ical technique, called Singular Value Decomposition or SVD, to a word-by-document matrix. SVD (and hence LSI) is a least-squares method. The projection into the latent semantic space is chosen such that the representations in the original space are changed as little as possible when measured by the sum of the squares of the differences. We first give a simple example of a least-squares method and then introduce SVD.

15.4.1 Least-squares methods

Before defining the particular least-squares method used in LSI, it is instructive to study the most common least-squares approximation: fitting a line to a set of points in the plane by way of *linear regression.*

LINEAR REGRESSION

Consider the following problem. We have a set of n points: (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), as in figure 15.7. We would like to find the line:

$$f(x) = mx + b$$

with parameters m and b that fits these points best. In a least-squares approximation, the best fit is the one that minimizes the sum of the squares of the differences:

$$SS(m,b) = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2 \qquad (15.8)$$

We compute b by solving $\frac{\partial SS(m,b)}{\partial b} = 0$, the value of b for which $SS(m,b)$ reaches its minimum:

$$\frac{\partial SS(m,b)}{\partial b} = \sum_{i=1}^{n} [2(y_i - mx_i - b)(-1)] = 0$$

$$\Leftrightarrow \quad [\textstyle\sum_{i=1}^{n} y_i] - [m \sum_{i=1}^{n} x_i] - [nb] = 0$$

$$\Leftrightarrow \quad b = \bar{y} - m\bar{x} \qquad (15.9)$$

where $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ and $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ are the means of the x and y coordinates, respectively.

We now substitute (15.9) for b in (15.8) and solve $\frac{\partial SS(m,b)}{\partial m} = 0$ for m:

$$\frac{\partial SS(m,b)}{\partial m} = \frac{\partial \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2}{\partial m} = 0$$

$$\Leftrightarrow \quad \sum_{i=1}^{n} 2(y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

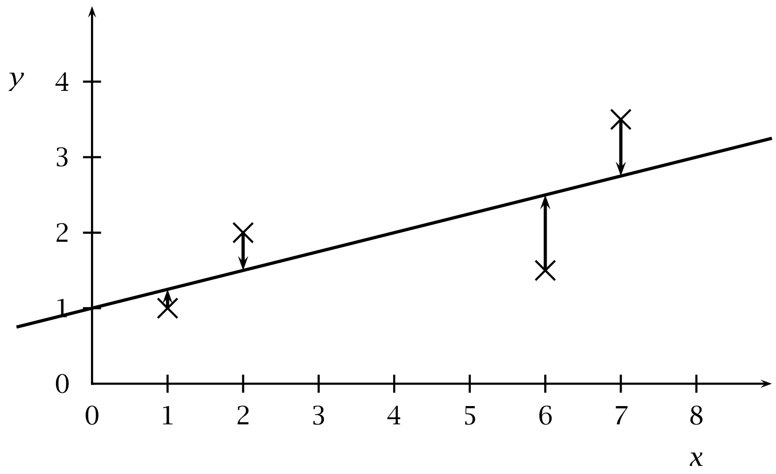

Figure 15.7 An example of linear regression. The line $y = 0.25x + 1$ is the best least-squares fit for the four points (1,1), (2,2), (6,1.5), (7,3.5). Arrows show which points on the line the original points are projected to.

$$\Leftrightarrow \qquad 0 = -\sum_{i=1}^{n}(\bar{y} - y_i)(\bar{x} - x_i) + m\sum_{i=1}^{n}(\bar{x} - x_i)^2$$

$$\Leftrightarrow \qquad m = \frac{\sum_{i=1}^{n}(\bar{y} - y_i)(\bar{x} - x_i)}{\sum_{i=1}^{n}(\bar{x} - x_i)^2} \tag{15.10}$$

Figure 15.7 shows an example of a least square fit for the four points $(1,1), (2,2), (6,1.5), (7,3.5)$. We have:

$$\bar{x} = 4, \bar{y} = 2, m = \frac{\sum_{i=1}^{n}(\bar{y} - y_i)(\bar{x} - x_i)}{\sum_{i=1}^{n}(\bar{x} - x_i)^2} = \frac{6.5}{26} = 0.25$$

and

$$b = \bar{y} - m\bar{x} = 2 - 0.25 \times 4 = 1$$

15.4.2 Singular Value Decomposition

As we have said, we can view Singular Value Decomposition or SVD as a method of word co-occurrence analysis. Instead of using a simple word overlap measure like the cosine, we instead use a more sophisticated similarity measure that makes better similarity judgements based on word co-occurrence. Equivalently, we can view SVD as a method for dimensionality reduction. The relation between these two viewpoints is that in the process of dimensionality reduction, co-occurring terms are mapped onto the same dimensions of the reduced space, thus increasing similarity in the representation of semantically similar documents.

Co-occurrence analysis and dimensionality reduction are two 'functional' ways of understanding LSI. We now look at the formal definition of LSI. LSI is the application of Singular Value Decomposition (SVD) to document-by-term matrices in information retrieval. SVD takes a matrix A and represents it as $\hat{A}$ in a lower dimensional space such that the "distance" between the two matrices as measured by the 2-norm is minimized:

$$\Delta = \|A - \hat{A}\|_2 \tag{15.11}$$

The 2-norm for matrices is the equivalent of Euclidean distance for vectors. SVD is in fact very similar to fitting a line, a one-dimensional object, to a set of points, which exists in the two-dimensional plane. Figure 15.7 shows which point on the one-dimensional line each of the original points corresponds to (see arrows).

Just as the linear regression in figure 15.7 can be interpreted as projecting a two-dimensional space onto a one-dimensional line, so does SVD project an n-dimensional space onto a k-dimensional space where $n \gg k$. In our application (word-document matrices), n is the number of word types in the collection. Values of k that are frequently chosen are 100 and 150. The projection transforms a document's vector in n-dimensional word space into a vector in the k-dimensional reduced space.

One possible source of confusion is that equation (15.11) compares the original matrix and a lower-dimensional approximation. Shouldn't the second matrix have fewer rows and columns, which would make equation (15.11) ill-defined? The analogy with line fitting is again helpful here. The fitted line exists in two dimensions, but it is a one-dimensional object. The same is true for $\hat{A}$: it is a matrix of lower rank, that is, it could be represented in a lower-dimensional space by transforming the axes of the space. But for the particular axes chosen it has the same number of rows and columns as A.

The SVD projection is computed by decomposing the document-by-term matrix $A_{t\times d}$ into the product of three matrices, [1] $T_{t\times n}$, $S_{n\times n}$, and

1. Technically, this is the definition of the so-called reduced SVD. The full SVD takes the

$$T = \begin{array}{l|rrrrr} & \text{Dim. 1} & \text{Dim. 2} & \text{Dim. 3} & \text{Dim. 4} & \text{Dim. 5} \\ \hline \text{cosmonaut} & -0.44 & -0.30 & 0.57 & 0.58 & 0.25 \\ \text{astronaut} & -0.13 & -0.33 & -0.59 & 0.00 & 0.73 \\ \text{moon} & -0.48 & -0.51 & -0.37 & 0.00 & -0.61 \\ \text{car} & -0.70 & 0.35 & 0.15 & -0.58 & 0.16 \\ \text{truck} & -0.26 & 0.65 & -0.41 & 0.58 & -0.09 \end{array}$$

Figure 15.8 The matrix T of the SVD decomposition of the matrix in figure 15.5. Values are rounded.

$$S = \begin{pmatrix} 2.16 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.59 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.28 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.39 \end{pmatrix}$$

Figure 15.9 The matrix of singular values of the SVD decomposition of the matrix in figure 15.5. Values are rounded.

$D_{d\times n}$:

(15.12) $$A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^{\mathrm{T}}$$

where $n = \min(t, d)$. We indicate dimensionality by subscripts: A has t rows and d columns, T has t rows and n columns and so on. D^{T} is the transpose of D, the matrix D rotated around its diagonal: $D_{ij} = (D^{\mathrm{T}})_{ji}$.

Examples of A, T, S, and D are given in figure 15.5 and figures 15.8 through 15.10. Figure 15.5 shows an example of A. A contains the document vectors with each column corresponding to one document. In other words, element a_{ij} of the matrix records how often term i occurs in document j. The counts should be appropriately weighted (as discussed in section 15.2). For simplicity of exposition, we have not applied weighting and assumed term frequencies of 1.

Figures 15.8 and 15.10 show T and D, respectively. These matrices have *orthonormal* columns. This means that the column vectors have

ORTHONORMAL

form $A_{t\times d} = T_{t\times t} S_{t\times d} \left(D_{d\times d}\right)^{\mathrm{T}}$, where the extra columns of T, S, D are zero vectors (?).

$D^{\mathrm{T}} =$

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

Figure 15.10 The matrix D^{T} of the SVD decomposition of the matrix in figure 15.5. Values are rounded.

unit length and are all orthogonal to each other. (If a matrix C has orthonormal columns, then $C^{\mathrm{T}}C = I$, where I is the diagonal matrix with a diagonal of 1's, and zeroes elsewhere. So we have $T^{\mathrm{T}}T = D^{\mathrm{T}}D = I$.)

We can view SVD as a method for rotating the axes of the n-dimensional space such that the first axis runs along the direction of largest variation among the documents, the second dimension runs along the direction with the second largest variation and so forth. The matrices T and D represent terms and documents in this new space. For example, the first column of T corresponds to the first row of A, and the first column of D corresponds to the first column of A.

The diagonal matrix S contains the singular values of A in descending order (as in figure 15.9). The i^{th} singular value indicates the amount of variation along the i^{th} axis. By restricting the matrices T, S, and D to their first $k < n$ columns one obtains the matrices $T_{t\times k}$, $S_{k\times k}$, and $(D_{d\times k})^{\mathrm{T}}$. Their product $\hat{A}$ is the best least square approximation of A by a matrix of rank k in the sense defined in equation (15.11). One can also prove that SVD is unique, that is, there is only one possible decomposition of a given matrix.[2]

See Golub and van Loan (1989) for an extensive treatment of SVD including a proof of the optimality property.

That SVD finds the optimal projection to a low-dimensional space is the key property for exploiting word co-occurrence patterns. SVD represents terms and documents in the lower dimensional space as well as possible.

2. For any given SVD solution, you can get a second non-identical one by flipping signs in the corresponding left and right singular vectors of T and D. If there are two or more identical singular values, one can also get different solutions for the singular vectors corresponding to the identical values. Apart from these cases, SVD is unique.

	d_1	d_2	d_3	d_4	d_5	d_6
Dimension 1	−1.62	−0.60	−0.04	−0.97	−0.71	−0.26
Dimension 2	−0.46	−0.84	−0.30	1.00	0.35	0.65

Figure 15.11 The matrix $B_{2\times d} = S_{2\times 2}D^{\mathrm{T}}{}_{2\times d}$ of documents after rescaling with singular values and reduction to two dimensions. Values are rounded.

In the process, some words that have similar co-occurrence patterns are projected (or collapsed) onto the same dimension. As a consequence, the similarity metric will make topically similar documents and queries come out as similar even if different words are used for describing the topic. If we restrict the matrix in figure 15.8 to the first two dimensions, we end up with two groups of terms: space exploration terms (*cosmonaut*, *astronaut*, and *moon*) which have negative values on the second dimension and automobile terms (*car* and *truck*) which have positive values on the second dimension. The second dimension directly reflects the different co-occurrence patterns of these two groups: space exploration terms only co-occur with other space exploration terms, automobile terms only co-occur with other automobile terms (with one exception: the occurrence of *car* in d_1). In some cases, we will be misled by such co-occurrences patterns and wrongly infer semantic similarity. However, in most cases co-occurrence is a valid indicator of topical relatedness.

These term similarities have a direct impact on document similarity. Let us assume a reduction to two dimensions. After rescaling with the singular values, we get the matrix $B = S_{2\times 2}D^{\mathrm{T}}{}_{2\times d}$ shown in figure 15.11, where $S_{2\times 2}$ is S restricted to two dimensions (with the diagonal elements 2.16, 1.59). Matrix B is a dimensionality reduction of the original matrix A and is what was shown in figure 15.6.

Table 15.9 shows the similarities between documents when they are represented in this new space. Not surprisingly, there is high similarity between d_1 and d_2 (0.78) and d_4, d_5, and d_6 (0.94, 0.93, 0.74). These document similarities are about the same in the original space (i.e., when we compute correlations for the original document vectors in figure 15.5). The key change is that d_2 and d_3, whose similarity is 0.00 in the original space, are now highly similar (0.88). Although d_2 and d_3 have no common terms, they are now recognized as being topically similar because of the co-occurrence patterns in the corpus.

Notice that we get the same similarity as in the original space (that is,

	d_1	d_2	d_3	d_4	d_5	d_6
d_1	1.00					
d_2	0.78	1.00				
d_3	0.40	0.88	1.00			
d_4	0.47	−0.18	−0.62	1.00		
d_5	0.74	0.16	−0.32	0.94	1.00	
d_6	0.10	−0.54	−0.87	0.93	0.74	1.00

Table 15.9 The matrix of document correlations $E^{\mathrm{T}}E$ where E is B with normalized columns. For example, the normalized correlation coefficient of documents d_3 and d_2 (when represented as in figure 15.11) is 0.88. Values are rounded.

zero similarity) if we compute similarity in the transformed space without any dimensionality reduction. Using the full vectors from figure 15.10 and rescaling them with the appropriate singular values we get:

$$-0.28 \times -0.20 \times 2.16^2 + -0.53 \times -0.19 \times 1.59^2 + \\ -0.75 \times 0.45 \times 1.28^2 + 0.00 \times 0.58 \times 1.00^2 + 0.29 \times 0.63 \times 0.39^2 \approx 0.00$$

(If you actually compute this expression, you will find that the answer is not quite zero, but this is only because of rounding errors. But this is as good a point as any to observe that many matrix computations are quite sensitive to rounding errors.)

We have computed document similarity in the reduced space using the product of S and D^{T}. The correctness of this procedure can be seen by looking at $A^{\mathrm{T}}A$, which is the matrix of all document correlations for the original space:

$$A^{\mathrm{T}}A = (TSD^{\mathrm{T}})^{\mathrm{T}}TSD^{\mathrm{T}} = DS^{\mathrm{T}}T^{\mathrm{T}}TSD^{\mathrm{T}} = DS^{\mathrm{T}}SD^{\mathrm{T}} = (SD^{\mathrm{T}})^{\mathrm{T}}(SD^{\mathrm{T}}) = B^{\mathrm{T}}B \tag{15.13}$$

Because T has orthonormal columns, we have $T^{\mathrm{T}}T = I$. Furthermore, since S is diagonal, $S = S^{\mathrm{T}}$. Term similarities are computed analogously since one observes that the term correlations are given by:

$$AA^{\mathrm{T}} = TSD^{\mathrm{T}}(TSD^{\mathrm{T}})^{\mathrm{T}} = TSD^{\mathrm{T}}DS^{\mathrm{T}}T^{\mathrm{T}} = (TS)(TS)^{\mathrm{T}} \tag{15.14}$$

One remaining problem for a practical application is how to fold queries and new documents into the reduced space. The SVD computation only gives us reduced representations for the document vectors in matrix A. We do not want to do a completely new SVD every time a new query is launched. In addition, in order to handle large corpora efficiently we may want to do SVD for only a sample of the documents (for example a third or a fourth). The remaining documents would then be folded in.

The equation for folding documents into the space can again be derived from the basic SVD equation:

$$
\begin{aligned}
& A = TSD^{\mathrm{T}} \\
\Leftrightarrow\quad & T^{\mathrm{T}}A = T^{\mathrm{T}}TSD^{\mathrm{T}} \\
\Leftrightarrow\quad & T^{\mathrm{T}}A = SD^{\mathrm{T}}
\end{aligned}
\tag{15.15}
$$

So we just multiply the query or document vector with the transpose of the term matrix T (after it has been truncated to the desired dimensionality). For example, for a query vector $\vec{q}$ and a reduction to dimensionality k, the query representation in the reduced space is ${T_{t\times k}}^{\mathrm{T}}\vec{q}$.

15.4.3 Latent Semantic Indexing in IR

LATENT SEMANTIC INDEXING

The application of SVD to information retrieval was originally proposed by a group of researchers at Bellcore (Deerwester et al. 1990) and called *Latent Semantic Indexing* (LSI) in this context. LSI has been compared to standard vector space search on several document collections. It was found that LSI performs better than vector space search in many cases, especially for high-recall searches (Deerwester et al. 1990; Dumais 1995). LSI's strength in high-recall searches is not surprising since a method that takes co-occurrence into account is expected to achieve higher recall. On the other hand, due to the noise added by spurious co-occurrence data one sometimes finds a decrease in precision.

The appropriateness of LSI also depends on the document collection. Recall the example of the vocabulary problem in figure 15.8. In a heterogeneous collection, documents may use different words to refer to the same topic like *HCI* and *user interface* in the figure. Here, LSI can help identify the underlying semantic similarity between seemingly dissimilar documents. However, in a collection with homogeneous vocabulary, LSI is less likely to be useful.

The application of SVD to information retrieval is called *Latent Semantic Indexing* because the document representations in the original term space are transformed to representations in a new reduced space. The dimensions in the reduced space are linear combinations of the original dimensions (this is so since matrix multiplications as in equation (15.16) are linear operations). The assumption here (and similarly for other forms of dimensionality reduction like principal component analysis) is that these new dimensions are a better representation of documents and queries. The metaphor underlying the term "latent" is that these new dimensions are the true representation. This true representation was then obscured by a generation process that expressed a particular dimension with one set of words in some documents and a different set of words in another document. LSI analysis recovers the original semantic structure of the space and its original dimensions. The process of assigning different words to the same underlying dimension is sometimes interpreted as a form of soft term clustering since it groups terms according to the dimensions that they are represented on in the reduced space.

One could also argue that the SVD representation is not only better (since it is based on the 'true' dimensions), but also more compact. Many documents have more than 150 unique terms. So the sparse vector representation will take up more storage space than the compact SVD representation if we reduce to 150 dimensions. However, the efficiency gain due to more compact representations is often outweighed by the additional cost of having to go through a high-dimensional matrix multiplication whenever we map a query or a new document to the reduced space. Another problem is that an inverted index cannot be constructed for SVD representations. If we have to compute the similarity between the query and every single document, then an SVD-based system can be slower than a term-based system that searches an inverted index.

The actual computation of SVD is quadratic in the rank of the document by term matrix (the rank is (bounded by) the smaller of the number of documents and the number of terms) and cubic in the number of singular values that are computed (Deerwester et al. 1990: 395).[3] For very large collections subsampling of documents and selection of terms according to frequency is often employed in order to reduce the cost of computing the Singular Value Decomposition.

NORMALITY ASSUMPTION

One objection to SVD is that, along with all other least-squares methods, it is really designed for normally-distributed data. But, as can be seen from the discussion earlier in this chapter, such a distribution is inappropriate for count data, and count data is, after all, what a term-by-document matrix consists of. The link between least squares and the

3. However, others have suggested that given the particular characteristics of the matrices that SVD is applied to in information retrieval the complexity is linear in the number of documents and (approximately) quadratic in the number of singular values. See (Oard and DeClaris 1996), (Berry et al. 1995), and (Berry and Young 1995) for discussion.

normal distribution can be easily seen by looking at the definition of the normal distribution (section 2.1.9):

$$\mathrm{n}(x;\ \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

where μ is the *mean* and σ the *covariance*. The smaller the squared deviation from the mean $(x-\mu)^2$, the higher the probability $n(x;\ \mu, \sigma)$. So the least squares solution is the maximum likelihood solution. But this is only true if the underlying data distribution is normal (or approximated well by a normal distribution). Other distributions like Poisson or negative binomial are more appropriate for term counts. One problematic feature of SVD is that, since the reconstruction $\hat{A}$ of the term-by-document matrix A is based on a normal distribution, it can have negative entries, clearly an inappropriate approximation for counts. A dimensionality reduction based on Poisson would not predict such impossible negative counts.

In defense of LSI (and the vector space model in general which can also be argued to assume a normal distribution), one can say that the matrix entries are not counts, but weights. Although this is not an issue that has been investigated systematically, the normal distribution could be appropriate for the weighted vectors even if it is not for count vectors.

From a practical point of view, LSI has been criticized for being computationally more expensive than other word co-occurrence methods while not being more effective. Another method that also uses co-occurrence

PSEUDO-FEEDBACK

is *pseudo-feedback* (also called *pseudo relevance feedback* and *two-stage retrieval*, Buckley et al. 1996; Kwok and Chan 1998). In pseudo-feedback, the top n documents (typically the top 10 or 20) returned by an ad-hoc query are assumed to be relevant and added to the query. Some of these top n documents will not actually be relevant, but a large enough proportion usually is to improve the quality of the query. Words that occur frequently with query words will be among the most frequent in the top n. So pseudo-feedback can be viewed as a cheap query-specific way of doing co-occurrence analysis and co-occurrence-based query modification.

Still, in contrast to many heuristic methods that incorporate term co-occurrence into information retrieval, LSI has a clean formal framework and a clearly defined optimization criterion (least squares) with one global optimum that can be efficiently computed. This conceptual simplicity and clarity make LSI one of the most interesting IR approaches that go beyond query-document term matching.

15.5 Discourse Segmentation

Text collections are increasingly heterogeneous. An important aspect of heterogeneity is length. On the world wide web, document sizes range from home pages with just one sentence to server logs of half a megabyte.

The weighting schemes discussed in section 15.2.2 take account of different lengths by applying cosine normalization. However, cosine normalization and other forms of normalization that discount term weights according to document length ignore the distribution of terms within a document. Suppose that you are looking for a short description of angioplasty. You would probably prefer a document in which the occurrences of angioplasty are concentrated in one or two paragraphs since such a concentration is most likely to contain a definition of what angioplasty is. On the other hand, a document of the same length in which the occurrences of angioplasty are scattered uniformly is less likely to be helpful.

We can exploit the structure of documents and search over structurally defined units like sections and paragraphs instead of full documents. However, the best subpart of a document to be returned to the user often encompasses several paragraphs. For example, in response to a query on angioplasty we may want to return the first two paragraphs of a subsection on angioplasty, which introduce the term and its definition, but not the rest of the subsection that goes into technical detail.

Some documents are not structured into paragraphs and sections. Or, in the case of documents structured by means of a markup language like HTML, it is not obvious how to break them apart into units that would be suitable for retrieval.

These considerations motivate an approach that breaks documents into topically coherent multi-paragraph subparts. In the rest of this subsection we will describe one approach to multiparagraph segmentation, the *TextTiling* algorithm (Hearst and Plaunt 1993; Hearst 1994, 1997).

TEXTTILING

15.5.1 TextTiling

The basic idea of this algorithm is to search for parts of a text where the vocabulary shifts from one subtopic to another. These points are then interpreted as the boundaries of multi-paragraph units.

SUBTOPIC

Sentence length can vary considerably. Therefore, the text is first divided into small fixed size units, the *token sequences*. Hearst suggests a size of 20 words for token sequences. We refer to the points between

TOKEN SEQUENCES