Acta Numerica (2006), pp. 327–384
doi: 10.1017/S0962492906240017

Numerical linear algebra in data mining

Lars Eldén
Department of Mathematics,
Linköping University, SE-581 83 Linköping, Sweden
E-mail: laeld@math.liu.se

Ideas and algorithms from numerical linear algebra are important in several areas of data mining. We give an overview of linear algebra methods in text mining (information retrieval), pattern recognition (classification of handwritten digits), and PageRank computations for web search engines. The emphasis is on rank reduction as a method of extracting information from a data matrix, low-rank approximation of matrices using the singular value decomposition and clustering, and on eigenvalue methods for network analysis.

CONTENTS

1. Introduction

1.1. Data mining

In modern society huge amounts of data are stored in databases with the purpose of extracting useful information. Often it is not known at the occasion of collecting the data what information is going to be requested, and therefore the database is often not designed for the distillation of

any particular information, but rather it is to a large extent unstructured. The science of extracting useful information from large data sets is usually referred to as 'data mining', sometimes along with 'knowledge discovery'.

There are numerous application areas of data mining, ranging from e-business (Berry and Linoff 2000, Mena 1999) to bioinformatics (Bergeron 2002), from scientific application such as astronomy (Burl, Asker, Smyth, Fayyad, Perona, Crumpler and Aubele 1998), to information retrieval (Baeza-Yates and Ribeiro-Neto 1999) and Internet search engines (Berry and Browne 2005).

Data mining is a truly interdisciplinary science, where techniques from computer science, statistics and data analysis, pattern recognition, linear algebra and optimization are used, often in a rather eclectic manner. Because of the practical importance of the applications, there are now numerous books and surveys in the area. We cite a few here: Christianini and Shawe-Taylor (2000), Cios, Pedrycz and Swiniarski (1998), Duda, Hart and Storck (2001), Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy (1996), Han and Kamber (2001), Hand, Mannila and Smyth (2001), Hastie, Tibshirani and Friedman (2001), Hegland (2001) and Witten and Frank (2000).

The purpose of this paper is not to give a comprehensive treatment of the areas of data mining, where linear algebra is being used, since that would be a far too ambitious undertaking. Instead we will present a few areas in which numerical linear algebra techniques play an important role. Naturally, the selection of topics is subjective, and reflects the research interests of the author.

This survey has three themes, as follows.

(1) *Information extraction from a data matrix by a rank reduction process.*

By determining the 'principal direction' of the data, the 'dominating' information is extracted first. Then the data matrix is deflated (explicitly or implicitly) and the same procedure is repeated. This can be formalized using the Wedderburn rank reduction procedure (Wedderburn 1934), which is the basis of many matrix factorizations.

The second theme is a variation of the rank reduction idea.

(2) *Data compression by low-rank approximation*: A data matrix $A \in \mathbb{R}^{m\times n}$, where m and n are large, will be approximated by a rank-k matrix,

$$A \approx WZ^T, \qquad W \in \mathbb{R}^{m\times k}, \quad Z \in \mathbb{R}^{n\times k},$$

where $k \ll \min(m, n)$.

In many applications the data matrix is huge, and difficult to use for storage and efficiency reasons. Thus, one evident purpose of compression is to obtain a representation of the data set that requires less memory than the

original data set, and that can be manipulated more efficiently. Sometimes one wishes to obtain a representation that can be interpreted as the 'main directions of variation' of the data, the *principal components*. This is done by building the low-rank approximation from the left and right singular vectors of A that correspond to the largest singular values. In some applications, *e.g.*, information retrieval (see Section 4) it is possible to obtain better search results from the compressed representation than from the original data. There the low-rank approximation also serves as a 'denoising device'.

(3) *Self-referencing definitions that can be formulated mathematically as eigenvalue and singular value problems.*

The most well-known example is the Google PageRank algorithm, which is based on the notion that the importance of a web page depends on how many inlinks it has from other important pages.

2. Vectors and matrices in data mining

Often the data are numerical, and the data points can be thought of as belonging to a high-dimensional vector space. Ensembles of data points can then be organized as matrices. In such cases it is natural to use concepts and techniques from linear algebra.

Example 2.1. Handwritten digit classification is a sub-area of *pattern recognition*. Here vectors are used to represent digits. The image of one digit is a 16×16 matrix of numbers, representing grey-scale. It can also be represented as a vector in $\mathbb{R}^{256}$, by stacking the columns of the matrix.

A set of n digits (handwritten 3s, say) can then be represented by matrix $A \in \mathbb{R}^{256\times n}$, and the columns of A can be thought of as a cluster. They also span a subspace of $\mathbb{R}^{256}$. We can compute an approximate basis of this subspace using the singular value decomposition (SVD) $A = U\Sigma V^T$. Three basis vectors of the '3-subspace' are illustrated in Figure 2.1. The digits are taken from the US Postal Service database (see, *e.g.*, Hastie *et al.* (2001)).

Let b be a vector representing an unknown digit, and assume that one wants to determine, automatically using a computer, which of the digits 0–9 the unknown digit represents. Given a set of basis vectors for 3s, $u_1, u_2, \ldots, u_k$, we may be able to determine whether b is a 3 or not, by checking if there is a linear combination of the k basis vectors, $\sum_{j=1}^{k} x_j u_j$, such that the residual $b - \sum_{j=1}^{k} x_j u_j$ is small. Thus, we determine the coordinates of b in the basis $\{u_j\}_{j=1}^{k}$, which is equivalent to solving a least squares problem with the data matrix $U_k = (u_1 \ \ldots \ u_k)$.

In Section 5 we discuss methods for classification of handwritten digits.

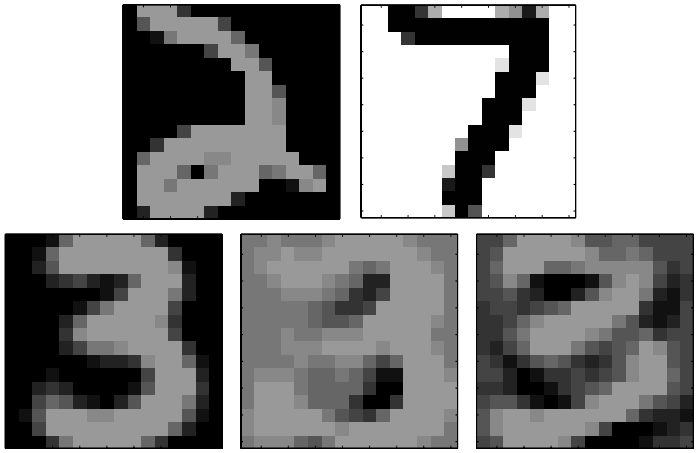


Figure 2.1. Handwritten digits from the US Postal Service data base, and basis vectors for 3s (bottom).

It also happens that there is a natural or perhaps clever way of encoding non-numerical data so that the data points become vectors. We will give a couple of such examples from text mining (information retrieval) and Internet search engines.

Example 2.2. *Term-document matrices* are used in information retrieval. Consider the following set of five documents. Key words, referred to as *terms*, are marked in boldface.[1]

Document 1:	The **Google matrix** P is a model of the **Internet.**
Document 2:	P_{ij} is nonzero if there is a **link** from **web page** j to i.
Document 3:	The **Google matrix** is used to **rank** all **web pages**
Document 4:	The **ranking** is done by solving a **matrix eigenvalue** problem.
Document 5:	**England** dropped out of the top 10 in the **FIFA ranking**.

Counting the frequency of terms in each document, we get the result shown in Table 2.1. The total set of terms is called the *dictionary*. Each document

[1] To avoid making the example too large, we have ignored some words that would normally be considered as terms. Note also that only the stem of a word is significant: 'ranking' is considered the same as 'rank'.

Table 2.1.

Term	Doc. 1	Doc. 2	Doc. 3	Doc. 4	Doc. 5
eigenvalue	0	0	0	1	0
England	0	0	0	0	1
FIFA	0	0	0	0	1
Google	1	0	1	0	0
Internet	1	0	0	0	0
link	0	1	0	0	0
matrix	1	0	1	1	0
page	0	1	1	0	0
rank	0	0	1	1	1
web	0	1	1	0	0

is represented by a vector in $\mathbb{R}^{10}$, and we can organize the data as a *term-document matrix*,

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{10\times 5}.$$

Assume that we want to find all documents that are relevant with respect to the query '**ranking** of **web pages**'. This is represented by a *query vector*, constructed in an analogous way to the term-document matrix, using the same dictionary,

$$q = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \in \mathbb{R}^{10}.$$

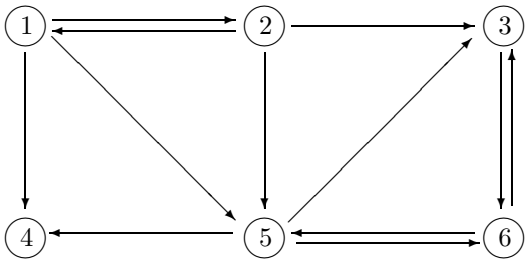


Figure 2.2.

Thus the query itself is considered as a document. The information retrieval task can now be formulated as a mathematical problem: *find the columns of A that are close to the vector q*. To solve this problem we use some distance measure in $\mathbb{R}^{10}$.

In information retrieval it is common that m is large, of the order 10^6, say. As most of the documents only contain a small fraction of the terms in the dictionary, the matrix is *sparse*.

In some methods for information retrieval, linear algebra techniques (*e.g.*, singular value decomposition (SVD)) are used for data compression and retrieval enhancement. We discuss vector space methods for information retrieval in Section 4.

The very idea of data mining is to extract useful information from large and often unstructured sets of data. Therefore it is necessary that the methods used are efficient and often specially designed for large problems. In some data mining applications huge matrices occur.

Example 2.3. The task of extracting information from all the web pages available on the Internet is performed by *search engines*. The core of the Google search engine[2] is a matrix computation, probably the largest that is performed routinely (Moler 2002). The Google matrix P is assumed to be of dimension of the order billions (2005), and it is used as a model of (all) the web pages on the Internet.

In the Google PageRank algorithm the problem of assigning ranks to all the web pages is formulated as a matrix eigenvalue problem. Let all web pages be ordered from 1 to n, and let i be a particular web page. Then O_i will denote the set of pages that i is linked to, the *outlinks*. The number of outlinks is denoted $N_i = |O_i|$. The set of *inlinks*, denoted I_i, are the pages

[2] `http://www.google.com`.

that have an outlink to i. Now define Q to be a square matrix of dimension n, and let

$$Q_{ij} = \begin{cases} 1/N_j, & \text{if there is a link from } j \text{ to } i, \\ 0, & \text{otherwise.} \end{cases}$$

This definition means that row i has nonzero elements in those positions that correspond to inlinks of i. Similarly, column j has nonzero elements equal to $1/N_j$ in those positions that correspond to the outlinks of j.

The link graph in Figure 2.2 illustrates a set of web pages with outlinks and inlinks. The corresponding matrix becomes

$$Q = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{1}{3} & 0 \end{pmatrix}.$$

Define a vector r, which holds the ranks of all pages. The vector r is then defined[3] as the eigenvector corresponding to the eigenvalue $\lambda = 1$ of Q:

$$\lambda r = Qr. \tag{2.1}$$

We shall discuss some numerical aspects of the PageRank computation in Section 6.1.

3. Data compression: low-rank approximation

3.1. Wedderburn rank reduction

One way of measuring the information contents in a data matrix is to compute its rank. Obviously, linearly dependent column or row vectors are redundant, as they can be replaced by linear combinations of the other, linearly independent columns. Therefore, one natural procedure for extracting information from a data matrix is to systematically determine a sequence of linearly independent vectors, and deflate the matrix by subtracting rank-one matrices, one at a time. It turns out that this *rank reduction procedure* is closely related to *matrix factorization*, *data compression*, *dimension reduction*, and *feature selection/extraction*. The key link between the concepts is the *Wedderburn rank reduction theorem*.

[3] This definition is provisional since it does not take into account the mathematical properties of Q: as the problem is formulated so far, there is usually no unique solution of the eigenvalue problem.

Theorem 3.1. (Wedderburn 1934) Suppose $A \in \mathbb{R}^{m\times n}$, $f \in \mathbb{R}^{n\times 1}$, and $g \in \mathbb{R}^{m\times 1}$. Then

$$\operatorname{rank}(A - \omega^{-1} A f g^T A) = \operatorname{rank}(A) - 1,$$

if and only if $\omega = g^T A f \neq 0$.

Based on Theorem 3.1 a stepwise rank reduction procedure can be defined: Let $A^{(1)} = A$, and define a sequence of matrices $\{A^{(i)}\}$

$$A^{(i+1)} = A^{(i)} - \omega_i^{-1} A^{(i)} f^{(i)} {g^{(i)}}^T A^{(i)}, \tag{3.1}$$

for any vectors $f^{(i)} \in \mathbb{R}^{n\times 1}$ and $g^{(i)} \in \mathbb{R}^{m\times 1}$, such that

$$\omega_i = {g^{(i)}}^T A^{(i)} f^{(i)} \neq 0. \tag{3.2}$$

The sequence defined in (3.1) terminates in $r = \operatorname{rank}(A)$ steps, since each time the rank of the matrix decreases by one. This process is called a *rank-reducing process* and the matrices $A^{(i)}$ are called Wedderburn matrices. For details, see Chu, Funderlic and Golub (1995). The process gives a matrix *rank-reducing decomposition*,

$$A = \hat{F} \Omega^{-1} \hat{G}^T, \tag{3.3}$$

where

$$\hat{F} = \left(\hat{f}_1, \ldots, \hat{f}_r\right) \in \mathbb{R}^{m\times r}, \quad \hat{f}_i = A^{(i)} f^{(i)}, \tag{3.4}$$

$$\Omega = \operatorname{diag}(\omega_1, \ldots, \omega_r) \in \mathbb{R}^{r\times r}, \tag{3.5}$$

$$\hat{G} = \left(\hat{g}_1, \ldots, \hat{g}_r\right) \in \mathbb{R}^{n\times r}, \quad \hat{g}_i = {A^{(i)}}^T g^{(i)}. \tag{3.6}$$

Theorem 3.1 can be generalized to the case where the reduction of rank is larger than one, as shown in the next theorem.

Theorem 3.2. (Guttman 1957) Suppose $A \in \mathbb{R}^{m\times n}$, $F \in \mathbb{R}^{n\times k}$, and $G \in \mathbb{R}^{m\times k}$. Then

$$\operatorname{rank}(A - AFR^{-1}G^T A) = \operatorname{rank}(A) - \operatorname{rank}(AFR^{-1}G^T A), \tag{3.7}$$

if and only if $R = G^T A F \in \mathbb{R}^{k\times k}$ is nonsingular.

Chu *et al.* (1995) discuss Wedderburn rank reduction from the point of view of solving linear systems of equations. There are many choices of F and G that satisfy the condition (3.7). Therefore, various rank-reducing decompositions (3.3) are possible. It is shown that several standard matrix factorizations in numerical linear algebra are instances of the Wedderburn formula: Gram–Schmidt orthogonalization, singular value decomposition, QR and Cholesky decomposition, as well as the Lanczos procedure.

A complementary view is taken in data analysis[4] (see Hubert, Meulman and Heiser (2000)), where the Wedderburn formula and matrix factorizations are considered as tools for data analysis: 'The major purpose of a matrix factorization in this context is to obtain some form of lower-rank approximation to A for understanding the structure of the data matrix ...'.

One important difference in the way the rank reduction is treated in data analysis and in numerical linear algebra, is that in algorithm descriptions in data analysis the subtraction of the rank-one matrix $\omega^{-1}Afg^TA$ is often done explicitly, whereas in numerical linear algebra it is mostly implicit. One notable example is the Partial Least Squares method (PLS) that is widely used in chemometrics. PLS is equivalent to Lanczos bidiagonalization: see Section 3.4. This difference in description is probably the main reason why the equivalence between PLS and Lanczos bidiagonalization has not been widely appreciated in either community, even though it was pointed out quite early (Wold, Ruhe, Wold and Dunn 1984).

The application of the Wedderburn formula to data mining is further discussed in Park and Eldén (2005).

3.2. SVD, Eckart–Young optimality, and principal component analysis

We will here give a brief account of the SVD, its optimality properties for low-rank matrix approximation, and its relation to *principal component analysis* (PCA). For a more detailed exposition, see, *e.g.*, Golub and Van Loan (1996).

Theorem 3.3. Any matrix $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, can be factorized

$$A = U\Sigma V^T, \qquad \Sigma = \begin{pmatrix} \Sigma_0 \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n}, \qquad \Sigma_0 = \operatorname{diag}(\sigma_1, \ldots, \sigma_n),$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal, and $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_n \geq 0$.

The assumption $m \geq n$ is no restriction. The σ_i are the *singular values*, and the columns of U and V are *left and right singular vectors*, respectively.

Suppose that A has rank r. Then $\sigma_r > 0$, $\sigma_{r+1} = 0$, and

$$A = U\Sigma V^T = \begin{pmatrix} U_r & \hat{U}_r \end{pmatrix} \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_r^T \\ \hat{V}_r^T \end{pmatrix} = U_r\Sigma_r V_r^T, \tag{3.8}$$

where $U_r \in \mathbb{R}^{m\times r}, \Sigma_r = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$, and $V_r \in \mathbb{R}^{n\times r}$. From (3.8)

[4] It is interesting to note that several of the linear algebra ideas used in data mining were originally conceived in applied statistics and data analysis, especially in psychometrics.

we see that the columns of U and V provide bases for all four *fundamental* subspaces of A:

U_r gives an orthogonal basis for Range(A),
$\hat{V}_r$ gives an orthogonal basis for Null(A),
V_r gives an orthogonal basis for Range(A^T),
$\hat{U}_r$ gives an orthogonal basis for Null(A^T),

where Range and Null denote the range space and the null space of the matrix, respectively.

Often it is convenient to write the SVD in *outer product* form, *i.e.*, express the matrix as a sum of rank-one matrices,

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T. \tag{3.9}$$

The SVD can be used to compute the rank of a matrix. However, in floating point arithmetic, the zero singular values usually appear as small numbers. Similarly, if A is made up from a rank-k matrix and additive noise of small magnitude, then it will have k singular values that will be significantly larger than the rest. In general, a large relative gap between two consecutive singular values is considered to reflect *numerical rank deficiency* of a matrix. Therefore, 'noise reduction' can be achieved via a *truncated SVD*. If trailing *small* diagonal elements of Σ are replaced by zeros, then a rank-k approximation A_k of A is obtained as

$$\begin{aligned} A &= \begin{pmatrix} U_k & \hat{U}_k \end{pmatrix} \begin{pmatrix} \Sigma_k & 0 \\ 0 & \hat{\Sigma}_k \end{pmatrix} \begin{pmatrix} V_k^T \\ \hat{V}_k^T \end{pmatrix} \\ &\approx \begin{pmatrix} U_k & \hat{U}_k \end{pmatrix} \begin{pmatrix} \Sigma_k & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_k^T \\ \hat{V}_k^T \end{pmatrix} = U_k \Sigma_k V_k^T =: A_k, \end{aligned} \tag{3.10}$$

where $\Sigma_k \in \mathbb{R}^{k\times k}$ and $\|\hat{\Sigma}_k\| < \epsilon$ for a *small* tolerance ϵ.

The low-rank approximation of a matrix obtained in this way from the SVD has an optimality property specified in the following theorem (Eckart and Young 1936, Mirsky 1960), which is the foundation of numerous important procedures in science and engineering. An *orthogonally invariant matrix norm* is one, for which $\|QAP\| = \|A\|$, where Q and P are arbitrary orthogonal matrices (of conforming dimensions). The matrix 2-norm and the Frobenius norm are orthogonally invariant.

Theorem 3.4. Let $\|\cdot\|$ denote any orthogonally invariant norm, and let the SVD of $A \in \mathbb{R}^{m\times n}$ be given as in Theorem 3.3. Assume that an integer k is given with $0 < k \le r = \text{rank}(A)$. Then

$$\min_{\text{rank}(B)=k} \|A - B\| = \|A - A_k\|,$$

where

$$A_k = U_k \Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T. \tag{3.11}$$

From the theorem we see that the singular values indicate how close a given matrix is to a matrix of lower rank.

The relation between the truncated SVD (3.11) and the Wedderburn matrix rank reduction process can be demonstrated as follows. In the rank reduction formula (3.7), define the error matrix E as

$$E = A - AF(G^T AF)^{-1} G^T A, \qquad F \in \mathbb{R}^{n\times k}, G \in \mathbb{R}^{m\times k}.$$

Assume that $k \leq \operatorname{rank}(A) = r$, and consider the problem

$$\min \|E\| = \min_{F\in\mathbb{R}^{n\times k}, G\in\mathbb{R}^{m\times k}} \|A - AF(G^T AF)^{-1} G^T A\|,$$

where the norm is orthogonally invariant. According to Theorem 3.4, the minimum error is obtained when

$$(AF)(G^T AF)^{-1}(G^T A) = U_k \Sigma_k V_k^T,$$

which is equivalent to choosing $F = V_k$ and $G = U_k$.

This same result can be obtained by a stepwise procedure, when k pairs of vectors $f^{(i)}$ and $g^{(i)}$ are to be found, where each pair reduces the matrix rank by 1.

The Wedderburn procedure helps to elucidate the equivalence between the SVD and *principal component analysis* (PCA) (Joliffe 1986). Let $X \in \mathbb{R}^{m\times n}$ be a data matrix, where each column is an observation of a real-valued random vector. The matrix is assumed to be centred, *i.e.*, the mean of each column is equal to zero. Let the SVD of X be $X = U\Sigma V^T$. The right singular vectors v_i are called *principal component directions* of X (Hastie *et al.* 2001, p. 62). The vector

$$z_1 = Xv_1 = \sigma_1 u_1$$

has the largest sample variance amongst all normalized linear combinations of the columns of X:

$$\operatorname{Var}(z_1) = \operatorname{Var}(Xv_1) = \frac{\sigma_1^2}{m}.$$

Finding the vector of maximal variance is equivalent, using linear algebra terminology, to maximizing the Rayleigh quotient:

$$\sigma_1^2 = \max_{v\neq 0} \frac{v^T X^T X v}{v^T v}, \qquad v_1 = \arg\max_{v\neq 0} \frac{v^T X^T X v}{v^T v}.$$

The normalized variable u_1 is called the *normalized first principal component* of X. The second principal component is the vector of largest sample

variance of the deflated data matrix $X - \sigma_1 u_1 v_1^T$, and so on. Any subsequent principal component is defined as the vector of maximal variance subject to the constraint that it is orthogonal to the previous ones.

Example 3.5. PCA is illustrated in Figure 3.1. 500 data points from a correlated normal distribution were generated, and collected in a data matrix $X \in \mathbb{R}^{3\times 500}$. The data points and the principal components are illustrated in the top plot. We then deflated the data matrix: $X_1 := X - \sigma_1 u_1 v_1^T$. The data points corresponding to X_1 are given in the bottom plot; they lie on a plane in $\mathbb{R}^3$, *i.e.*, X_1 has rank 2.

The concept of principal components has been generalized to *principal curves and surfaces*: see Hastie (1984) and Hastie *et al.* (2001, Section 14.5.2). A recent paper along these lines is Einbeck, Tutz and Evers (2005).

3.3. Generalized SVD

The SVD can be used for low-rank approximation involving one matrix. It often happens that two matrices are involved in the criterion that determines the dimension reduction: see Section 4.4. In such cases a generalization of the SVD to two matrices can be used to analyse and compute the dimension reduction transformation.

Theorem 3.6. (GSVD) Let $A \in \mathbb{R}^{m\times n}$, $m \geq n$, and $B \in \mathbb{R}^{p\times n}$. Then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{p\times p}$, and a nonsingular $X \in \mathbb{R}^{n\times n}$, such that

$$U^T A X = C = \operatorname{diag}(c_1, \ldots, c_n), \quad 1 \geq c_1 \geq \cdots \geq c_n \geq 0, \tag{3.12}$$

$$V^T B X = S = \operatorname{diag}(s_1, \ldots, s_q), \quad 0 \leq s_1 \leq \cdots \leq s_q \leq 1, \tag{3.13}$$

where $q = \min(p, n)$ and

$$C^T C + S^T S = I.$$

A proof can be found in Golub and Van Loan (1996, Section 8.7.3); see also Van Loan (1976) and Paige and Saunders (1981).

The generalized SVD is sometimes called the *Quotient SVD*.[5] There is also a different generalization, called the *Product SVD*: see, *e.g.*, De Moor and Van Dooren (1992), Golub, Sølna and Van Dooren (2000).

3.4. Partial least squares: Lanczos bidiagonalization

Linear least squares (regression) problems occur frequently in data mining. Consider the minimization problem

$$\min_{\beta} \|y - X\beta\|, \tag{3.14}$$

[5] Assume that B is square and nonsingular. Then the GSVD gives the SVD of AB^{-1}.

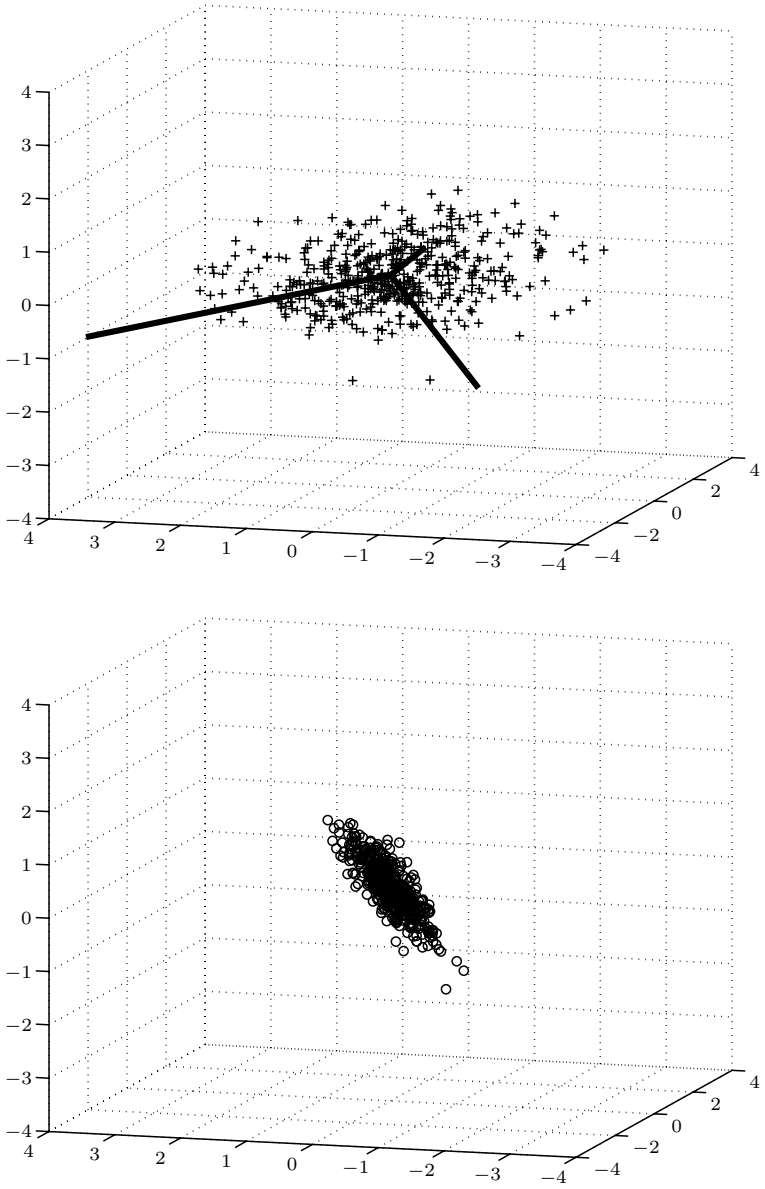


Figure 3.1. Cluster of points in $\mathbb{R}^3$ with (scaled) principal components (top). The same data with the contributions along the first principal component deflated (bottom).

where X is an $m \times n$ real matrix, and the norm is the Euclidean vector norm. This is the *linear least squares problem* in numerical linear algebra, and the *multiple linear regression problem* in statistics. Using regression terminology, the vector y consists of observations of a response variable, and the columns of X contain the values of the explanatory variables. Often the matrix is large and ill-conditioned: the column vectors are (almost) linearly dependent. Sometimes, in addition, the problem is under-determined, *i.e.*, $m < n$. In such cases the straightforward solution of (3.14) may be physically meaningless (from the point of view of the application at hand) and difficult to interpret. Then one may want to express the solution by projecting it onto a lower-dimensional subspace: let W be an $n \times k$ matrix with orthonormal columns. Using this as a basis for the subspace, one considers the approximate minimization

$$\min_{\beta} \|y - X\beta\| \approx \min_{z} \|y - XWz\|. \tag{3.15}$$

One obvious method for projecting the solution onto a low-dimensional subspace is *principal components regression* (*PCR*) (Massy 1965), where the columns of W are chosen as right singular vectors from the SVD of X. In numerical linear algebra this is called *truncated singular value decomposition* (TSVD). Another such projection method, the *partial least squares* (PLS) method (Wold 1975), is standard in chemometrics (Wold, Sjöström and Eriksson 2001). It has been known for quite some time (Wold *et al.* 1984) (see also Helland (1988), Di Ruscio (2000), Phatak and de Hoog (2002)) that PLS is equivalent to Lanczos (Golub–Kahan) bidiagonalization (Golub and Kahan 1965, Paige and Saunders 1982) (we will refer to this as LBD). The equivalence is further discussed in Eldén (2004*b*), and the properties of PLS are analysed using the SVD.

There are several variants of PLS: see, *e.g.*, Frank and Friedman (1993). The following is the so-called NIPALS version.

The NIPALS PLS algorithm

---

1 $X_0 = X$

2 **for** $\boldsymbol{i = 1, 2, \ldots, k}$

(a) $w_i = \frac{1}{\|X_{i-1}^T y\|} X_{i-1}^T y$

(b) $t_i = \frac{1}{\|X_{i-1} w_i\|} X_{i-1} w_i$

(c) $p_i = X_{i-1}^T t_i$

(d) $X_i = X_{i-1} - t_i p_i^T$

In the statistics/chemometrics literature the vectors w_i, t_i, and p_i are called *weight*, *score*, and *loading vectors*, respectively.

It is obvious that PLS is a Wedderburn procedure. One advantage of PLS for regression is that the basis vectors in the solution space (the columns of W (3.15)) are influenced by the right-hand side.[6] This is not the case in PCR, where the basis vectors are singular vectors of X. Often PLS gives a higher reduction of the norm of the residual $y - X\beta$ for small values of k than does PCR.

The Lanczos bidiagonalization procedure can be started in different ways: see, *e.g.*, Björck (1996, Section 7.6). It turns out that PLS corresponds to the following formulation.

Lanczos Bidiagonalization (LBD)

1 $v_1 = \frac{1}{\|X^T y\|} X^T y; \quad \alpha_1 u_1 = X v_1$
2 **for** $\boldsymbol{i = 2, \ldots, k}$
(a) $\gamma_{i-1} v_i = X^T u_{i-1} - \alpha_{i-1} v_{i-1}$
(b) $\alpha_i u_i = X v_i - \gamma_{i-1} u_{i-1}$

The coefficients γ_{i-1} and α_i are determined so that $\|v_i\| = \|u_i\| = 1$.

Both algorithms generate two sets of orthogonal basis vectors: $(w_i)_{i=1}^k$ and $(t_i)_{i=1}^k$ for PLS, $(v_i)_{i=1}^k$ and $(u_i)_{i=1}^k$ for LBD. It is straightforward to show (directly using the equations defining the algorithm – see Eldén (2004*b*)) that the two methods are equivalent.

Proposition 3.7. The PLS and LBD methods generate the same orthogonal bases, and the same approximate solution, $\beta_{\text{pls}}^{(k)} = \beta_{\text{lbd}}^{(k)}$.

4. Text mining

By text mining we understand methods for extracting useful information from large and often unstructured collections of texts. A related term is *information retrieval*. A typical application is search in databases of abstract of scientific papers. For instance, in medical applications one may want to find all the abstracts in the database that deal with a particular syndrome. So one puts together a search phrase, a *query*, with key words that are relevant to the syndrome. Then the retrieval system is used to match the query to the documents in the database, and present to the user all the documents that are relevant, preferably ranked according to relevance.

[6] However, it is not always appreciated that the dependence of the basis vectors on the right-hand side is non-linear and quite complicated. For a discussion of these aspects of PLS, see Eldén (2004*b*).

Example 4.1. The following is a typical query:

9. *the use of induced hypothermia in heart surgery, neurosurgery, head injuries and infectious diseases.*

The query is taken from a test collection of medical abstracts, called Medline.[7] We will refer to this query as Q9 from here on.

Another well-known area of text mining is web search engines. There the search phrase is usually very short, and often there are so many relevant documents that it is out of the question to present them all to the user. In that application the ranking of the search result is critical for the efficiency of the search engine. We will come back to this problem in Section 6.1.

For overviews of information retrieval, see, *e.g.*, Korfhage (1997) and Grossman and Frieder (1998). In this section we will describe briefly one of the most common methods for text mining, namely the *vector space model* (Salton, Yang and Wong 1975). In Example 2.2 we demonstrated the basic ideas of the construction of a term-document matrix in the vector space model. Below we first give a very brief overview of the preprocessing that is usually done before the actual term-document matrix is set up. Then we describe a variant of the vector space model: *latent semantic indexing* (LSI) (Deerwester, Dumais, Furnas, Landauer and Harsman 1990), which is based on the SVD of the term-document matrix. For a more detailed account of the different techniques used in connection with the vector space model, see Berry and Browne (2005).

4.1. Vector space model: preprocessing and query matching

In information retrieval, key words that carry information about the contents of a document are called *terms*. A basic task is to create a list of all the terms in alphabetic order, a so-called *index*. But before the index is made, two preprocessing steps should be done: (1) eliminate all stop words, (2) perform stemming.

Stop words are extremely common words. The occurrence of such a word in a document does not distinguish it from other documents. The following is the beginning of one stop list:[8]

a, a's, able, about, above, according, accordingly, across, actually, after, afterwards, again, against, ain't, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, ...

[7] See, *e.g.*, `http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/`
[8] `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`

Stemming is the process of reducing each word that is conjugated or has a suffix to its stem. Clearly, from the point of view of information retrieval, no information is lost in the following reduction:

$$\left.\begin{array}{c}\textbf{comput}\text{able}\\ \textbf{comput}\text{ation}\\ \textbf{comput}\text{ing}\\ \textbf{comput}\text{ed}\\ \textbf{comput}\text{ational}\end{array}\right\} \longrightarrow \quad \textbf{comput}$$

Public domain stemming algorithms are available on the Internet.[9]

A number of pre-processed documents are parsed,[10] giving a term-document matrix $A \in \mathbb{R}^{m\times n}$, where m is the number of terms in the dictionary and n is the number of documents. It is common not only to count the occurrence of terms in documents but also to apply a *term-weighting scheme*, where the elements of A are weighted depending on the characteristics of the document collection. Similarly, document weighting is usually done. A number of schemes are described in Berry and Browne (2005, Section 3.2.1). For example, one can define the elements in A by

$$a_{ij} = f_{ij}\log(n/n_i), \tag{4.1}$$

where f_{ij} is the term frequency, the number of times term i appears in document j, and n_i is the number of documents that contain term i (inverse document frequency). If a term occurs frequently in only a few documents, then both factors are large. In this case the term discriminates well between different groups of documents, and it gets a large weight in the documents where it appears.

Normally, the term-document matrix is *sparse*: most of the matrix elements are equal to zero. Then, of course, one avoids storing all the zeros, and instead uses a sparse matrix storage scheme (see, *e.g.*, Saad (2003, Chapter 3) and Goharian, Jain and Sun (2003)).

Example 4.2. For the stemmed Medline collection (*cf.* Example 4.1) the matrix (including 30 query columns) is 4163×1063 with 48263 nonzero elements, *i.e.*, approximately 1%. The first 500 rows and columns of the matrix are illustrated in Figure 4.1.

The query (*cf.* Example 4.1) is parsed using the same dictionary as the documents, giving a vector $q \in \mathbb{R}^m$. *Query matching* is the process of finding

[9] `http://www.tartarus.org/~martin/PorterStemmer/`.

[10] Public domain text parsers are described in Giles, Wo and Berry (2003) and Zeimpekis and Gallopoulos (2005).

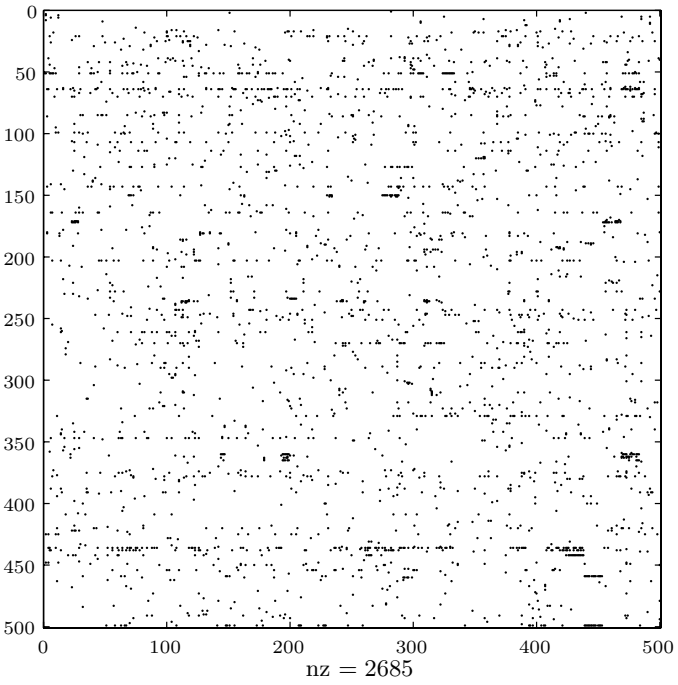


Figure 4.1. The first 500 rows and columns of the Medline matrix. Each dot represents a nonzero element.

all documents that are considered relevant to a particular query q. This is often done using the cosine distance measure: *all documents are returned for which*

$$\frac{q^T a_j}{\|q\|_2 \, \|a_j\|_2} > \text{tol}, \tag{4.2}$$

where tol is a predefined tolerance. If the tolerance is lowered, then more documents are returned, and then it is likely that more of the documents that are relevant to the query are returned. But at the same time there is a risk that more documents that are not relevant are also returned.

Example 4.3. We did query matching for query Q9 in the stemmed Medline collection. With tol = 0.19 only document 409 was considered relevant. When the tolerance was lowered to 0.17, then documents 409, 415, and 467 were retrieved.

We illustrate the different categories of documents in a query matching for two values of the tolerance in Figure 4.2. The query matching produces a good result when the intersection between the two sets of returned and

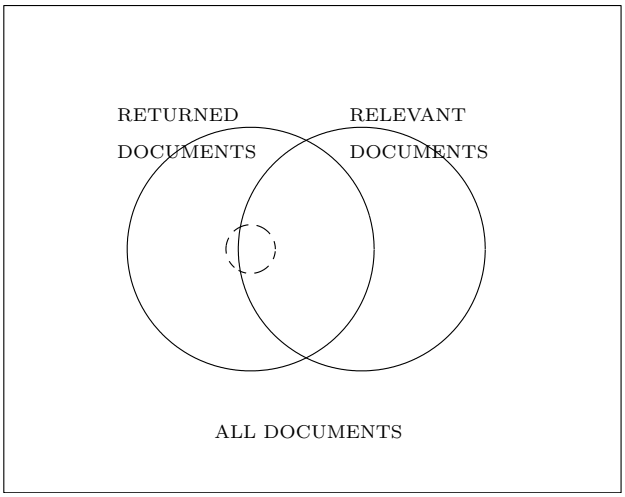


Figure 4.2. Returned and relevant documents for two values of the tolerance. The dashed circle represents the returned documents for a high value of the cosine tolerance.

relevant documents is as large as possible, and the number of returned irrelevant documents is small. For a high value of the tolerance, the retrieved documents are likely to be relevant (the small circle in Figure 4.2). When the cosine tolerance is lowered, then the intersection is increased, but at the same time, more irrelevant documents are returned.

In performance modelling for information retrieval we define the following measures:

$$\textit{precision} \qquad P = \frac{D_r}{D_t},$$

where D_r is the number of relevant documents retrieved, and D_t the total number of documents retrieved; and

$$\textit{recall} \qquad R = \frac{D_r}{N_r},$$

where N_r is the total number of relevant documents in the data base. With the cosine measure, we see that with a large value of tol we have high precision, but low recall. For a small value of tol we have high recall, but low precision.

In the evaluation of different methods and models for information retrieval usually a number of queries are used. For testing purposes all documents have been read by a human and those that are relevant to a certain query are marked.

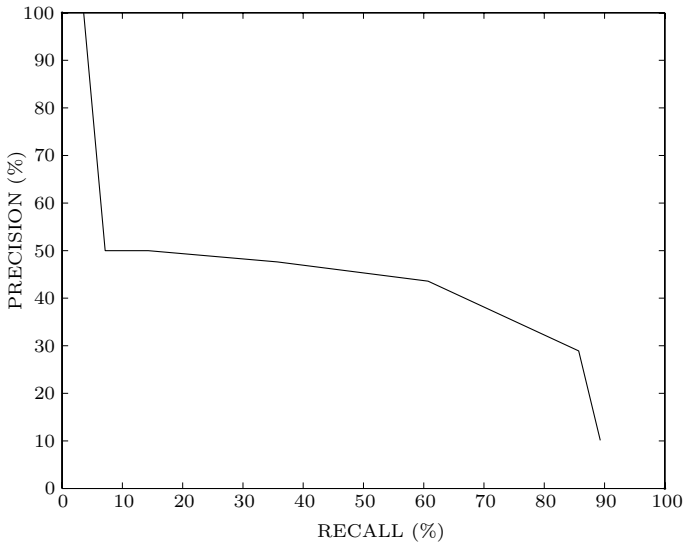


Figure 4.3. Query matching for Q9 using the vector space method. Recall versus precision.

Example 4.4. We did query matching for query Q9 in the Medline collection (stemmed) using the cosine measure, and obtained recall and precision as illustrated in Figure 4.3. In the comparison of different methods it is more illustrative to draw the recall versus precision diagram. Ideally a method has high recall at the same time as the precision is high. Thus, the closer the curve is to the upper right corner, the better the method.

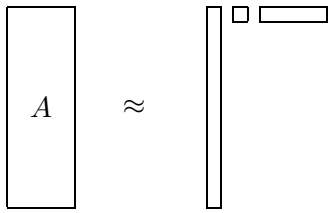
In this example and the following examples the matrix elements were computed using term frequency and inverse document frequency weighting (4.1).

4.2. LSI: latent semantic indexing

Latent semantic indexing[11] (LSI) 'is based on the assumption that there is some underlying latent semantic structure in the data ... that is corrupted by the wide variety of words used ...' (quoted from Park, Jeon and Rosen (2001)) and that this semantic structure can be enhanced by projecting the data (the term-document matrix and the queries) onto a lower-dimensional space using the singular value decomposition. LSI is discussed in Deerwester *et al.* (1990), Berry, Dumais and O'Brien (1995), Berry, Drmac and Jessup (1999), Berry (2001), Jessup and Martin (2001) and Berry and Browne (2005).

[11] Sometimes also called *latent semantic analysis* (LSA) (Jessup and Martin 2001).

Let $A = U\Sigma V^T$ be the SVD of the term-document matrix and approximate it by a matrix of rank k:



$$= U_k(\Sigma_k V_k) =: U_k D_k.$$

The columns of U_k live in the document space and are an orthogonal basis that we use to approximate all the documents: column j of D_k holds the coordinates of document j in terms of the orthogonal basis. With this k-dimensional approximation the term-document matrix is represented by $A_k = U_k D_k$, and in query matching we compute $q^T A_k = q^T U_k D_k = (U_k^T q)^T D_k$. Thus, we compute the coordinates of the query in terms of the new document basis and compute the cosines from

$$\cos\theta_j = \frac{q_k^T (D_k e_j)}{\|q_k\|_2 \, \|D_k e_j\|_2}, \qquad q_k = U_k^T q. \tag{4.3}$$

This means that the query matching is performed in a k-dimensional space.

Example 4.5. We did query matching for Q9 in the Medline collection, approximating the matrix using the truncated SVD of rank 100. The recall–precision curve is given in Figure 4.4. It is seen that for this query LSI improves the retrieval performance. In Figure 4.5 we also demonstrate a fact that is common to many term-document matrices: it is rather well conditioned, and there is no gap in the sequence of singular values. Therefore, we cannot find a suitable rank of the LSI approximation by inspecting the singular values: it must be determined by retrieval experiments.

Another remarkable fact is that with $k = 100$ the approximation error in the matrix approximation,

$$\frac{\|A - A_k\|_F}{\|A\|_F} \approx 0.8,$$

is large, and we still get *improved retrieval performance*. In view of the large approximation error in the truncated SVD approximation of the term-document matrix, one may question whether the 'optimal' singular vectors constitute the best basis for representing the term-document matrix. On the other hand, since we get such good results, perhaps a more natural conclusion may be that the Frobenius norm is not a good measure of the information contents in the term-document matrix.

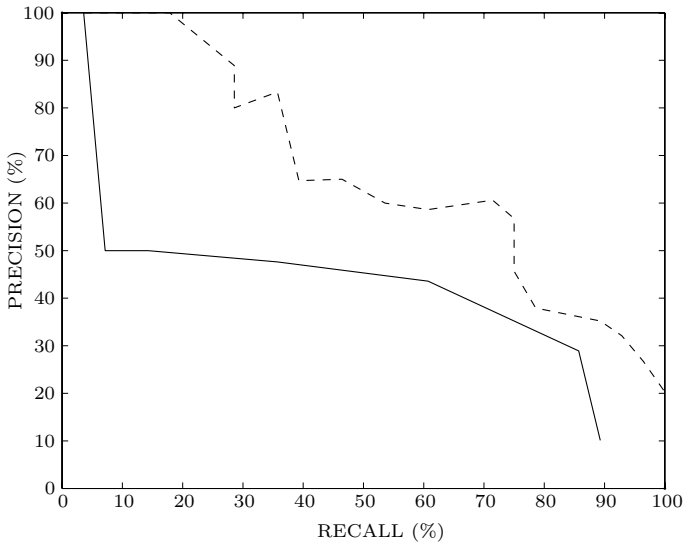


Figure 4.4. Query matching for Q9. Recall versus precision for the full vector space model (solid line) and the rank-100 approximation (dashed).

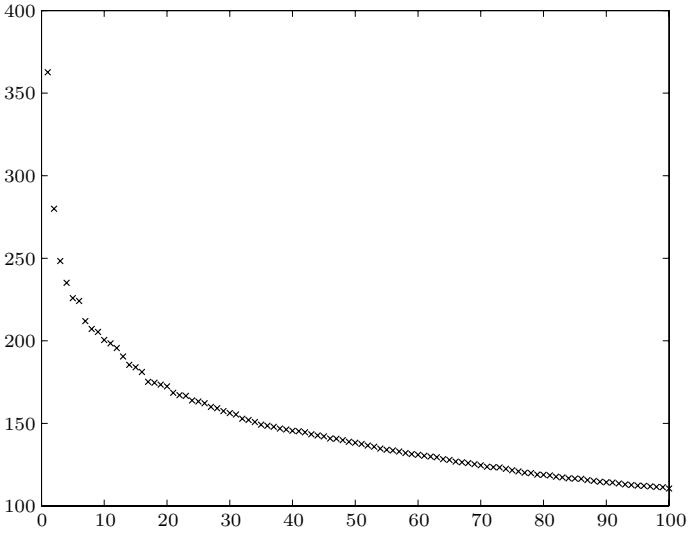


Figure 4.5. First 100 singular values of the Medline (stemmed) matrix.

It is also interesting to see what are the most important 'directions' in the data. From Theorem 3.4 we know that the first few left singular vectors are the dominant directions in the document space, and their largest components should indicate what these directions are. The Matlab statements `find(abs(U(:,k))>0.13)`, combined with look-up in the dictionary of terms, gave the results shown in Table 4.1, for `k=1,2`.

Table 4.1.

`U(:,1)`	`U(:,2)`
cell	case
growth	cell
hormone	children
patient	defect
	dna
	growth
	patient
	ventricular

It should be said that LSI does not give significantly better results for all queries in the Medline collection: there are some where it gives results comparable to the full vector model, and some where it gives worse performance. However, it is often the average performance that matters.

Jessup and Martin (2001) made a systematic study of different aspects of LSI. They showed that LSI improves retrieval performance for surprisingly small values of the reduced rank k. At the same time the relative matrix approximation errors are large. It is probably not possible to prove any general results for LSI that explain how and for which data it can improve retrieval performance. Instead we give an artificial example (constructed using ideas similar to those of a corresponding example in Berry and Browne (2005)) that gives a partial explanation.

Example 4.6. Consider the term-document matrix from Example 2.2, and the query '**ranking** of **web pages**'. Obviously, Documents 1–4 are relevant with respect to the query, while Document 5 is totally irrelevant. However, we obtain the following cosines for query and the original data:

$$\begin{pmatrix} 0 & 0.6667 & 0.7746 & 0.3333 & 0.3333 \end{pmatrix}.$$

We then compute the SVD of the term-document matrix, and use a rank-two approximation. After projection to the two-dimensional subspace the

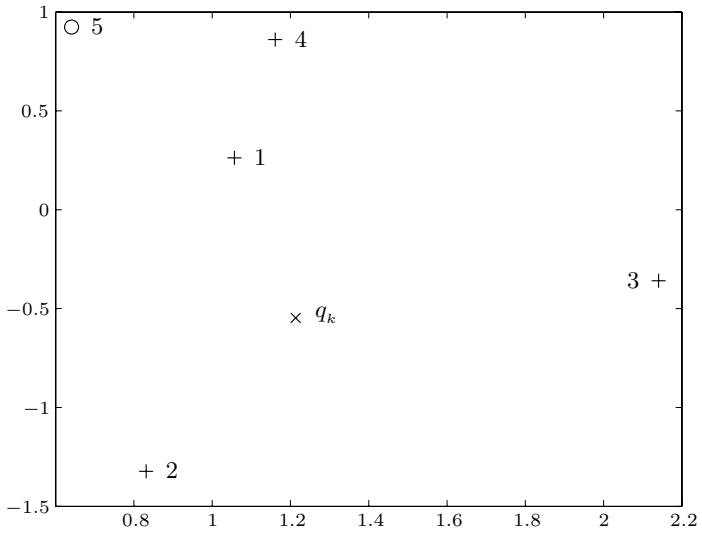


Figure 4.6. The documents and the query projected to the coordinate system of the first two left singular vectors.

cosines, computed according to (4.3), are

$$\begin{pmatrix} 0.7857 & 0.8332 & 0.9670 & 0.4873 & 0.1819 \end{pmatrix}.$$

It turns out that Document 1, which was deemed totally irrelevant to the query in the original representation, is now highly relevant. In addition, the scores for the relevant Documents 2–4 have been reinforced. At the same time, the score for Document 5 has been significantly reduced. Thus, in this artificial example, the dimension reduction enhanced the retrieval performance. The improvement may be explained as follows.

In Figure 4.6 we plot the five documents and the query in the coordinate system of the first two left singular vectors. Obviously, in this representation, the first document is is closer to the query than Document 5. The first two left singular vectors are

$$u_1 = \begin{pmatrix} 0.1425 \\ 0.0787 \\ 0.0787 \\ 0.3924 \\ 0.1297 \\ 0.1020 \\ 0.5348 \\ 0.3647 \\ 0.4838 \\ 0.3647 \end{pmatrix}, \qquad \begin{pmatrix} 0.2430 \\ 0.2607 \\ 0.2607 \\ -0.0274 \\ 0.0740 \\ -0.3735 \\ 0.2156 \\ -0.4749 \\ 0.4023 \\ -0.4749 \end{pmatrix},$$

and the singular values are $\Sigma = \mathrm{diag}(2.8546, 1.8823, 1.7321, 1.2603, 0.8483)$. The first four columns in A are strongly coupled via the words *Google*, *matrix*, *etc.*, and those words are the dominating contents of the document collection (*cf.* the singular values). This shows in the composition of u_1. So even if none of the words in the query is matched by Document 1, that document is so strongly correlated to the the dominating direction that it becomes relevant in the reduced representation.

4.3. Clustering and least squares

Clustering is widely used in pattern recognition and data mining. We give here a brief account of the application of clustering to text mining.

Clustering is the grouping together of similar objects. In the vector space model for text mining, similarity is defined as the distance between points in $\mathbb{R}^m$, where m is the number of terms in the dictionary. There are many clustering methods, *e.g.*, the k-means method, agglomerative clustering, self-organizing maps, and multi-dimensional scaling: see the references in Dhillon (2001), Dhillon, Fan and Guan (2001).

The relation between the SVD and clustering is explored in Dhillon (2001); see also Zha, Ding, Gu, He and Simon (2002) and Dhillon, Guan and Kulis (2005). Here the approach is graph-theoretic. The sparse term-document matrix represents a bi-partite graph, where the two sets of vertices are the documents $\{d_j\}$ and the terms $\{t_i\}$. An edge (t_i, d_j) exists if term t_i occurs in document d_j, *i.e.*, if the element in position (i, j) is nonzero. Clustering the documents is then equivalent to partitioning the graph. A *spectral partitioning* method is described, where the eigenvectors of a Laplacian of the graph are optimal partitioning vectors. Equivalently, the singular vectors of a related matrix can be used. It is of some interest that spectral clustering methods are related to algorithms for the partitioning of meshes in parallel finite element computations: see, *e.g.*, Simon, Sohn and Biswas (1998).

Clustering for text mining is discussed in Dhillon and Modha (2001) and Park, Jeon and Rosen (2003), and the similarities between LSI and clustering are pointed out in Dhillon and Modha (2001).

Given a partitioning of a term-document matrix into k clusters,

$$A = \begin{pmatrix} A_1 & A_2 & \cdots & A_k \end{pmatrix}, \tag{4.4}$$

where $A_j \in \mathbb{R}^{n_j}$, one can take the *centroid* of each cluster,[12]

$$c^{(j)} = \frac{1}{n_j} A_j e^{(j)}, \qquad e^{(j)} = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}^T, \tag{4.5}$$

with $e^{(j)} \in \mathbb{R}^{n_j}$, as a representative of the class. Together the centroid

[12] In Dhillon and Modha (2001) normalized centroids are called *concept vectors*.

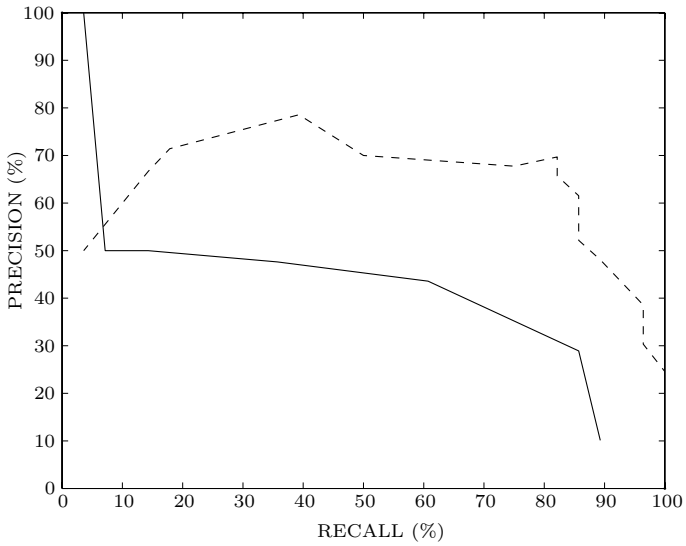


Figure 4.7. Query matching for Q9. Recall versus precision for the full vector space model (solid line) and the rank-50 centroid approximation (dashed).

vectors can be used as an approximate basis for the document collection, and the coordinates of each document with respect to this basis can be computed by solving the least squares problem

$$\min_D \|A - CD\|_F, \qquad C = \begin{pmatrix} c^{(1)} & c^{(2)} & \cdots & c^{(k)} \end{pmatrix}. \tag{4.6}$$

Example 4.7. We did query matching for Q9 in the Medline collection. Before computing the clustering we normalized the columns to equal Euclidean length. We approximated the matrix using the orthonormalized centroids from a clustering into 50 clusters. The recall–precision diagram is given in Figure 4.7. We see that for high values of recall, the centroid method is as good as the LSI method with double the rank: see Figure 4.4.

For rank 50 the approximation error in the centroid method,

$$\|A - CD\|_F / \|A\|_F \approx 0.9,$$

is even higher than for LSI of rank 100.

The improved performance can be explained in a similar way as for LSI. Being the 'average document' of a cluster, the centroid captures the main links between the dominant documents in the cluster. By expressing all documents in terms of the centroids, the dominant links are emphasized.

4.4. Clustering and linear discriminant analysis

When centroids are used as basis vectors, the coordinates of the documents are computed from (4.6) as

$$D := G^T A, \qquad G^T = R^{-1}Q^T,$$

where $C = QR$ is the thin QR decomposition[13] of the centroid matrix. The criterion for choosing G is based on approximating the term-document matrix A as well as possible in the Frobenius norm. As we have seen earlier (Examples 4.5 and 4.7), a good approximation of A is not always necessary for good retrieval performance, and it may be natural to look for other criteria for determining the matrix G in a dimension reduction.

Linear discriminant analysis (LDA) is frequently used for classification (Duda *et al.* 2001). In the context of cluster-based text mining, LDA is used to derive a transformation G, such that the cluster structure is as well preserved as possible in the dimension reduction.

In Howland, Jeon and Park (2003) and Howland and Park (2004) the application of LDA to text mining is explored, and it is shown how the GSVD (Theorem 3.6) can be used to extend the dimension reduction procedure to cases where the standard LDA criterion is not valid.

Assume that a clustering of the documents has been made as in (4.4) with centroids (4.5). Define the overall centroid

$$c = Ae, \qquad e = \frac{1}{\sqrt{n}}\begin{pmatrix}1 & 1 & \cdots & 1\end{pmatrix}^T,$$

the three matrices[14]

$$\begin{aligned}
\mathbb{R}^{m\times n} \ni H_w &= \begin{pmatrix} A_1 - c^{(1)}e^{(1)^T} & A_2 - c^{(2)}e^{(2)^T} & \cdots & A_k - c^{(k)}e^{(k)^T}\end{pmatrix},\\
\mathbb{R}^{m\times k} \ni H_b &= \begin{pmatrix}\sqrt{n_1}(c^{(1)} - c) & \sqrt{n_2}(c^{(2)} - c) & \cdots & \sqrt{n_k}(c^{(k)} - c)\end{pmatrix},\\
\mathbb{R}^{m\times n} \ni H_m &= A - c\,e^T,
\end{aligned}$$

and the corresponding *scatter matrices*

$$\begin{aligned}
S_w &= H_w H_w^T,\\
S_b &= H_b H_b^T,\\
S_m &= H_m H_m^T.
\end{aligned}$$

Assume that we want to use the (dimension-reduced) clustering for classifying new documents, *i.e.*, determine to which cluster they belong. The 'quality of the clustering' with respect to this task depends on how 'tight'

[13] The *thin QR decomposition* of a matrix $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, is $A = QR$, where $Q \in \mathbb{R}^{m\times n}$ has orthonormal columns and R is upper triangular.

[14] Note: subscript w for 'within classes', b for 'between classes'.

or coherent each cluster is, and how well separated the clusters are. The overall tightness ('within-class scatter') of the clustering can be measured as

$$J_w = \mathrm{tr}(S_w) = \|H_w\|_F^2,$$

and the separateness ('between-class scatter') of the clusters by

$$J_b = \mathrm{tr}(S_b) = \|H_b\|_F^2.$$

Ideally, the clusters should be separated at the same time as each cluster is tight. Different quality measures can be defined. Often in LDA one uses

$$J = \frac{\mathrm{tr}(S_b)}{\mathrm{tr}(S_w)}, \tag{4.7}$$

with the motivation that if all the clusters are tight then S_w is small, and if the clusters are well separated then S_b is large. Thus the quality of the clustering with respect to classification is high if J is large. Similar measures are considered in Howland and Park (2004).

Now assume that we want to determine a dimension reduction transformation, represented by the matrix $G \in \mathbb{R}^{m\times d}$, such that the quality of the reduced representation is as high as possible. After the dimension reduction, the tightness and separateness are

$$J_b(G) = \|G^T H_b\|_F^2 = \mathrm{tr}(G^T S_b G),$$
$$J_w(G) = \|G^T H_w\|_F^2 = \mathrm{tr}(G^T S_w G).$$

Since $\mathrm{rank}(H_b) \le k-1$, it is only meaningful to choose $d = k-1$: see Howland *et al.* (2003).

The question arises whether it is possible to determine G so that, in a consistent way, the quotient $J_b(G)/J_w(G)$ is maximized. The answer is derived using the GSVD of H_w^T and H_b^T. We assume that $m > n$; see Howland *et al.* (2003) for a treatment of the general (but with respect to the text mining application more restrictive) case. We further assume

$$\mathrm{rank}\begin{pmatrix} H_b^T \\ H_w^T \end{pmatrix} = t.$$

Under these assumptions the GSVD has the form (Paige and Saunders 1981)

$$H_b^T = U^T \Sigma_b (Z\ 0) Q^T, \tag{4.8}$$
$$H_w^T = V^T \Sigma_w (Z\ 0) Q^T, \tag{4.9}$$

where U and V are orthogonal, $Z \in \mathbb{R}^{t\times t}$ is nonsingular, and $Q \in \mathbb{R}^{m\times m}$ is orthogonal. The diagonal matrices Σ_b and Σ_w will be specified shortly. We first see that, with

$$\widetilde{G} = Q^T G = \begin{pmatrix} \widetilde{G}_1 \\ \widetilde{G}_2 \end{pmatrix}, \qquad \widetilde{G}_1 \in \mathbb{R}^{t\times d},$$

we have

$$J_b(G) = \|\Sigma_b Z\widetilde{G}_1\|_F^2, \qquad J_w(G) = \|\Sigma_w Z\widetilde{G}_1\|_F^2. \tag{4.10}$$

Obviously, we should not waste the degrees of freedom in G by choosing a nonzero $\widetilde{G}_2$, since that would not affect the quality of the clustering after dimension reduction. Next we specify

$$\mathbb{R}^{(k-1)\times t} \ni \Sigma_b = \begin{pmatrix} I_b & 0 & 0 \\ 0 & D_b & 0 \\ 0 & 0 & 0_b \end{pmatrix},$$

$$\mathbb{R}^{n\times t} \ni \Sigma_w = \begin{pmatrix} 0_w & 0 & 0 \\ 0 & D_w & 0 \\ 0 & 0 & I_w \end{pmatrix},$$

where $I_b \in \mathbb{R}^{(t-s)\times(t-s)}$ and $I_w \in \mathbb{R}^{r\times r}$ are identity matrices with data-dependent values of r and s, and $0_b \in \mathbb{R}^{1\times r}$ and $0_w \in \mathbb{R}^{(n-s)\times(t-s)}$ are zero matrices. The diagonal matrices satisfy

$$D_b = \operatorname{diag}(\alpha_{r+1}, \ldots, \alpha_{r+s}), \quad \alpha_{r+1} \geq \cdots \geq \alpha_{r+s} > 0, \tag{4.11}$$

$$D_w = \operatorname{diag}(\beta_{r+1}, \ldots, \beta_{r+s}), \quad 0 < \beta_{r+1} \leq \cdots \leq \beta_{r+s}, \tag{4.12}$$

and $\alpha_i^2 + \beta_i^2 = 1,\ i = r+1, \ldots, r+s$. Note that the column-wise partitionings of Σ_b and Σ_w are identical. Now we define

$$\widehat{G} = Z\widetilde{G}_1 = \begin{pmatrix} \widehat{G}_1 \\ \widehat{G}_2 \\ \widehat{G}_3 \end{pmatrix},$$

where the partitioning conforms with that of Σ_b and Σ_w. Then we have

$$J_b(G) = \|\Sigma_b\widehat{G}\|_F^2 = \|\widehat{G}_1\|_F^2 + \|D_b\widehat{G}_2\|_F^2,$$
$$J_w(G) = \|\Sigma_w\widehat{G}\|_F^2 = \|D_w\widehat{G}_2\|_F^2 + \|\widehat{G}_3\|_F^2.$$

At this point we see that the maximization of

$$\frac{J_b(G)}{J_w(G)} = \frac{\operatorname{tr}(\widehat{G}^T\Sigma_b^T\Sigma_b\widehat{G})}{\operatorname{tr}(\widehat{G}^T\Sigma_w^T\Sigma_w\widehat{G})} \tag{4.13}$$

is not a well-defined problem: We can make $J_b(G)$ large simply by choosing $\widehat{G}_1$ large, without changing $J_w(G)$. On the other hand, (4.13) can be considered as the Rayleigh quotient of a generalized eigenvalue problem (see, *e.g.*, Golub and Van Loan (1996, Section 8.7.2)), where the largest set of eigenvalues are infinite (since the first eigenvalues of $\Sigma_b^T\Sigma_b$ and $\Sigma_w^T\Sigma_w$ are 1 and 0, respectively), and the following are $\alpha_{r+i}^2/\beta_{r+i}^2,\ i = 1, 2, \ldots, s$. With this in mind it is natural to constrain the data of the problem so that

$$\widehat{G}^T\widehat{G} = I. \tag{4.14}$$

We see that, under this constraint,

$$\widehat{G} = \begin{pmatrix} I \\ 0 \end{pmatrix}, \tag{4.15}$$

is a (non-unique) solution of the maximization of (4.13). Consequently, the transformation matrix G is chosen as

$$G = Q \begin{pmatrix} Z^{-1}\widehat{G} \\ 0 \end{pmatrix} = Q \begin{pmatrix} Y_1 \\ 0 \end{pmatrix},$$

where Y_1 denotes the first $k-1$ columns of Z^{-1}.

LDA-based dimension reduction was tested in Howland *et al.* (2003) on data (abstracts) from the Medline database. Classification results were obtained for the compressed data, with much better precision than using the full vector space model.

4.5. Text mining using Lanczos bidiagonalization (PLS)

In LSI and cluster-based methods, the dimension reduction is determined completely from the term-document matrix, and therefore it is the same for all query vectors. In chemometrics it has been known for a long time that PLS (Lanczos bidiagonalization) often gives considerably more efficient compression (in terms of the dimensions of the subspaces used) than PCA (LSI/SVD), the reason being that the right-hand side (of the least squares problem) determines the choice of basis vectors.

In a series of papers (see Blom and Ruhe (2005)), the use of Lanczos bidiagonalization for text mining has been investigated. The recursion starts with the normalized query vector and computes two orthonormal bases[15] P and Q.

Lanczos Bidiagonalization

---

1 $q_1 = q/\|q\|_2, \quad \beta_1 = 0, \quad p_0 = 0.$

2 **for** $\boldsymbol{i = 2, \ldots, k}$

(a) $\alpha_i p_i = A^T q_i - \beta_i p_{i-1}.$

(b) $\beta_{i+1} q_{i+1} = Apk - \alpha_i q_i.$

The coefficients α_i and β_{i+1} are determined so that $\|p_i\|_2 = \|q_{i+1}\|_2 = 1$.

---

[15] We use a slightly different notation here to emphasize that the starting vector is different from that in Section 3.4.

Define the matrices

$$Q_i = \begin{pmatrix} q_1 & q_2 & \cdots & q_i \end{pmatrix},$$
$$P_i = \begin{pmatrix} p_1 & p_2 & \cdots & p_i \end{pmatrix},$$
$$B_{i+1,i} = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \alpha_i & \\ & & \beta_{i+1} & \end{pmatrix}.$$

The recursion can be formulated as matrix equations,

$$A^T Q_i = P_i B_{i,i}^T,$$
$$AP_i = Q_{i+1} B_{i+1,i}. \tag{4.16}$$

If we compute the thin QR decomposition of $B_{i+1,i}$,

$$B_{i+1,i} = H_{i+1,i+1} R,$$

then we can write (4.16)

$$AP_i = W_i R, \qquad W_i = Q_{i+1} H_{i+1,i+1},$$

which means that the columns of W_i are an approximate orthogonal basis of the document space (*cf.* the corresponding equation $AV_i = U_i \Sigma_i$ for the LSI approximation, where we use the columns of U_i as basis vectors). Thus we have

$$A \approx W_i D_i, \qquad D_i = W_i^T A, \tag{4.17}$$

and we can use this low-rank approximation in the same way as in the LSI method.

The convergence of the recursion can be monitored by computing the residual $\|AP_i z - q\|_2$. It is easy to show (see, *e.g.*, Blom and Ruhe (2005)) that this quantity is equal in magnitude to a certain element in the matrix $H_{i+1,i+1}$. When the residual is smaller than a prescribed tolerance, the approximation (4.17) is deemed good enough for this particular query.

In this approach the matrix approximation is recomputed for every query. This has the following advantages.

(1) Since the right-hand side influences the choice of basis vectors, only a very few steps of the bidiagonalization algorithm need be taken. Blom and Ruhe (2005) report tests for which this algorithm performed better, with $k = 3$, than LSI with subspace dimension 259.

(2) The computation is relatively cheap, the dominating cost being a small number of matrix-vector multiplications.

(3) Most information retrieval systems change with time, when new documents are added. In LSI this necessitates the updating of the SVD of the term-document matrix. Unfortunately, it is quite expensive to update an SVD. The Lanczos-based method, on the other hand, adapts immediately and at no extra cost to changes of A.

5. Classification and pattern recognition

5.1. Classification of handwritten digits using SVD bases

Computer classification of handwritten digits is a standard problem in pattern recognition. The typical application is automatic reading of zip codes on envelopes. A comprehensive review of different algorithms is given in LeCun, Bottou, Bengio and Haffner (1998).

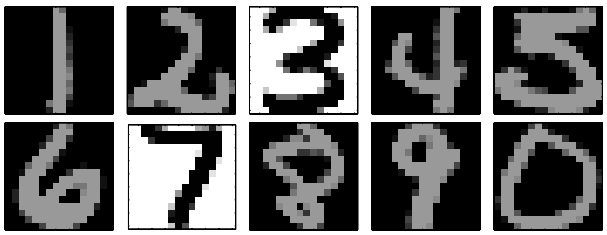


Figure 5.1. Handwritten digits from the US Postal Service database.

In Figure 5.1 we illustrate handwritten digits that we will use in the examples in this section.

We will treat the digits in three different, but equivalent ways:

(1) 16×16 grey-scale images,

(2) functions of two variables,

(3) vectors in $\mathbb{R}^{256}$.

In the classification of an unknown digit it is necessary to compute the distance to known digits. Different distance measures can be used, perhaps the most natural is Euclidean distance: stack the columns of the image in a vector and identify each digit as a vector in $\mathbb{R}^{256}$. Then define the distance function

$$\text{dist}(x, y) = \|x - y\|_2.$$

An alternative distance function can be based on the cosine between two vectors.

In a real application of recognition of handwritten digits, *e.g.*, zip code reading, there are hardware and real time factors that must be taken into account. In this section we will describe an idealized setting. The problem is:

Given a set of of manually classified digits (the training set), classify a set of unknown digits (the test set).

In the US Postal Service database, the training set contains 7291 handwritten digits, and the test set has 2007 digits.

When we consider the training set digits as vectors or points, then it is reasonable to assume that all digits of one kind form a cluster of points in a Euclidean 256-dimensional vector space. Ideally the clusters are well separated and the separation depends on how well written the training digits are.

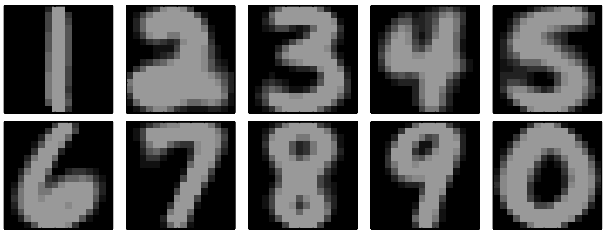


Figure 5.2. The means (centroids) of all digits in the training set.

In Figure 5.2 we illustrate the means (centroids) of the digits in the training set. From this figure we get the impression that a majority of the digits are well written (if there were many badly written digits this would demonstrate itself as diffuse means). This means that the clusters are rather well separated. Therefore it is likely that a simple algorithm that computes the distance from each unknown digit to the means should work rather well.

A simple classification algorithm

Training. Given the training set, compute the mean (centroid) of all digits of one kind.

Classification. For each digit in the test set, compute the distance to all ten means, and classify as the closest.

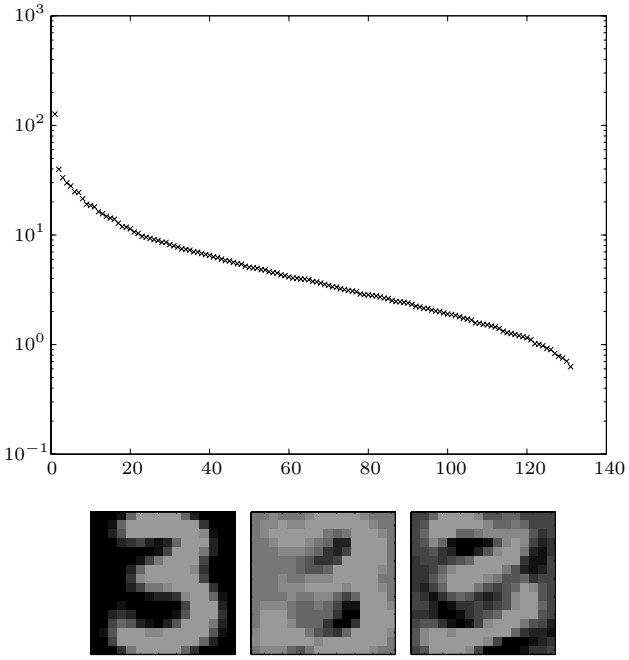


Figure 5.3. Singular values (top), and the first three singular images (vectors) computed using the 131 3s of the training set (bottom).

It turns out that for our test set the success rate of this algorithm is around 75%, which is not good enough. The reason is that the algorithm does not use any information about the variation of the digits of one kind. This variation can be modelled using the SVD.

Let $A \in \mathbb{R}^{m\times n}$, with $m = 256$, be the matrix consisting of all the training digits of one kind, the 3s, say. The columns of A span a linear subspace of $\mathbb{R}^m$. However, this subspace cannot be expected to have a large dimension, because if it had, then the subspaces of the different kinds of digits would intersect.

The idea now is to 'model' the variation within the set of training digits of one kind using an orthogonal basis of the subspace. An orthogonal basis can be computed using the SVD, and A can be approximated by a sum of rank-one matrices (3.9),

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

for some value of k. Each column in A is an image of a digit 3, and therefore the left singular vectors u_i are an orthogonal basis in the 'image space of 3s'. We will refer to the left singular vectors as 'singular images'. From the matrix approximation properties of the SVD (Theorem 3.4) we know that the first singular vector represents the 'dominating' direction of the data matrix. Therefore, if we fold the vectors u_i back to images, we expect the first singular vector to look like a 3, and the following singular images should represent the dominating variations of the training set around the first singular image. In Figure 5.3 we illustrate the singular values and the first three singular images for the training set 3s.

The SVD basis classification algorithm will be based on the following assumptions.

(1) Each digit (in the training and test sets) is well characterized by a few of the first singular images of its own kind. The more precise meaning of 'a few' should be investigated by experiment.

(2) An expansion in terms of the first few singular images discriminates well between the different classes of digits.

(3) If an unknown digit can be better approximated in one particular basis of singular images, the basis of 3s say, than in the bases of the other classes, then it is likely that the unknown digit is a 3.

Thus we should compute how well an unknown digit can be represented in the ten different bases. This can be done by computing the residual vector in *least squares problems* of the type

$$\min_{\alpha_i} \left\| z - \sum_{i=1}^{k} \alpha_i u_i \right\|,$$

where z represents an unknown digit, and u_i the singular images. We can write this problem in the form

$$\min_{\alpha} \|z - U_k \alpha\|_2,$$

where $U_k = \begin{pmatrix} u_1 & u_2 & \cdots & u_k \end{pmatrix}$. Since the columns of U_k are orthogonal, the solution of this problem is given by $\alpha = U_k^T z$, and the norm of the residual vector of the least squares problems is

$$\|(I - U_k U_k^T) z\|_2. \tag{5.1}$$

It is interesting to see how the residual depends on the number of terms in the basis. In Figure 5.4 we illustrate the approximation of a nicely written 3 in terms of the 3-basis with different numbers of basis images. In Figure 5.5 we show the approximation of a nice 3 in the 5-basis.

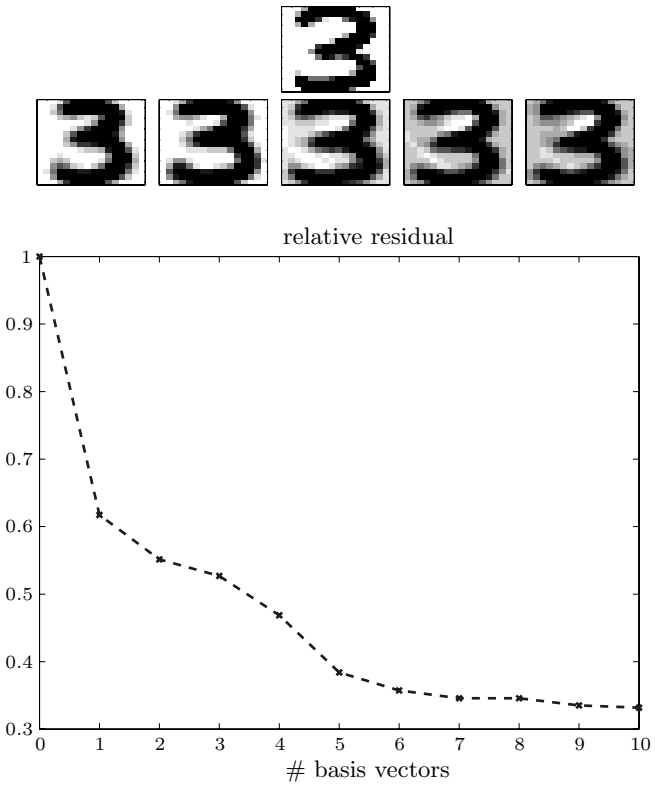


Figure 5.4. Unknown digit (nice 3) and approximations using 1, 3, 5, 7, and 9 terms in the 3-basis (top). Relative residual $\|(I - U_k U_k^T)z\|_2/\|z\|_2$ in least squares problem (bottom).

From Figures 5.4 and 5.5 we see that the relative residual is considerably smaller for the nice 3 in the 3-basis than in the 5-basis.

It is possible to devise several classification algorithm based on the model of expanding in terms of SVD bases. Below we give a simple variant.

An SVD basis classification algorithm

Training. For the training set of known digits, compute the SVD of each class of digits, and use k basis vectors for each class.

Classification. For a given test digit, compute its relative residual in all ten bases. If one residual is significantly smaller than all the others, classify as that. Otherwise give up.

The algorithm is closely related to the SIMCA method (Wold 1976, Sjöström and Wold 1980).

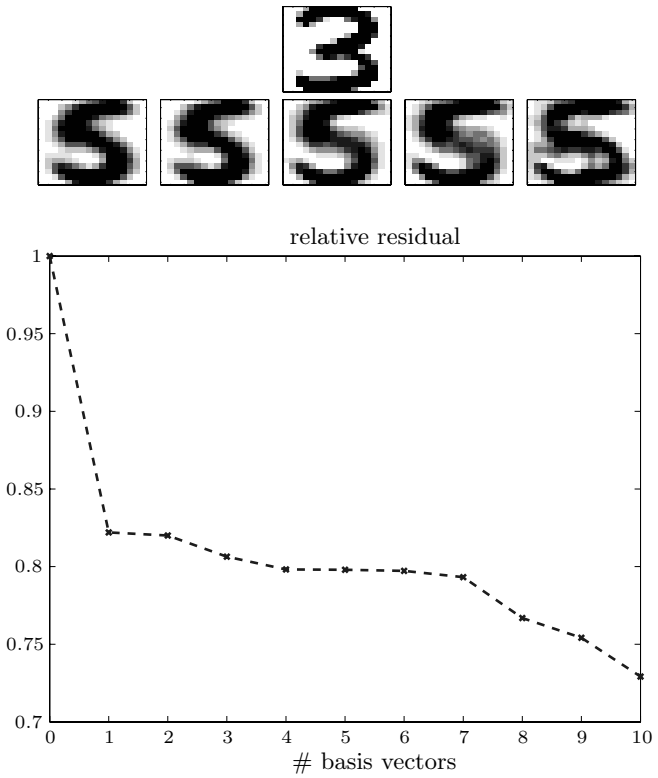


Figure 5.5. Unknown digit (nice 3) and approximations using 1, 3, 5, 7, and 9 terms in the 5-basis (top). Relative residual in least squares problem (bottom).

We next give some test results[16] for the US Postal Service data set, with 7291 training digits and 2007 test digits (Hastie *et al.* 2001). In Table 5.1 we give classification results as a function of the number of basis images for each class.

Table 5.1. Correct classifications as a function of the number of basis images (for each class).

# basis images	1	2	4	6	8	10
correct (%)	80	86	90	90.5	92	93

[16] From Savas (2002).

Even if there is a very significant improvement in performance compared to the method where one only used the centroid images, the results are not good enough, as the best algorithms reach about 97% correct classifications.

5.2. Tangent distance

Apparently it is the large variation in the way digits are written that makes it difficult to classify correctly. In the preceding subsection we used SVD bases to model the variation. An alternative model can be based on describing mathematically what are common and acceptable variations. We illustrate a few such variations in Figure 5.6. In the first column of modified

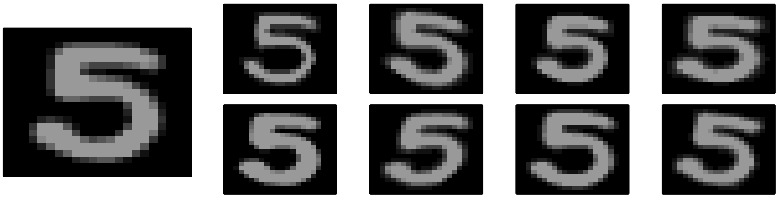


Figure 5.6. A digit and acceptable transformations.

digits, the digits appear to be written[17] with a thinner and thicker pen, in the second the digits have been stretched diagonal-wise, in the third they have been compressed and elongated vertically, and in the fourth they have been rotated. Such transformations constitute no difficulties for a human reader and ideally they should be easy to deal with in automatic digit recognition. A distance measure, *tangent distance*, that is invariant under small transformations of this type is described in Simard *et al.* (1993, 2001).

For now we interpret 16×16 images as points in $\mathbb{R}^{256}$. Let p be a fixed pattern in an image. We shall first consider the case of only one permitted transformation, diagonal stretching, say. The transformation can be thought of as moving the pattern along a curve in $\mathbb{R}^{256}$. Let the curve be parametrized by a real parameter α so that the curve is given by $s(p, \alpha)$, and in such a way that $s(p, 0) = p$. In general, such curves are nonlinear, and can be approximated by the first two terms in the Taylor expansion,

$$s(p, \alpha) = s(p, 0) + \frac{\mathrm{d}s}{\mathrm{d}\alpha}(p, 0)\,\alpha + O(\alpha^2) \approx p + t_p\alpha,$$

where $t_p = \frac{\mathrm{d}s}{\mathrm{d}\alpha}(p, 0)$ is a vector in $\mathbb{R}^{256}$. By varying α slightly around 0, we

[17] Note that the modified digits have not been written manually but using the techniques described later in this section. The presentation in this section is based on the papers by Simard, LeCun and Denker (1993) and Simard, LeCun, Denker and Victorri (2001), and the master's thesis of Savas (2002).

make a small movement of the pattern along the tangent at the point p on the curve. Assume that we have another pattern e that is approximated similarly,

$$s(e, \alpha) \approx e + t_e \alpha.$$

Since we consider small movements along the curves as allowed, such small movements should not influence the distance function. Therefore, ideally we would like to define our measure of closeness between p and e as the closest distance between the two curves.

In general we cannot compute the distance between the curves, but we can use the first-order approximations. Thus we move the patterns independently along their respective tangents, until we find the smallest distance. If we measure this distance in the usual Euclidean norm, we shall solve the least squares problem

$$\min_{\alpha_p, \alpha_e} \|p + t_p \alpha_p - e - t_e \alpha_e\|_2 = \min_{\alpha_p, \alpha_e} \left\| (p - e) - \begin{pmatrix} -t_p & t_e \end{pmatrix} \begin{pmatrix} \alpha_p \\ \alpha_e \end{pmatrix} \right\|_2.$$

Consider now the case when we are allowed to move the pattern p along l different curves in $\mathbb{R}^{256}$, parametrized by $\alpha = (\alpha_1 \cdots \alpha_l)^T$. This is equivalent to moving the pattern on an l-dimensional surface (manifold) in $\mathbb{R}^{256}$. Assume that we have two patterns, p and e, each of which can move on its surface of allowed transformations. Ideally we would like to find the closest distance between the surfaces, but instead, since this is not possible to compute, we now define a distance measure where we compute the distance between the two *tangent planes* of the surface in the points p and e.

As before, the tangent plane is given by the first two terms in the Taylor expansion of the function $s(p, \alpha)$:

$$s(p, \alpha) = s(p, 0) + \sum_i^l \frac{\mathrm{d}s}{\mathrm{d}\alpha_i}(p, 0)\, \alpha_i + O(\|\alpha\|_2^2) \approx p + T_p \alpha,$$

where T_p is the matrix

$$T_p = \begin{pmatrix} \frac{\mathrm{d}s}{\mathrm{d}\alpha_1} & \frac{\mathrm{d}s}{\mathrm{d}\alpha_2} & \cdots & \frac{\mathrm{d}s}{\mathrm{d}\alpha_l} \end{pmatrix},$$

and the derivatives are all evaluated in the point $(p, 0)$.

Thus the *tangent distance* between the points p and e is defined as the smallest possible residual in the least squares problem

$$\min_{\alpha_p, \alpha_e} \|p + T_p \alpha_p - e - T_e \alpha_e\|_2 = \min_{\alpha_p, \alpha_e} \left\| (p - e) - \begin{pmatrix} -T_p & T_e \end{pmatrix} \begin{pmatrix} \alpha_p \\ \alpha_e \end{pmatrix} \right\|_2.$$

The least squares problem can be solved, *e.g.*, using the QR decomposition of $A = \begin{pmatrix} -T_p & T_e \end{pmatrix}$. Note that we are in fact not interested in the solution

itself but only in the norm of the residual. Write the least squares problem in the form

$$\min_{\alpha} \|b - A\alpha\|_2, \qquad b = p - e, \quad \alpha = \begin{pmatrix} \alpha_p \\ \alpha_e \end{pmatrix}.$$

With the QR decomposition[18]

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = (Q_1\, Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1 R,$$

the norm of the residual is given by

$$\begin{aligned} \min_{\alpha} \|b - A\alpha\|_2^2 &= \min_{\alpha} \left\| \begin{pmatrix} Q_1^T b - R\alpha \\ Q_2^T b \end{pmatrix} \right\|^2 \\ &= \min_{\alpha} \{ \| \left(Q_1^T b - R\alpha \right) \|_2^2 + \|Q_2^T b\|_2^2 \} = \|Q_2^T b\|_2^2. \end{aligned}$$

The case when the matrix A does not have full column rank is easily dealt with using the SVD. The probability that the columns of the tangent matrix are almost linearly dependent is high when the two patterns are close.

The most important property of this distance function is that it is *invariant under movements of the patterns on the tangent planes*. For instance, if we make a small translation in the x-direction of a pattern, then with this measure the distance it has been moved is equal to zero.

Simard *et al.* (1993) and (2001) considered the following transformation: *horizontal* and *vertical translation*, *rotation*, *scaling*, *parallel* and *diagonal hyperbolic transformation*, and *thickening*. If we consider the image pattern as a function of two variables, $p = p(x, y)$, then the derivative of each transformation can be expressed as a differentiation operator that is a linear combination of the derivatives $p_x = \frac{\mathrm{d}p}{\mathrm{d}x}$ and $p_y = \frac{\mathrm{d}p}{\mathrm{d}y}$. For instance, the rotation derivative is

$$yp_x - xp_y,$$

and the scaling derivative is

$$xp_x + yp_y.$$

The derivative of the diagonal hyperbolic transformation is

$$yp_x + xp_y,$$

and the 'thickening' derivative is

$$(p_x)^2 + (p_y)^2.$$

[18] A has dimension $256 \times 2l$; since the number of transformations is usually less than 10, the linear system is over-determined.

The algorithm is summarized as follows.

A tangent distance classification algorithm

Training. For each digit in the training set, compute its tangent matrix T_p.

Classification. For each test digit:

- Compute its tangent matrix.
- Compute the tangent distance to all training digits and classify it as the one with shortest distance.

This algorithm is quite good in terms of classification performance (96.9% correct classification for the US Postal Service data set (Savas 2002)), but it is very expensive, since each test digit is compared to all the training digits. In order to make it competitive it must be combined with some other algorithm that reduces the number of tangent distance comparisons to make.

We end this section by remarking that it is necessary to pre-process the digits in different ways in order to enhance the classification: see LeCun *et al.* (1998). For instance, performance is improved if the images are smoothed (convolved with a Gaussian kernel): see Simard *et al.* (2001). In Savas (2002) the derivatives p_x and p_y are computed numerically by finite differences.

6. Eigenvalue methods in data mining

When an Internet search is made using a search engine, there is first a traditional text processing part, where the aim is to find all the web pages containing the words of the query. Because of the massive size of the Web, the number of hits is likely to be much too large to be handled by the user. Therefore, some measure of quality is needed to sort out the pages that are likely to be most relevant to the particular query.

When one uses a web search engine, then typically the search phrase is under-specified.

Example 6.1. A Google search conducted on September 29, 2005, using the search phrase *university*, gave as a result links to the following well-known universities: *Harvard*, *Stanford*, *Cambridge*, *Yale*, *Cornell*, *Oxford*. The total number of web pages relevant to the search phrase was more than 2 billion.

Obviously Google uses an algorithm for ranking all the web pages that agrees rather well with a common-sense quality measure. Loosely speaking, Google assigns a high rank to a web page if it has inlinks from other pages that have a high rank. We will see that this 'self-referencing' statement can be formulated mathematically as an eigenvalue equation for a certain matrix.

In the context of automatic text summarization, similar self-referencing statements can be formulated mathematically as the defining equations of a singular value problem. We treat this application briefly in Section 6.3.

6.1. PageRank

It is of course impossible to define a generally valid measure of relevance that would be acceptable for a majority of users of a search engine. Google uses the concept of *PageRank* as a quality measure of web pages. It is based on the assumption that the number of links to and from a page give information about the importance of a page. We will give a description of PageRank based primarily on Page, Brin, Motwani and Winograd (1998). Concerning Google, see Brin and Page (1998).

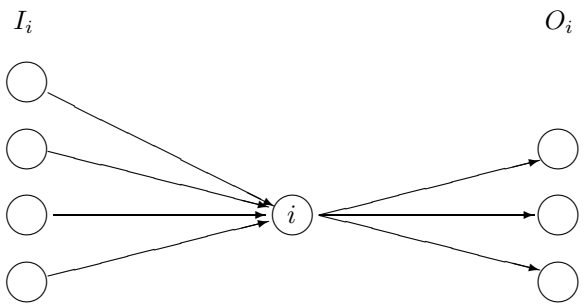


Figure 6.1. Inlinks and outlinks.

Let all web pages be ordered from 1 to n, and let i be a particular web page. Then O_i will denote the set of pages that i is linked to, the *outlinks*. The number of outlinks is denoted by $N_i = |O_i|$. The set of *inlinks*, denoted by I_i, are the pages that have an outlink to i: see Figure 6.1.

In general, a page i can be considered as more important the more inlinks it has. However, a ranking system based only on the number of inlinks is easy to manipulate.[19] When you design a web page i that (*e.g.*, for commercial reasons) you would like to be seen by as many as possible, you could simply create a large number of (information-less and unimportant) pages that have outlinks to i. In order to discourage this, one may define

[19] For an example of attempts to fool a search engine: see Totty and Mangalindan (2003).

the rank of i in such a way that if a highly ranked page j, has an outlink to i, this should add to the importance of i in the following way: the rank of page i is a weighted sum of the ranks of the pages that have outlinks to i. The weighting is such that the rank of a page j is divided evenly among its outlinks. The preliminary definition of PageRank is

$$r_i = \sum_{j \in I_i} \frac{r_j}{N_j}, \qquad i = 1, 2, \ldots, n. \tag{6.1}$$

The definition (6.1) is recursive, so PageRank cannot be computed directly. The equation can be reformulated as an eigenvalue problem for a matrix representing the graph of the Internet. Let Q be a square matrix of dimension n, and let

$$Q_{ij} = \begin{cases} 1/N_j, & \text{if there is a link from } j \text{ to } i, \\ 0, & \text{otherwise.} \end{cases}$$

This definition means that row i has nonzero elements in those positions that correspond to inlinks of i. Similarly, column j has nonzero elements equal to N_j in those positions that correspond to the outlinks of j, and, provided that the page has outlinks, the sum of all the elements in column j is equal to one. In the following symbolic picture of the matrix Q, nonzero elements are denoted $*$:

$$\begin{array}{ccc} & j & \\ & \downarrow & \\ i \rightarrow & \begin{pmatrix} & * & \\ & 0 & \\ & \vdots & \\ 0 \ * \ \cdots & * & * \ \cdots \\ & \vdots & \\ & 0 & \\ & * & \end{pmatrix} & \leftarrow \text{ inlinks} \\ & \uparrow & \\ & \text{outlinks} & \end{array}$$

Obviously, (6.1) can be written as

$$\lambda r = Qr, \tag{6.2}$$

i.e., r is an eigenvector of Q with eigenvalue $\lambda = 1$. However, at this point it is not clear that PageRank is well defined, as we do not know if there exists an eigenvalue equal to 1.

It turns out that the theory of random walk and Markov chains gives an intuitive explanation of the concepts involved. Assume that a surfer

visiting a web page always chooses the next page among the outlinks with equal probability. This random walk induces a Markov chain with transition matrix Q^T: see, *e.g.*, Meyer (2000) and Langville and Meyer (2005*a*).[20] A Markov chain is a random process for which the next state is determined completely by the present state; the process has no memory. The eigenvector r of the transition matrix with eigenvalue 1 corresponds to a stationary probability distribution for the Markov chain: The element in position i, r_i, is the asymptotic probability that the random walker is at web page i.

The random surfer should never get stuck. In other words, there should be no web pages without outlinks (such a page corresponds to a zero column in Q). Therefore the model is modified so that zero columns are replaced by a constant value in each position (equal probability to go to any other page in the net). Define the vectors

$$d_j = \begin{cases} 1, & \text{if } N_j = 0, \\ 0, & \text{otherwise,} \end{cases}$$

for $j = 1, \ldots, n$, and

$$e = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n.$$

Then the modified matrix is defined by

$$P = Q + \frac{1}{n} e d^T. \tag{6.3}$$

Now P is a proper *column-stochastic matrix*: it has nonnegative elements ($P \geq 0$), and

$$e^T P = e^T. \tag{6.4}$$

By analogy with (6.2), we would like to define the PageRank vector as a unique eigenvector of P with eigenvalue 1,

$$Pr = r.$$

However, uniqueness is still not guaranteed. To ensure this, the directed graph corresponding to the matrix must be *strongly connected*: given any two nodes (N_i, N_j), in the graph, there must exist a path leading from N_i

[20] Note that we use a slightly different notation to that common in the theory of stochastic processes.

to N_j. In matrix terms, P must be *irreducible*.[21] Equivalently, there must not exist any subgraph that has no outlinks.

The uniqueness of the eigenvalue is now guaranteed by the Perron–Frobenius theorem; we state it for the special case treated here.

Theorem 6.2. Let A be an irreducible column-stochastic matrix. Then the largest eigenvalue in magnitude is equal to 1. There is a unique corresponding eigenvector r satisfying $r > 0$, and $\|r\|_1 = 1$; this is the only eigenvector that is nonnegative. If $A > 0$, then $|\lambda_i| < 1$, $i = 2, 3, \ldots, n$.

Proof. Because A is column-stochastic we have $e^T A = e^T$, which means that 1 is an eigenvalue of A. The rest of the statement can be proved using Perron–Frobenius theory (Meyer 2000, Chapter 8). □

Given the size of the Internet and reasonable assumptions about its structure, it is highly probable that the link graph is *not* strongly connected, which means that the PageRank eigenvector of P is not well defined. To ensure connectedness, *i.e.*, to make it impossible for the random walker to get trapped in a subgraph, one can add, artificially, a link from every web page to all the other. In matrix terms, this can be made by taking a convex combination of P and a rank-one matrix,

$$A = \alpha P + (1 - \alpha)\frac{1}{n} e e^T, \tag{6.5}$$

for some α satisfying $0 \le \alpha \le 1$. Obviously A is irreducible (since $A > 0$) and column-stochastic:

$$e^T A = \alpha e^T P + (1 - \alpha)\frac{1}{n} e^T e e^T = \alpha e^T + (1 - \alpha) e^T = e^T.$$

The random walk interpretation of the additional rank-one term is that each time step a page is visited, the surfer will jump to any page in the whole web with probability $1 - \alpha$ (sometimes referred to as *teleportation*).

For the convergence of the numerical eigenvalue algorithm, it is essential to know how the eigenvalues of P are changed by the rank one modification (6.5).

Proposition 6.3. Given that the eigenvalues of the column-stochastic matrix P are $\{1, \lambda_2, \lambda_3 \ldots, \lambda_n\}$, the eigenvalues of $A = \alpha P + (1 - \alpha)\frac{1}{n} e e^T$ are $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$.

Several proofs of the proposition have been published (Haveliwala and Kamvar 2003*b*, Langville and Meyer 2005*a*). An elementary and simple variant (Eldén 2004*a*) is given here.

[21] A matrix P is *reducible* if there exist a permutation matrix Π such that $\Pi P \Pi^T = \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}$, where both X and Z are square matrices.

Proof. Define $\hat{e}$ to be e normalized to Euclidean length 1, and let $U_1 \in \mathbb{R}^{n\times(n-1)}$ be such that $U = \begin{pmatrix}\hat{e} & U_1\end{pmatrix}$ is orthogonal. Then, since $\hat{e}^T P = \hat{e}^T$,

$$U^T P U = \begin{pmatrix}\hat{e}^T P \\ U_1^T P\end{pmatrix}\begin{pmatrix}\hat{e} & U_1\end{pmatrix} = \begin{pmatrix}\hat{e}^T \\ U_1^T P\end{pmatrix}\begin{pmatrix}\hat{e} & U_1\end{pmatrix}$$
$$= \begin{pmatrix}\hat{e}^T\hat{e} & \hat{e}^T U_1 \\ U_1^T P\hat{e} & U_1^T P^T U_1\end{pmatrix} = \begin{pmatrix}1 & 0 \\ w & T\end{pmatrix}, \tag{6.6}$$

where $w = U_1^T P\hat{e}$, and $T = U_1^T P^T U_1$. Since we have made a similarity transformation, the matrix T has the eigenvalues $\lambda_2, \lambda_3, \ldots, \lambda_n$. We further have

$$U^T v = \begin{pmatrix}1/\sqrt{n}\, e^T v \\ U_1^T v\end{pmatrix} = \begin{pmatrix}1/\sqrt{n} \\ U_1^T v\end{pmatrix}.$$

Therefore,

$$U^T A U = U^T(\alpha P + (1-\alpha)ve^T)U$$
$$= \alpha\begin{pmatrix}1 & 0 \\ w & T\end{pmatrix} + (1-\alpha)\begin{pmatrix}1/\sqrt{n} \\ U_1^T v\end{pmatrix}\begin{pmatrix}\sqrt{n} & 0\end{pmatrix}$$
$$= \alpha\begin{pmatrix}1 & 0 \\ w & T\end{pmatrix} + (1-\alpha)\begin{pmatrix}1 & 0 \\ \sqrt{n}\,U_1^T v & 0\end{pmatrix} =: \begin{pmatrix}1 & 0 \\ w_1 & \alpha T\end{pmatrix}.$$

The statement now follows immediately. □

This means that even if P has a multiple eigenvalue equal to 1, the second-largest eigenvalue in magnitude of A is always equal to α.

The vector e in (6.5) can be replaced by a nonnegative vector v with $\|v\|_1 = 1$ that can be chosen in order to make the search biased towards certain kinds of web pages. Therefore, it is referred to as a *personalization vector* (Page *et al.* 1998, Haveliwala and Kamvar 2003*a*). The vector v can also be used for avoiding manipulation by so-called link farms (Langville and Meyer 2005*a*). Proposition 6.3 also holds in this case.

We want to solve the eigenvalue problem

$$Ar = r,$$

where r is normalized, $\|r\|_1 = 1$. Because of the sparsity and the dimension of A it is not possible to use sparse eigenvalue algorithms that require the storage of more than a very few vectors. The only viable method so far for PageRank computations on the whole Web seems to be the *power method*.

It is well known (see, *e.g.*, Golub and Van Loan (1996, Section 7.3)) that the rate of convergence of the power method depends on the ratio of the second-largest and the largest eigenvalue in magnitude. Here we have

$$|\lambda^{(k)} - 1| = O(\alpha^k),$$

due to Proposition 6.3.

In view of the huge dimension of the Google matrix, it is nontrivial to compute the matrix-vector product $y = Az$, where $A = \alpha P + (1-\alpha)\frac{1}{n}ee^T$. First, we see that if the vector z satisfies $\|z\|_1 = e^T z = 1$, then

$$\|y\|_1 = e^T y = e^T Az = e^T z = 1, \tag{6.7}$$

since A is column-stochastic ($e^T A = e^T$). Therefore normalization of the vectors produced in the power iteration is unnecessary.

Then recall that P was constructed from the actual link matrix Q as

$$P = Q + \frac{1}{n}ed^T,$$

where the row vector d has an element 1 in all those positions that correspond to web pages with no outlinks, see (6.3). This means that to represent P as a sparse matrix, we insert a large number of full vectors in Q, each of the same dimension as the total number of web pages. Consequently, one cannot afford to represent P explicitly. Let us look at the multiplication $y = Az$ in some more detail:

$$y = \alpha\left(Q + \frac{1}{n}ed^T\right)z + \frac{(1-\alpha)}{n}e(e^T z) = \alpha Qz + \beta\frac{1}{n}\,e, \tag{6.8}$$

where

$$\beta = \alpha d^T z + (1-\alpha)e^T z.$$

However, we do not need to compute β from this equation. Instead we can use (6.7) in combination with (6.8):

$$1 = e^T(\alpha Qz) + \beta e^T\left(\frac{1}{n}\,e\right) = e^T(\alpha Qz) + \beta.$$

Thus, we have $\beta = 1 - \|\alpha Qz\|_1$. An extra bonus is that we do not use the vector d, *i.e.*, we do not need to know which pages lack outlinks.

The following Matlab code implements the matrix vector multiplication.

```
yhat=alpha*Q*z;
beta=1-norm(yhat,1);
y=yhat+beta*v;
residual=norm(y-z,1);
```

Here $v = (1/n)\,e$, or a personalized teleportation vector: see p. 372.

From Proposition 6.3 we know that the second eigenvalue of the Google matrix satisfies $\lambda_2 = \alpha$. A typical value of α is 0.85. Approximately $k = 57$ iterations are needed to make the factor 0.85^k equal to 10^{-4}. This is reported (Langville and Meyer 2005*a*) to be close the number of iterations used by Google.

In view of the fact that one PageRank calculation using the power method can take several days, several enhancements of the iteration procedure have been proposed. Kamvar, Haveliwala and Golub (2003*a*) describe an adaptive method that checks the convergence of the components of the PageRank vector and avoids performing the power iteration for those components. The block structure of the Web is used in Kamvar, Haveliwala, Manning and Golub (2003*b*), and speed-ups of a factor 2 have been reported. An acceleration method based on Aitken extrapolation is discussed in Kamvar, Haveliwala, Manning and Golub (2003*c*). Aggregation methods are discussed in several papers by Langville and Meyer and in Ipsen and Kirkland (2006).

If the PageRank is computed for a subset of the Internet, one particular domain, say, then the matrix A may be of sufficiently small dimension to use methods other than the power method: *e.g.*, the Arnoldi method (Golub and Greif 2004).

A variant of PageRank is proposed in Gyöngyi, Garcia-Molina and Pedersen (2004). Further properties of the PageRank matrix are given in Serra-Capizzano (2005).

6.2. HITS

Another method based on the link structure of the Web was introduced at the same time as PageRank (Kleinberg 1999). It is called HITS (hypertext induced topic search), and is based on the concepts of *authorities* and *hubs*. An authority is a web page with several inlinks and a hub has several outlinks. The basic idea is: *good hubs point to good authorities and good authorities are pointed to by good hubs*. Each web page is assigned both a hub score y and an authority score x.

Let L be the adjacency matrix of the directed web graph. Then two equations are given that mathematically define the relation between the two scores, based on the basic idea:

$$x = L^T y, \qquad y = Lx. \tag{6.9}$$

The algorithm for computing the scores is the power method, which converges to the left and right singular vectors corresponding to the largest singular value of L. In the implementation of HITS it is not the adjacency matrix of the whole web that is used, but of all the pages relevant to the query.

There is now an extensive literature on PageRank, HITS and other ranking methods. For overviews, see Langville and Meyer (2005*b*, 2005*c*) and Berkin (2005). A combination of HITS and PageRank has been proposed in Lempel and Moran (2001).

Obviously the ideas underlying PageRank and HITS are not restricted

to web applications, but can be applied to other network analyses. For instance, a variant of the HITS method was recently used in a study of Supreme Court precedent (Fowler and Jeon 2005). A generalization of HITS is given in Blondel, Gajardo, Heymans, Senellart and Dooren (2004), which also treats synonym extraction.

6.3. Text summarization

Because of the explosion in the amount of textual information available, there is a need to develop automatic procedures for text summarization. One typical situation is when a web search engine presents a small amount of text from each document that matches a certain query. Another relevant area is the summarization of news articles.

Automatic text summarization is an active research field with connections to several other research areas such as information retrieval, natural language processing, and machine learning. Informally, the goal of text summarization is to *extract content from a text document, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's need* (Mani 2001). In this section we will have a considerably less ambitious goal: we present a method (Zha 2002), related to HITS, for automatically extracting key words and key sentences from a text. There are connections to the vector space model in information retrieval, and to the concept of PageRank.

Consider a text from which we want to extract key words and key sentences. As one of the preprocessing steps, one should perform stemming and eliminate stop words. Similarly, if the text carries special symbols, *e.g.*, mathematics, or mark-up language tags (HTML, LaTeX), it may be necessary to remove those. Since we want to compare word frequencies in different sentences, we must consider each sentence as a separate document (in the terminology of information retrieval). After the preprocessing has been done, we parse the text, using the same type of parser as in information retrieval. This way a term-document matrix is prepared, which in this section we will refer to as a *term-sentence* matrix. Thus we have a matrix $A \in \mathbb{R}^{m\times n}$, where m denotes the number of different terms, and n the number of sentences. The element a_{ij} is defined as the frequency[22] of term i in document j.

The basis of the procedure in Zha (2002) is the simultaneous, but separate *ranking* of the terms and the sentences. Thus, term i is given a nonnegative *saliency score*, denoted u_i. The higher the saliency score, the more important the term. The saliency score of sentence j is denoted by v_j.

[22] Naturally, a term and document weighting scheme (see Berry and Browne (2005, Section 3.2.1)) should be used.

The assignment of saliency scores is made based on the *mutual reinforcement principle* (Zha 2002):

A term should have a high saliency score if it appears in many sentences with high saliency scores. A sentence should have a high saliency score if it contains many words with high saliency scores.

More precisely, we assert that the saliency score of term i is proportional to the sum of the scores of the sentences where it appears; in addition, each term is weighted by the corresponding matrix element,

$$u_i \propto \sum_{j=1}^{n} a_{ij} v_j, \qquad i = 1, 2, \ldots, m.$$

Similarly, the saliency score of sentence j is defined to be proportional to the scores of its words, weighted by the corresponding a_{ij},

$$v_j \propto \sum_{i=1}^{m} a_{ij} u_i, \qquad j = 1, 2, \ldots, n.$$

Collecting the saliency scores in two vectors, $u \in \mathbb{R}^m$, and $v \in \mathbb{R}^n$, these two equations can be written as

$$\sigma_u u = Av, \tag{6.10}$$

$$\sigma_v v = A^T u, \tag{6.11}$$

where σ_u and σ_v are proportionality constants. In fact, the constants must be equal: inserting one equation into the other, we get

$$\sigma_u\, u = \frac{1}{\sigma_v} AA^T u,$$

$$\sigma_v\, v = \frac{1}{\sigma_u} A^T A v,$$

which shows that u and v are singular vectors corresponding to the same singular value. If we choose the largest singular value, then we are guaranteed that the components of u and v are nonnegative.

Example 6.4. We created a term-sentence matrix using (a slightly earlier version of) the text in Section 4. Since the text is written using LaTeX, we first had to remove all LaTeX typesetting commands. This was done using a lexical scanner called `detex`.[23] Then the text was stemmed and stop words were removed. A term-sentence matrix A was constructed using a text parser: there turned out to be 435 terms in 218 sentences. The first singular vectors were computed in Matlab.

[23] `http://www.cs.purdue.edu/homes/trinkle/detex/`

By determining the ten largest components of u_1, and using the dictionary produced by the text parser, we found that the following ten words are the most important in the section.

document, *term*, *matrix*, *approximation* (*approximate*), *query*, *vector*, *space*, *number*, *basis*, *cluster*.

The three most important sentences are, in order, as follows.

(1) *Latent semantic indexing* (LSI) 'is based on the assumption that there is some underlying latent semantic structure in the data ... that is corrupted by the wide variety of words used ...' (quoted from Park, Jeon and Rosen (2001)) and that this semantic structure can be enhanced by projecting the data (the term-document matrix and the queries) onto a lower-dimensional space using the singular value decomposition.

(2) In view of the large approximation error in the truncated SVD approximation of the term-document matrix, one may question whether the 'optimal' singular vectors constitute the best basis for representing the term-document matrix.

(3) For example, one can define the elements in A by

$$a_{ij} = f_{ij}\log(n/n_i),$$

where f_{ij} is the term frequency, the number of times term i appears in document j, and n_i is the number of documents that contain term i (inverse document frequency).

It is apparent that this method prefers long sentences. On the other hand, these sentences are undeniably key sentences for the text.

7. New directions

Multidimensional arrays (tensors) have been used for data analysis in psychometrics and chemometrics since the 1960s; for overviews see, *e.g.*, Kroonenberg (1992), Smilde, Bro and Geladi (2004) and the 'Three-Mode Company' web page.[24] In fact, 'three-mode analysis' appears to be a standard tool in those areas. Only in recent years has there been an increased interest among the numerical linear algebra community in tensor computations, especially for applications in signal processing and data mining. A particular generalization of the SVD, the *higher order SVD* (HOSVD), was studied in Lathauwer, Moor and Vandewalle (2000*a*).[25] This is a tensor decomposition

[24] `http://three-mode.leidenuniv.nl/`.

[25] However, related concepts had already been considered in Tucker (1964) and (1966), and are referred to as the *Tucker model* in psychometrics.

in terms of orthogonal matrices, which 'orders' the tensor in a way similar to that in which the singular values of the SVD are ordered, but which does not satisfy an Eckart–Young optimality property (Theorem 3.4); see Lathauwer, Moor and Vandewalle (2000*b*). Owing to the ordering property, this decomposition can be used for compression and dimensionality reduction, and it has been successfully applied to face recognition (Vasilescu and Terzopoulos 2002*a*, 2002*b*, 2003).

The SVD expansion (3.9) of a matrix, as a sum of rank-one matrices, has been generalized to tensors (Harshman 1970, Carroll and Chang 1970), and is called the *PARAFAC/CANDECOMP model*. For overviews, see Bro (1997) and Smilde *et al.* (2004). This does not give an exact decomposition of the tensor, and its theoretical properties are much more involved (*e.g.*, degeneracies occur – see Kruskal (1976, 1977), Bro (1997) and Sidiropoulos and Bro (2000)). A recent application of PARAFAC to network analysis is presented in Kolda, Bader and Kenny (2005*a*), where the hub and authority scores of the HITS method are complemented with topic scores for the anchor text of the web pages.

Recently several papers have appeared where standard techniques in data analysis and machine learning are generalized to tensors: see, *e.g.*, Yan, Xu, Yang, Zhang, Tang and Zhang (2005) and Cai, He and Han (2005).

It is not uncommon in the data mining/machine learning literature for data compression and rank reduction problems to be presented as matrix problems, while they can in fact be considered as tensor approximation problems: for examples see Tenenbaum and Freeman (2000) and Ye (2005).

Novel data mining applications, especially in link structure analysis, are presented and suggested in Kolda, Brown, Corones, Critchlow, Eliassi-Rad, Getoor, Hendrickson, Kumar, Lambert, Matarazzo, McCurley, Merrill, Samatova, Speck, Srikant, Thomas, Wertheimer and Wong (2005*b*).

REFERENCES

R. Baeza-Yates and B. Ribeiro-Neto (1999), *Modern Information Retrieval*, ACM Press, Addison-Wesley, New York.

B. Bergeron (2002), *Bioinformatics Computing*, Prentice-Hall.

P. Berkin (2005), 'A survey on PageRank computing', *Internet Mathematics* **2**, 73–120.

M. Berry, ed. (2001), *Computational Information Retrieval*, SIAM, Philadelphia, PA.

M. Berry and M. Browne (2005), *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, 2nd edn, SIAM, Philadelphia, PA.

M. Berry and G. Linoff (2000), *Mastering Data Mining: The Art and Science of Customer Relationship Management*, Wiley, New York.

M. Berry, Z. Drmac and E. Jessup (1999), 'Matrices, vector spaces and information retrieval', *SIAM Review* **41**, 335–362.

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

Å. Björck (1996), *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA.

K. Blom and A. Ruhe (2005), 'A Krylov subspace method for information retrieval', *SIAM J. Matrix Anal. Appl.* **26**, 566–582.

V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart and P. V. Dooren (2004), 'A measure of similarity between graph vertices: Applications to synonym extraction and web searching', *SIAM Review* **46**, 647–666.

S. Brin and L. Page (1998), 'The anatomy of a large-scale hypertextual web search engine', *Computer Networks and ISDN Systems* **30**, 107–117.

R. Bro (1997), 'PARAFAC: Tutorial and applications', *Chemometrics and Intelligent Laboratory Systems* **38**, 149–171.

M. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, L. Crumpler and J. Aubele (1998), 'Learning to recognize volcanoes on Venus', *Machine Learning* **30**, 165–195.

D. Cai, X. He and J. Han (2005), Subspace learning based on tensor analysis. Technical Report UIUCDCS-R-2005-2572, UILU-ENG-2005-1767, Computer Science Department, University of Illinois, Urbana-Champaign.

J. D. Carroll and J. J. Chang (1970), 'Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart–Young" decomposition', *Psychometrika* **35**, 283–319.

N. Christianini and J. Shawe-Taylor (2000), *An Introduction to Support Vector Machines*, Cambridge University Press.

M. Chu, R. Funderlic and G. Golub (1995), 'A rank-one reduction formula and its applications to matrix factorization', *SIAM Review* **37**, 512–530.

K. Cios, W. Pedrycz and R. Swiniarski (1998), *Data Mining: Methods for Knowledge Discovery*, Kluwer, Boston.

B. De Moor and P. Van Dooren (1992), 'Generalizations of the singular value and QR decompositions', *SIAM J. Matrix Anal. Appl.* **13**, 993–1014.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer and R. Harsman (1990), 'Indexing by latent semantic analysis', *J. Amer. Soc. Information Science* **41**, 391–407.

I. Dhillon (2001), Co-clustering documents and words using bipartite spectral graph partitioning, in *Proc. 7th ACM–SIGKDD Conference*, pp. 269–274.

I. Dhillon and D. Modha (2001), 'Concept decompositions for large sparse text data using clustering', *Machine Learning* **42**, 143–175.

I. Dhillon, J. Fan and Y. Guan (2001), Efficient clustering of very large document collections, in *Data Mining For Scientific and Engineering Applications* (V. Grossman, C. Kamath and R. Namburu, eds), Kluwer.

I. Dhillon, Y. Guan and B. Kulis (2005), A unified view of kernel k-means, spectral clustering and graph partitioning. Technical Report UTCS TR-04-25, University of Texas at Austin, Department of Computer Sciences.

D. Di Ruscio (2000), 'A weighted view on the partial least-squares algorithm', *Automatica* **36**, 831–850.

R. Duda, P. Hart and D. Storck (2001), *Pattern Classification*, 2nd edn, Wiley-Interscience.

G. Eckart and G. Young (1936), 'The approximation of one matrix by another of lower rank', *Psychometrika* **1**, 211–218.
J. Einbeck, G. Tutz and L. Evers (2005), 'Local principal curves', *Statistics and Computing* **15**, 301–313.
L. Eldén (2004*a*), The eigenvalues of the Google matrix. Technical Report LiTH-MAT-R–04-01, Department of Mathematics, Linköping University.
L. Eldén (2004*b*), 'Partial least squares vs. Lanczos bidiagonalization I: Analysis of a projection method for multiple regression', *Comput. Statist. Data Anal.* **46**, 11–31.
U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, eds (1996), *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press, Menlo Park, CA.
J. Fowler and S. Jeon (2005), The authority of supreme court precedent: A network analysis. Technical report, Department of Political Science, UC Davis.
I. Frank and J. Friedman (1993), 'A statistical view of some chemometrics regression tools', *Technometrics* **35**, 109–135.
J. Giles, L. Wo and M. Berry (2003), GTP (General Text Parser) software for text mining, in *Statistical Data Mining and Knowledge Discovery* (H. Bozdogan, ed.), CRC Press, Boca Raton, pp. 455–471.
N. Goharian, A. Jain and Q. Sun (2003), 'Comparative analysis of sparse matrix algorithms for information retrieval', *J. Systemics, Cybernetics and Informatics* **1**(1).
G. Golub and C. Greif (2004), Arnoldi-type algorithms for computing stationary distribution vectors, with application to PageRank. Technical Report SCCM-04-15, Department of Computer Science, Stanford University.
G. H. Golub and W. Kahan (1965), 'Calculating the singular values and pseudo-inverse of a matrix', *SIAM J. Numer. Anal. Ser. B* **2**, 205–224.
G. H. Golub and C. F. Van Loan (1996), *Matrix Computations*, 3rd edn, Johns Hopkins Press, Baltimore, MD.
G. Golub, K. Sølna and P. Van Dooren (2000), 'Computing the SVD of a general matrix product/quotient', *SIAM J. Matrix Anal. Appl.* **22**, 1–19.
D. Grossman and O. Frieder (1998), *Information Retrieval: Algorithms and Heuristics*, Kluwer.
L. Guttman (1957), 'A necessary and sufficient formula for matrix factoring', *Psychometrika* **22**, 79–81.
Z. Gyöngyi, H. Garcia-Molina and J. Pedersen (2004), Combating web spam with TrustRank, in *Proc. 30th International Conference on Very Large Databases*, Morgan Kaufmann, pp. 576–587.
J. Han and M. Kamber (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco.
D. Hand, H. Mannila and P. Smyth (2001), *Principles of Data Mining*, MIT Press, Cambridge, MA.
R. A. Harshman (1970), 'Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis', *UCLA Working Papers in Phonetics* **16**, 1–84.
T. Hastie (1984), Principal curves and surfaces. Technical report, Stanford University.

T. Hastie, R. Tibshirani and J. Friedman (2001), *The Elements of Statistical Learning: Data mining, Inference and Prediction*, Springer, New York.

T. Haveliwala and S. Kamvar (2003*a*), An analytical comparison of approaches to personalizing PageRank. Technical report, Computer Science Department, Stanford University.

T. Haveliwala and S. Kamvar (2003*b*), The second eigenvalue of the Google matrix. Technical report, Computer Science Department, Stanford University.

M. Hegland (2001), 'Data mining techniques', in *Acta Numerica*, Vol. 10, Cambridge University Press, pp. 313–355.

I. Helland (1988), 'On the structure of partial least squares regression', *Commun. Statist. Simulation* **17**, 581–607.

P. Howland and H. Park (2004), 'Generalizing discriminant analysis using the generalized singular value decomposition', *IEEE Trans. Pattern Anal. Machine Intelligence* **26**, 995– 1006.

P. Howland, M. Jeon and H. Park (2003), 'Structure preserving dimension reduction based on the generalized singular value decomposition', *SIAM J. Matrix Anal. Appl.* **25**, 165–179.

L. Hubert, J. Meulman and W. Heiser (2000), 'Two purposes for matrix factorization: A historical appraisal', *SIAM Review* **42**, 68–82.

I. C. Ipsen and S. Kirkland (2006), 'Convergence analysis of a Pagerank updating algorithm by Langville and Meyer', *SIAM J. Matrix Anal. Appl.* **27**, 952–967.

E. Jessup and J. Martin (2001), Taking a new look at the latent semantic analysis approach to information retrieval, in *Computational Information Retrieval* (M. Berry, ed.), SIAM, Philadelphia, PA, pp. 121–144.

I. Joliffe (1986), *Principal Component Analysis*, Springer, New York.

S. Kamvar, T. Haveliwala and G. Golub (2003*a*), 'Adaptive methods for the computation of PageRank', *Linear Algebra Appl.* **386**, 51–65.

S. Kamvar, T. Haveliwala, C. Manning and G. Golub (2003*b*), Exploiting the block structure of the Web for computing PageRank. Technical report, Computer Science Department, Stanford University.

S. Kamvar, T. Haveliwala, C. Manning and G. Golub (2003*c*), Extrapolation methods for accelerating PageRank computations, in *Proc. 12th International World Wide Web Conference, Budapest, May 2003*, pp. 261–270.

J. M. Kleinberg (1999), 'Authoritative sources in a hyperlinked environment', *J. Assoc. Comput. Mach.* **46**, 604–632.

T. Kolda, B. Bader and J. Kenny (2005*a*), Higher-order web link analysis using multilinear algebra, in *Proc. 5th IEEE International Conference on Data Mining, ICDM05*, IEEE Computer Society Press.

T. Kolda, D. Brown, J. Corones, T. Critchlow, T. Eliassi-Rad, L. Getoor, B. Hendrickson, V. Kumar, D. Lambert, C. Matarazzo, K. McCurley, M. Merrill, N. Samatova, D. Speck, R. Srikant, J. Thomas, M. Wertheimer and P. C. Wong (2005*b*), Data sciences technology for homeland security information management and knowledge discovery. Technical Report SAND2004-6648, Sandia National Laboratories.

R. Korfhage (1997), *Information Storage and Retrieval*, Wiley, New York.

P. M. Kroonenberg (1992), 'Three-mode component models: A survey of the literature', *Statistica Applicata* **4**, 619–633.

J. B. Kruskal (1976), 'More factors than subjects, tests and treatments: An indeterminacy theorem for canonical decomposition and individual differences scaling', *Psychometrika* **41**, 281–293.

J. B. Kruskal (1977), 'Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics (Corrections, 17-1-1984; available from author)', *Linear Algebra Appl.* **18**, 95–138.

A. Langville and C. Meyer (2005*a*), 'Deeper inside PageRank', *Internet Mathematics* **1**, 335–380.

A. N. Langville and C. D. Meyer (2005*b*), 'A survey of eigenvector methods for web information retrieval', *SIAM Review* **47**, 135–161.

A. N. Langville and C. D. Meyer (2005*c*), *Understanding Web Search Engine Rankings: Google's PageRank, Teoma's HITS, and Other Ranking Algorithms*, Princeton University Press.

L. D. Lathauwer, B. D. Moor and J. Vandewalle (2000*a*), 'A multilinear singular value decomposition', *SIAM J. Matrix Anal. Appl.* **21**, 1253–1278.

L. D. Lathauwer, B. D. Moor and J. Vandewalle (2000*b*), 'On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensor', *SIAM J. Matrix Anal. Appl.* **21**, 1324–1342.

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner (1998), 'Gradient-based learning applied to document recognition', *Proc. IEEE* **86**, 2278–2324.

R. Lempel and S. Moran (2001), 'Salsa: the stochastic approach for link-structure analysis', *ACM Trans. Inf. Syst.* **19**, 131–160.

I. Mani (2001), *Automatic Summarization*, John Benjamins.

W. Massy (1965), 'Principal components regression in exploratory statistical research', *J. Amer. Statist. Assoc.* **60**, 234–246.

J. Mena (1999), *Data Mining Your Website*, Digital Press, Boston.

C. Meyer (2000), *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia.

L. Mirsky (1960), 'Symmetric gauge functions and unitarily invariant norms', *Quart. J. Math. Oxford* **11**, 50–59.

C. Moler (2002), 'The world's largest matrix computation', *Matlab News and Notes*, October 2002, pp. 12–13.

L. Page, S. Brin, R. Motwani and T. Winograd (1998), 'The PageRank citation ranking: Bringing order to the Web', Stanford Digital Library Working Papers.

C. Paige and M. Saunders (1981), 'Towards a generalized singular value decomposition', *SIAM J. Numer. Anal.* **18**, 398–405.

C. Paige and M. Saunders (1982), 'LSQR: An algorithm for sparse linear equations and sparse least squares', *ACM Trans. Math. Software* **8**, 43–71.

H. Park and L. Eldén (2005), Matrix rank reduction for data analysis and feature extraction, in *Handbook of Parallel Computing and Statistics* (E. Kontoghiorghes, ed.), CRC Press, Boca Raton.

H. Park, M. Jeon and J. B. Rosen (2001), Lower dimensional representation of text data in vector space based information retrieval, in *Computational Information Retrieval* (M. Berry, ed.), SIAM, Philadelphia, PA, pp. 3–23.

H. Park, M. Jeon and J. B. Rosen (2003), 'Lower dimensional representation of text data based on centroids and least squares', *BIT* **43**, 427–448.

A. Phatak and F. de Hoog (2002), 'Exploiting the connection between PLS, Lanczos methods and conjugate gradients: Alternative proofs of some properties of PLS', *J. Chemometrics* **16**, 361–367.

Y. Saad (2003), *Iterative Methods for Sparse Linear Systems*, 2nd edn, SIAM.

G. Salton, C. Yang and A. Wong (1975), 'A vector-space model for automatic indexing', *Comm. Assoc. Comput. Mach.* **18**, 613–620.

B. Savas (2002), Analyses and test of handwritten digit algorithms. Master's thesis, Mathematics Department, Linköping University.

S. Serra-Capizzano (2005), 'Jordan canonical form of the Google matrix: A potential contribution to the Pagerank computation', *SIAM J. Matrix Anal. Appl.* **27**, 305–312.

N. D. Sidiropoulos and R. Bro (2000), 'On the uniqueness of multilinear decomposition of N-way arrays', *J. Chemometrics* **14**, 229–239.

P. Simard, Y. LeCun and J. Denker (1993), Efficient pattern recognition using a new transformation distance, in *Advances in Neural Information Processing Systems 5* (J. Cowan, S. Hanson and C. Giles, eds), Morgan Kaufmann, pp. 50–58.

P. Simard, Y. LeCun, J. Denker and B. Victorri (2001), 'Transformation invariance in pattern recognition: Tangent distance and tangent propagation', *Internat. J. Imaging System Techn.* **11**, 181–194.

H. D. Simon, A. Sohn and R. Biswas (1998), 'Harp: A dynamic spectral partitioner.', *J. Parallel Distrib. Comput.* **50**, 83–103.

M. Sjöström and S. Wold (1980), SIMCA: A pattern recognition method based on principal components models, in *Pattern Recognition in Practice* (E. S. Gelsema and L. N. Kanal, eds), North-Holland, pp. 351–359.

A. Smilde, R. Bro and P. Geladi (2004), *Multi-Way Analysis: Applications in the Chemical Sciences*, Wiley.

J. Tenenbaum and W. Freeman (2000), 'Separating style and content with bilinear models', *Neural Computation* **12**, 1247–1283.

M. Totty and M. Mangalindan (2003), 'As Google becomes Web's gatekeeper, sites fight to get in', *Wall Street Journal* **CCXLI**.

L. Tucker (1964), The extension of factor analysis to three-dimensional matrices, in *Contributions to Mathematical Psychology* (H. Gulliksen and N. Frederiksen, eds), Holt, Rinehart and Winston, New York, pp. 109–127.

L. Tucker (1966), 'Some mathematical notes on three-mode factor analysis', *Psychometrika* **31**, 279–311.

C. Van Loan (1976), 'Generalizing the singular value decomposition', *SIAM J. Numer. Anal.* **13**, 76–83.

M. Vasilescu and D. Terzopoulos (2002*a*), Multilinear analysis of image ensembles: Tensorfaces, in *Proc. 7th European Conference on Computer Vision* (ECCV'02), Vol. 2350 of *Lecture Notes in Computer Science*, Springer, Copenhagen, Denmark, pp. 447–460.

M. Vasilescu and D. Terzopoulos (2002*b*), Multilinear image analysis for facial recognition, in *International Conference on Pattern Recognition, Quebec City, Canada* (ICPR '02), IEEE Computer Society, pp. 511–514.

M. Vasilescu and D. Terzopoulos (2003), Multilinear subspace analysis of image ensembles, in *IEEE Conference on Computer Vision and Pattern Recognition, Madison WI* (CVPR'03), pp. 93–99.

J. Wedderburn (1934), *Lectures on Matrices*, Colloquium Publications, AMS, New York.

I. Witten and E. Frank (2000), *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, San Francisco.

H. Wold (1975), Soft modeling by latent variables: The nonlinear iterative partial least squares approach, in *Perspectives in Probability and Statistics: Papers in Honour of M. S. Bartlett* (J. Gani, ed.), Academic Press, London.

S. Wold (1976), 'Pattern recognition by means of disjoint principal components models', *Pattern Recognition* **8**, 127–139.

S. Wold, A. Ruhe, H. Wold and W. Dunn (1984), 'The collinearity problem in linear regression: The partial least squares (PLS) approach to generalized inverses', *SIAM J. Sci. Statist. Comput.* **5**, 735–743.

S. Wold, M. Sjöström and L. Eriksson (2001), 'PLS-regression: A basic tool of chemometrics', *Chemometrics and Intell. Lab. Systems* **58**, 109–130.

S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang and H. Zhang (2005), Discriminant analysis with tensor representation, in *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR'05).

J. Ye (2005), 'Generalized low rank approximations of matrices', *Machine Learning* **61**, 167–191.

D. Zeimpekis and E. Gallopoulos (2005), Design of a MATLAB toolbox for term-document matrix generation, in *Proc. Workshop on Clustering High Dimensional Data and its Applications* (I. Dhillon, J. Kogan and J. Ghosh, eds), Newport Beach, CA, pp. 38–48.

H. Zha (2002), Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering, in *Proc. 25th Annual International ACM–SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, pp. 113–120.

H. Zha, C. Ding, M. Gu, X. He and H. Simon (2002), Spectral relaxation for k-means clustering, in *Advances in Neural Information Processing Systems* (T. Dietterich, S. Becker and Z. Ghahramani, eds), MIT Press, pp. 1057–1064.