18 Matrix decompositions and latent semantic indexing

On page 113, we introduced the notion of a *term-document matrix:* an $M \times N$ matrix C, 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 C is likely to have several tens of thousands of rows and columns. In Section 18.1.1, we first develop a class of operations from linear algebra, known as *matrix decomposition*. In Section 18.2, we use a special form of matrix decomposition to construct a *low-rank* approximation to the term-document matrix. In Section 18.3 we examine the application of such low-rank approximations to indexing and retrieving documents, a technique referred to as *latent semantic indexing*. Although latent semantic indexing has not been established as a significant force in scoring and ranking for information retrieval (IR), it remains an intriguing approach to clustering in a number of domains including for collections of text documents (Section 16.6, page 343). 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, although Example 18.1 is especially recommended as it highlights a property of eigenvalues that we exploit later in the chapter.

18.1 Linear algebra review

We briefly review some necessary background in linear algebra. Let C be an $M \times N$ matrix with real-valued entries; for a term–document matrix, all
RANK entries are in fact non-negative. The *rank* of a matrix is the number of linearly independent rows (or columns) in it; thus, $rank(C) \leq \min\{M, N\}$. 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 nonzero diagonal entries. If all r 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 C and a vector $\vec{x}$ that is not all zeros, the values of λ satisfying

$$C\vec{x} = \lambda\vec{x} \tag{18.1}$$

EIGENVALUE are called the *eigenvalues* of C. The N-vector $\vec{x}$ satisfying Equation (18.1) for an eigenvalue λ 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 C are the M-vectors y such that

$$\vec{y}^T C = \lambda\vec{y}^T. \tag{18.2}$$

The number of nonzero eigenvalues of C is at most rank(C).

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

Example 18.1: Consider the matrix

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

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

$$\vec{x_1} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{x_2} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ and } \vec{x_3} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

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 can always express $\vec{v}$ as a linear combination of the three

eigenvectors of S; in the current example we have

$$\vec{v} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} = 2\vec{x_1} + 4\vec{x_2} + 6\vec{x_3}.$$

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

$$\begin{aligned} S\vec{v} &= S(2\vec{x_1} + 4\vec{x_2} + 6\vec{x_3}) \\ &= 2S\vec{x_1} + 4S\vec{x_2} + 6S\vec{x_3} \\ &= 2\lambda_1\vec{x_1} + 4\lambda_2\vec{x_2} + 6\lambda_3\vec{x_3} \\ &= 60\vec{x_1} + 80\vec{x_2} + 6\vec{x_3}. \end{aligned} \tag{18.3}$$

Example 18.1 shows that 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 Equation (18.3) that the product $S\vec{v}$ is relatively unaffected by terms arising from the small eigenvalues of S; in our example, because $\lambda_3 = 1$, the contribution of the third term on the right hand side of Equation (18.3) is small. In fact, if we were to completely ignore the contribution in Equation (18.3) from the third eigenvector corresponding to $\lambda_3 = 1$, then the product $S\vec{v}$ would be computed to be $\begin{pmatrix} 60 \\ 80 \\ 0 \end{pmatrix}$ rather than the correct product, which is $\begin{pmatrix} 60 \\ 80 \\ 6 \end{pmatrix}$; these two vectors are relatively close to each other by any of various metrics one could apply (such as the length of their vector difference).

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.

Example 18.2: Consider the real, symmetric matrix

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

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*. Matrix decompositions similar to the ones in this section forms the basis of our principal text-analysis technique in Section 18.3, where we will look at decompositions of nonsquare term–document matrices. The square decompositions in this section are simpler and can be treated with sufficient mathematical rigor to help the reader to understand how such decompositions work. The detailed mathematical derivation of the more complex decompositions in Section 18.2 are beyond the scope of this book.

MATRIX DECOMPOSITION

We begin by giving 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

EIGEN DECOMPOSITION

$$S = U\Lambda U^{-1}, \tag{18.5}$$

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

$$\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_M \end{pmatrix}, \lambda_i \geq \lambda_{i+1}. \tag{18.6}$$

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

$$U = (\vec{u_1}\ \vec{u_2} \cdots \vec{u_M}). \tag{18.7}$$

Then we have

$$\begin{aligned} SU &= S(\vec{u_1}\ \vec{u_2} \cdots \vec{u_M}) \\ &= (\lambda_1\vec{u_1}\ \lambda_2\vec{u_2} \cdots \lambda_M\vec{u_M}) \\ &= (\vec{u_1}\ \vec{u_2} \cdots \vec{u_M}) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_M \end{pmatrix}. \end{aligned}$$

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

We next state a closely related decomposition of a symmetric square matrix into the product of matrices derived from its eigenvectors. This will pave the way for the development of our main tool for text analysis, the singular value decomposition (Section 18.2).

SYMMETRIC DIAGONAL DECOMPOSITION

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 diagonal decomposition

$$S = Q\Lambda Q^T, \tag{18.8}$$

where the columns of Q are the orthogonal and normalized (unit length, real) eigenvectors of S, and Λ 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 diagonal decomposition to build low-rank approximations to term–document matrices.

?

Exercise 18.1 What is the rank of the 3×3 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

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

Find the corresponding eigenvector.

Exercise 18.3 Compute the unique eigen decomposition of the 2×2 matrix in (18.4).

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 C where (barring a rare coincidence) $M \neq N$; furthermore, C is very unlikely to be symmetric. To this end we first describe an extension of the symmetric diagonal 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 C. It is beyond the scope of this book to develop a full treatment of the mathematics underlying singular value decompositions; following the statement of Theorem 18.3 we relate the singular value

SINGULAR VALUE DECOMPOSITION

SYMMETRIC DIAGONAL DECOMPOSITION

decomposition to the symmetric diagonal decompositions from Section 18.1.1. Given C, let U be the $M \times M$ matrix whose columns are the orthogonal eigenvectors of CC^T, and V be the $N \times N$ matrix whose columns are the orthogonal eigenvectors of C^TC. Denote by C^T the transpose of a matrix C.

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

SVD

$$C = U\Sigma V^T, \tag{18.9}$$

where

1. *The eigenvalues $\lambda_1, \ldots, \lambda_r$ of CC^T are the same as the eigenvalues of C^TC;*
2. *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 Σ is composed by setting $\Sigma_{ii} = \sigma_i$ for $1 \leq i \leq r$, and zero otherwise.*

The values σ_i are referred to as the *singular values* of C. It is instructive to examine the relationship of Theorem 18.3 to Theorem 18.2; we do this rather than derive the general proof of Theorem 18.3, which is beyond the scope of this book.

By multiplying Equation (18.9) by its transposed version, we have

$$CC^T = U\Sigma V^T \; V\Sigma U^T = U\Sigma^2 U^T. \tag{18.10}$$

Note now that in Equation (18.10), the left-hand side is a square symmetric matrix real-valued matrix, and the right-hand side represents its symmetric diagonal decomposition as in Theorem 18.2. What does the left-hand side CC^T represent? It is a square matrix with a row and a column corresponding to each of the M terms. The entry (i, j) in the matrix is a measure of the overlap between the ith and jth terms, based on their co-occurrence in documents. The precise mathematical meaning depends on the manner in which C is constructed based on term weighting. Consider the case where C is the term–document incidence matrix of page 3, illustrated in Figure 1.1. Then the entry (i, j) in CC^T is the number of documents in which both term i and term j occur.

When writing down the numerical values of the SVD, it is conventional to represent Σ as an $r \times r$ matrix with the singular values on the diagonals, because all its entries outside this submatrix are zeros. Accordingly, it is conventional to omit the rightmost $M - r$ columns of U corresponding to these omitted rows of Σ; likewise the rightmost $N - r$ columns of V are omitted because they correspond in V^T to the rows that will be multiplied by the $N - r$ columns of zeros in Σ. This written form of the SVD is sometimes known as the *reduced SVD* or *truncated SVD* and we will encounter it again in Exercise 18.9. Henceforth, our numerical examples and exercises will use this reduced form.

REDUCED SVD
TRUNCATED SVD

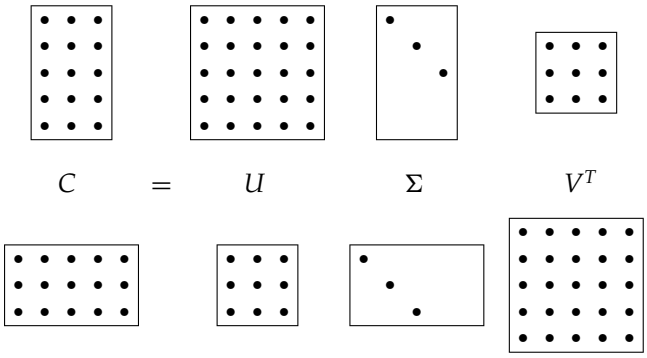

Figure 18.1 Illustration of the SVD. In this schematic illustration of (18.9), we see two cases illustrated. In the top half of the figure, we have a matrix C for which $M > N$. The lower half illustrates the case $M < N$.

Example 18.3: We now illustrate the singular-value decomposition of a 4×2 matrix of rank 2; the singular values are $\Sigma_{11} = 2.236$ and $\Sigma_{22} = 1$.

$$C = \begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} -0.632 & 0.000 \\ 0.316 & -0.707 \\ -0.316 & -0.707 \\ 0.632 & 0.000 \end{pmatrix} \begin{pmatrix} 2.236 & 0.000 \\ 0.000 & 1.000 \end{pmatrix} \begin{pmatrix} -0.707 & 0.707 \\ -0.707 & -0.707 \end{pmatrix}. \tag{18.11}$$

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 18.5.

Exercise 18.4 Let

$$C = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{18.12}$$

be the term–document incidence matrix for a collection. Compute the co-occurrence matrix CC^T. What is the interpretation of the diagonal entries of CC^T when C is a term–document incidence matrix?

Exercise 18.5 Verify that the SVD of the matrix in Equation (18.12) is

$$U = \begin{pmatrix} -0.816 & 0.000 \\ -0.408 & -0.707 \\ -0.408 & 0.707 \end{pmatrix}, \Sigma = \begin{pmatrix} 1.732 & 0.000 \\ 0.000 & 1.000 \end{pmatrix} \text{ and } V^T = \begin{pmatrix} -0.707 & -0.707 \\ 0.707 & -0.707 \end{pmatrix}, \tag{18.13}$$

by verifying all of the properties in the statement of Theorem 18.3.

Exercise 18.6 Suppose that C is a term–document incidence matrix. What do the entries of C^TC represent?

Exercise 18.7 Let

$$C = \begin{pmatrix} 0 & 2 & 1 \\ 0 & 3 & 0 \\ 2 & 1 & 0 \end{pmatrix} \tag{18.14}$$

be a term–document matrix whose entries are term frequencies; thus term 1 occurs twice in document 2 and once in document 3. Compute CC^T; observe that its entries are largest where two terms have their most frequent occurrences together in the same document.

18.3 Low-rank approximations

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 SVD, then develop its application to IR.

FROBENIUS NORM

Given an $M \times N$ matrix C and a positive integer k, we wish to find an $M \times N$ matrix C_k of rank at most k, so as to minimize the *Frobenius norm* of the matrix difference $X = C - C_k$, defined to be

$$||X||_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} X_{ij}^2}. \tag{18.15}$$

Thus, the Frobenius norm of X measures the discrepancy between C_k and C; our goal is to find a matrix C_k that minimizes this discrepancy, while constraining C_k to have rank at most k. If r is the rank of C, clearly $C_r = C$ and the Frobenius norm of the discrepancy is zero in this case. When k is far

LOW-RANK APPROXIMATION

smaller than r, we refer to C_k as a *low-rank approximation*.

The SVD can be used to solve the low-rank matrix approximation problem. We then derive from it an application to approximating term–document matrices. We invoke the following three-step procedure to this end:

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

The rank of C_k is at most k: This follows from the fact that Σ_k has at most k nonzero values. Next, we recall the intuition of Example 18.1: 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 C. 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.

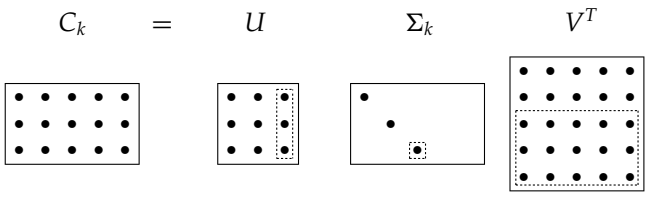

Figure 18.2 Illustration of low rank approximation using the SVD. The dashed boxes indicate the matrix entries affected by "zeroing out" the smallest singular values.

Theorem 18.4.

$$\min_{Z|\ rank(Z)=k} ||C - Z||_F = ||C - C_k||_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}. \tag{18.16}$$

Recalling that the singular values are in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots$, we learn from Theorem 18.4 that C_k is the best rank-k approximation to C, incurring an error (measured by the Frobenius norm of $C - C_k$) equal to σ_{k+1}. Thus, the larger k is, the smaller this error (and in particular, for $k = r$, the error is zero since $\Sigma_r = \Sigma$; provided $r < M, N$, then $\sigma_{r+1} = 0$ and thus $C_r = C$).

To derive further insight into why the process of truncating the smallest $r - k$ singular values in Σ helps to generate a rank-k approximation of low error, we examine the form of C_k:

$$C_k = U\Sigma_k V^T \tag{18.17}$$

$$= U \begin{pmatrix} \sigma_1 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \sigma_k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots \end{pmatrix} V^T \tag{18.18}$$

$$= \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T, \tag{18.19}$$

where $\vec{u}_i$ and $\vec{v}_i$ are the ith columns of U and V, respectively. Thus, $\vec{u}_i \vec{v}_i^T$ is a rank-1 matrix, so that we have just expressed C_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 $\vec{u}_i \vec{v}_i^T$ is weighted by a sequence of shrinking singular values σ_i.

Exercise 18.8 Compute a rank-1 approximation C_1 to the matrix C in Example 18.12, using the SVD as in Exercise 18.13. What is the Frobenius norm of the error of this approximation?

Exercise 18.9 Consider now the computation in Exercise 18.8. Following the schematic in Figure 18.2, notice that for a rank-1 approximation we have

σ_1 being a scalar. Denote by U_1 the first column of U and by V_1 the first column of V. Show that the rank-1 approximation to C can then be written as $U_1\sigma_1 V_1^T = \sigma_1 U_1 V_1^T$.

Exercise 18.10 Exercise 18.9 can be generalized to rank k approximations: We let U_k' and V_k' denote the "reduced" matrices 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

$$C_k = U_k' \Sigma_k' V_k'^T, \tag{18.20}$$

where Σ_k' is the square $k \times k$ submatrix of Σ_k with the singular values $\sigma_1, \ldots, \sigma_k$ on the diagonal. The primary advantage of using (18.20) is to eliminate a lot of redundant columns of zeros in U and V, thereby explicitly eliminating multiplication by columns that do not affect the low-rank approximation; this version of the SVD is sometimes known as the reduced SVD or truncated SVD and is a computationally simpler representation from which to compute the low rank approximation.

For the matrix C in Example 18.3, write down both Σ_2 and Σ_2'.

18.4 Latent semantic indexing

We now discuss the approximation of a term-document matrix C by one of lower rank using the SVD. The low-rank approximation to C yields a new representation for each document in the collection. We will cast queries into this low-rank representation as well, enabling us to compute query–document similarity scores in this low-rank representation. This process is known as *latent semantic indexing* (generally abbreviated LSI).

LATENT SEMANTIC INDEXING

But first, we motivate such an approximation. Recall the vector space representation of documents and queries introduced in Chapter 6. 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: *synonymy* and *polysemy*. *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. Consequently, the computed similarity $\vec{q} \cdot \vec{d}$ between a query $\vec{q}$ (say, car) and a document $\vec{d}$ containing both car and automobile 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 $\vec{q} \cdot \vec{d}$ 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 C is likely to have several tens of thousand of rows and columns, and a rank in the tens of thousands as well. In LSI (sometimes referred to as *latent semantic analysis (LSA)*), we use the SVD to construct a low-rank approximation C_k to the term–document matrix, for a value of k that is far smaller than the original rank of C. In the experimental work cited later in this section, 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 CC^T and C^TC. Note that the matrix C_k is itself still an $M \times N$ matrix, irrespective of k.

LSA

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

$$\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}. \tag{18.21}$$

Now, we may use cosine similarities as in Chapter 6 to compute the similarity between a query and a document, between two documents, or between two terms. Note especially that Equation (18.21) does not in any way depend on $\vec{q}$ being a query; it is simply a vector in the space of terms. This means that if we have an LSI representation of a collection of documents, a new document not in the collection can be "folded in" to this representation using Equation (18.21). This allows us to incrementally add documents to an LSI representation. Of course, such incremental addition fails to capture the co-occurrences of the newly added documents (and even ignores any new terms they contain). As such, the quality of the LSI representation will degrade as more documents are added and will eventually require a recomputation of the LSI representation.

The fidelity of the approximation of C_k to C 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.

Example 18.4: Consider the term-document matrix $C =$

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

Its SVD is the product of three matrices as below. First we have U, which in this example is:

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
voyage	−0.70	0.35	0.15	−0.58	0.16
trip	−0.26	0.65	−0.41	0.58	−0.09

When applying the SVD to a term–document matrix, U is known as the *SVD term matrix*. The singular values are $\Sigma =$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

Finally we have V^T, which in the context of a term-document matrix is known as the *SVD document matrix*:

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

By "zeroing out" all but the two largest singular values of Σ, we obtain $\Sigma_2 =$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

From this, we compute $C_2 =$

	d_1	d_2	d_3	d_4	d_5	d_6
1	−1.62	−0.60	−0.44	−0.97	−0.70	−0.26
2	−0.46	−0.84	−0.30	1.00	0.35	0.65
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Notice that the low-rank approximation, unlike the original matrix C, can have negative entries.

Examination of C_2 and Σ_2 in Example 18.4 shows that the last three rows of each of these matrices are populated entirely by zeros. This suggests that the SVD product $U\Sigma V^T$ in Equation (18.18) can be carried out with only two rows in the representations of Σ_2 and V^T; we may then replace these matrices by their truncated versions Σ_2' and $(V')^T$. For instance, the truncated SVD document matrix $(V')^T$ in this example is:

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41

Figure 18.3 illustrates the documents in $(V')^T$ in two dimensions. Note also that C_2 is dense relative to C.

We may in general view the low-rank approximation of C by C_k as a *constrained optimization* problem, subject to the constraint that C_k have rank at

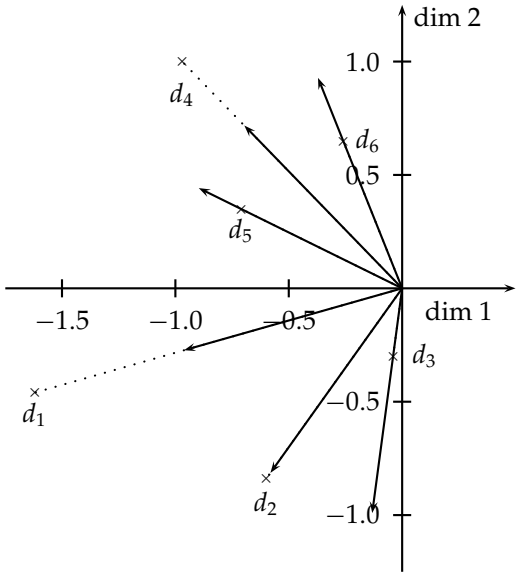

Figure 18.3 The documents of Example 18.4 reduced to two dimensions in $(V')^T$.

most k, we seek a representation of the terms and documents comprising C with low Frobenius norm for the error $C - C_k$. 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*.

Dumais (1993) and Dumais (1995) conducted experiments with LSI on TREC documents and tasks, using the commonly used Lanczos algorithm to compute the SVD. At the time of their work in the early 1990s, the LSI computation on tens of thousands of documents took approximately one day on one machine. In these experiments, they achieved precision at or above that of the median TREC participant. In about 20% of TREC topics, their system was the top scorer, and reportedly slightly better on average than standard vector spaces for LSI at about 350 dimensions. Here are some conclusions on LSI first suggested by their work, and subsequently verified by many other experiments.

- The computational cost of the SVD is significant; at the time of this writing, we know of no successful experiment with over one million documents. This has been the biggest obstacle to the widespread adoption to LSI. One approach to this obstacle is to build the LSI representation on a randomly sampled subset of the documents in the collection, following which the remaining documents are "folded in" as detailed with Equation (18.21).
- As we reduce k, recall tends to increase, as expected.
- Most surprisingly, a value of k in the low hundreds can actually *increase* precision on some query benchmarks. This suggests that, for a suitable value of k, LSI addresses some of the challenges of synonymy.
- LSI works best in applications where there is little overlap between queries and documents.

The experiments also documented some modes where LSI failed to match the effectiveness of more traditional indexes and score computations. 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), and no way of enforcing Boolean conditions.

SOFT CLUSTERING

LSI can be viewed as *soft clustering* by interpreting each dimension of the reduced space as a cluster and the value that a document has on that dimension as its fractional membership in that cluster.

? **Exercise 18.11** Assume you have a set of documents each of which is in either English or in Spanish. 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:

DocID	document text
1	hello
2	open house
3	mi casa
4	hola Profesor
5	hola y bienvenido
6	hello and welcome

Figure 18.4 Documents for Exercise 18.11.

Spanish	English
mi	my
casa	house
hola	hello
profesor	professor
y	and
bienvenido	welcome

Figure 18.5 Glossary for Exercise 18.11.

1. Construct the appropriate term–document matrix C to use for a collection consisting of these 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, Σ_2' and V_2 and from these derive the rank 2 approximation C_2.
3. State succinctly what the (i, j) entry in the matrix $C^T C$ represents.
4. State succinctly what the (i, j) entry in the matrix $C_2^T C_2$ represents, and why it differs from that in $C^T C$.

18.5 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 IR 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. www.cs.utk.edu/~berry/lsi++/ and http://lsi.argreenhouse.com/lsi/LSIpapers.html offer comprehensive pointers to the literature and software of LSI. Schütze and Silverstein (1997) evaluate LSI and truncated representations of centroids for efficient K-means

clustering (Section 16.4). Bast and Majumdar (2005) detail the role of the reduced dimension k in LSI and how different pairs of terms get coalesced together at differing values of k. Applications of LSI to *cross-language information retrieval* (where documents in two or more different languages are indexed, and a query posed in one language is expected to retrieve documents in other languages) are developed in Berry and Young (1995) and Littman et al. (1998). LSI (referred to as LSA in more general settings) has been applied to host of other problems in computer science ranging from memory modeling to computer vision.

CROSS-LANGUAGE INFORMATION RETRIEVAL

Hofmann (1999a, 1999b) provides an initial probabilistic extension of the basic LSI technique. A more satisfactory formal basis for a probabilistic latent variable model for dimensionality reduction is the Latent Dirichlet Allocation (LDA) model (Blei et al. 2003), which is generative and assigns probabilities to documents outside of the training set. This model is extended to a hierarchical clustering by Rosen-Zvi et al. (2004). Wei and Croft (2006) present the first large scale evaluation of LDA, finding it to significantly outperform the query likelihood model of Section 12.2 (page 223), but to not perform quite as well as the relevance model mentioned in Section 12.4 (page 230) – but the latter does additional per-query processing unlike LDA. Teh et al. (2006) generalize further by presenting Hierarchical Dirichlet processes, a probabilistic model that allows a group (for us, a document) to be drawn from an infinite mixture of latent topics, while still allowing these topics to be shared across documents.