Matrix decompositions and Latent Semantic Indexing

In Chapter 7 we introduced the notion of a term-document matrix: an \( m \times n \) matrix \( M \), each of whose rows represents a term and each of whose columns represents a document in the collection. Even for a collection of modest size, the term-document matrix \( M \) is likely to have at least several tens of thousand of rows and columns. In this chapter we first develop a class of operations from linear algebra, known as matrix decomposition. We then use a special form of matrix decomposition to construct a low-rank approximation to the term-document matrix. We then examine the application of such low-rank approximations to indexing and retrieving documents, a technique referred to as latent semantic indexing. While latent semantic indexing has not been established as a significant force in scoring and ranking for information retrieval, it remains an intriguing approach to clustering in a number of domains including for collections of text documents. Understanding its full potential remains an area of active research.

Readers who do not require a refresher on linear algebra may skip Section 18.1.

18.1 Linear algebra review

We briefly review some necessary background in linear algebra. Let \( M \) be an \( m \times n \) matrix with real-valued entries; for a term-document matrix, all entries are in fact non-negative. The rank of a matrix is the number of linearly independent rows (or columns) in it; thus, \( \text{rank}(M) \leq \min\{m, n\} \). A square \( r \times r \) matrix all of whose off-diagonal entries are zero is called a diagonal matrix; its rank is equal to the number of non-zero diagonal entries. If all the diagonal entries of such a diagonal matrix are 1, it is called the identity matrix of dimension \( r \) and represented by \( I_r \).

For a square \( m \times m \) matrix \( M \), the values of \( \lambda \) satisfying

\[
M \vec{x} = \lambda \vec{x}.
\]
are called the eigenvalues of $M$. The $n$-vector $\vec{x}$ satisfying (18.1) for an eigenvalue $\lambda$ is the corresponding right eigenvector. The eigenvector corresponding to the eigenvalue of largest magnitude is called the principal eigenvector. In a similar fashion, the left eigenvectors of $M$ are the $m$-vectors $\vec{y}$ such that

$$\vec{y} M = \lambda \vec{y}. \quad (18.2)$$

The number of non-zero eigenvalues of $M$ is $\text{rank}(M)$.

**Exercise 18.1**
What is the rank of the $3 \times 3$ diagonal matrix below?

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}$$

**Exercise 18.2**
Show that $\lambda = 2$ is an eigenvalue of

$$M = \begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix}.$$ 

Find the corresponding eigenvector.

The eigenvalues of a matrix are found by solving the characteristic equation, which is obtained by rewriting (18.1) in the form $(M - \lambda I_n)\vec{x} = 0$. The eigenvalues of $M$ are then the solutions of $|M - \lambda I_m| = 0$, where $|S|$ denotes the determinant of a square matrix $S$. The equation $|M - \lambda I_m| = 0$ is an $m$th order polynomial equation in $\lambda$ and can have at most $m$ roots, which are the eigenvalues of $M$. These eigenvalues can in general be complex, even if all entries of $M$ are real.

We now examine some further properties of eigenvalues and eigenvectors, to set up the central idea of singular value decompositions in Section 18.2 below. First, we look at the relationship between matrix-vector multiplication and eigenvalues. Consider the matrix

$$S = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$ 

Clearly the matrix has rank 3, and therefore has 3 non-zero eigenvalues $\lambda_1 = 30$, $\lambda_2 = 20$ and $\lambda_3 = 1$, with the three corresponding eigenvectors

$$\vec{v}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{v}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ and } \vec{v}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Preliminary draft (c) 2007 Cambridge UP
For each of the eigenvectors, multiplication by $S$ acts as if we were multiplying the eigenvector by a multiple of the identity matrix; the multiple is different for each eigenvector. Now, consider an arbitrary vector, such as $\vec{v} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$. We may express $\vec{v}$ as a linear combination of the three eigenvectors of $S$:

$$\vec{x} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} = 2\vec{x}_1 + 4\vec{x}_2 + 6\vec{x}_3.$$ 

Suppose we multiply the arbitrary vector $\vec{v}$ by $S$:

$$S\vec{v} = S(2\vec{x}_1 + 4\vec{x}_2 + 6\vec{x}_3) = 2\lambda_1\vec{x}_1 + 4\lambda_2\vec{x}_2 + 6\lambda_3\vec{x}_3 = 60\vec{x}_1 + 80\vec{x}_2 + 6\vec{x}_3.$$ (18.3)

Even though $\vec{v}$ is an arbitrary vector, the effect of multiplication by $S$ is determined by the eigenvalues and eigenvectors of $S$. Furthermore, it is intuitively apparent from (18.3) that the product $S\vec{v}$ is relatively unaffected by terms arising from the small eigenvalues of $S$; in our example, since $\lambda_3 = 1$, the contribution of the third term on the right hand side of (18.3) is small. This suggests that the effect of small eigenvalues (and their eigenvectors) on a matrix-vector product is small. We will carry forward this intuition when studying matrix decompositions and low-rank approximations in Section 18.2. Before doing so, we examine the eigenvectors and eigenvalues of special forms of matrices that will be of particular interest to us.

For a symmetric matrix $S$, the eigenvectors corresponding to distinct eigenvalues are orthogonal. Further, if $S$ is both real and symmetric, the eigenvalues are all real. We say that $S$ is positive semidefinite if for any real-valued column vector $\vec{v}$, we have $\vec{v}^T S \vec{v} \geq 0$. Then, all the eigenvalues of $S$ are non-negative.

**Example 18.1:** Consider the real, symmetric matrix

$$S = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$ (18.4)

From the characteristic equation $|S - \lambda I| = 0$, we have the quadratic $(2 - \lambda)^2 - 1 = 0$, whose solutions yield the eigenvalues 3 and 1. The corresponding eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ are orthogonal.
18.1.1 Matrix decompositions

In this section we examine ways in which a square matrix can be factored into the product of matrices derived from its eigenvectors; we refer to this process as matrix decomposition. This will form the basis of our principal text-analysis technique in Section 18.3. We begin by proving two theorems on the decomposition of a square matrix into the product of three matrices of a special form. The first of these, Theorem 18.1, gives the basic factorization of a square real-valued matrix into three factors. The second, Theorem 18.2, applies to square symmetric matrices and is the basis of the singular value decomposition described in Theorem 18.3.

Theorem 18.1 (Matrix diagonalization theorem) Let $S$ be a square real-valued $m \times m$ matrix with $m$ linearly independent eigenvectors. Then there exists an eigen decomposition

$$S = U \Lambda U^{-1},$$

(18.5)

where the columns of $U$ are the eigenvectors of $S$ and $\Lambda$ is a diagonal matrix whose diagonal entries are the eigenvalues of $S$ in decreasing order

$$\begin{pmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_m
\end{pmatrix}, \lambda_i \geq \lambda_{i+1}.
$$

(18.6)

If the eigenvalues are distinct, then this decomposition is unique.

To understand how Theorem 18.1 works, we note that $U$ has the eigenvectors of $S$ as columns

$$U = (\vec{u}_1 \ \vec{u}_2 \ \cdots \ \vec{u}_m).$$

(18.7)

Then we have

$$SU = S (\vec{u}_1 \ \vec{u}_2 \ \cdots \ \vec{u}_m) = (\lambda_1 \vec{u}_1 \ \lambda_2 \vec{u}_2 \ \cdots \ \lambda_m \vec{u}_m) = (\vec{u}_1 \ \vec{u}_2 \ \cdots \ \vec{u}_m) \begin{pmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_m
\end{pmatrix}.$$ 

Thus, we have $SU = U \Lambda$, or $S = U \Lambda U^{-1}$.

Exercise 18.3

Compute the unique eigen decomposition of the $2 \times 2$ matrix in (18.4).
We next describe how a symmetric square matrix can be decomposed into the product of matrices derived from its eigenvectors. This will pave the way for our development of our main tool for text analysis, the singular value decomposition (Section 18.2).

**Theorem 18.2** (Symmetric diagonalization theorem) Let $S$ be a square, symmetric real-valued $m \times m$ matrix with $m$ linearly independent eigenvectors. Then there exists a symmetric eigen decomposition

$$S = Q\Lambda Q^T,$$

where the columns of $Q$ are the orthogonal and normalized (unit length) eigenvectors of $S$, and $\Lambda$ is the diagonal matrix whose entries are the eigenvalues of $S$. Further, all entries of $Q$ are real and we have $Q^{-1} = Q^T$.

We will build on this symmetric eigen decomposition to build low-rank approximations to term-document matrices.

## 18.2 Term-document matrices and singular value decompositions

The decompositions we have been studying thus far apply to square matrices. However, the matrix we are interested in is the $m \times n$ term-document matrix where (barring a rare coincidence) $m \neq n$. To this end we first describe an extension of the symmetric eigen decomposition known as the singular value decomposition. We then show in Section 18.3 how this can be used to construct an approximate version of $M$.

**Theorem 18.3** Let $r$ be the rank of the $m \times n$ matrix $M$. Then, there is a singular-value decomposition (SVD for short) of $M$ of the form

$$M = U\Sigma V^T,$$

where

1. $U$ is an $m \times m$ matrix whose columns are the orthogonal eigenvectors of $MM^T$;
2. $V$ is an $n \times n$ matrix whose columns are the orthogonal eigenvectors of $M^TM$, and $V^T$ its transpose;
3. The eigenvalues $\lambda_1, \ldots, \lambda_r$ of $MM^T$ are the same as the eigenvalues of $M^TM$;
4. For $1 \leq i \leq r$, let $\sigma_i = \sqrt{\lambda_i}$, with $\lambda_i \geq \lambda_{i+1}$. Then the $m \times n$ matrix $\Sigma$ is composed by setting $\Sigma_{ij} = \sigma_i$ for $1 \leq i \leq r$, and zero otherwise.

The values $\sigma_i$ are referred to as the singular values of $M$. 
$M = U \Sigma V^T$

Figure 18.1 Illustration of the singular-value decomposition. In this schematic illustration of (18.9), we see two cases illustrated. In the top half of the figure, we have a matrix $M$ for which $m > n$. The lower half illustrates the case $m < n$.

Example 18.2: We now illustrate the singular-value decomposition of a $3 \times 2$ matrix of rank 2; the singular values are $\Sigma_{11} = \sqrt{3}$ and $\Sigma_{22} = 1$.

\[
M = \begin{pmatrix}
1 & -1 \\
0 & 1 \\
1 & 0
\end{pmatrix} = \begin{pmatrix}
2/\sqrt{6} & 0 & 1/\sqrt{3} \\
-1/\sqrt{6} & 1/\sqrt{2} & 1/\sqrt{3} \\
1/\sqrt{6} & 1/\sqrt{2} & -1/\sqrt{3}
\end{pmatrix} \begin{pmatrix}
\sqrt{3} & 0 \\
0 & 1 \\
1/\sqrt{2} & 1/\sqrt{2}
\end{pmatrix} \begin{pmatrix}
1/\sqrt{2} & -1/\sqrt{2} \\
1/\sqrt{2} & 1/\sqrt{2}
\end{pmatrix}.
\]

As with the matrix decompositions defined in Section 18.1.1, the singular value decomposition of a matrix can be computed by a variety of algorithms, many of which have been publicly available software implementations; pointers to these are given in Section 18.4.

18.3 Low-rank approximations and latent semantic indexing

We next state a matrix approximation problem that at first seems to have little to do with information retrieval. We describe a solution to this matrix problem using singular-value decompositions, then develop its application to information retrieval.

Given an $m \times n$ matrix $M$ and a positive integer $k$, we wish to find an $m \times n$ matrix $M_k$ of rank $\leq k$, so as to minimize the Frobenius norm of the
matrix difference \( X = M - M_k \), defined to be

\[
||X||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}^2}.
\]

(18.11)

Thus, the Frobenius norm of \( X \) measures the discrepancy between \( M_k \) and \( M \); our goal is to find a matrix \( M_k \) that minimizes this discrepancy, while constraining \( M_k \) to have rank at most \( k \). If \( r \) is the rank of \( M \), clearly \( M_r = M \) and the Frobenius norm of the discrepancy is zero in this case. When \( k \) is far smaller than \( r \), we refer to \( M_k \) as a low-rank approximation.

The singular value decomposition can be used to solve this matrix approximation problem, then derive from it an application to approximating term-document matrices. We invoke the following three-step procedure to this end:

1. Given \( M \), construct its SVD in the form shown in (18.9); thus, \( M = U \Sigma V^T \).
2. Derive from \( \Sigma \) the matrix \( \Sigma_k \) formed by replacing by zeros the \( r - k \) smallest singular values on the diagonal of \( \Sigma \).
3. Compute and output \( M_k = U \Sigma_k V^T \) as the rank-\( k \) approximation to \( M \).

The rank of \( M_k \) is at most \( k \); this follows from the fact that \( \Sigma_k \) has at most \( k \) non-zero values. Next, we recall the intuition of the example in (18.3): the effect of small eigenvalues on matrix products is small. Thus, it seems plausible that replacing these small eigenvalues by zero will not substantially alter the product, leaving it “close” to \( M \). The following theorem due to Eckart and Young tells us that, in fact, this procedure yields the matrix of rank \( k \) with the lowest possible Frobenius error.

\[
\min_{X: \text{rank}(X) = k} ||M - X||_F = ||M - M_k||_F = \sigma_{k+1}.
\]

(18.12)

Recalling that the singular values are in decreasing order \( \sigma_1 \geq \sigma_2 \geq \cdots \), we learn from Theorem 18.4 that \( M_k \) is the best rank-\( k \) approximation to \( M \), incurring an error (measured by the Frobenius norm of \( M - M_k \)) equal to \( \sigma_{k+1} \). Thus, the larger \( k \) is the smaller this error (and in particular, for \( k = r \), the error is zero since \( \Sigma_r = \Sigma \) and thus \( M_r = M \)).

To derive further insight into why the process of truncating the smallest \( r - k + 1 \) singular values in \( \Sigma \) helps generate a rank-\( k \) approximation of low error, we examine the form of \( M_k \):

\[
M_k = U \Sigma_k V^T
\]

(18.13)
Figure 18.2 Illustration of low rank approximation using the singular-value decomposition. The dashed boxes indicate the matrix entries affected by “zeroing out” the smallest singular values.

\[ M_k = U \Sigma_k V^T \]

where \( \bar{u}_i \) and \( \bar{v}_i \) are the \( i \)th columns of \( U \) and \( V \), respectively. Thus, \( \bar{u}_i \bar{v}_i^T \) is a rank-1 matrix, so that we have just expressed \( M_k \) as the sum of \( k \) rank-1 matrices each weighted by a singular value. As \( i \) increases, the contribution of the rank-1 matrix \( \bar{u}_i \bar{v}_i^T \) is weighted by a sequence of shrinking singular values \( \sigma_i \).

Exercise 18.4
Compute a rank 1 approximation to the matrix \( M \) in Example 18.2, using the SVD as above. What is the Frobenius norm of the error of this approximation?

Exercise 18.5
Consider now the computation in Exercise 18.4. Following the schematic in Figure 18.2, notice that for a rank 1 approximation we have \( \sigma_1 \) being a scalar. Denote by \( \bar{u}_1 \) the first column of \( U \) and by \( \bar{v}_1 \) the first column of \( V \). Show that the rank-1 approximation to \( M \) can then be written as \( \bar{u}_1 \sigma_1 \bar{v}_1^T \).

In fact, Exercise 18.5 can be generalized to rank \( k \) approximations: we let \( U_k \) and \( V_k \) denote the matrix formed by retaining only the first \( k \) columns of \( U \) and \( V \), respectively. Thus \( U_k \) is an \( m \times k \) matrix while \( V_k^T \) is a \( k \times n \) matrix. Then, we have

\[ M_k = U_k \Sigma_k V_k^T, \]

Exercise 18.4
Compute a rank 1 approximation to the matrix \( M \) in Example 18.2, using the SVD as above. What is the Frobenius norm of the error of this approximation?

Exercise 18.5
Consider now the computation in Exercise 18.4. Following the schematic in Figure 18.2, notice that for a rank 1 approximation we have \( \sigma_1 \) being a scalar. Denote by \( \bar{u}_1 \) the first column of \( U \) and by \( \bar{v}_1 \) the first column of \( V \). Show that the rank-1 approximation to \( M \) can then be written as \( \bar{u}_1 \sigma_1 \bar{v}_1^T \).

In fact, Exercise 18.5 can be generalized to rank \( k \) approximations: we let \( U_k \) and \( V_k \) denote the matrix formed by retaining only the first \( k \) columns of \( U \) and \( V \), respectively. Thus \( U_k \) is an \( m \times k \) matrix while \( V_k^T \) is a \( k \times n \) matrix. Then, we have

\[ M_k = U_k \Sigma_k V_k^T, \]
18.3 Low-rank approximations and latent semantic indexing

where $\Sigma_k'$ is the square $k \times k$ submatrix of $\Sigma_k$ with the singular values $\sigma_1, \ldots, \sigma_k$ on the diagonal. The primary advantage of using (18.16) is to eliminate a lot of redundant columns in $U$ and $V$, thereby explicitly eliminating multiplication by columns that will contribute zero to the low-rank approximation.

Exercise 18.6
For the matrix $M$ in Example 18.2, write down both $\Sigma_2$ and $\Sigma_2'$.

Exercise 18.7
Under what conditions is $\Sigma_k = \Sigma_k'$?

Before discussing the approximation of a term-document matrix $M$ by one of lower rank, we first motivate such an approximation. Recall the vector space representation of documents and queries introduced in Chapter 7. This vector space representation enjoys a number of advantages including the uniform treatment of queries and documents as vectors, the induced score computation based on cosine similarity, the ability to weight different terms differently, and its extension beyond document retrieval to such applications as clustering and classification. The vector space representation suffers, however, from its inability to cope with two classic problems arising in natural languages: 

1. Synonymy refers to a case where two different words (say car and automobile) have the same meaning. Because the vector space representation fails to capture the relationship between synonymous terms such as car and automobile – according each a separate dimension in the vector space – we may have a situation where the computed similarity $q \cdot d$ between a query $q$ (say, car) and a document $d$ containing these terms underestimates the true similarity that a user would perceive. Polysemy on the other hand refers to the case where a term such as charge has multiple meanings, so that the computed similarity $q \cdot d$ overestimates the similarity that a user would perceive. Could we use the co-occurrences of terms (whether, for instance, charge occurs in a document containing steed versus in a document containing electron) to capture the latent semantic associations of terms and alleviate these problems?

Even for a collection of modest size, the term-document matrix $M$ is likely to have several tens of thousand of rows and columns, and a rank in the tens of thousands as well. In latent semantic indexing (generally abbreviated LSI), we use the SVD to construct a low-rank approximation $M_k$ to the term-document matrix, for a value of $k$ that is far smaller than the original rank of $M$. In the experimental work cited below, $k$ is generally chosen to be in the low hundreds. We thus map each row/column (respectively corresponding to a term/document) to a $k$-dimensional space; this space is defined by the $k$ principal eigenvectors (corresponding to the largest eigenvalues) of $MM^T$ and $M^TM$. Note that the matrix $M_k$ is itself still an $m \times n$ matrix, irrespective of $k$. 

LATENT SEMANTIC INDEXING
Next, we use the new $k$-dimensional LSI representation as we did the original representation— to compute similarities between vectors. A query vector $\vec{q}$ is mapped into its representation in the LSI space by the transformation

$$\vec{q}_k = \vec{q}^T U_k \Sigma_k^{-1}.$$  \hfill (18.17)

Now, we may use cosine similarities as in Chapter 7 to compute the similarity between a query and a document, or between two documents.

The fidelity of the approximation of $M_k$ to $M$ leads us to hope that the relative values of cosine similarities are preserved: if a query is close to a document in the original space, it remains relatively close in the $k$-dimensional space. But this in itself is not sufficiently interesting, especially given that the sparse query vector $\vec{q}$ turns into a dense query vector $\vec{q}_k$ in the low-dimensional space. This has a significant computational cost, when compared with the cost of processing $\vec{q}$ in its native form.

We may view the low-rank approximation of $M$ by $M_k$ as a constrained optimization problem: subject to the constraint that $M_k$ have rank at most $k$, we seek a representation of the terms and documents comprising $M$ with low Frobenius norm for the error $M - M_k$. When forced to squeeze the terms/documents down to a $k$-dimensional space, the SVD should bring together terms with similar co-occurrences. This intuition suggests, then, that not only should retrieval quality not suffer too much from the dimension reduction, but in fact may improve.

Experiments with LSI tend to consistently bear out the following conclusions:

- The computational cost of the SVD is significant; at the time of this writing, we know of no successful experiment with over one million docu-
ments. This has been the biggest obstacle to the widespread adoption to LSI.

- As we reduce $k$, recall tends to increase, as expected.
- Most surprisingly, a value of $k$ in the low hundreds actually increases precision on many query benchmarks. This appears to confirm that for a suitable value of $k$, LSI addresses some of the challenges of synonymy and polysemy.

The experiments also documented some modes where LSI failed to match the performance of more traditional indexes and query languages. Most notably (and perhaps obviously), LSI shares two basic drawbacks of vector space retrieval: there is no good way of expressing negations (find documents that contain german but not shepherd), or Boolean conditions.

18.4 References and further reading

Strang (1986) provides an excellent introductory overview of matrix decompositions including the singular value decomposition. Theorem 18.4 is due to Eckart and Young (1936). The connection between information retrieval and low-rank approximations of the term-document matrix was introduced in Deerwester et al. (1990), with a subsequent survey of results in Berry et al. (1995). Dumais (1993) and Dumais (1995) describe experiments on TREC benchmarks giving evidence that at least on some benchmarks, LSI can produce better precision and recall than standard vector-space retrieval. http://www.cs.utk.edu/berri/lisi++/ and http://lsi.argreenhouse.com/lsi/LSIpapers.html offer comprehensive pointers to the literature and software of LSI. Hofmann Hofmann (1999a;b) provides a probabilistic extension of the basic latent semantic indexing technique.

Exercise 18.8

Assume you have a set of documents each of which is in either English or in Spanish. You want to be able to support cross-language retrieval; in other words, users with some information need should be able to issue queries in either English or Spanish, and retrieve documents of either language satisfying the information need. The collection is given in Figure 18.4.

Figure 18.5 gives a glossary relating the Spanish and English words above for your own information. This glossary is NOT available to the retrieval system:

1. Construct below the appropriate term-document matrix $M$ to use for a collection consisting of the above documents. For simplicity, use raw term frequencies rather than normalized tf.idf weights. Make sure to clearly label the dimensions of your matrix.
2. Write down the matrices $U_2, \Sigma'_2$ and $V_2$ and from these derive the rank 2 approximation $M_2$.

3. State succinctly what the $(i, j)$ entry in the matrix $M^T M$ represents.

4. State succinctly what the $(i, j)$ entry in the matrix $M_2^T M_2$ represents, and why it differs from that in $M^T M$. 