CS297 Report

Text Summarization Using Lanczos Algorithm

By

Youn sang, Kim

Advisor: Dr. Chris Pollett

Department of Computer Science

San José State University

Spring 2010

Table of Contents

Introduction

When browsing the Internet, one can easily get overwhelmed by the flood of text available. No one wants to waste time reading long texts on a web page. By summarizing a website, one can quickly read an accurate representation of its contents and save time by knowing what is worth reading and what is not. For this reason, an efficient text summarizer is needed that will provide a non-redundant extract from the original site. All the major search engines (Google, Yahoo, etc.) use automated text summarization and present condensed descriptions of the search results.

My project also deals with text summarization. Though many summarization methods exist, I will make use of the Lanczos algorithm for the computation of a few eigenvalues and eigenvectors of a large sparse matrix and SVD (Singular Value Decomposition), taking a high-dimensional set of data and reducing it to a lower-dimensional set of data. This makes it possible to identify the best approximation of the original text by using the information acquired from the Lanczos algorithm earlier.

For CS 297, I produced a total of four deliverables. My goal was to understand linear algebra concepts such as the Lanczos algorithm and SVD, implement them using Java, and produce a Java program that summarizes original texts. I will discuss how I approached each deliverable in greater detail, including the math concepts I used in each section. In the section named “Deliverable 1,” I will define matrix, matrix properties and operations, eigenvalues and eigenvectors, determinants, the Newton-Raphson method, and the calculation of SVD in the brute force approach. In “Deliverable 2,” I will talk about the Lanczos algorithm

and its implementation. In “Deliverable 3,” I will examine the QR algorithm for finding eigenvalues and the improvement to the previous implementation. In “Deliverable 4,” I will show how the Lanczos algorithm and SVD are related to the summarization of texts. Finally, I will give my conclusions and explore areas for future work.

Deliverable 1

Calculating SVD in the brute force approach

The main objective of Deliverable 1 was to understand SVD and implement it. In this deliverable, my program decomposed an original matrix into three matrices (an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V) via SVD, and verified the result by multiplying the three decomposed matrices together and reconstructing the original matrix. I call this the “brute force approach” because my program does not use any algorithms to quickly find eigenvalues and eigenvectors, but instead follows mathematical steps for solving them, as one would do on a piece of paper.

Since I deal with matrices throughout all deliverables, I need to define what a matrix is. A matrix is a table that contains data, consisting of rows and columns. A matrix can also be viewed as a collection of row vectors or column vectors. A vector is a sequence of numbers corresponding to measurements along one dimension. For example, for the matrix word by sentence, depending on choosing column vectors or row vectors, the dimensions will be words if we choose the former and sentences if we choose the latter. Common matrix operations include addition, subtraction, multiplication, and transposition.

To calculate SVD, it is essential to find eigenvalues and eigenvectors. If a nonzero vector satisfies the equation below, vector v is called an eigenvector and scalar λ is called an eigenvalue.

$A\vec{v}$ = $\lambda\vec{v}$ where A is square matrix

To compute eigenvalues and eigenvectors, we can rewrite the equation $A\vec{v}$ = $\lambda\vec{v}$ as $(A-\lambda I)\vec{v}$ = 0, where I is the n x n identity matrix. In order for a nonzero vector v to exist, it is necessary and sufficient that $A-\lambda I$ must not be invertible. Otherwise, if $A-\lambda I$ has an inverse,

$$(A-\lambda I)^{-1}(A-\lambda I)v = (A-\lambda I)^{-1}0$$
$$v = 0.$$

Since we want a nonzero vector v, the determinant of $A-\lambda I$ should equal 0, as the matrix is invertible if it has a nonzero determinant. A determinant is a function of a square matrix that reduces it to a special number, and is denoted as det(A) or |A|. Thus, if matrix A is a 2 x 2 matrix, then

|A| = $\begin{vmatrix} a & b \\ c & c \end{vmatrix} = ad - bc.$

To determine eigenvalues, we can simply find the roots of the associated characteristic equation given by det($A-\lambda I$)= 0. For example, if A is $\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$, then

det$\begin{bmatrix} 1-\lambda & 3 \\ 3 & 1-\lambda \end{bmatrix}$ = $(1-\lambda)(1-\lambda)-9=0$.

If we solve the characteristic equation above for the eigenvalues of A, they are λ_1= -2 and λ_2= 4. But since a matrix can be larger than 2 x 2, I used another technique to compute the determinants, which is expansion by minor. I found the

determinant of matrix A by $\sum_{i=1}^{k}(-1)^{i+j}a_{ij}M_{ij}$, where M_{ij} is minor of A, obtained by taking the determinant of A with row i and column j crossed out. In my program, this is done by a recursive call. For a 3 x 3 matrix, the above technique will produce:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = a_{11}\begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix} - a_{12}\begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix} + a_{13}\begin{bmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

Moreover, to find complex roots, I used the Newton-Raphson method, which is a fast way to compute roots of polynomial equations. However, the Newton-Raphson method usually works well only once an initial guess is reasonably close to the root. Otherwise, it may diverge.

To find eigenvectors v corresponding to each eigenvalue λ, we simply solve the system of linear equations given by $(A-\lambda I)$v = 0. But since I often needed to solve large homogeneous systems of linear equations where the constant term for each equation is zero, converting the matrix into upper triangular form via Gaussian Elimination and doing the back substitution was required.

SVD is based on a linear algebra theorem that a rectangular matrix A can be decomposed into the product of three matrices (an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V), and reconstructed by multiplying the three matrices together. The SVD theorem is usually presented as:

$A_{nxp} = U_{nxn}S_{nxp}V_{pxp}^{T}$ where $UU^{T} = U^{T}U = I$ and $V^{T}V = VV^{T} = I$

Calculating SVD consists of finding the eigenvalues and eigenvectors of AA^{T} and $A^{T}A$. The eigenvectors of AA^{T} make up the columns of U and those of $A^{T}A$ make up the columns of V. The singular values in S contain square roots of

eigenvalues from AA^T or A^TA and are placed along the diagonal of S in descending order.

For example, if we have a matrix A such that A = $\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$, then in order to find U,

We must start with AA^T. The transpose of A is $\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$. Thus,

AA^T = $\begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix}$. Next, we must find the eigenvalues and corresponding eigenvectors of AA^T. Suppose the eigenvalues are 16 and 4. Then for each eigenvalue λ_1 = 16 and λ_2 = 4, we must compute eigenvectors by solving $(A^TA - 16I)X_1 = 0$ and $(A^TA - 4I)X_2 = 0$. Once computed, we convert eigenvectors to unit vectors by normalizing their lengths. We construct U by placing vectors along its columns, keeping in mind that we must preserve the order in which the singular values were placed along the diagonal of S. Constructing S is done by placing the square roots of 16 and 4 along the diagonal of S in descending order. Constructing V is a bit tricky. If X is the eigenvector of AA^T with an eigenvalue of k, then

AA^T X = k X.

We can rewrite the above equation as $A^TAA^TX = A^T(kX)$ by multiplying both sides by A^T, which can be reparenthesized as $(A^TA)(A^TX) = k(A^TX)$. Thus, A^TX is an eigenvector of A^TA with an eigenvalue of k, which gives us V.

Deliverable 2

Implementing the Lanczos algorithm

In this section, I studied the Lanczos algorithm, which uses a single vector Lanczos recursion, and made a program in Java, which takes a symmetric matrix as an input and generates a tridiagonal matrix; I also tested it successfully on a 4x4 symmetric matrix. My main objective was to understand the Lanczos algorithm so I could implement it in SQL for CS 298. Learning and implementing the Lanczos algorithm is necessary because I deal with a large sparse matrix when summarizing texts. The Lanczos algorithm is used to calculate eigenvalues of a large sparse matrix quickly, thus greatly improving the speed of the calculation of SVD.

The Lanczos algorithm can determine the eigenvalues for a large sparse matrix by using Lanczos recursion, which converts original matrix A into tridiagonal matrix T through a finite number of orthogonal similarity transformations.

Suppose we have orthonormal vectors $q_1, q_2, q_3, \ldots\ldots q_k$ and let Q = $[q_1, q_2, \ldots . q_k]$.

Thus, $Q^T Q = I$.

Since we want to change A to a tridiagonal matrix T, we apply a similarity transformation.

Then T can be decomposed into $Q^T A Q$ [1].

We can rewrite the equation as AQ=QT by multiplying both sides by Q.

The jth column of the above equation can be viewed as:

$$\begin{bmatrix} & A & \end{bmatrix}\begin{bmatrix} & q_j & \end{bmatrix} = \begin{bmatrix} q_{j-1} & q_j & q_{j+1} \end{bmatrix}\begin{bmatrix} \cdot & \cdot & \cdot & & & \\ & \cdot & \cdot & \beta_{j-1} & & \\ & & \cdot & \alpha_j & \cdot & \\ & & & \beta_j & \cdot & \cdot \\ & & & & \cdot & \cdot & \cdot \end{bmatrix}$$

$$\mathrm{A}\, q_j = \beta_{j-1} q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \qquad \text{(i)}$$

If we rewrite the above equation by multiplying both sides by q_j^T,

$$q_j^T \mathrm{A}\, q_j = \beta_{j-1} q_j^T q_{j-1} + \alpha_j q_j^T q_j + \beta_j q_j^T q_{j+1}$$

$q_j^T q_{j-1}$ and $q_j^T q_{j+1}$ will be zero, and we get $\alpha_j = q_j^T \mathrm{A}\, q_j$.

If we rearrange (i), we can obtain β_j.

$$r_j = \beta_j q_{j+1} = \mathrm{A}\, q_j - \alpha_j q_j - \beta_{j-1} q_{j-1} \qquad \text{(ii)}$$

$$\| r_j \| = \beta_j (q_{j+1}^T q_{j+1}) = \beta_j$$

We can decide the next orthonormal vector by rearranging (ii),

$$q_{j+1} = r_j \,/\, \beta_j$$

continuing the recursion until j =n, where n is the size of A, T_n is an orthogonal similarity transformation of A and therefore has the same eigenvalues.

By summarizing the above, the Lanczos algorithm can be written as:

r_0 = initial vector

$\beta_0 = \| r_0 \|$

For j =1, 2, 3, ….n

$q_j = r_{j-1} / \beta_{j-1}$

$\mathrm{a} = \mathrm{A}\, q_j$

$\alpha_j = q_j^T \mathrm{a}$

r_j = a - $\beta_{j-1} q_{j-1}$ - $\alpha_j q_j$

$\beta_j = \|r_j\|$

After n number of Lanczos recursions, nxn tridiagonal matrix is generated. The eigenvalues for the tridiagonal matrix are approximate to those of the original matrix A.

Moreover, the eigenvectors of A can be found by multiplying the eigenvectors of T by the Lanczos vectors acquired from the recursion. I chose the Lanczos algorithm because the number of arithmetical operations required to generate a Lanczos matrix is proportional to the number of nonzero entries of A [2]. This saves running time for a large sparse matrix.

Deliverable 3

Implementing SVD using the Lanczos algorithm

The main objective of this deliverable was to implement SVD using a Lanczos algorithm. In this section, my program decomposed an original matrix into three matrices (an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V) via SVD, and verified the result by multiplying the three decomposed matrices together to reconstruct the original. The big difference between this and Deliverable 1 was that I transformed an original matrix into a tridiagonal matrix via Lanczos algorithm in the process of calculating SVD.

I further improved this process by using a QR algorithm on the tridiagonal matrix to find the eigenvalues, which was a technique developed in 1961 by John G.F. Francis and Vera N. Kublanovskaya. First, the matrix must be factored into a

product of an orthogonal matrix Q_1 and an upper triangular matrix R_1 with positive entries along the diagonal, using the Gram-Schmidt orthogonalization process, a method for converting a set of vectors into orthonormal vectors. Next, we multiply the two factors(R_1, Q_1)in reverse order. Thus, we have $A_2 = R_1 Q_1$. We then repeat the steps above until all entries below the subdiagonal are zero. After a sufficient number of iterations k, eigenvalues appear along its diagonal on $A_{k.}$.

The rest of the calculations for SVD are the same as in Deliverable 1. I also made some corrections on the Lanczos algorithm in my code. It turned out that even after a small number of iterations, I quickly lost orthogonality of the Lanczos vectors due to not being able to completely reduce them to tridiagonal form. During each Lanczos recursion, full reorthogonalization of the current vector was needed in relation to all previous vectors. The full reorthogonalization was carried out using a Gram-Schmidt process, which can be described as:

$$r_j = r_j - \sum_{k=1}^{j-1} \left(q_k^T r_j\right) q_k \qquad \text{j = Lanczos step}$$

Deliverable 4

Extracting summaries from original texts using the Lanczos algorithm and SVD

The main objective of Deliverable 4 was to use the Lanczos algorithm and SVD to put the theory into practice. It is very important to emphasize the role of SVD and the Lanczos algorithm in data reduction.

SVD is a method for reducing a high-dimensional set of data to a lower-dimensional set of data. It allows us to identify which data exhibit the most

variation by ordering the dimensions. It gives us the best approximation of the original data by simply ignoring other dimensions below certain thresholds to reduce the volume of content, while maintaining the main relationships present in the original data. For Deliverable 4 and my master project, we deal with a word by sentence matrix A, where A_{ij} represents the frequency of a particular word appearing in each sentence. Earlier, I mentioned that A can be decomposed into three matrices (U, S, and V^T). Hence, in our case, the word matrix U consists of one row vector for each word, the sentence matrix V^T consists of one column vector for each sentence, and the singular matrix S consists of single values along its diagonal, reflecting the importance of each dimension. To be specific, each number U_{ij} indicates how strongly related word i is to the topic or concept represented by semantic dimension j, while each number V_{ij} indicates how strongly related sentence i is to the topic represented by semantic dimension j. Each number, S_{ii} on diagonal entries of S, indicates the magnitude of the importance of the corresponding semantic dimension.

I am using the Lanczos algorithm because a word by sentence matrix is very sparse, meaning that it is populated primarily by zeros, because the same words seldom reappear in adjoining sentences. The Lanczos algorithm is the fastest method for solving eigenvalues on large sparse matrices.

Conclusion

The purpose of all my deliverables was to learn SVD and the Lanczos algorithm and to be able to implement them myself, even if the implementation required

another language.

I produced four deliverables in total, each of which helped me gain knowledge of various linear algebra theorems and apply them systematically. From Deliverable 1, I learned how to calculate SVD, eigenvalues, and eigenvectors. Deliverable 2 taught me how to convert a symmetric matrix into a tridiagonal matrix by implementing the Lanczos algorithm. From Deliverable 3, I learned to incorporate the Lanczos algorithm into calculating SVD and different techniques to make my implementation more stable. Deliverable 4 showed me how all of the previous deliverables could be put together to reach my final objective–text summarization. The deliverables I have completed in this semester have helped me implement SVD using the Lanczos algorithm.

For CS 298, I will extend the deliverables I developed in CS 297 to create a program that generates a text summary of a web site. Since I will deal with large data sets of words and sentences from different pages in a web site, it is natural that I will use a database. Thus I will create SQL scripts that will calculate SVD using the Lanczos algorithm, and will be able to extract a summary from a web site.

References

1. Golub, G. H. and C. F. Van Loan, Matrix Computations, The John Hopkins University Press, Baltimore, 1983.

2. Cullum, J., and Willoughby, R. (2002). *Lanczos Algorithms for Large Symmetric Eigenvalue Computations Vol.1: Theory*. SIAM, Philadelphia.