 1) compute the SVD of A 2) solve the DCP problem exactly for the points projected into this low dimensional space 3) return the cluster given in step 2) as the clusters for the points A. The basic idea behind this algorithm is to first find the vector space V which is closest to all the points in A and then let A_k be this set of points projected onto V. Since the points that are the columns of A_k are in a space with constant dimensions, there is a polynomial time algorithm (details in [8]) to find the best cluster centers for the points A_k; further a good clustering in V is a good clustering in the larger space. Below we present the proof of the approximation bound for the algorithm just described.

Theorem 3 *Let $\mathcal{O}$ be the set of k points which optimizes $f_{\mathcal{A}\mathcal{B}}$ over all sets $\mathcal{B}$ and let $\mathcal{C}$ be the set of points returned by the above algorithm then $f_{\mathcal{A}}(\mathcal{C}) \leq 2 f_{\mathcal{A}}(\mathcal{O})$.*

Proof:

Let $\mathcal{O} = \{o_1, ...o_k\}$ be the optimal cluster centers and $a_1, ...a_n$ be the points we want to cluster. Let A be the matrix whose columns are the points in $\mathcal{A}$, let $SVD_k(A) = U_k D_k V_k^T$ and for any vector v, let $v' = U_k U_k^T v$ (this is the projection of v onto $< U_k >$), let $\mathcal{O}' = \{o'_1, \ldots, o'_k\}$ and let O be the matrix whose columns are points in $\mathcal{O}$. We note that:

$f_{\mathcal{A}}(\mathcal{O}) = \sum_{i=1}^n dist(a_i, \mathcal{O}) \geq \sum_{i=1}^n (dist(a'_i, \mathcal{O}'))^2$

since projecting into a lower dimensional space can only reduce distances. We also note that:

$f_{\mathcal{A}}(\mathcal{O}) = \sum_{i=1}^n (dist(a_i, \mathcal{O})^2 \geq |a_i - OO^T a_i|^2 \geq \sum_{i=1}^n |a_i - a'_i|^2$

Where the first inequality holds since the distance of a point x to a set of points $\mathcal{P}$ is always greater than or equal to the distance of x to the space spanned by $\mathcal{P}$ (if we think of the points as vectors). The second inequality holds by Fact 1.

Putting these two equations together, we get that:

$2f_{\mathcal{A}}(\mathcal{O}) \geq \sum_{i=1}^n (dist(a'_i, \mathcal{O}')^2 + \sum_{i=1}^n |a_i - a'_i|^2$

Now we note that the vector $a'_i - o'_j$ (where o_j is the closest point in $\mathcal{O}$ to a'_i is orthogonal to $a_i - a'_i$ since the first vector exists in the subspace $< U_k >$ and the second vector is orthogonal to $< U_k >$. Then since if $v1 \perp v2$ we know that $|v1|^2 + |v2|^2 = |v1 + v2|^2$, we can say that:

$$\begin{aligned} 2f_{\mathcal{A}}(\mathcal{O}) &\geq \sum_{i=1}^n dist(a_i, \mathcal{O}') \\ 2f_{\mathcal{A}}(\mathcal{O}) &\geq \sum_{i=1}^n dist(a_i, C) \\ &= f_{\mathcal{A}}(C)\Box \end{aligned}$$

6 Speeding up the Singular Value Decomposition Algorithm

The success of the SVD on the previous four problems gives ample motivation for its use. In this section, we'll discuss ways to make its computation fast. Standard algorithms for computing the SVD take $O(mn^2)$ time where m, n are the dimensions of the matrix and n is the smaller dimension. For the case where we are interested in just the top k singular vectors where k is a constant, there are other methods which achieve better but super-linear runtimes. We do not discuss these methods here.

For most any data mining problem, the input size is huge so algorithms which run in near-linear time in the size of the input are required. For this reason, any use of spectral techniques for data mining problems

Table 7: Approximation Algorithms for SVD

Algorithm(Reference)	Approximation Guarantee	Running Time
SAMPLE1 [7]	$\lvert A-\bar{A}_k\rvert_F \leq \lvert A-A_k\rvert_F + \epsilon\lvert A\rvert_F$	$O(mn + k^{12}/\epsilon^9$
SAMPLE2 [8]	$\lvert A-\bar{A}_k\rvert_F^2 \leq \lvert A-A_k\rvert_F^2 + \epsilon\lvert A\rvert_F^2$	$O(mn + \frac{k^3}{\epsilon^6} + m\frac{k^2}{\epsilon^4})$
PROJECTION [5]	$\lvert A-\bar{A}_k\rvert_F^2 \leq \lvert A-A_{\lceil k/2\rceil}\rvert_F^2 + 2\epsilon\lvert A\rvert_F^2$	$O(mn(\log^2 n))$

Table 8: Algorithm SAMPLE2

Inputs to the algorithm: A,k,ϵ,δ, c

1. $p_i \leftarrow |a_i|^2/|A|_F^2\ \forall i, 1 \leq i \leq n$, $x \leftarrow 4k/(\epsilon^2\delta)$.
2. Sample with replacement x columns of A according to distribution p_i.
   Include the column $a_i/\sqrt{xp_i}$ in S if column a_i is sampled.
3. Compute the matrix of left singular vectors of S, U_S, and return $\bar{A}_k = (U_S {U_S}^T)A$.

will have to deal with the problem of speeding up the computation of the SVD. The most common empirical techniques used for speeding up SVD include 1) computing the SVD only on a subset of the original matrix that matches some criteria 2) randomly sampling the columns of the matrix with uniform probability.

In this section, we look at several linear or near linear time algorithms for computing an approximation to the SVD that have guaranteed approximation bounds.

6.1 Theoretical Results

Three major theoretical results in approximating the SVD are presented Table 7. In each case, the algorithms return approximations to the top k singular vector triples and the approximation bounds are given for the rank-k matrix $\bar{A}_k$ which we get from projecting A onto the space spanned by the approximated singular vectors. The approximation matrices in the first two cases are compared against the optimal approximation matrix of the same rank but the third bound compares against the optimal matrix of rank $\lceil k/2\rceil$. All of the approximation results hold with probability $1-o(1)$. The run times reported assume $m \leq n$.

The first two results are from algorithms that randomly sample the columns of the matrix and the third result is from an algorithm that projects the matrix onto a low rank random subspace. We note that the second result is always better than the third in both approximation accuracy and running time. 7.

The algorithm for the second result in Table 7 is presented in Table 8. The inputs to the algorithm are A, a $m \times n$ matrix, k the rank of the output matrix, ϵ the error bound between 0 and 1 and δ the tolerated probability of error.

By way of intuition, we show below that for any unit vector u, $E(|u^T S|^2) = |u^T A|^2$. This fact suggests that in particular, the singular vectors of A will expand as much in S as they did in A relative to other vectors so as long as the variance of $|u^T S|^2$ is small, we'd expect the eigenvectors to be the same.

If we use any sampling distribution and divide columns in S by $\sqrt{xp_i}$, we'll get an unbiased sampler in the above sense. However in [11], they state that there is an as yet unpublished proof that the probability distribution given above for $c = 1$ is optimal in the sense that it gives the lowest variance among all unbiased

Table 9: Experimental Results for Sampling Algorithms

Experiment 1: m varies, $n = 5,000$, $k = 40$, $s = 100$		Experiment 2: $m = 115,000$ $n = 78,000$, $k = 900$, $s = 10,000$	
	Approximation Error		Approximation Error
Weighted Sampling:	.08	Weighted Sampling:	.06
Exact SVD:	.07	Uniform Sampling:	.09

Experiment 3: m, n, k, s as in Experiment 2		
	Precision(All Queries)	Precision(Special Queries)
Exact SVD	.187	.280
Weighted Sampling	.170	.266
Uniform Sampling	.185	.235
Baseline	.213	.166

samplers for the random variable $|u^T S|^2$. The intuition behind this probability distribution is that it samples the more important columns of A more frequently and these weightier columns will determine much of the weight of $|u^T A|$

$$\begin{aligned} E(|u^T S|^2) &= \sum_{j=1}^{x} E((u^T s_j)^2) \\ &= \sum_{j=1}^{x} \sum_{i=1}^{n} p_i (\tfrac{u_i \cdot a_i}{\sqrt{p_i x}})^2 \\ &= \sum_{i=1}^{n} (u_i \cdot a_i)^2 \\ &= |u^T A|^2 \end{aligned}$$

where the first step holds by linearity of expectation.

The proof of the approximation bound for algorithm SAMPLE2 is given in the appendix.

6.2 Empirical Results for Sampling Algorithms

The approximation guarantee for the sampling based SVD algorithm given in the last section is not good. In particular, it doesn't seem amenable to plugging into other proofs for algorithms which use the slow SVD algorithm to obtain faster algorithms which are provably nearly as good. In this section we present some very positive empirical results for this sampling algorithm which suggest that it does much better than the theoretical guarantee.

Experimental results from [11, 1] for the sampling algorithm of the last section are presented in table 6.2. In this table, m is the number of rows of the matrix, n is the number of columns, k is the rank of the matrix and s is the number of samples used by the sampling algorithm.

Approximation Error in Experiments 1 and 2 is defined to be $1 - \frac{|A-X|_f^2}{|A|_F^2}$ where A is the original matrix and X is the approximation to A. The number of rows in the matrix was not reported in Experiment 1 and approximation Error for Exact SVD was not reported Experiment 2. In these experiments, "Weighted Sampling" is the SAMPLE2 algorithm and "Uniform Sampling" is the more common sampling method of just sampling all columns of the matrix with equal probability. In both of these experiments, the sampling algorithms were run 10 times and the numbers reported are the averages over these runs.

A column of the matrix approximated in Experiment 1 is a word count vector for the first several lines of a document on the web. $5,000$ documents were collected for each of 10 different web search queries. The number of rows in the matrix is equal to the number of unique terms in the collection of documents and so varies from run to run.

The average Approximation Error for the queries is reported and it's seen that "Weighted Sampling" is comparable in error to the exact k rank SVD. "Weighted Sampling" always ran at least 20 times faster than the Exact SVD. Prima Facie evidence also suggest that the singular vector "clusters" returned by the sampling algorithm correspond well to different topics in the query.

The columns of the matrix in Experiment 2 and 3 are word count vectors for documents from the TREC database (a collection of AP news articles) [14]. The matrix is very sparse with only .13% of the matrix nonzero. Experiment 2 compares approximation error for this new domain on both "Weighted Sampling" and "Uniform Sampling". As we'd expect, "Weighted Sampling" outperforms "Uniform Sampling". The approximation error results for "Weighted Sampling" are much better than we'd expect from the theoretical bounds.

In Experiment 3, the approximation results for SVD are used in the problem of identifying relevant documents for a particular query. The database of news articles used included information on the relevance of all documents in the collection to 49 different queries. The task of the algorithms was to return a fixed size collection(the number is unreported in [11] but [14] puts it at 200) of documents for each query of which a large percentage were actually relevant.

The precision of a query is the percentage the documents returned that are actually relevant. The SVD using query retrieval algorithms for this experiment are unreported but are assumed to use the cosine distance in rank 900 "SVD space". Precision results for a non-SVD based algorithm(SMART) are also reported in the table as "baseline" for comparison. The special queries are a subset of 15 queries for which the SVD based algorithms out perform the non-SVD based algorithm. 10 independent runs were done for each query. Average Precision over these runs is reported for the sampling based algorithms.

Surprisingly Uniform Sampling outperforms Weighted Sampling on this task despite the fact that it produces a worse approximation to the original matrix. However, for the special queries for which the LSI technique seems to be particularly effective, Weighted Sampling outperforms Uniform Sampling.

7 Conclusion

7.1 Analysis of Theoretical Results on Spectral Methods

The theoretical successes in the area of applying spectral techniques to data mining and information retrieval include: 1) Development of a core algorithm and analytic tools are applicable to a wide variety of practical problems 2)Generally good correlation between theoretical and empirical results (we must be making reasonable assumptions)

Limitations of the theoretical results include 1) the fact that there is no general model telling us what types of problems spectral techniques can be applied to 2) A bad tradeoff between speed of SVD and loss in accuracy (must sacrifice a lot of accuracy to get linear times) 3) lack of knowledge about which assumptions are strictly necessary for those algorithms that have theoretical guarantees (for example, do we need to know, k, the best rank to cut off the SVD.)

Empirical successes of spectral methods include: 1)the wide success of one basic algorithm on many kinds of problems [13] 2) the general robustness of spectral techniques to noise and general tweaking of the algorithm(i.e. term weighting, folding in), 3) Empirical success in creating fast algorithms for SVD that use sampling but don't reduce accuracy too much.

Limitations of the empirical results include: 1) lack of comparison of spectral techniques with other data mining techniques (e.g. association rules, machine learning algorithms) 2) the fact that spectral based algorithms are still slower than many other algorithms for data mining 3) the fact that eigenvectors and eigenvalues are not as easy for a human to interpret as results from other algorithms that can be used for data mining (i.e. association rules, decision trees).

8 Open Problems

8.1 Separating pages on the web

The link analysis techniques of [2] don't give a very satisfying way of combining both link structure of web pages and the terms contained in a document. In general, we'd like to be able to combine the link and term information about documents to better solve problems in both LSI and link analysis. To this end, we present a preliminary idea here which uses both terms and links to try to split a collection of documents into two categories which are sparsely connected and don't contain the same terms.

Assume there is a collection of web pages and terms and we are concerned with classifying the pages and terms by how much they belong to one of two "opposite" classes C and D. For a given page or term x, let $C(x), D(x)$ be real numbers between -1 and 1 which gives the degree to which each page or term belongs to classification C and D respectively. The two classes C and D are opposite in the sense that: 1) for any page or term x, $C(x) = -D(x)$ 2) pages with high C classification do not contain terms, point to pages or contain pointers to them from pages with high D classification and vice versa.

As a motivating example, we may have a collection of pages on abortion and we wish to find the pages that are pro-life and pro-choice. A less extreme case may be that we have a collection of pages on automobiles and we wish to divide this set into say cars and trucks.

If a page contains a term or links to another pages, we say there is a connection between the page and that term or link. If we let the score of a connection between two objects(pages or terms) x and y be $C(x)C(y)$ (which equals $D(x)D(y)$) then if we want to partition these pages, we are interested in assigning $C(x), D(x)$ to all pages x such that 1) $\sum_x C(x) = \sum_x D(x)$ and 2) maximizing the score of the partition which equals the sum of the scores of all of the links.

Let A be the adjacency matrix of the graph of connections between pages and terms and let P be the projection matrix which projects onto the space orthogonal to the all 1's vector. We present the following basic fact without proof.

Fact 3 *The partition with maximal score satisfying the constraint that* $\sum_x C(x) = \sum_x D(x)$ *is the first singular vector pair of the matrix* PA.

8.2 Hierarchical Clustering

Very frequently when we are given a large collection of documents, the topical information about these documents can best be represented in a hierarchy. For example, hierarchical web directories are extremely useful for finding an web page appropriate to a given query. One promising open problem is whether we can use spectral techniques on terms and/or link structure to recover this underlying hierarchical structure.

In this section, we focus just on recovering this heirarchical topic structure. A key observation for this problem is that documents which share a parent topic tend to have a number of terms which they share in common. For example the topics "dogs" and "cats" would tend to share the terms "pet", "companion", "shelter"(and they'd also tend to share some link structure).

For a collection of documents and terms, we can say that the "important" terms in the collection correspond to high values for the left singular vector and the "important" documents correspond to high terms for the right singular vector if we use the conductance argument for defining "importance". If we assume that these "important" terms are the terms shared by the two or more subtopics in the collection of documents then the following algorithm is suggested for creating a hierarchy in a collection of n documents and m terms:

Let u_1, v_1 be the first left and right singular vectors of a document term matrix A. Let the documents associated with the top cn (for some constant c) values of u_1 be put in the parent topic. Let the terms associated with the top cm values of v_1 also be put in the parent topic. Now remove all of the documents and terms that have been classified from the collection, apply the separation algorithm given in the previous section to get two a partition of the remaining documents into child topics and then recursively call this procedure on the two child sets of documents and terms.

8.3 Ratio Rules

8.3.1 Prediction using Ratio Rules

In this problem, we are given as in the Ratio Rules section matrices B and C_H which are randomly generated from a customer matrix A by splitting the columns into B and C and then letting C_H be the matrix C with some hidden values. A prediction algorithm can be described as a function f which takes the matrix B and C_H and generates a prediction matrix C'. We are interested in finding a function f which minimizes $E(|C - f(B, C_H)|_F^2)$ over the random variables B and C. To motivate the problem, we can assume that the matrix B is obtained by exhaustive market research of say customer preferences and we wish to use that information to make predictions about the real world.

We assume the same model is used for generation of customers in the matrix B and the matrix C and that some values of C are hidden by an adversary which has no knowledge of the customer generation model. There are several plausible customer generation models we can assume for the generation of C. One of these is model1: assume there are k orthonormal vectors that define a "topic space" of the customers and that customers are generated first as random points in this space and then are perturbed by an error matrix in the larger space.

Conjecture 1 *Under model1, for some function f which uses SVD to do prediction, $E(|C - f(B, C_H)|_F^2 / |C|_F^2$ is $o(1)$ with high probability as the matrix A grows large and the size of B is a constant fraction of the size of A.*

8.3.2 Ratio Rule Recovery

For this problem, we are given a matrix A whose columns represent the amount purchased by each customer, assume that these columns are generated by a constant number of underlying rules and are interested in recovering these rules. One important open problem in this area is can we *prove* that the rules can be recovered under reasonable models of customer generation? The very simplest case here would be the case where the rules are indeed ratios and that no item occurs under any two rules

A richer type of rule for customer generation would be an arbitrary polynomial equation of small degree. For example let m be the amount of milk that a customer buys, a be the amount of apples and c be the amount of cookies then there may be a rule of the form $ma = 1$ or $m^2 + ac = 2$. We call rules of this form polynomial rules. For a model which generates the customers based on a fixed number of such polynomials(even in the presence of noise), we present the following conjecture

Conjecture 2 *The underlying polynomial rules used to generate A under model1 are in the space spanned by a constant number of the* smallest *left singular vectors of an appropriate matrix(described below).*

To get some basic intuition about why this might be true, let's first consider the case when the rules are linear - say of the form $2a + b = c$. We can write all of these rules as equations that equal 0 and represent them as vectors - so for the case above, if a,b,c occur in the first three columns then the vector $(2, 1, -1)$ corresponds to the example rule. Now since most of the customers obey this rule, we'd expect that the dot product of most of the customers with the rule vector would equal 0 and so we would expect that the vector form of the rule would be in the space spanned by a constant number of the *smallest* left singular vectors of A(since most of the points don't have a large component in its direction).

On the other hand if we are given a collection of the smallest left singular vectors of the matrix A, we know that the dot product of most customer vectors with them is close to 0 so we'd expect any vector in their span to represent a linear equation that is usually satisfied.

Now if we are interested in higher degree polynomials (and constants), we need to create a different kind of matrix. Assume for concreteness that all of the polynomial rules are of degree ≤ 2. Then let A' be the matrix A with rows added in the following way: for every pair of rows of x and y of A, add the row z to A' where $z[i] = x[i] * y[i]$ for all i, for every row x of A, add the row z to A' where $z[i] = x[i]^2$ and finally add k rows of all 1's to A'. Now we'd expect the above analysis to continue to hold true if we establish the obvious correspondence between vectors and polynomials.

Other problems include: Can we recover the polynomial rules in the presence of noise? What if the rules are only obeyed by most of the customers?

8.4 Speeding Up SVD

There are not as many promising open problems in this area as in the others - improving the approximation results for the SVD for general matrices seems difficult. One possible direction is to come up with approximation algorithms for just finding say the top c *components* of the top singular vector. This would be useful for example in an algorithm to return the top c authoritative sites for a user query in real time. Other possibilities include making certain assumptions about the matrix - i.e. the column points in "k-rank SVD space" are well spread out and devising a fast, accurate SVD algorithm for these types of matrices.

References

[1] R. Kannan and S. Vempala. Real Time clustering and ranking of documents on the web Unpublished manuscript, 1999.

[2] J. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pp. 668-677, 1998

[3] K. Bharat and M. Henzinger Improved Algorithms for Topic Distillation in a Hyperlinked Environment

[4] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, S. Rajagopalan Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text Proc. of 7th World Wide Web Conference, pp. 65-74

[5] C. Papadiitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent Semantic Indexing: A Probabilistic Analysis. In *Proceedings of ACM Symposium on Principles of Database Systems.* 1997.

[6] Y. Azar, A. Fiat, A. Karlin, F. McSherry, J. Saia Data Mining through Spectral Analysis Submitted to FOCS 2001.

[7] A. Frieze, R. Kannan and S. Vempala. Fast Monte-Carlo Algorithms for finding low-rank approximations In *Proc. 39th IEEE Symp. on Foundations of Computer Science*, 1998.

[8] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay. Clustering in large graphs and matrices In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[9] F. Korn, A. Labrinidis, Y. Kotidis and C. Faloutsos. Ratio Rules: A New Paradigm for Fast, Quantifiable Data Mining In *VLDB* New York, NY, 1998.

[10] G. Golub and C. Van Loan. Matrix Computations, Johns Hopkins University Press, 1989.

[11] F. Jiang, R. Kannan, M. Littman, S. Vempala Efficient Singular Value Decomposition via Improved Document Sampling *Technical Report Duke University Dept. of Computer Science*, 1999.

[12] S. Guattery, G. Miller On the Performance of Spectral Graph Partitioning Methods In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1995

[13] M. Berry, S. Dumais and G. O'Brien Using linear algebra for intelligent information retrieval. In *SIAM Review 37(4)*, pp.573–595, 1995.

[14] S. Dumais LSI meets TREC: A status report In *Proceedings of the First Text REtrieval Conference*, 1992

[15] M. Berry, Z. Drmac and E. Jessup Matrices, Vector Spaces and Information Retrieval In *SIAM Review Volume 41, Number 2* pp. 335-362, 1999.

[16] A. Arning, R. Agrawal, P. Raghavan. A Linear Method for Deviation Detection in Large Databases

[17] M. Berry and G. Linoff. Data Mining Techniques for Marketing, Sales and Customer Support, Wiley Computer Publishing, 1997.

[18] R. Agrawal, T. Imielinski, A. Swami Mining Association Rules Between Sets of Items in Large Databases' In *Proc. of the 1993 ACM SIGMOD Conference* pp 207-216, 1993.

A Basic Linear Algebra Facts

We will be dealing with vector space over the real numbers only. Most of these facts and proofs can be found in a standard linear algebra text such as [10].

A.1 Vectors and Vector Norms

Definition 2 *For a vector x with length n, let $x[i]$ be the i-th component of x. We use $|x|$ to denote the 2-norm of x where $|x| = \sqrt{\sum_{i=1}^{n} x[i]^2}$. We use $|x|_1$ to denote the 1-norm of x where $|x|_1 = \sum_{i=1}^{n} |x[i]|$ where*

Fact 4 *The cosine of the angle between two vectors x and x is $\frac{x \cdot y}{|x||y|}$*

Fact 5 $x \cdot y \leq |x||y|$

For Euclidean space, this follows from the Fact 4 and the fact that the cosine function is always between -1 and 1

Fact 6 $|x + y| \leq |x| + |y|$

This follows by squaring both sides of the inequality, substituting $v^T v$ for $|v|^2$ for all vectors v and then using Fact 5

Fact 7 $|x - y| \geq ||x| - |y||$

This follows by using the same procedure as in Fact 6

Fact 8 $|x|_1 \leq \sqrt{n}|x|$

A.2 Matrix Norms

Definition 3 *For any $m \times n$ matrix A the standard matrix norm is denoted $|A|$ and $|A| = max_{|x|=1}(|Ax|)$ where x is a vector. The Froebenius norm of a matrix is denoted $|A|_F$ where $|A|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j}^2}$*

The following facts are true for any matrix A and vector x:

Fact 9 $|Ax| \leq |A||x|$

Fact 10 $|A|_F^2 = \sum_{i=1}^{n} |a_i|^2$

Fact 11 $|a_i| \leq |A|$

Fact 12 $|A|_F^2 = trace(A^T A)$ *this is direct from Fact 10*

Fact 13 $|A|_F = |A^T|_F$ *and* $|A| = |A^T|$

A.3 Orthonormal Matrices

Definition 4 *A matrix Q is said to be* Orthonormal *if $Q^T Q = I$.*

The following facts hold for any orthonormal matrix Q, any matrix A and any vectors x and y:

Fact 14 *The Froebenius norm is orthonormally invariant i.e.* $|QA|_F = |AQ|_F = |A|_F$

Fact 15 *The 2-norm is orthonormally invariant i.e.* $|QA| = |AQ| = |A|$

Fact 16 $Ax \cdot Qy = x \cdot y$

Fact 17 *The angle between Qx and Qy equals the angle between x and y*

A.4 Projections

Fact 18 *A matrix P is a projection matrix if it has the following two properties: 1) $P^T = P$ 2) $P^2 = P$.*

Fact 19 *If Q is a matrix whose columns are orthonormal then QQ^T is a projection matrix which projects onto the subspace spanned by the columns of Q. We note that the above is true even if Q is padded with* 0 *vectors.*

Fact 20 *If P is a projection matrix then so is $I - P$.*

Fact 21 *If the vector x is in the column space of a projection matrix P, then $Px = x$.*

A.5 Singular Vectors and Eigenvectors

The following facts are true for any $m \times n$ real matrix A with rank r.

Fact 22 *We can write $A = UDV^T$ where U,V are mxm and nxn orthonormal matrices respectively and D is a diagonal matrix with real valued, nonincreasing entries $\sigma_1, \ldots \sigma_r$ and $u_i^T A = \sigma_i v_i$, $Av_i = \sigma_i u_i$. $U = V$ iff A is symmetric.*

Fact 23 *For a fixed k, let $A_k = SVD(A) = U_k D_k V_k^T$ be the k rank SVD then we will define the i^{th} column of A in "k-rank LSI space" as $U_k^T a_i$ which equals the i^{th} column of $D_k V_k^T$. The i^{th} column of A_k is $U_k U_k^T a_i$.*

Fact 24 *The matrix A_k minimizes $|A - B|$ over all matrices B that have rank $\leq k$ and $|A - A_k| = \sigma_{k+1}(A)$*

Proof:

It's straigtforward to show that $|A - A_k| = \sigma_{k+1}(A)$. Assume there is some matrix M which has rank $\leq k$ and that $|A - M| \leq \sigma_{k+1}(A)$. Then it must be the case that $Mu_i \neq 0$ for all $1 \leq i \leq k+1$ where u_i is the ith right singular vector of A.

But then since $dim(Range(M)) \leq k$, we know there exist α_i not all 0 such that $\sum_{i=1}^{k+1} \alpha_i M u_i = 0$ which means that $M \sum_{i=1}^{k+1} \alpha_i u_i = 0$. But then $(A - M) \sum_{i=1}^{k+1} \alpha_i u_i = A \sum_{i=1}^{k+1} \alpha_i u_i$ and since $\sum_{i=1}^{k+1} \alpha_i u_i$ is in the subspace spanned by the top $k+1$ singular vectors and all of these are positive, this vector must be expanded by $\geq \sigma_{k+1}$ which is a contradiction.

□

Fact 25 *The matrix A_k minimizes $|A - B|_F$ over all matrices B that have rank $\leq k$ and $|A - A_k|_F^2 = \sum_{i=k+1}^{r} (\sigma_i(A))^2$*

Fact 26 *Define for a symmetric matrix B, $\lambda_i(B)$ to be the i-th largest eigenvalue then $\sigma_i(A) = \sqrt{\lambda_i(AA^T)} = \sqrt{\lambda_i(A^T A)}$*

Fact 27 *For any symmetric matrix B, define $e_i(B)$ to be the i^{th} largest eigenvector of B and define $u_i(C)$, $v_i(C)$ to be the i^{th} left and right singular vectors of C for any matrix C then for the matrix A, $u_i(A) = e_i(AA^T)$ and $v_i(A) = e_i(A^T A)$*

Fact 28 $maxarg_{|x|=1} |Ax| = v_1(A)$ *and* $max_{|x|=1} |Ax| = \sigma_1(A)$

Fact 29 $maxarg_{|x|=1,|y|=1} x^T A y = u_1(A), v_1(A)$ *and* $max_{|x|=1,|y|=1} x^T A y = \sigma_1(A)$.

Fact 30 *If A has all entries nonnegative then $u_1(A)$ and $v_1(A)$ have all entries nonnegative.*

Fact 31 $|A|_F^2 = \sum_{i=1}^{n} \sigma^2(A)$

this is true since $|A|^2 = |U_A^T A V_A|^2 = |\Lambda|^2 = \sum_{i=1}^{n} \sigma^2(A)$ where U_A and V_A are the orthonormal matrices in the SVD of A.

B Proof of Link Analysis Theorem

Theorem 4 *As $i \rightarrow \infty$ $x_i \rightarrow u_1$ and $y_i \rightarrow v_1$ where u_1,v_1 are the left and right singular vectors of A. if $\sigma_1(A) > \sigma_2(A)$.*

Proof Sketch:

We give the proof that x_i converges to u_1. The proof for the convergence of y_i is similar.

The key observation is that the n dimensional vector of all 1's, call it x_0 can be written as

$$x_0 = \sum_{j=1}^{n} \alpha_j u_j$$

for some scalars $\alpha_1, \ldots \alpha_n$ since the left singular vectors span R^n. So if we want to just get the direction of x_i without the normalization, we can look at:

$$(AA^T)^i w = \lambda_1^i \alpha_1 u_1 + \lambda_2^i \alpha_2 u_2 + \cdots + \lambda_n^i \alpha_n u_n$$

We note that x_i is just the vector on the left hand side above divided by some constant in front which normalizes it. We will look at what happens if we divide the vector on the left hand side by $\lambda_1^n \alpha_1$ which is valid if the entries of AA^T are nonnegative since the first eigenvector of AA^T is then all positive and hence can not be orthogonal to u_0 and hence $\lambda_1^n \alpha_1 \neq 0$

$$\frac{(AA^T)^i w}{\lambda_1^i \alpha_1} = u_1 + (\lambda_2/\lambda_1)^i \frac{\alpha_2}{\alpha_1} u_2 + \cdots + (\lambda_n/\lambda_1)^i \frac{\alpha_n}{\alpha_1} u_n$$

We note that all of the coefficients in front of $u_2, \ldots u_n$ in the above converge to 0 as l grows large if the first eigenvalue is at all greater than the second eigenvalue. So the direction of the vector on the left hand side must converge to u_1 in the limit. $\Box$

C Proof of LSI Theorem

A key lemma is stated here without proof.

Lemma 1 *Let B and $\widehat{B} = B + E$ be $m \times n$ real matrices. Let*

$$\begin{aligned} B &= UDV^T \\ \widehat{B} &= \widehat{U}\widehat{D}\widehat{V} \end{aligned}$$

be the SVDs of B and $\widehat{B}$ respectively. Let $\delta = \sigma_k - \sigma_{k+1}$ where σ_i is the i^{th} singular value of B. If E is $o(\delta)$ then there is an orthonormal matrix R such that:

$$\begin{aligned} \widehat{V}_k &= V_k R + E_V \\ \widehat{U}_k &= U_k R + E_U \end{aligned}$$

where V_k, U_k, $\widehat{V}_k$, $\widehat{U}_k$ are the first k columns of the matrices V, U, $\widehat{U}$, $\widehat{V}$ respectively and E_V, E_U are matrices with norms that are $O(|E|/\delta)$

If for any matrix B, we define $< B >$ to be the vector space spanned by the columns of B, then this lemma just says that if the error matrix is sufficiently small, there exists some E_U such that $< \widehat{U}_k > = < U_k + E_U >$ where $|E_U|$ is a δ fraction of $|E|$.

The theorem is restated here:

Theorem 5 *Let A be a $m \times n$ matrix with SVD $U_k D_k V_k^T$ and let $\widehat{A} = A + E$, $\gamma = |E|$ and $\epsilon = \gamma/\delta$ where $\epsilon \leq 1$. If for two columns i, j, $1 \leq i, j \leq n$ the following statements are true where x stands for either i or j: 1) $|e_x| \leq |a_x|$ 2) $|a'_x| = \Theta(|a_x|)$ 3) $|\widehat{a}'_x| = \Theta(|\widehat{a}_x|)$ 4) $|U_k^T e_x| = O(\epsilon |\widehat{a}_x|)$ then $\widehat{a}'_i$ and $\widehat{a}'_j$ are $O(\epsilon)$-skewed with respect to A.*

We will let $A = UDV^T$ be the SVD of A and $\widehat{A} = \widehat{U}\widehat{D}\widehat{V}$ be the SVD of $\widehat{A}$ and will let U_k, $\widehat{U}_k$ be the first k columns of U and $\widehat{U}$.

We will need to prove several lemmas before giving the proof of this theorem.

The first lemma we will show is that for the matrix R given by Lemma 1, we have the following where x again stands for i or j.

Lemma 2

$$|\widehat{a}'_x - Ra'_x| = O(\epsilon|\widehat{a}'_x|)$$

and

$$|\widehat{a}'_x - Ra'_x| = O(\epsilon|a_x{}'|)$$

Proof:

$$\begin{aligned}
|\widehat{a}'_x - R^T a'_x| &= |\widehat{U}_k^T \widehat{a}_x - R^T U_k^T a_x| \\
&= |(\widehat{U}_k^T - R^T U_k^T)\widehat{a}_x + R^T U_k^T(\widehat{a}_x - a_x)| \\
&\leq |(\widehat{U}_k^T - R^T U_k^T)\widehat{a}_x| + |R^T U_k^T(\widehat{a}_x - a_x)| \\
&\leq |(\widehat{U}_k^T - R^T U_k^T)||\widehat{a}_x| + |R^T||U_k^T e_x)|
\end{aligned}$$

where the last two lines follow from the triangle inequality, the matrix-vector norm inequality and the fact that $\widehat{a}_x - a_x = e_x$. By our assumption $|U_k^T e_x| = O(\epsilon|\widehat{a}_x|)$ and by Lemma 1(if we take transposes),$|(\widehat{U}_k^T - R^T U_k^T)| \leq \gamma/\delta$ so we have:

$$\begin{aligned}
&\leq (\gamma/\delta)|\widehat{a}_x| + O(\epsilon|\widehat{a}_x| \\
&= O(\epsilon|\widehat{a}_i|) \\
&= O(\epsilon|\widehat{a}'_i|)
\end{aligned}$$

Where the last step follows by our assumption that $|\widehat{a}'_x| = \Theta(|\widehat{a}_x|)$

This establishes the first part of the above lemma, the second follows by a substitution and use of the triangle inequality:

$$\begin{aligned}
O(\epsilon|\widehat{a}_x|) &= O(\epsilon|a_x + e_x|) \\
&\leq O(\epsilon(|a_x| + |e_x|)) \\
&= O(\epsilon|a_x|) \\
&= O(\epsilon|a'_x|)
\end{aligned}$$

where the second to last step follows by the assumption that $|e_x| \leq |a_x|$ and the last step follows because of our assumption that $|a'_x| = \Theta(|a_x|)$.

□

The following lemma shows that the lengths of the original and perturbed columns in LSI space are relatively close.

Lemma 3

$$|\widehat{a}'_i| = |a'_i| \pm O(\epsilon|a'_i|)$$

$$|a'_i| = |\widehat{a}'_i| \pm O(\epsilon|\widehat{a}'_i|)$$

Proof:

We show how the first part of this fact can be established from Fact 7 and Lemma 2.

$$
\begin{aligned}
|\widehat{a}'_i - Ra'_i| &\geq ||\widehat{a}'_i| - |Ra'_i|| \\
O(\epsilon|a_i{}') &= ||\widehat{a}'_i| - |Ra'_i|| \\
|\widehat{a}'_i| &= |a'_i| \pm O(\epsilon|a'_i|)
\end{aligned}
$$

The second part of this lemma follows if we plug in the second value of $O(\epsilon|\widehat{a}'_i|)$ from Lemma 2 in the second step.

□

Now we will use the above two lemmas to show:

Lemma 4

$$|\widehat{a}'_i \cdot \widehat{a}'_j - a_i{}' \cdot a_j{}'| = O(\epsilon|\widehat{a}'_i||\widehat{a}'_j|)$$

Proof:

$$
\begin{aligned}
|\widehat{a}'_i \cdot \widehat{a}'_j - a_i{}' \cdot a_j{}'| &= |\widehat{a}'_i \cdot \widehat{a}'_j - R^T a_i{}' \cdot R^T a_j{}'| \\
&= |(\widehat{a}'_i - R^T a_i{}') \cdot \widehat{a}'_j + R^T a_i{}' \cdot \widehat{a}'_j - R^T a_j{}'| \\
&= |v_1 \cdot \widehat{a}'_j + v_2 \cdot R^T a_i{}'|
\end{aligned}
$$

where the first step follows by Fact 16 and v_1 and v_2 in the last step are vectors with norms $O(\epsilon|\widehat{a}'_i|$ and $O(\epsilon|\widehat{a}'_j|)$ respectively by Lemma 2. If we use the Cauchy-Schwartz inequality on this last step and the fact that $|x+y| \leq |x| + |y|$ when x and y are reals, we can continue:

$$
\begin{aligned}
|v_1 \cdot \widehat{a}'_j + v_2 \cdot R^T a_i{}'| &\leq |v_1 \cdot \widehat{a}'_j| + |v_2 \cdot R^T \widehat{a}'_i| \\
&\leq |v_1||\widehat{a}'_j| + |v_2||\widehat{a}'_i| \\
&= O(\epsilon|\widehat{a}'_i||\widehat{a}'_j|)
\end{aligned}
$$

□

Finally, we can prove the statement of the theorem:

If $a'_i \cdot a'_j = 0$ then $\widehat{a}'_i \cdot \widehat{a}'_j = O(\epsilon|\widehat{a}'_i||\widehat{a}'_j|)$
If $a'_i \cdot a'_j = |a_i{}'||a_j{}'|$ then $\widehat{a}'_i \cdot \widehat{a}'_j = (1 - O(\epsilon))|\widehat{a}'_i||\widehat{a}'_j|)$

Proof:

For the case where $a'_i \cdot a'_j = 0$, if we plug into Lemma 4, we have directly:

$$|\widehat{a}'_i \cdot \widehat{a}'_j| = O(\epsilon|\widehat{a}'_i||\widehat{a}'_j|)$$

For the case where $a'_i \cdot a'_j = |a_i{}'||a_j{}'|$, if we plug into Lemma 4, we get:

$$
\begin{aligned}
|\widehat{a}'_i \cdot \widehat{a}'_j - |a_i{}'||a_j{}'|| &= O(\epsilon|\widehat{a}'_i||\widehat{a}'_j|) \\
\widehat{a}'_i \cdot \widehat{a}'_j &= |a_i{}'||a_j{}'| \pm O(\epsilon|\widehat{a}'_i||\widehat{a}'_j|) \\
\widehat{a}'_i \cdot \widehat{a}'_j &= |\widehat{a}'_i||\widehat{a}'_j| \pm O(\epsilon|\widehat{a}'_i||\widehat{a}'_j|)
\end{aligned}
$$

Where the last step follows by Lemma 3.

□

D Proof of Fast SVD Lemma and Theorem

Lemma 5 (Good Sample) *If $p_1 \dots p_n$ is a probability distribution with which columns of A are sampled with replacement and for all $1 \le i \le n$*

$$p_i = \frac{|a_i|^2}{|A|_F^2}$$

and x samples according to this distribution are the columns of a matrix S, then for all $\Theta > 0$

$$Pr(|AA^T - SS^T|_F \ge \Theta |A|_F^2) \le \frac{1}{\Theta^2 x}$$

Proof:
We note that A is a m by n matrix and S is m by x and that the columns are documents in both matrices. We will first bound $|SS^T - AA^T|^2$ which is equal to $|AA^T - SS^T|^2$. At a high level, we will show that the mean of the u, v entry of SS^T is the u, v entry of AA^T and hence that $E(([SS^T]_{u,v} - [AA^T]_{u,v})^2)$ is the *variance* of the random variable which is entry u, v of SS^T and so $E(([SS^T]_{u,v} - [AA^T]_{u,v})^2) \le E(([SS^T]_{u,v})^2)$ which we show is relatively small. Finally we'll use Markov's inequality to get the desired bound.

First we show that $E([SS^T]_{u,v}) = [AA^T]_{u,v}$. We will liberally use linearity of expectation to show this.

$$\begin{array}{rcl} E([SS^T]_{u,v}) & = & \sum_{j=1}^{x} E(s_{u,j} s_{v,j}) \\ & = & \sum_{j=1}^{x} \sum_{i=1}^{n} p_i \frac{a_{u,i} a_{v,i}}{p_i x} \\ & = & \sum_{j=1}^{n} a_{u,i} a_{v,i} \\ & = & [AA^T]_{u,v} \end{array}$$

Now we recognize that $E(([SS^T]_{u,v} - [AA^T]_{u,v})^2) = variance([SS^T]_{u,v}) = E(([SS^T]_{u,v})^2) - [AA^T]_{u,v}$. So we have:

$$\begin{array}{rcl} E(([SS^T]_{u,v} - [AA^T]_{u,v})^2) & \le & E(([SS^T]_{u,v})^2) \\ & = & \sum_{j=1}^{x} E(s_{u,j} s_{v,j}) \\ & = & \frac{|A|_F^2}{x^2 x^2} \sum_{j=1}^{x} \sum_{i=1}^{n} \frac{a_{u,i}^2 a_{v,i}^2}{|a_i|^2} \\ & = & \frac{|A|_F^2}{x} \sum_{i=1}^{n} \frac{a_{u,i}^2 a_{v,i}^2}{|a_i|^2} \end{array}$$

where the third step follows by plugging in the probabilities.

Now we can use the result to calculate the expected norm of $SS^T - AA^T$ since:

$$\begin{array}{rcl} E(|SS^T - AA^T|^2) & = & \sum_{u=1}^{m} \sum_{v=1}^{n} E((SS_{u,v}^T - AA_{u,v}^T)^2) \\ & \le & \frac{|A|_F^2}{x} \sum_{i=1}^{n} \frac{1}{|a_i|^2} \sum_{u=1}^{m} \sum v = 1^n a_{u,i}^2 a_{v,i}^2 \\ & = & \frac{|A|_F^2}{x} \sum_{i=1}^{n} \frac{1}{|a_i|^2} (|a_i|^2 |a_i|^2) \\ & = & \frac{|A|_F^4}{x} \end{array}$$

Now the Markov inequality says that, $Pr(X \ge t) \le \frac{E(X)}{t}$ so we know that:

$$Pr(|AA^T - SS^T|^2 \ge \Theta^2 |A|_F^4) \le \frac{1}{x\Theta^2}$$

taking square roots of both sides of the inequality, we have that:

$$Pr(|AA^T - SS^T| \geq \Theta|A|_F^2) \leq \frac{1}{x\Theta^x}$$

□

Theorem 6 (Good Approximation) *For any positive real ϵ,*

$$|A - \bar{A}_k|_F^2 \leq |A - A_k|_F^2 + \epsilon|A|_F^2$$

with probability at least $1-\delta$ where x, the sample size in the algorithm, is $4k/(\epsilon^2\delta)$.

Proof:

Let $q_1 \ldots q_k$ be the eigenvectors of SS^T and let Q be a $m \times n$ matrix whose first k columns are the q_i and whose remaining columns are 0.

We first simplify the quantity we wish to calculate $|A - QQ^TA|$ by noting that QQ^T is a projection matrix and therefore so is $I - QQ^T$ by Fact 20. Let $P = I - QQ^T$ then we have:

$$\begin{aligned}
|A - QQ^TA|_F^2 &= |PA|_F^2 \\
&= trace((PA)^T(PA)) \\
&= trace(A^TPA) \\
&= trace(A^TA) - trace(A^TQQ^TA) \\
&= \textstyle\sum_{i=1}^n \sigma^2(A) - |A^Tq_i|^2
\end{aligned}$$

We note that since $|A - A_k|_F^2 = \sum_{i=k+1}^n \sigma^2(A)$, the statement of the theorem reduces to proving that:

$$\sum_{i=1}^k \sigma^2(A) - |A^Tq_i| \leq \epsilon|A|_F^2$$

In the remainder of the proof, we will show that the above quantity is less than or equal to a multiple of $|AA^T - SS^T|$. We'll do this by upperbounding two quantities by $|AA^T - SS^T|$ which two quantities when combined will give something comparable to $\sum_{i=1}^k \sigma^2(A) - |A^Tq_i|$

The first upperbound is:

$$\begin{aligned}
|AA^T - SS^T|_F^2 &\geq |Q^T(AA^T - SS^T)Q|^2 \\
&\geq \textstyle\sum_{i=1}^k (q_i^T(AA^T - SS^T)q_i)^2) \\
&= \textstyle\sum_{i=1}^k (|A^Tq_i|^2 - \sigma_i^2(S))^2
\end{aligned}$$

The first line holds because the Froebenius norm is orthonormally invariant and Q contains an orthonormal matrix. The formula on the second line is just the squares of the diagonal elements of the matrix in the first line.

The second upperbound is:

$$\begin{aligned}
|AA^T - SS^T|_F^2 &\geq \textstyle\sum_{i=1}^n (\sigma_i(AA^T) - \sigma_i(SS^T))^2 \\
&\geq \textstyle\sum_{i=1}^k (\sigma_i^2(S) - \sigma_i^2(A))^2
\end{aligned}$$

Where the first line is the Hoffman-Wielande inequility [10] which states that $|X - Y|^2 \geq \sum_{i=1}^n (\sigma_i(X) - \sigma_i(Y))^2$ for any matrices x and y with n singular values each.

So we have:

$$\begin{aligned}\sum_{i=1}^{k}(|A^T q_i|^2 - \sigma_i^2(S))^2 &\leq |AA^T - SS^T|_F^2 \\ \sum_{i=1}^{k}(\sigma_i^2(S) - \sigma_i^2(A))^2 &\leq |AA^T - SS^T|_F^2\end{aligned}$$

If we let $x_i = |A^T q_i|^2 - \sigma_i^2(S)$, $y_i = \sigma_i^2(S) - \sigma_i^2(A))^2$ and $c = |AA^T - SS^T|_F^2$ then by adding these two inequalities and multiplying by 2, we can rewrite this as: $\sum_{i=1}^{k} 2x_i^2 + 2y_i^2 \leq 4c$ which implies since $(a+b)^2 \leq 2a^2 + 2b^2$ for all real a,b that $\sum_{i=1}^{k}(x_i + y_i)^2 \leq 4c$. Expanding these values again gives the following chain of inequalities:

$$\begin{aligned}\sum_{i=1}^{k}(|A^T q_i|^2 - \sigma_i^2(A))^2 &\leq 4|AA^T - SS^T|_F^2 \\ \sqrt{\sum_{i=1}^{k}(|A^T q_i|^2 - \sigma_i^2(A))^2} &\leq 2|AA^T - SS^T|_F \\ \sum_{i=1}^{k}|A^T q_i|^2 - \sigma_i^2(A) &\leq 2\sqrt{k}|AA^T - SS^T|_F\end{aligned}$$

Where the last inequality holds since for any sequence of reals $x_1 \ldots x_k$, $\sum_{i=1}^{k} x_i \leq \sqrt{k}\sqrt{\sum_{i=1}^{k} x_i^2}$ (which is easy to see if we let a vector x contain the x_i values, a vector y contain k 1's and then apply the Cauchy-Schwartz inequality to x and y).

Starting with Lemma 5 , we get the following chain of inequalities for all $\Theta > 0$,

$$\begin{gathered}Pr(2\sqrt{k}|AA^T - SS^T| \geq 2\sqrt{k}\Theta|A|_F^2) \leq \frac{1}{\Theta^2 x} \\ Pr(\sum_{i=1}^{k}|A^T q_i|^2 - \sigma_i^2(A) \geq 2\sqrt{k}\Theta|A|_F^2) \leq \frac{1}{\Theta^2 x}\end{gathered}$$

If we plug in $\Theta = \epsilon/2\sqrt{k}$, we get that

$$Pr(\sum_{i=1}^{k}|A^T q_i|^2 - \sigma_i^2(A) \geq \epsilon|A|_F^2) \leq \delta$$

□