3 Best-Fit Subspaces and Singular Value Decomposition (SVD)

3.1 Introduction

In this chapter, we examine the *Singular Value Decomposition* (SVD) of a matrix. Consider each row of an $n \times d$ matrix A as a point in d-dimensional space. The singular value decomposition finds the best-fitting k-dimensional subspace for $k = 1, 2, 3, \ldots$, for the set of n data points. Here, "best" means minimizing the sum of the squares of the perpendicular distances of the points to the subspace, or equivalently, maximizing the sum of squares of the lengths of the projections of the points onto this subspace.[4] We begin with a special case where the subspace is 1-dimensional, namely a line through the origin. We then show that the best-fitting k-dimensional subspace can be found by k applications of the best fitting line algorithm, where on the i^{th} iteration we find the best fit line perpendicular to the previous $i-1$ lines. When k reaches the rank of the matrix, from these operations we get an exact decomposition of the matrix called the *singular value decomposition*.

In matrix notation, the singular value decomposition of a matrix A with real entries (we assume all our matrices have real entries) is the factorization of A into the product of three matrices, $A = UDV^T$, where the columns of U and V are orthonormal[5] and the matrix D is diagonal with positive real entries. The columns of V are the unit length vectors defining the best fitting lines described above (the i^{th} column being the unit-length vector in the direction of the i^{th} line). The coordinates of a row of U will be the fractions of the corresponding row of A along the direction of each of the lines.

The SVD is useful in many tasks. Often a data matrix A is close to a low rank matrix and it is useful to find a good low rank approximation to A. For any k, the singular value decomposition of A gives the best rank-k approximation to A in a well-defined sense.

If $\mathbf{u_i}$ and $\mathbf{v_i}$ are columns of U and V respectively, then the matrix equation $A = UDV^T$ can be rewritten as

$$A = \sum_i d_{ii} \mathbf{u_i} \mathbf{v_i}^T.$$

Since $\mathbf{u_i}$ is a $n \times 1$ matrix and $\mathbf{v_i}$ is a $d \times 1$ matrix, $\mathbf{u_i}\mathbf{v_i}^T$ is an $n \times d$ matrix with the same dimensions as A. The i^{th} term in the above sum can be viewed as giving the components of the rows of A along direction $\mathbf{v_i}$. When the terms are summed, they reconstruct A.

[4] This equivalence is due to the Pythagorean Theorem. For each point, its squared length (its distance to the origin squared) is exactly equal to the squared length of its projection onto the subspace plus the squared distance of the point to its projection; therefore, maximizing the sum of the former is equivalent to minimizing the sum of the latter. For further discussion see Section 3.2.

[5] A set of vectors is orthonormal if each is of length one and they are pairwise orthogonal.

This decomposition of A can be viewed as analogous to writing a vector $\mathbf{x}$ in some orthonormal basis $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_d}$. The coordinates of $\mathbf{x} = (\mathbf{x} \cdot \mathbf{v_1}, \mathbf{x} \cdot \mathbf{v_2} \ldots, \mathbf{x} \cdot \mathbf{v_d})$ are the projections of $\mathbf{x}$ onto the $\mathbf{v_i}$'s. For SVD, this basis has the property that for any k, the first k vectors of this basis produce the least possible total sum of squares error for that value of k.

In addition to the singular value decomposition, there is an eigenvalue decomposition. Let A be a square matrix. A vector $\mathbf{v}$ such that $A\mathbf{v} = \lambda\mathbf{v}$ is called an eigenvector and λ the eigenvalue. When A is symmetric, the eigenvectors are orthogonal and A can be expressed as $A = VDV^T$ where the eigenvectors are the columns of V and D is a diagonal matrix with the corresponding eigenvalues on its diagonal. For a symmetric matrix A the singular values and eigenvalues are identical. If the singular values are distinct, then A's singular vectors and eigenvectors are identical. If a singular value has multiplicity d greater than one, the corresponding singular vectors span a subspace of dimension d and any orthogonal basis of the subspace can be used as the eigenvectors or singular vectors.[6]

The singular value decomposition is defined for all matrices, whereas the more familiar eigenvector decomposition requires that the matrix A be square and certain other conditions on the matrix to ensure orthogonality of the eigenvectors. In contrast, the columns of V in the singular value decomposition, called the *right-singular vectors* of A, always form an orthogonal set with no assumptions on A. The columns of U are called the *left-singular vectors* and they also form an orthogonal set (see Section 3.6). A simple consequence of the orthonormality is that for a square and invertible matrix A, the inverse of A is $VD^{-1}U^T$.

Eigenvalues and eignevectors satisfy $A\mathbf{v} = \lambda\mathbf{v}$. We will show that singular values and vectors satisfy a somewhat analogous relationship. Since $A\mathbf{v_i}$ is a $n \times 1$ matrix (vector), the matrix A cannot act on it from the left. But A^T, which is a $d \times n$ matrix, can act on this vector. Indeed, we will show that

$$A\mathbf{v_i} = d_{ii}\mathbf{u_i} \quad \text{and} \quad A^T\mathbf{u_i} = d_{ii}\mathbf{v_i}.$$

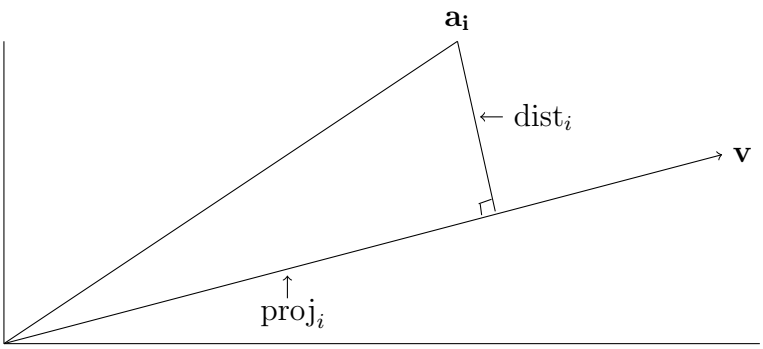
In words, A acting on $\mathbf{v_i}$ produces a scalar multiple of $\mathbf{u_i}$ and A^T acting on $\mathbf{u_i}$ produces the same scalar multiple of $\mathbf{v_i}$. Note that $A^TA\mathbf{v_i} = d_{ii}^2\mathbf{v_i}$. The i^{th} singular vector of A is the i^{th} eigenvector of the square symmetric matrix A^TA.

3.2 Preliminaries

Consider projecting a point $\mathbf{a_i} = (a_{i1}, a_{i2}, \ldots, a_{id})$ onto a line through the origin. Then

$$a_{i1}^2 + a_{i2}^2 + \cdots + a_{id}^2 = (\text{length of projection})^2 + (\text{distance of point to line})^2.$$

[6]When $d = 1$ there are actually two possible singular vectors, one the negative of the other. The subspace spanned is unique.



Minimizing $\sum_i \text{dist}_i^2$ is equivalent to maximizing $\sum_i \text{proj}_i^2$

Figure 3.1: The projection of the point $\mathbf{a_i}$ onto the line through the origin in the direction of $\mathbf{v}$.

This holds by the Pythagorean Theorem (see Figure 3.1). Thus

$$(\text{distance of point to line})^2 = a_{i1}^2 + a_{i2}^2 + \cdots + a_{id}^2 - (\text{length of projection})^2 .$$

Since $\sum_{i=1}^{n} (a_{i1}^2 + a_{i2}^2 + \cdots + a_{id}^2)$ is a constant independent of the line, minimizing the sum of the squares of the distances to the line is equivalent to maximizing the sum of the squares of the lengths of the projections onto the line. Similarly for best-fit subspaces, maximizing the sum of the squared lengths of the projections onto the subspace minimizes the sum of squared distances to the subspace.

Thus we have two interpretations of the best-fit subspace. The first is that it minimizes the sum of squared distances of the data points to it. This first interpretation and its use are akin to the notion of least-squares fit from calculus.[7] The second interpretation of best-fit-subspace is that it maximizes the sum of projections squared of the data points on it. This says that the subspace contains the maximum content of data among all subspaces of the same dimension. The choice of the objective function as the sum of squared distances seems a bit arbitrary and in a way it is. But the square has many nice mathematical properties. The first of these, as we have just seen, is that minimizing the sum of squared distances is equivalent to maximizing the sum of squared projections.

3.3 Singular Vectors

We now define the *singular vectors* of an $n \times d$ matrix A. Consider the rows of A as n points in a d-dimensional space. Consider the best fit line through the origin. Let $\mathbf{v}$ be a unit vector along this line. The length of the projection of $\mathbf{a_i}$, the i^{th} row of A, onto $\mathbf{v}$ is $|\mathbf{a_i} \cdot \mathbf{v}|$. From this we see that the sum of the squared lengths of the projections is

[7] But there is a difference: here we take the perpendicular distance to the line or subspace, whereas, in the calculus notion, given n pairs, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, we find a line $l = \{(x, y)|y = mx + b\}$ minimizing the vertical squared distances of the points to it, namely, $\sum_{i=1}^{n} (y_i - mx_i - b)^2$.

$|A\mathbf{v}|^2$. The best fit line is the one maximizing $|A\mathbf{v}|^2$ and hence minimizing the sum of the squared distances of the points to the line.

With this in mind, define the *first singular vector* $\mathbf{v_1}$ of A as

$$\mathbf{v_1} = \arg\max_{|\mathbf{v}|=1} |A\mathbf{v}|.$$

Technically, there may be a tie for the vector attaining the maximum and so we should not use the article "the"; in fact, $-\mathbf{v_1}$ is always as good as $\mathbf{v_1}$. In this case, we arbitrarily pick one of the vectors achieving the maximum and refer to it as "the first singular vector" avoiding the more cumbersome "one of the vectors achieving the maximum". We adopt this terminology for all uses of arg max .

The value $\sigma_1(A) = |A\mathbf{v_1}|$ is called the *first singular value* of A. Note that $\sigma_1^2 = \sum_{i=1}^{n} (\mathbf{a_i} \cdot \mathbf{v_1})^2$ is the sum of the squared lengths of the projections of the points onto the line determined by $\mathbf{v_1}$.

If the data points were all either on a line or close to a line, intuitively, $\mathbf{v_1}$ should give us the direction of that line. It is possible that data points are not close to one line, but lie close to a 2-dimensional subspace or more generally a low dimensional space. Suppose we have an algorithm for finding $\mathbf{v_1}$ (we will describe one such algorithm later). How do we use this to find the best-fit 2-dimensional plane or more generally the best fit k-dimensional space?

The greedy approach begins by finding $\mathbf{v_1}$ and then finds the best 2-dimensional subspace containing $\mathbf{v_1}$. The sum of squared distances helps. For every 2-dimensional subspace containing $\mathbf{v_1}$, the sum of squared lengths of the projections onto the subspace equals the sum of squared projections onto $\mathbf{v_1}$ plus the sum of squared projections along a vector perpendicular to $\mathbf{v_1}$ in the subspace. Thus, instead of looking for the best 2-dimensional subspace containing $\mathbf{v_1}$, look for a unit vector $\mathbf{v_2}$ perpendicular to $\mathbf{v_1}$ that maximizes $|A\mathbf{v}|^2$ among all such unit vectors. Using the same greedy strategy to find the best three and higher dimensional subspaces, defines $\mathbf{v_3}, \mathbf{v_4}, \ldots$ in a similar manner. This is captured in the following definitions. There is no apriori guarantee that the greedy algorithm gives the best fit. But, in fact, the greedy algorithm does work and yields the best-fit subspaces of every dimension as we will show.

The *second singular vector*, $\mathbf{v_2}$, is defined by the best fit line perpendicular to $\mathbf{v_1}$.

$$\mathbf{v_2} = \arg\max_{\substack{\mathbf{v} \perp \mathbf{v_1} \\ |\mathbf{v}|=1}} |A\mathbf{v}|$$

The value $\sigma_2(A) = |A\mathbf{v_2}|$ is called the *second singular value* of A**.** The *third singular*

vector $\mathbf{v}_3$ and the *third singular value* are defined similarly by

$$\mathbf{v_3} = \arg\max_{\substack{\mathbf{v}\perp\mathbf{v_1},\mathbf{v_2}\\ |\mathbf{v}|=1}} |A\mathbf{v}|$$

and

$$\sigma_3(A) = |A\mathbf{v_3}|,$$

and so on. The process stops when we have found singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$, singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$, and

$$\max_{\substack{\mathbf{v}\perp\mathbf{v_1},\mathbf{v_2},\ldots,\mathbf{v_r}\\ |\mathbf{v}|=1}} |A\mathbf{v}| = 0.$$

The greedy algorithm found the $\mathbf{v_1}$ that maximized $|A\mathbf{v}|$ and then the best fit 2-dimensional subspace containing $\mathbf{v_1}$. Is this necessarily the best-fit 2-dimensional subspace overall? The following theorem establishes that the greedy algorithm finds the best subspaces of every dimension.

Theorem 3.1 (The Greedy Algorithm Works) *Let A be an $n \times d$ matrix with singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$. For $1 \leq k \leq r$, let V_k be the subspace spanned by $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$. For each k, V_k is the best-fit k-dimensional subspace for A.*

Proof: The statement is obviously true for $k = 1$. For $k = 2$, let W be a best-fit 2-dimensional subspace for A. For any orthonormal basis $(\mathbf{w_1}, \mathbf{w_2})$ of W, $|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2$ is the sum of squared lengths of the projections of the rows of A onto W. Choose an orthonormal basis $(\mathbf{w}_1, \mathbf{w_2})$ of W so that $\mathbf{w_2}$ is perpendicular to $\mathbf{v_1}$. If $\mathbf{v_1}$ is perpendicular to W, any unit vector in W will do as $\mathbf{w_2}$. If not, choose $\mathbf{w_2}$ to be the unit vector in W perpendicular to the projection of $\mathbf{v_1}$ onto W. This makes $\mathbf{w_2}$ perpendicular to $\mathbf{v_1}$.[8] Since $\mathbf{v_1}$ maximizes $|A\mathbf{v}|^2$, it follows that $|A\mathbf{w_1}|^2 \leq |A\mathbf{v_1}|^2$. Since $\mathbf{v_2}$ maximizes $|A\mathbf{v}|^2$ over all $\mathbf{v}$ perpendicular to $\mathbf{v_1}$, $|A\mathbf{w_2}|^2 \leq |A\mathbf{v_2}|^2$. Thus

$$|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2 \leq |A\mathbf{v_1}|^2 + |A\mathbf{v_2}|^2.$$

Hence, V_2 is at least as good as W and so is a best-fit 2-dimensional subspace.

For general k, proceed by induction. By the induction hypothesis, V_{k-1} is a best-fit k-1 dimensional subspace. Suppose W is a best-fit k-dimensional subspace. Choose an orthonormal basis $\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_k}$ of W so that $\mathbf{w_k}$ is perpendicular to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k-1}}$. Then

$$|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2 + \cdots + |A\mathbf{w_{k-1}}|^2 \leq |A\mathbf{v_1}|^2 + |A\mathbf{v_2}|^2 + \cdots + |A\mathbf{v_{k-1}}|^2$$

since V_{k-1} is an optimal $k-1$ dimensional subspace. Since $\mathbf{w_k}$ is perpendicular to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k-1}}$, by the definition of $\mathbf{v_k}$, $|A\mathbf{w_k}|^2 \leq |A\mathbf{v_k}|^2$. Thus

$$|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2 + \cdots + |A\mathbf{w_{k-1}}|^2 + |A\mathbf{w_k}|^2 \leq |A\mathbf{v_1}|^2 + |A\mathbf{v_2}|^2 + \cdots + |A\mathbf{v_{k-1}}|^2 + |A\mathbf{v_k}|^2,$$

proving that V_k is at least as good as W and hence is optimal. ∎

[8] This can be seen by noting that $\mathbf{v_1}$ is the sum of two vectors that each are individually perpendicular to $\mathbf{w_2}$, namely the projection of $\mathbf{v_1}$ to W and the portion of $\mathbf{v_1}$ orthogonal to W.

Note that the n-dimensional vector $A\mathbf{v_i}$ is a list of lengths (with signs) of the projections of the rows of A onto $\mathbf{v_i}$. Think of $|A\mathbf{v_i}| = \sigma_i(A)$ as the *component* of the matrix A along $\mathbf{v_i}$. For this interpretation to make sense, it should be true that adding up the squares of the components of A along each of the $\mathbf{v_i}$ gives the square of the "whole content of A". This is indeed the case and is the matrix analogy of decomposing a vector into its components along orthogonal directions.

Consider one row, say $\mathbf{a_j}$, of A. Since $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ span the space of all rows of A, $\mathbf{a_j} \cdot \mathbf{v} = 0$ for all $\mathbf{v}$ perpendicular to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$. Thus, for each row $\mathbf{a_j}$, $\sum_{i=1}^{r} (\mathbf{a_j} \cdot \mathbf{v_i})^2 = |\mathbf{a_j}|^2$. Summing over all rows j,

$$\sum_{j=1}^{n} |\mathbf{a_j}|^2 = \sum_{j=1}^{n} \sum_{i=1}^{r} (\mathbf{a_j} \cdot \mathbf{v_i})^2 = \sum_{i=1}^{r} \sum_{j=1}^{n} (\mathbf{a_j} \cdot \mathbf{v_i})^2 = \sum_{i=1}^{r} |A\mathbf{v_i}|^2 = \sum_{i=1}^{r} \sigma_i^2(A).$$

But $\sum_{j=1}^{n} |\mathbf{a_j}|^2 = \sum_{j=1}^{n} \sum_{k=1}^{d} a_{jk}^2$, the sum of squares of all the entries of A. Thus, the sum of squares of the singular values of A is indeed the square of the "whole content of A", i.e., the sum of squares of all the entries. There is an important norm associated with this quantity, the Frobenius norm of A, denoted $||A||_F$ defined as

$$||A||_F = \sqrt{\sum_{j,k} a_{jk}^2}.$$

Lemma 3.2 *For any matrix A, the sum of squares of the singular values equals the square of the Frobenius norm. That is, $\sum \sigma_i^2(A) = ||A||_F^2$.*

Proof: By the preceding discussion. ∎

The vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ are called the *right-singular vectors*. The vectors $A\mathbf{v_i}$ form a fundamental set of vectors and we normalize them to length one by

$$\mathbf{u_i} = \frac{1}{\sigma_i(A)} A\mathbf{v_i}.$$

Later we will show that $\mathbf{u}_i$ similarly maximizes $|\mathbf{u}^T A|$ over all $\mathbf{u}$ perpendicular to $\mathbf{u}_1, \ldots, \mathbf{u}_{i-1}$. These $\mathbf{u}_i$ are called the *left-singular vectors*. Clearly, the right-singular vectors are orthogonal by definition. We will show later that the left-singular vectors are also orthogonal.

3.4 Singular Value Decomposition (SVD)

Let A be an $n \times d$ matrix with singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ and corresponding singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$. The left-singular vectors of A are $\mathbf{u_i} = \frac{1}{\sigma_i} A\mathbf{v_i}$ where $\sigma_i \mathbf{u_i}$ is

a vector whose coordinates correspond to the projections of the rows of A onto $\mathbf{v_i}$. Each $\sigma_i \mathbf{u_i} \mathbf{v_i^T}$ is a rank one matrix whose rows are the "$\mathbf{v_i}$ components" of the rows of A, i.e., the projections of the rows of A in the $\mathbf{v_i}$ direction. We will prove that A can be decomposed into a sum of rank one matrices as

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

Geometrically, each point is decomposed in A into its components along each of the r orthogonal directions given by the $\mathbf{v_i}$. We will also prove this algebraically. We begin with a simple lemma that two matrices A and B are identical if $A\mathbf{v} = B\mathbf{v}$ for all $\mathbf{v}$.

Lemma 3.3 *Matrices A and B are identical if and only if for all vectors $\mathbf{v}$, $A\mathbf{v} = B\mathbf{v}$.*

Proof: Clearly, if $A = B$ then $A\mathbf{v} = B\mathbf{v}$ for all $\mathbf{v}$. For the converse, suppose that $A\mathbf{v} = B\mathbf{v}$ for all $\mathbf{v}$. Let $\mathbf{e_i}$ be the vector that is all zeros except for the i^{th} component which has value one. Now $A\mathbf{e_i}$ is the i^{th} column of A and thus $A = B$ if for each i, $A\mathbf{e_i} = B\mathbf{e_i}$. ∎

Theorem 3.4 *Let A be an $n \times d$ matrix with right-singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$, left-singular vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}$, and corresponding singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$. Then*

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

Proof: We first show that multiplying both A and $\sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ by $\mathbf{v_j}$ results in equality.

$$\sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T \mathbf{v_j} = \sigma_j \mathbf{u_j} = A\mathbf{v_j}$$

Since any vector $\mathbf{v}$ can be expressed as a linear combination of the singular vectors plus a vector perpendicular to the $\mathbf{v_i}$, $A\mathbf{v} = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T \mathbf{v}$ for all $\mathbf{v}$ and by Lemma 3.3, $A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$. ∎

The decomposition $A = \sum_i \sigma_i \mathbf{u_i} \mathbf{v_i^T}$ is called the *singular value decomposition*, *SVD*, of A. We can rewrite this equation in matrix notation as $A = UDV^T$ where $\mathbf{u_i}$ is the i^{th} column of U, $\mathbf{v_i^T}$ is the i^{th} row of V^T, and D is a diagonal matrix with σ_i as the i^{th} entry on its diagonal. For any matrix A, the sequence of singular values is unique and if the singular values are all distinct, then the sequence of singular vectors is unique up to signs. However, when some set of singular values are equal, the corresponding singular vectors span some subspace. Any set of orthonormal vectors spanning this subspace can be used as the singular vectors.

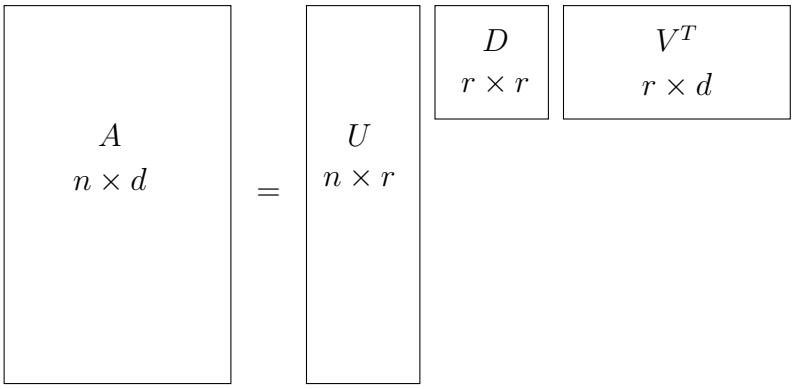


Figure 3.2: The SVD decomposition of an $n \times d$ matrix.

3.5 Best Rank-k Approximations

Let A be an $n \times d$ matrix and think of the rows of A as n points in d-dimensional space. Let

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

be the SVD of A. For $k \in \{1, 2, \ldots, r\}$, let

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

be the sum truncated after k terms. It is clear that A_k has rank k. We show that A_k is the best rank k approximation to A, where error is measured in the Frobenius norm. Geometrically, this says that $\mathbf{v_1}, \ldots, \mathbf{v_k}$ define the k-dimensional space minimizing the sum of squared distances of the points to the space. To see why, we need the following lemma.

Lemma 3.5 *The rows of A_k are the projections of the rows of A onto the subspace V_k spanned by the first k singular vectors of A.*

Proof: Let $\mathbf{a}$ be an arbitrary row vector. Since the $\mathbf{v_i}$ are orthonormal, the projection of the vector $\mathbf{a}$ onto V_k is given by $\sum_{i=1}^{k} (\mathbf{a} \cdot \mathbf{v_i}) \mathbf{v_i}^T$. Thus, the matrix whose rows are the projections of the rows of A onto V_k is given by $\sum_{i=1}^{k} A \mathbf{v_i} \mathbf{v_i}^T$. This last expression simplifies to

$$\sum_{i=1}^{k} A \mathbf{v_i} \mathbf{v_i}^T = \sum_{i=1}^{k} \sigma_i \mathbf{u_i} \mathbf{v_i}^T = A_k.$$

∎

Theorem 3.6 *For any matrix B of rank at most k*

$$\|A - A_k\|_F \leq \|A - B\|_F$$

Proof: Let B minimize $\|A - B\|_F^2$ among all rank k or less matrices. Let V be the space spanned by the rows of B. The dimension of V is at most k. Since B minimizes $\|A - B\|_F^2$, it must be that each row of B is the projection of the corresponding row of A onto V: Otherwise replace the row of B with the projection of the corresponding row of A onto V. This still keeps the row space of B contained in V and hence the rank of B is still at most k. But it reduces $\|A - B\|_F^2$, contradicting the minimality of $||A - B||_F$.

Since each row of B is the projection of the corresponding row of A, it follows that $\|A - B\|_F^2$ is the sum of squared distances of rows of A to V. Since A_k minimizes the sum of squared distance of rows of A to any k-dimensional subspace, from Theorem 3.1, it follows that $\|A - A_k\|_F \leq \|A - B\|_F$. ∎

In addition to the Frobenius norm, there is another matrix norm of interest. Consider an $n \times d$ matrix A and a large number of vectors where for each vector $\mathbf{x}$ we wish to compute $A\mathbf{x}$. It takes time $O(nd)$ to compute each product $A\mathbf{x}$ but if we approximate A by $A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ and approximate $A\mathbf{x}$ by $A_k\mathbf{x}$ it requires only k dot products of d-dimensional vectors, followed by a sum of k n-dimensional vectors, and takes time $O(kd + kn)$, which is a win provided $k \ll \min(d, n)$. How is the error measured? Since $\mathbf{x}$ is unknown, the approximation needs to be good for every $\mathbf{x}$. So we take the maximum over all $\mathbf{x}$ of $|(A_k - A)\mathbf{x}|$. Since this would be infinite if $|\mathbf{x}|$ could grow without bound, we restrict the maximum to $|\mathbf{x}| \leq 1$. Formally, we define a new norm of a matrix A by

$$||A||_2 = \max_{|\mathbf{x}| \leq 1} |A\mathbf{x}|.$$

This is called the 2-norm or the spectral norm. Note that it equals $\sigma_1(A)$.

As an application consider a large database of documents that form rows of an $n \times d$ matrix A. There are d terms and each document is a d-dimensional vector with one component for each term, which is the number of occurrences of the term in the document. We are allowed to "preprocess" A. After the preprocessing, we receive queries. Each query $\mathbf{x}$ is an d-dimensional vector which specifies how important each term is to the query. The desired answer is an n-dimensional vector which gives the similarity (dot product) of the query to each document in the database, namely $A\mathbf{x}$, the "matrix-vector" product. Query time is to be much less than preprocessing time, since the idea is that we need to answer many queries for the same database. There are many other applications where one performs many matrix vector products with the same matrix. This technique is applicable to these situations as well.

3.6 Left Singular Vectors

The left singular vectors are also pairwise orthogonal. Intuitively if $\mathbf{u_i}$ and $\mathbf{u_j}$, $i < j$, were not orthogonal, one would suspect that the right singular vector $\mathbf{v_j}$ had a component of $\mathbf{v_i}$

which would contradict that $\mathbf{v_i}$ and $\mathbf{v_j}$ were orthogonal. Let i be the smallest integer such that $\mathbf{u_i}$ is not orthogonal to all other $\mathbf{u_j}$. Then to prove that $\mathbf{u_i}$ and $\mathbf{u_j}$ are orthogonal, we add a small component of $\mathbf{v_j}$ to $\mathbf{v_i}$, normalize the result to be a unit vector

$$\mathbf{v_i'} = \frac{\mathbf{v_i} + \epsilon \mathbf{v_j}}{|\mathbf{v_i} + \epsilon \mathbf{v_j}|}$$

and show that $|A\mathbf{v_i'}| > |A\mathbf{v_i}|$, a contradiction.

Theorem 3.7 *The left singular vectors are pairwise orthogonal.*

Proof: Let i be the smallest integer such that $\mathbf{u_i}$ is not orthogonal to some other $\mathbf{u_j}$. Without loss of generality assume that $\mathbf{u_i}^T\mathbf{u_j} = \delta > 0$. If $\mathbf{u_i}^T\mathbf{u_j} < 0$ then just replace $\mathbf{u_i}$ with $-\mathbf{u_i}$. Clearly $j > i$ since i was selected to be the smallest such index. For $\varepsilon > 0$, let

$$\mathbf{v_i'} = \frac{\mathbf{v_i} + \epsilon \mathbf{v_j}}{|\mathbf{v_i} + \epsilon \mathbf{v_j}|}.$$

Notice that $\mathbf{v_i'}$ is a unit-length vector.

$$A\mathbf{v_i'} = \frac{\sigma_i \mathbf{u_i} + \varepsilon\sigma_j \mathbf{u_j}}{\sqrt{1+\varepsilon^2}}$$

has length at least as large as its component along $\mathbf{u_i}$ which is

$$\mathbf{u_i^T}\left(\frac{\sigma_i \mathbf{u_i} + \varepsilon\sigma_j \mathbf{u_j}}{\sqrt{1+\varepsilon^2}}\right) > (\sigma_i + \varepsilon\sigma_j\delta)\left(1 - \tfrac{\varepsilon^2}{2}\right) > \sigma_i - \tfrac{\varepsilon^2}{2}\sigma_i + \varepsilon\sigma_j\delta - \tfrac{\varepsilon^3}{2}\sigma_j\delta > \sigma_i,$$

for sufficiently small ϵ, a contradiction since $\mathbf{v_i} + \varepsilon\mathbf{v_j}$ is orthogonal to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{i-1}}$ since $j > i$ and σ_i is defined to be the maximum of $|A\mathbf{v}|$ over such vectors. ∎

Next we prove that A_k is the best rank k, 2-norm approximation to A. We first show that the square of the 2-norm of $A - A_k$ is the square of the $(k+1)^{st}$ singular value of A. This is essentially by definition of A_k; that is, A_k represents the projections of the rows in A onto the space spanned by the top k singular vectors, and so $A - A_k$ is the remaining portion of those rows, whose top singular value will be σ_{k+1}.

Lemma 3.8 $\|A - A_k\|_2^2 = \sigma_{k+1}^2$.

Proof: Let $A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i}\mathbf{v_i}^T$ be the singular value decomposition of A. Then $A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^T$ and $A - A_k = \sum_{i=k+1}^{r} \sigma_i \mathbf{u_i}\mathbf{v_i}^T$. Let $\mathbf{v}$ be the top singular vector of $A - A_k$. Express $\mathbf{v}$ as a linear combination of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$. That is, write $\mathbf{v} = \sum_{j=1}^{r} c_j \mathbf{v_j}$. Then

$$\begin{aligned} |(A - A_k)\mathbf{v}| &= \left|\sum_{i=k+1}^{r} \sigma_i \mathbf{u_i}\mathbf{v_i}^T \sum_{j=1}^{r} c_j \mathbf{v_j}\right| = \left|\sum_{i=k+1}^{r} c_i \sigma_i \mathbf{u_i}\mathbf{v_i}^T \mathbf{v_i}\right| \\ &= \left|\sum_{i=k+1}^{r} c_i \sigma_i \mathbf{u_i}\right| = \sqrt{\sum_{i=k+1}^{r} c_i^2 \sigma_i^2}, \end{aligned}$$

since the $\mathbf{u_i}$ are orthonormal. The $\mathbf{v}$ maximizing this last quantity, subject to the constraint that $|\mathbf{v}|^2 = \sum_{i=1}^{r} c_i^2 = 1$, occurs when $c_{k+1} = 1$ and the rest of the c_i are zero. Thus, $\|A - A_k\|_2^2 = \sigma_{k+1}^2$ proving the lemma. ∎

Finally, we prove that A_k is the best rank k, 2-norm approximation to A:

Theorem 3.9 *Let A be an $n \times d$ matrix. For any matrix B of rank at most k*

$$\|A - A_k\|_2 \leq \|A - B\|_2 .$$

Proof: If A is of rank k or less, the theorem is obviously true since $\|A - A_k\|_2 = 0$. Assume that A is of rank greater than k. By Lemma 3.8, $\|A - A_k\|_2^2 = \sigma_{k+1}^2$. The null space of B, the set of vectors $\mathbf{v}$ such that $B\mathbf{v} = 0$, has dimension at least $d - k$. Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}$ be the first $k+1$ singular vectors of A. By a dimension argument, it follows that there exists a $\mathbf{z} \neq 0$ in

$$\text{Null}\,(B) \cap \text{Span}\left\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}\right\}.$$

Scale $\mathbf{z}$ to be of length one.

$$\|A - B\|_2^2 \geq |(A - B)\,\mathbf{z}|^2 .$$

Since $B\mathbf{z} = 0$,

$$\|A - B\|_2^2 \geq |A\mathbf{z}|^2 .$$

Since $\mathbf{z}$ is in the Span $\left\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}\right\}$

$$|A\mathbf{z}|^2 = \left|\sum_{i=1}^{n} \sigma_i \mathbf{u_i} \mathbf{v_i}^T \mathbf{z}\right|^2 = \sum_{i=1}^{n} \sigma_i^2 \left(\mathbf{v_i}^T \mathbf{z}\right)^2 = \sum_{i=1}^{k+1} \sigma_i^2 \left(\mathbf{v_i}^T \mathbf{z}\right)^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} \left(\mathbf{v_i}^T \mathbf{z}\right)^2 = \sigma_{k+1}^2.$$

It follows that $\|A - B\|_2^2 \geq \sigma_{k+1}^2$ proving the theorem. ∎

For a square symmetric matrix A and eigenvector $\mathbf{v}$, $A\mathbf{v} = \lambda\mathbf{v}$. We now prove the analog for singular values and vectors we discussed in the introduction.

Lemma 3.10 (Analog of eigenvalues and eigenvectors)

$$A\mathbf{v_i} = \sigma_i \mathbf{u_i} \text{ and } A^T \mathbf{u_i} = \sigma_i \mathbf{v_i}.$$

Proof: The first equation follows from the definition of left singular vectors. For the second, note that from the SVD, we get $A^T \mathbf{u_i} = \sum_j \sigma_j \mathbf{v_j} \mathbf{u_j}^T \mathbf{u_i}$, where since the $\mathbf{u_j}$ are orthonormal, all terms in the summation are zero except for $j = i$. ∎

3.7 Power Method for Singular Value Decomposition

Computing the singular value decomposition is an important branch of numerical analysis in which there have been many sophisticated developments over a long period of time. The reader is referred to numerical analysis texts for more details. Here we present an "in-principle" method to establish that the approximate SVD of a matrix A can be computed in polynomial time. The method we present, called the *power method*, is simple and is in fact the conceptual starting point for many algorithms. Let A be a matrix whose SVD is $\sum_i \sigma_i \mathbf{u_i}\mathbf{v_i}^T$. We wish to work with a matrix that is square and symmetric. Let $B = A^T A$. By direct multiplication, using the orthogonality of the $\mathbf{u_i}$'s that was proved in Theorem 3.7,

$$\begin{aligned} B = A^T A &= \left(\sum_i \sigma_i \mathbf{v_i}\mathbf{u_i}^T\right)\left(\sum_j \sigma_j \mathbf{u_j}\mathbf{v_j}^T\right) \\ &= \sum_{i,j} \sigma_i\sigma_j \mathbf{v_i}(\mathbf{u}_i^T \cdot \mathbf{u_j})\mathbf{v_j}^T = \sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T. \end{aligned}$$

The matrix B is square and symmetric, and has the same left and right-singular vectors. In particular, $B\mathbf{v_j} = (\sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T)\mathbf{v_j} = \sigma_j^2 \mathbf{v_j}$, so $\mathbf{v_j}$ is an eigenvector of B with eigenvalue σ_j^2. If A is itself square and symmetric, it will have the same right and left-singular vectors, namely $A = \sum_i \sigma_i \mathbf{v_i}\mathbf{v_i}^T$ and computing B is unnecessary.

Now consider computing B^2.

$$B^2 = \left(\sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T\right)\left(\sum_j \sigma_j^2 \mathbf{v_j}\mathbf{v_j}^T\right) = \sum_{ij} \sigma_i^2\sigma_j^2 \mathbf{v_i}(\mathbf{v_i}^T\mathbf{v_j})\mathbf{v_j}^T$$

When $i \neq j$, the dot product $\mathbf{v_i}^T\mathbf{v_j}$ is zero by orthogonality.[9] Thus, $B^2 = \sum_{i=1}^{r} \sigma_i^4 \mathbf{v_i}\mathbf{v_i}^T$. In computing the k^{th} power of B, all the cross product terms are zero and

$$B^k = \sum_{i=1}^{r} \sigma_i^{2k} \mathbf{v_i}\mathbf{v_i}^T.$$

If $\sigma_1 > \sigma_2$, then the first term in the summation dominates, so $B^k \to \sigma_1^{2k}\mathbf{v_1}\mathbf{v_1}^T$. This means a close estimate to $\mathbf{v_1}$ can be computed by simply taking the first column of B^k and normalizing it to a unit vector.

3.7.1 A Faster Method

A problem with the above method is that A may be a very large, sparse matrix, say a $10^8 \times 10^8$ matrix with 10^9 nonzero entries. Sparse matrices are often represented by just

[9] The "outer product" $\mathbf{v_i}\mathbf{v_j}^T$ is a matrix and is not zero even for $i \neq j$.

a list of nonzero entries, say a list of triples of the form (i, j, a_{ij}). Though A is sparse, B need not be and in the worse case may have all 10^{16} entries nonzero[10] and it is then impossible to even write down B, let alone compute the product B^2. Even if A is moderate in size, computing matrix products is costly in time. Thus, a more efficient method is needed.

Instead of computing B^k, select a random vector $\mathbf{x}$ and compute the product $B^k\mathbf{x}$. The vector $\mathbf{x}$ can be expressed in terms of the singular vectors of B augmented to a full orthonormal basis as $\mathbf{x} = \sum_{i=1}^{d} c_i\mathbf{v_i}$. Then

$$B^k\mathbf{x} \approx (\sigma_1^{2k}\mathbf{v_1}\mathbf{v_1}^T)\left(\sum_{i=1}^{d} c_i\mathbf{v_i}\right) = \sigma_1^{2k}c_1\mathbf{v_1}.$$

Normalizing the resulting vector yields $\mathbf{v_1}$, the first singular vector of A. The way $B^k\mathbf{x}$ is computed is by a series of matrix vector products, instead of matrix products. $B^k\mathbf{x} = A^TA\ldots A^TA\mathbf{x}$, which can be computed right-to-left. This consists of $2k$ vector times sparse matrix multiplications.

To compute k singular vectors, one selects a random vector $\mathbf{r}$ and finds an orthonormal basis for the space spanned by $\mathbf{r}, A\mathbf{r}, \ldots, A^{k-1}\mathbf{r}$. Then compute A times each of the basis vectors, and find an orthonormal basis for the space spanned by the resulting vectors. Intuitively, one has applied A to a subspace rather than a single vector. One repeatedly applies A to the subspace, calculating an orthonormal basis after each application to prevent the subspace collapsing to the one dimensional subspace spanned by the first singular vector. The process quickly converges to the first k singular vectors.

An issue occurs if there is no significant gap between the first and second singular values of a matrix. Take for example the case when there is a tie for the first singular vector and $\sigma_1 = \sigma_2$. Then, the above argument fails. We will overcome this hurdle. Theorem 3.11 below states that even with ties, the power method converges to some vector in the span of those singular vectors corresponding to the "nearly highest" singular values. The theorem assumes it is given a vector $\mathbf{x}$ which has a component of magnitude at least δ along the first right singular vector $\mathbf{v_1}$ of A. We will see in Lemma 3.12 that a random vector satisfies this condition with fairly high probability.

Theorem 3.11 *Let A be an $n \times d$ matrix and $\mathbf{x}$ a unit length vector in $\mathbf{R}^d$ with $|\mathbf{x}^T\mathbf{v_1}| \geq \delta$, where $\delta > 0$. Let V be the space spanned by the right singular vectors of A corresponding to singular values greater than $(1-\varepsilon)\,\sigma_1$. Let $\mathbf{w}$ be the unit vector after $k = \frac{\ln(1/\varepsilon\delta)}{2\varepsilon}$ iterations of the power method, namely,*

$$\mathbf{w} = \frac{\left(A^TA\right)^k\mathbf{x}}{\left|\left(A^TA\right)^k\mathbf{x}\right|}.$$

[10] E.g., suppose each entry in the first row of A is nonzero and the rest of A is zero.

Then $\mathbf{w}$ *has a component of at most* ε *perpendicular to* V*.*

Proof: Let

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

be the SVD of A. If the rank of A is less than d, then for convenience complete $\{\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v}_r\}$ into an orthonormal basis $\{\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v}_d\}$ of d-space. Write $\mathbf{x}$ in the basis of the $\mathbf{v_i}$'s as

$$\mathbf{x} = \sum_{i=1}^{d} c_i \mathbf{v_i}.$$

Since $(A^T A)^k = \sum_{i=1}^{d} \sigma_i^{2k} \mathbf{v_i} \mathbf{v_i}^T$, it follows that $(A^T A)^k \mathbf{x} = \sum_{i=1}^{d} \sigma_i^{2k} c_i \mathbf{v_i}$. By hypothesis, $|c_1| \geq \delta$.

Suppose that $\sigma_1, \sigma_2, \ldots, \sigma_m$ are the singular values of A that are greater than or equal to $(1-\varepsilon)\,\sigma_1$ and that $\sigma_{m+1}, \ldots, \sigma_d$ are the singular values that are less than $(1-\varepsilon)\,\sigma_1$. Now

$$|(A^T A)^k \mathbf{x}|^2 = \left| \sum_{i=1}^{d} \sigma_i^{2k} c_i \mathbf{v_i} \right|^2 = \sum_{i=1}^{d} \sigma_i^{4k} c_i^2 \geq \sigma_1^{4k} c_1^2 \geq \sigma_1^{4k} \delta^2.$$

The component of $|(A^T A)^k \mathbf{x}|^2$ perpendicular to the space V is

$$\sum_{i=m+1}^{d} \sigma_i^{4k} c_i^2 \leq (1-\varepsilon)^{4k}\, \sigma_1^{4k} \sum_{i=m+1}^{d} c_i^2 \leq (1-\varepsilon)^{4k}\, \sigma_1^{4k}$$

since $\sum_{i=1}^{d} c_i^2 = |\mathbf{x}| = 1$. Thus, the component of $\mathbf{w}$ perpendicular to V has squared length at most $\frac{(1-\varepsilon)^{4k} \sigma_1^{4k}}{\sigma_1^{4k} \delta^2}$ and so its length is at most

$$\frac{(1-\varepsilon)^{2k} \sigma_1^{2k}}{\delta \sigma_1^{2k}} = \frac{(1-\varepsilon)^{2k}}{\delta} \leq \frac{e^{-2k\varepsilon}}{\delta} = \varepsilon$$

since $k = \frac{\ln(1/\epsilon\delta}{2\epsilon}$. ∎

Lemma 3.12 *Let* $\mathbf{y} \in \mathbf{R}^n$ *be a random vector with the unit variance spherical Gaussian as its probability density. Normalize* y *to be a unit length vector by setting* $\mathbf{x} = \mathbf{y}/|\mathbf{y}|$*. Let* $\mathbf{v}$ *be any unit length vector. Then*

$$Prob\left(|\mathbf{x}^T \mathbf{v}| \leq \frac{1}{20\sqrt{d}} \right) \leq \frac{1}{10} + 3e^{-d/96}.$$

Proof: Proving for the unit length vector $\mathbf{x}$ that Prob $\left(|\mathbf{x}^T\mathbf{v}| \leq \frac{1}{20\sqrt{d}}\right) \leq \frac{1}{10} + 3e^{-d/96}$ is equivalent to proving for the unnormalized vector $\mathbf{y}$ that Prob$(|\mathbf{y}| \geq 2\sqrt{d}) \leq 3e^{-d/96}$ and Prob$(|\mathbf{y}^T\mathbf{v}| \leq \frac{1}{10}) \leq 1/10$. That Prob$(|\mathbf{y}| \geq 2\sqrt{d})$ is at most $3e^{-d/96}$ follows from Theorem (2.9) with $\sqrt{d}$ substituted for β. The probability that $|\mathbf{y}^T\mathbf{v}| \leq \frac{1}{10}$ is at most 1/10 follows from the fact that $\mathbf{y}^T\mathbf{v}$ is a random, zero mean, unit variance Gaussian with density is at most $1/\sqrt{2\pi} \leq 1/2$ in the interval $[-1/10, 1/10]$, so the integral of the Gaussian over the interval is at most 1/10. ∎

3.8 Singular Vectors and Eigenvectors

For a square matrix B, if $B\mathbf{x} = \lambda\mathbf{x}$, then $\mathbf{x}$ is an *eigenvector* of B and λ is the corresponding *eigenvalue*. We saw in Section 3.7, if $B = A^TA$, then the right singular vectors $\mathbf{v_j}$ of A are eigenvectors of B with eigenvalues σ_j^2. The same argument shows that the left singular vectors $\mathbf{u_j}$ of A are eigenvectors of AA^T with eigenvalues σ_j^2.

The matrix $B = A^TA$ has the property that for any vector $\mathbf{x}$, $\mathbf{x}^TB\mathbf{x} \geq 0$. This is because $B = \sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T$ and for any $\mathbf{x}$, $\mathbf{x}^T\mathbf{v_i}\mathbf{v_i}^T\mathbf{x} = (\mathbf{x}^T\mathbf{v_i})^2 \geq 0$. A matrix B with the property that $\mathbf{x}^TB\mathbf{x} \geq 0$ for all $\mathbf{x}$ is called *positive semi-definite*. Every matrix of the form A^TA is positive semi-definite. In the other direction, any positive semi-definite matrix B can be decomposed into a product A^TA, and so its eigenvalue decomposition can be obtained from the singular value decomposition of A. The interested reader should consult a linear algebra book.

3.9 Applications of Singular Value Decomposition

3.9.1 Centering Data

Singular value decomposition is used in many applications and for some of these applications it is essential to first center the data by subtracting the centroid of the data from each data point.[11] If you are interested in the statistics of the data and how it varies in relationship to its mean, then you would center the data. On the other hand, if you are interested in finding the best low rank approximation to a matrix, then you do not center the data. The issue is whether you are finding the best fitting subspace or the best fitting affine space. In the latter case you first center the data and then find the best fitting subspace. See Figure 3.3.

We first show that the line minimizing the sum of squared distances to a set of points, if not restricted to go through the origin, must pass through the centroid of the points. This implies that if the centroid is subtracted from each data point, such a line will pass through the origin. The best fit line can be generalized to k dimensional "planes". The operation of subtracting the centroid from all data points is useful in other contexts as well. We give it the name "centering data".

[11] The centroid of a set of points is the coordinate-wise average of the points.

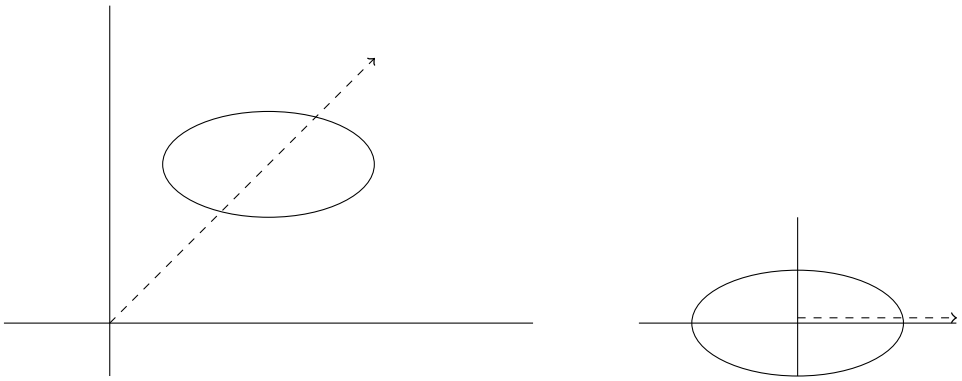


Figure 3.3: If one wants statistical information relative to the mean of the data, one needs to center the data. If one wants the best low rank approximation, one would not center the data.

Lemma 3.13 *The best-fit line (minimizing the sum of perpendicular distances squared) of a set of data points must pass through the centroid of the points.*

Proof: Subtract the centroid from each data point so that the centroid is $\mathbf{0}$. After centering the data let ℓ be the best-fit line and assume for contradiction that ℓ does not pass through the origin. The line ℓ can be written as $\{\mathbf{a} + \lambda\mathbf{v} | \lambda \in \mathbf{R}\}$, where $\mathbf{a}$ is the closest point to $\mathbf{0}$ on ℓ and $\mathbf{v}$ is a unit length vector in the direction of ℓ, which is perpendicular to $\mathbf{a}$. For a data point $\mathbf{a_i}$, let $dist(\mathbf{a_i}, \ell)$ denote its perpendicular distance to ℓ. By the Pythagorean theorem, we have $|\mathbf{a_i} - \mathbf{a}|^2 = dist(\mathbf{a_i}, \ell)^2 + (\mathbf{v} \cdot \mathbf{a_i})^2$, or equivalently, $dist(\mathbf{a_i}, \ell)^2 = |\mathbf{a_i} - \mathbf{a}|^2 - (\mathbf{v} \cdot \mathbf{a_i})^2$. Summing over all data points:

$$\sum_{i=1}^{n} dist(\mathbf{a_i}, \ell)^2 = \sum_{i=1}^{n} \left(|\mathbf{a_i} - \mathbf{a}|^2 - (\mathbf{v} \cdot \mathbf{a_i})^2 \right) = \sum_{i=1}^{n} \left(|\mathbf{a_i}|^2 + |\mathbf{a}|^2 - 2\mathbf{a_i} \cdot \mathbf{a} - (\mathbf{v} \cdot \mathbf{a_i})^2 \right)$$

$$= \sum_{i=1}^{n} |\mathbf{a_i}|^2 + n|\mathbf{a}|^2 - 2\mathbf{a} \cdot \left(\sum_i \mathbf{a_i} \right) - \sum_{i=1}^{n} (\mathbf{v} \cdot \mathbf{a_i})^2 = \sum_i |\mathbf{a_i}|^2 + n|\mathbf{a}|^2 - \sum_i (\mathbf{v} \cdot \mathbf{a_i})^2,$$

where we used the fact that since the centroid is $\mathbf{0}$, $\sum_i \mathbf{a_i} = 0$. The above expression is minimized when $\mathbf{a} = \mathbf{0}$, so the line $\ell' = \{\lambda\mathbf{v} : \lambda \in \mathbf{R}\}$ through the origin is a better fit than ℓ, contradicting ℓ being the best-fit line. ∎

A statement analogous to Lemma 3.13 holds for higher dimensional objects. Define an *affine space* as a subspace translated by a vector. So an affine space is a set of the form

$$\{\mathbf{v_0} + \sum_{i=1}^{k} c_i \mathbf{v_i} | c_1, c_2, \ldots, c_k \in \mathbf{R}\}.$$

Here, $\mathbf{v_0}$ is the translation and $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ form an orthonormal basis for the subspace.

Lemma 3.14 *The k dimensional affine space which minimizes the sum of squared perpendicular distances to the data points must pass through the centroid of the points.*

Proof: We only give a brief idea of the proof, which is similar to the previous lemma. Instead of $(\mathbf{v} \cdot \mathbf{a_i})^2$, we will now have $\sum_{j=1}^{k} (\mathbf{v_j} \cdot \mathbf{a_i})^2$, where the $\mathbf{v_j}, j = 1, 2, \ldots, k$ are an orthonormal basis of the subspace through the origin parallel to the affine space. ∎

3.9.2 Principal Component Analysis

The traditional use of SVD is in Principal Component Analysis (PCA). PCA is illustrated by a movie recommendation setting where there are n customers and d movies. Let matrix A with elements a_{ij} represent the amount that customer i likes movie j. One hypothesizes that there are only k underlying basic factors that determine how much a given customer will like a given movie, where k is much smaller than n or d. For example, these could be the amount of comedy, drama, and action, the novelty of the story, etc. Each movie can be described as a k-dimensional vector indicating how much of these basic factors the movie has, and each customer can be described as a k-dimensional vector indicating how important each of these basic factors is to that customer. The dot-product of these two vectors is hypothesized to determine how much that customer will like that movie. In particular, this means that the $n \times d$ matrix A can be expressed as the product of an $n \times k$ matrix U describing the customers and a $k \times d$ matrix V describing the movies. Finding the best rank k approximation A_k by SVD gives such a U and V. One twist is that A may not be exactly equal to UV, in which case $A - UV$ is treated as noise. Another issue is that SVD gives a factorization with negative entries. Nonnegative matrix factorization (NMF) is more appropriate in some contexts where we want to keep entries nonnegative. NMF is discussed in Chapter 9

In the above setting, A was available fully and we wished to find U and V to identify the basic factors. However, in a case such as movie recommendations, each customer may have seen only a small fraction of the movies, so it may be more natural to assume that we are given just a few elements of A and wish to estimate A. If A was an arbitrary matrix of size $n \times d$, this would require $\Omega(nd)$ pieces of information and cannot be done with a few entries. But again hypothesize that A was a small rank matrix with added noise. If now we also assume that the given entries are randomly drawn according to some known distribution, then there is a possibility that SVD can be used to estimate the whole of A. This area is called collaborative filtering and one of its uses is to recommend movies or to target an ad to a customer based on one or two purchases. We do not describe it here.

3.9.3 Clustering a Mixture of Spherical Gaussians

Clustering is the task of partitioning a set of points into k subsets or clusters where each cluster consists of nearby points. Different definitions of the quality of a clustering lead to different solutions. Clustering is an important area which we will study in detail in Chapter 7. Here we will see how to solve a particular clustering problem using singular value decomposition.

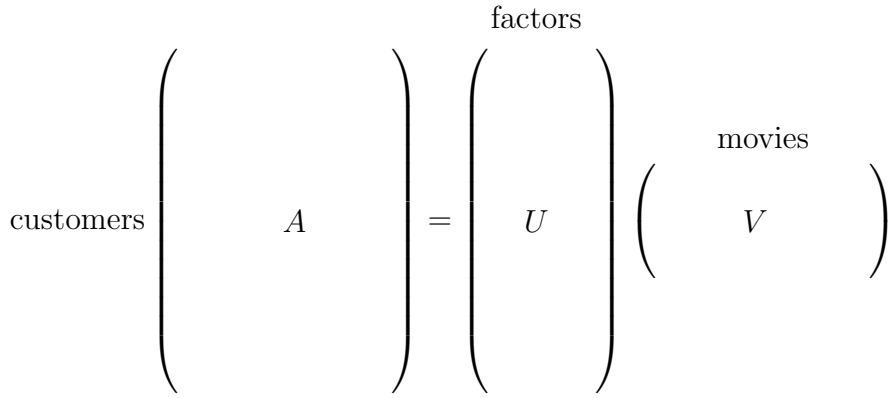


Figure 3.4: Customer-movie data

Mathematical formulations of clustering tend to have the property that finding the highest quality solution to a given set of data is NP-hard. One way around this is to assume stochastic models of input data and devise algorithms to cluster data generated by such models. Mixture models are a very important class of stochastic models. A mixture is a probability density or distribution that is the weighted sum of simple component probability densities. It is of the form

$$f = w_1 p_1 + w_2 p_2 + \cdots + w_k p_k,$$

where $p_1, p_2, \ldots, p_k$ are the basic probability densities and $w_1, w_2, \ldots, w_k$ are positive real numbers called mixture weights that add up to one. Clearly, f is a probability density and integrates to one.

The *model fitting problem* is to fit a mixture of k basic densities to n independent, identically distributed samples, each sample drawn according to the same mixture distribution f. The class of basic densities is known, but various parameters such as their means and the component weights of the mixture are not. Here, we deal with the case where the basic densities are all spherical Gaussians. There are two equivalent ways of thinking of the hidden sample generation process when only the samples are given:

1. Pick each sample according to the density f on $\mathbf{R}^d$.

2. Pick a random i from $\{1, 2, \ldots, k\}$ where probability of picking i is w_i. Then, pick a sample according to the density p_i.

One approach to the model-fitting problem is to break it into two subproblems:

1. First, cluster the set of samples into k clusters $C_1, C_2, \ldots, C_k$, where C_i is the set of samples generated according to p_i (see (2) above) by the hidden generation process.

2. Then fit a single Gaussian distribution to each cluster of sample points.

The second problem is relatively easier and indeed we saw the solution in Chapter 2, where we showed that taking the empirical mean (the mean of the sample) and the empirical standard deviation gives us the best-fit Gaussian. The first problem is harder and this is what we discuss here.

If the component Gaussians in the mixture have their centers very close together, then the clustering problem is unresolvable. In the limiting case where a pair of component densities are the same, there is no way to distinguish between them. What condition on the inter-center separation will guarantee unambiguous clustering? First, by looking at 1-dimensional examples, it is clear that this separation should be measured in units of the standard deviation, since the density is a function of the number of standard deviation from the mean. In one dimension, if two Gaussians have inter-center separation at least six times the maximum of their standard deviations, then they hardly overlap. This is summarized in the question: How many standard deviations apart are the means? In one dimension, if the answer is at least six, we can easily tell the Gaussians apart. What is the analog of this in higher dimensions?

We discussed in Chapter 2 distances between two sample points from the same Gaussian as well the distance between two sample points from two different Gaussians. Recall from that discussion that if

- If $\mathbf{x}$ and $\mathbf{y}$ are two independent samples from the same spherical Gaussian with standard deviation[12] σ then

$$|\mathbf{x} - \mathbf{y}|^2 \approx 2\big(\sqrt{d} \pm O(1)\big)^2\sigma^2.$$

- If $\mathbf{x}$ and $\mathbf{y}$ are samples from different spherical Gaussians each of standard deviation σ and means separated by distance Δ, then

$$|\mathbf{x} - \mathbf{y}|^2 \approx 2\big(\sqrt{d} \pm O(1)\big)^2\sigma^2 + \Delta^2.$$

To ensure that points from the same Gaussian are closer to each other than points from different Gaussians, we need

$$2\big(\sqrt{d} - O(1)\big)^2\sigma^2 + \Delta^2 > 2\big(\sqrt{d} + O(1)\big)^2\sigma^2.$$

Expanding the squares, the high order term $2d$ cancels and we need that

$$\Delta > cd^{1/4},$$

for some constant c. While this was not a completely rigorous argument, it can be used to show that a distance based clustering approach (see Chapter 2 for an example) requires an

[12]Since a spherical Gaussian has the same standard deviation in every direction, we call it the standard deviation of the Gaussian.

inter-mean separation of at least $cd^{1/4}$ standard deviations to succeed, thus unfortunately not keeping with mnemonic of a constant number of standard deviations separation of the means. Here, indeed, we will show that $\Omega(1)$ standard deviations suffice provided the number k of Gaussians is $O(1)$.

The central idea is the following. Suppose we can find the subspace spanned by the k centers and project the sample points to this subspace. The projection of a spherical Gaussian with standard deviation σ remains a spherical Gaussian with standard deviation σ (Lemma 3.15). In the projection, the inter-center separation remains the same. So in the projection, the Gaussians are distinct provided the inter-center separation in the whole space is at least $ck^{1/4}\,\sigma$ which is less than $cd^{1/4}\,\sigma$ for $k \ll d$. Interestingly, we will see that the subspace spanned by the k-centers is essentially the best-fit k-dimensional subspace that can be found by singular value decomposition.

Lemma 3.15 *Suppose p is a d-dimensional spherical Gaussian with center $\boldsymbol{\mu}$ and standard deviation σ. The density of p projected onto a k-dimensional subspace V is a spherical Gaussian with the same standard deviation.*

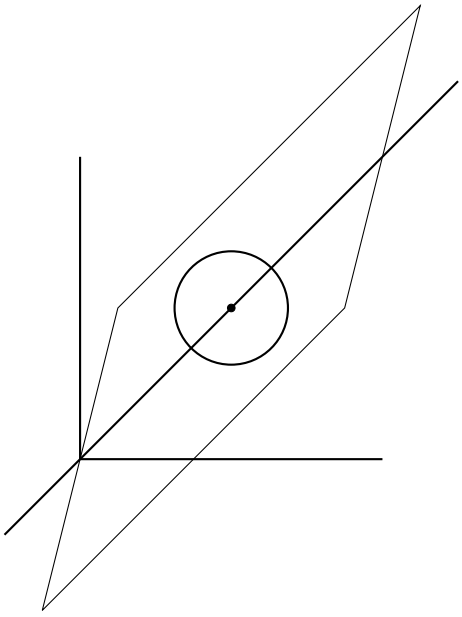
Proof: Rotate the coordinate system so V is spanned by the first k coordinate vectors. The Gaussian remains spherical with standard deviation σ although the coordinates of its center have changed. For a point $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, we will use the notation $\mathbf{x}' = (x_1, x_2, \ldots x_k)$ and $\mathbf{x}'' = (x_{k+1}, x_{k+2}, \ldots, x_n)$. The density of the projected Gaussian at the point $(x_1, x_2, \ldots, x_k)$ is

$$ce^{-\frac{|\mathbf{x}'-\boldsymbol{\mu}'|^2}{2\sigma^2}} \int_{\mathbf{x}''} e^{-\frac{|\mathbf{x}''-\boldsymbol{\mu}''|^2}{2\sigma^2}} d\mathbf{x}'' = c'e^{-\frac{|\mathbf{x}'-\boldsymbol{\mu}'|^2}{2\sigma^2}}.$$

This implies the lemma. ∎

We now show that the top k singular vectors produced by the SVD span the space of the k centers. First, we extend the notion of best fit to probability distributions. Then we show that for a single spherical Gaussian whose center is not the origin, the best fit 1-dimensional subspace is the line though the center of the Gaussian and the origin. Next, we show that the best fit k-dimensional subspace for a single Gaussian whose center is not the origin is any k-dimensional subspace containing the line through the Gaussian's center and the origin. Finally, for k spherical Gaussians, the best fit k-dimensional subspace is the subspace containing their centers. Thus, the SVD finds the subspace that contains the centers.

Recall that for a set of points, the best-fit line is the line passing through the origin that maximizes the sum of squared lengths of the projections of the points onto the line. We extend this definition to probability densities instead of a set of points.



1. The best fit 1-dimension subspace to a spherical Gaussian is the line through its center and the origin.
2. Any k-dimensional subspace containing the line is a best fit k-dimensional subspace for the Gaussian.
3. The best fit k-dimensional subspace for k spherical Gaussians is the subspace containing their centers.

Figure 3.5: Best fit subspace to a spherical Gaussian.

Definition 3.1 *If p is a probability density in d space, the best fit line for p is the line in the $\mathbf{v_1}$ direction where*

$$\mathbf{v_1} = \arg\max_{|\mathbf{v}|=1} \mathop{E}_{\mathbf{x}\sim p} \left[(\mathbf{v}^T\mathbf{x})^2\right].$$

■

For a spherical Gaussian centered at the origin, it is easy to see that any line passing through the origin is a best fit line. Our next lemma shows that the best fit line for a spherical Gaussian centered at $\boldsymbol{\mu} \neq 0$ is the line passing through $\boldsymbol{\mu}$ and the origin.

Lemma 3.16 *Let the probability density p be a spherical Gaussian with center $\boldsymbol{\mu} \neq 0$. The unique best fit 1-dimensional subspace is the line passing through $\boldsymbol{\mu}$ and the origin. If $\boldsymbol{\mu} = 0$, then any line through the origin is a best-fit line.*

Proof: For a randomly chosen $\boldsymbol{x}$ (according to p) and a fixed unit length vector $\mathbf{v}$,

$$\begin{aligned}
\mathop{E}_{\mathbf{x}\sim p}\left[(\mathbf{v}^T\mathbf{x})^2\right] &= \mathop{E}_{\mathbf{x}\sim p}\left[\left(\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu}) + \mathbf{v}^T\boldsymbol{\mu}\right)^2\right] \\
&= \mathop{E}_{\mathbf{x}\sim p}\left[\left(\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu})\right)^2 + 2\left(\mathbf{v}^T\boldsymbol{\mu}\right)\left(\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu})\right) + \left(\mathbf{v}^T\boldsymbol{\mu}\right)^2\right] \\
&= \mathop{E}_{\mathbf{x}\sim p}\left[\left(\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu})\right)^2\right] + 2\left(\mathbf{v}^T\boldsymbol{\mu}\right)E\left[\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu})\right] + \left(\mathbf{v}^T\boldsymbol{\mu}\right)^2 \\
&= \mathop{E}_{\mathbf{x}\sim p}\left[\left(\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu})\right)^2\right] + \left(\mathbf{v}^T\boldsymbol{\mu}\right)^2 \\
&= \sigma^2 + \left(\mathbf{v}^T\boldsymbol{\mu}\right)^2
\end{aligned}$$

where the fourth line follows from the fact that $E[\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu})] = 0$, and the fifth line follows from the fact that $E[(\mathbf{v}^T(\mathbf{x}-\boldsymbol{\mu}))^2]$ is the variance in the direction $\mathbf{v}$. The best fit line $\mathbf{v}$ maximizes $E_{\mathbf{x}\sim p}[(\mathbf{v}^T\mathbf{x})^2]$ and therefore maximizes $\left(\mathbf{v}^T\boldsymbol{\mu}\right)^2$. This is maximized when $\mathbf{v}$ is aligned with the center $\boldsymbol{\mu}$. To see uniqueness, just note that if $\boldsymbol{\mu} \neq 0$, then $\mathbf{v}^T\boldsymbol{\mu}$ is strictly less when $\mathbf{v}$ is not aligned with the center. ∎

We now extend Definition 3.1 to k-dimensional subspaces.

Definition 3.2 *If p is a probability density in d-space then the best-fit k-dimensional subspace V_k is*

$$V_k = \underset{\substack{V \\ dim(V)=k}}{\operatorname{argmax}} \underset{\mathbf{x}\sim p}{E}\left(|\operatorname{proj}(\mathbf{x}, V)|^2\right),$$

where proj$(\mathbf{x}, V)$ is the orthogonal projection of $\mathbf{x}$ onto V. ∎

Lemma 3.17 *For a spherical Gaussian with center $\boldsymbol{\mu}$, a k-dimensional subspace is a best fit subspace if and only if it contains $\boldsymbol{\mu}$.*

Proof: If $\boldsymbol{\mu} = \mathbf{0}$, then by symmetry any k-dimensional subspace is a best-fit subspace. If $\boldsymbol{\mu} \neq \mathbf{0}$, then, the best-fit line must pass through $\boldsymbol{\mu}$ by Lemma 3.16. Now, as in the greedy algorithm for finding subsequent singular vectors, we would project perpendicular to the first singular vector. But after the projection, the mean of the Gaussian becomes $\mathbf{0}$ and any vectors will do as subsequent best-fit directions. ∎

This leads to the following theorem.

Theorem 3.18 *If p is a mixture of k spherical Gaussians, then the best fit k-dimensional subspace contains the centers. In particular, if the means of the Gaussians are linearly independent, the space spanned by them is the unique best-fit k dimensional subspace.*

Proof: Let p be the mixture $w_1p_1 + w_2p_2 + \cdots + w_kp_k$. Let V be any subspace of dimension k or less. Then,

$$\underset{\mathbf{x}\sim p}{E}\left(|\operatorname{proj}(\mathbf{x}, V)|^2\right) = \sum_{i=1}^{k} w_i \underset{\mathbf{x}\sim p_i}{E}\left(|\operatorname{proj}(\mathbf{x}, V)|^2\right)$$

If V contains the centers of the densities p_i, by Lemma 3.17, each term in the summation is individually maximized, which implies the entire summation is maximized, proving the theorem. ∎

For an infinite set of points drawn according to the mixture, the k-dimensional SVD subspace gives exactly the space of the centers. In reality, we have only a large number of samples drawn according to the mixture. However, it is intuitively clear that as the number of samples increases, the set of sample points will approximate the probability density and so the SVD subspace of the sample will be close to the space spanned by the centers. The details of how close it gets as a function of the number of samples are technical and we do not carry this out here.

3.9.4 Ranking Documents and Web Pages

An important task for a document collection is to rank the documents according to their intrinsic relevance to the collection. A good candidate definition of "intrinsic relevance" is a document's projection onto the best-fit direction for that collection, namely the top left-singular vector of the term-document matrix. An intuitive reason for this is that this direction has the maximum sum of squared projections of the collection and so can be thought of as a synthetic term-document vector best representing the document collection.

Ranking in order of the projection of each document's term vector along the best fit direction has a nice interpretation in terms of the power method. For this, we consider a different example, that of the web with hypertext links. The World Wide Web can be represented by a directed graph whose nodes correspond to web pages and directed edges to hypertext links between pages. Some web pages, called *authorities*, are the most prominent sources for information on a given topic. Other pages called *hubs*, are ones that identify the authorities on a topic. Authority pages are pointed to by many hub pages and hub pages point to many authorities. One is led to what seems like a circular definition: a hub is a page that points to many authorities and an authority is a page that is pointed to by many hubs.

One would like to assign hub weights and authority weights to each node of the web. If there are n nodes, the hub weights form an n-dimensional vector $\mathbf{u}$ and the authority weights form an n-dimensional vector $\mathbf{v}$. Suppose A is the adjacency matrix representing the directed graph. Here a_{ij} is 1 if there is a hypertext link from page i to page j and 0 otherwise. Given hub vector $\mathbf{u}$, the authority vector $\mathbf{v}$ could be computed by the formula

$$v_j \propto \sum_{i=1}^{d} u_i a_{ij}$$

since the right hand side is the sum of the hub weights of all the nodes that point to node j. In matrix terms,

$$\mathbf{v} = A^T\mathbf{u}/|A^T\mathbf{u}|.$$

Similarly, given an authority vector $\mathbf{v}$, the hub vector $\mathbf{u}$ could be computed by $\mathbf{u} = A\mathbf{v}/|A\mathbf{v}|$. Of course, at the start, we have neither vector. But the above discussion suggests a power iteration. Start with any $\mathbf{v}$. Set $\mathbf{u} = A\mathbf{v}$, then set $\mathbf{v} = A^T\mathbf{u}$, then renormalize and repeat the process. We know from the power method that this converges to the left and right-singular vectors. So after sufficiently many iterations, we may use the left vector $\mathbf{u}$ as the hub weights vector and project each column of A onto this direction and rank columns (authorities) in order of this projection. But the projections just form the vector $A^T\mathbf{u}$ which equals a multiple of $\mathbf{v}$. So we can just rank by order of the v_j. This is the basis of an algorithm called the HITS algorithm, which was one of the early proposals for ranking web pages.

A different ranking called *pagerank* is widely used. It is based on a random walk on the graph described above. We will study random walks in detail in Chapter 4.

3.9.5 An Application of SVD to a Discrete Optimization Problem

In clustering a mixture of Gaussians, SVD was used as a dimension reduction technique. It found a k-dimensional subspace (the space of centers) of a d-dimensional space and made the Gaussian clustering problem easier by projecting the data to the subspace. Here, instead of fitting a model to data, we consider an optimization problem where applying dimension reduction makes the problem easier. The use of SVD to solve discrete optimization problems is a relatively new subject with many applications. We start with an important NP-hard problem, the maximum cut problem for a directed graph $G(V, E)$.

The maximum cut problem is to partition the nodes of an n-node directed graph into two subsets S and $\bar{S}$ so that the number of edges from S to $\bar{S}$ is maximized. Let A be the adjacency matrix of the graph. With each vertex i, associate an indicator variable x_i. The variable x_i will be set to 1 for $i \in S$ and 0 for $i \in \bar{S}$. The vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is unknown and we are trying to find it or equivalently the cut, so as to maximize the number of edges across the cut. The number of edges across the cut is precisely

$$\sum_{i,j} x_i(1 - x_j)a_{ij}.$$

Thus, the maximum cut problem can be posed as the optimization problem

$$\text{Maximize} \sum_{i,j} x_i(1 - x_j)a_{ij} \quad \text{subject to } x_i \in \{0, 1\}.$$

In matrix notation,

$$\sum_{i,j} x_i(1 - x_j)a_{ij} = \mathbf{x}^T A(\mathbf{1} - \mathbf{x}),$$

where $\mathbf{1}$ denotes the vector of all 1's . So, the problem can be restated as

$$\text{Maximize } \mathbf{x}^T A(\mathbf{1} - \mathbf{x}) \quad \text{subject to } x_i \in \{0, 1\}. \tag{3.1}$$

This problem is NP-hard. However we will see that for dense graphs, that is, graphs with $\Omega(n^2)$ edges and therefore whose optimal solution has size $\Omega(n^2)$,[13] we can use the SVD to find a near optimal solution in polynomial time. To do so we will begin by computing the SVD of A and replacing A by $A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ in (3.1) to get

$$\text{Maximize } \mathbf{x}^T A_k(\mathbf{1} - \mathbf{x}) \quad \text{subject to } x_i \in \{0, 1\}. \tag{3.2}$$

Note that the matrix A_k is no longer a 0-1 adjacency matrix.

We will show that:

[13] Any graph of m edges has a cut of size at least $m/2$. This can be seen by noting that the expected size of the cut for a *random* $\mathbf{x} \in \{0, 1\}^n$ is exactly $m/2$.

1. For each 0-1 vector $\mathbf{x}$, $\mathbf{x}^T A_k(\mathbf{1}-\mathbf{x})$ and $\mathbf{x}^T A(\mathbf{1}-\mathbf{x})$ differ by at most $\frac{n^2}{\sqrt{k+1}}$. Thus, the maxima in (3.1) and (3.2) differ by at most this amount.

2. A near optimal $\mathbf{x}$ for (3.2) can be found in time $n^{O(k)}$ by exploiting the low rank of A_k, which is polynomial time for constant k. By Item 1 this is near optimal for (3.1) where near optimal means with additive error of at most $\frac{n^2}{\sqrt{k+1}}$.

First, we prove Item 1. Since $\mathbf{x}$ and $\mathbf{1}-\mathbf{x}$ are 0-1 n-vectors, each has length at most $\sqrt{n}$. By the definition of the 2-norm, $|(A-A_k)(\mathbf{1}-\mathbf{x})| \leq \sqrt{n}||A-A_k||_2$. Now since $\mathbf{x}^T(A-A_k)(\mathbf{1}-\mathbf{x})$ is the dot product of the vector $\mathbf{x}$ with the vector $(A-A_k)(\mathbf{1}-\mathbf{x})$,

$$|\mathbf{x}^T(A-A_k)(\mathbf{1}-\mathbf{x})| \leq n||A-A_k||_2.$$

By Lemma 3.8, $||A-A_k||_2 = \sigma_{k+1}(A)$. The inequalities,

$$(k+1)\sigma_{k+1}^2 \leq \sigma_1^2+\sigma_2^2+\cdots\sigma_{k+1}^2 \leq ||A||_F^2 = \sum_{i,j} a_{ij}^2 \leq n^2$$

imply that $\sigma_{k+1}^2 \leq \frac{n^2}{k+1}$ and hence $||A-A_k||_2 \leq \frac{n}{\sqrt{k+1}}$ proving Item 1.

Next we focus on Item 2. It is instructive to look at the special case when k=1 and A is approximated by the rank one matrix A_1. An even more special case when the left and right-singular vectors $\mathbf{u}$ and $\mathbf{v}$ are identical is already NP-hard to solve exactly because it subsumes the problem of whether for a set of n integers, $\{a_1, a_2, \ldots, a_n\}$, there is a partition into two subsets whose sums are equal. However, for that problem, there is an efficient dynamic programming algorithm that finds a near-optimal solution. We will build on that idea for the general rank k problem.

For Item 2, we want to maximize $\sum_{i=1}^k \sigma_i(\mathbf{x}^T\mathbf{u_i})(\mathbf{v_i}^T(\mathbf{1}-\mathbf{x}))$ over 0-1 vectors $\mathbf{x}$. A piece of notation will be useful. For any $S \subseteq \{1,2,\ldots n\}$, write $\mathbf{u_i}(S)$ for the sum of coordinates of the vector $\mathbf{u_i}$ corresponding to elements in the set S, that is, $\mathbf{u}_i(S) = \sum_{j\in S} u_{ij}$, and similarly for $\mathbf{v_i}$. We will find S to maximize $\sum_{i=1}^k \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(\bar{S})$ using dynamic programming.

For a subset S of $\{1,2,\ldots,n\}$, define the $2k$-dimensional vector

$$\mathbf{w}(S) = \big(\mathbf{u_1}(S), \mathbf{v_1}(\bar{S}), \mathbf{u_2}(S), \mathbf{v_2}(\bar{S}), \ldots, \mathbf{u_k}(S), \mathbf{v_k}(\bar{S})\big).$$

If we had the list of all such vectors, we could find $\sum_{i=1}^k \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(\bar{S})$ for each of them and take the maximum. There are 2^n subsets S, but several S could have the same $\mathbf{w}(S)$ and in that case it suffices to list just one of them. Round each coordinate of each $\mathbf{u_i}$ to the nearest integer multiple of $\frac{1}{nk^2}$. Call the rounded vector $\tilde{\mathbf{u}}_\mathbf{i}$. Similarly obtain $\tilde{\mathbf{v}}_\mathbf{i}$. Let $\tilde{\mathbf{w}}(S)$ denote the vector $(\tilde{\mathbf{u}}_\mathbf{1}(S), \tilde{\mathbf{v}}_\mathbf{1}(\bar{S}), \tilde{\mathbf{u}}_\mathbf{2}(S), \tilde{\mathbf{v}}_\mathbf{2}(\bar{S}), \ldots, \tilde{\mathbf{u}}_\mathbf{k}(S), \tilde{\mathbf{v}}_\mathbf{k}(\bar{S}))$. We will construct a list of all possible values of the vector $\tilde{\mathbf{w}}(S)$. Again, if several different S's lead to the

same vector $\tilde{\mathbf{w}}(S)$, we will keep only one copy on the list. The list will be constructed by dynamic programming. For the recursive step, assume we already have a list of all such vectors for $S \subseteq \{1, 2, \ldots, i\}$ and wish to construct the list for $S \subseteq \{1, 2, \ldots, i+1\}$. Each $S \subseteq \{1, 2, \ldots, i\}$ leads to two possible $S' \subseteq \{1, 2, \ldots, i+1\}$, namely, S and $S \cup \{i+1\}$. In the first case, the vector $\tilde{\mathbf{w}}(S') = (\tilde{\mathbf{u}}_\mathbf{1}(S), \tilde{\mathbf{v}}_\mathbf{1}(\bar{S}) + \tilde{v}_{1,i+1}, \tilde{\mathbf{u}}_\mathbf{2}(S), \tilde{\mathbf{v}}_\mathbf{2}(\bar{S}) + \tilde{v}_{2,i+1}, \ldots, ...)$. In the second case, it is $\tilde{\mathbf{w}}(S') = (\tilde{\mathbf{u}}_\mathbf{1}(S) + \tilde{u}_{1,i+1}, \tilde{\mathbf{v}}_\mathbf{1}(\bar{S}), \tilde{\mathbf{u}}_\mathbf{2}(S) + \tilde{u}_{2,i+1}, \tilde{\mathbf{v}}_\mathbf{2}(\bar{S}), \ldots, ...)$. We put in these two vectors for each vector in the previous list. Then, crucially, we prune - i.e., eliminate duplicates.

Assume that k is constant. Now, we show that the error is at most $\frac{n^2}{\sqrt{k+1}}$ as claimed. Since $\mathbf{u_i}$ and $\mathbf{v_i}$ are unit length vectors, $|\mathbf{u_i}(S)|, |\mathbf{v_i}(\bar{S})| \leq \sqrt{n}$. Also $|\tilde{\mathbf{u}}_\mathbf{i}(S) - \mathbf{u_i}(S)| \leq \frac{n}{nk^2} = \frac{1}{k^2}$ and similarly for $\mathbf{v_i}$. To bound the error, we use an elementary fact: if a and b are reals with $|a|, |b| \leq M$ and we estimate a by a' and b by b' so that $|a-a'|, |b-b'| \leq \delta \leq M$, then $a'b'$ is an estimate of ab in the sense

$$|ab - a'b'| = |a(b-b') + b'(a-a')| \leq |a||b-b'| + (|b| + |b-b'|)|a-a'| \leq 3M\delta.$$

Using this,

$$\left| \sum_{i=1}^{k} \sigma_i \tilde{\mathbf{u}}_\mathbf{i}(S) \tilde{\mathbf{v}}_\mathbf{i}(\bar{S}) \quad - \quad \sum_{i=1}^{k} \sigma_i \mathbf{u_i}(S) \mathbf{v_i}(S) \right| \leq 3k\sigma_1 \sqrt{n}/k^2 \leq 3n^{3/2}/k \leq n^2/k,$$

and this meets the claimed error bound.

Next, we show that the running time is polynomially bounded. First, $|\tilde{\mathbf{u}}_\mathbf{i}(S)|, |\tilde{\mathbf{v}}_\mathbf{i}(S)| \leq 2\sqrt{n}$. Since $\tilde{\mathbf{u}}_\mathbf{i}(S)$ and $\tilde{\mathbf{v}}_\mathbf{i}(S)$ are all integer multiples of $1/(nk^2)$, there are at most $2n^{3/2}k^2$ possible values of $\tilde{\mathbf{u}}_\mathbf{i}(S)$ and $\tilde{\mathbf{v}}_\mathbf{i}(S)$ from which it follows that the list of $\tilde{\mathbf{w}}(S)$ never gets larger than $(2n^{3/2}k^2)^{2k}$ which for fixed k is polynomially bounded.

We summarize what we have accomplished.

Theorem 3.19 *Given a directed graph $G(V, E)$, a cut of size at least the maximum cut minus $O\left(\frac{n^2}{\sqrt{k}}\right)$ can be computed in time polynomial in n for any fixed k.*

Note that achieving the same accuracy in time polynomial in n and k would give an exact max cut in polynomial time.

3.10 Bibliographic Notes

Singular value decomposition is fundamental to numerical analysis and linear algebra. There are many texts on these subjects and the interested reader may want to study these. A good reference is [GvL96]. The material on clustering a mixture of Gaussians in Section 3.9.3 is from [VW02]. Modeling data with a mixture of Gaussians is a standard tool in statistics. Several well-known heuristics like the expectation-minimization

algorithm are used to learn (fit) the mixture model to data. Recently, in theoretical computer science, there has been modest progress on provable polynomial-time algorithms for learning mixtures. Some references are [DS07], [AK05], [AM05], and [MV10]. The application to the discrete optimization problem is from [FK99]. The section on ranking documents/webpages is from two influential papers, one on hubs and authorities by Jon Kleinberg [Kle99] and the other on pagerank by Page, Brin, Motwani and Winograd [BMPW98].

3.11 Exercises

Exercise 3.1 (Least squares vertical error) *In many experiments one collects the value of a parameter at various instances of time. Let y_i be the value of the parameter y at time x_i. Suppose we wish to construct the best linear approximation to the data in the sense that we wish to minimize the mean square error. Here error is measured vertically rather than perpendicular to the line. Develop formulas for m and b to minimize the mean square error of the points $\{(x_i, y_i) \,|1 \leq i \leq n\}$ to the line $y = mx + b$.*

Exercise 3.2 *Given five observed variables, height, weight, age, income, and blood pressure of n people, how would one find the best least squares fit affine subspace of the form*

$$a_1 (height) + a_2 (weight) + a_3 (age) + a_4 (income) + a_5 (blood\ pressure) = a_6$$

Here $a_1, a_2, \ldots, a_6$ are the unknown parameters. If there is a good best fit 4-dimensional affine subspace, then one can think of the points as lying close to a 4-dimensional sheet rather than points lying in 5-dimensions. Why might it be better to use the perpendicular distance to the affine subspace rather than vertical distance where vertical distance is measured along the coordinate axis corresponding to one of the variables?

Exercise 3.3 *Manually find the best fit lines (not subspaces which must contain the origin) through the points in the sets below. Subtract the center of gravity of the points in the set from each of the points in the set and find the best fit line for the resulting points. Does the best fit line for the original data go through the origin?*

1. *(4,4) (6,2)*
2. *(4,2) (4,4) (6,2) (6,4)*
3. *(3,2.5) (3,5) (5,1) (5,3.5)*

Exercise 3.4 *Manually determine the best fit line through the origin for each of the following sets of points. Is the best fit line unique? Justify your answer for each of the subproblems.*

1. $\{(0,1), (1,0)\}$
2. $\{(0,1), (2,0)\}$

Exercise 3.5 *Manually find the left and right-singular vectors, the singular values, and the SVD decomposition of the matrices in Figure 3.6.*

Exercise 3.6 *Consider the matrix*

$$A = \begin{pmatrix} 1 & 2 \\ -1 & 2 \\ 1 & -2 \\ -1 & -2 \end{pmatrix}$$

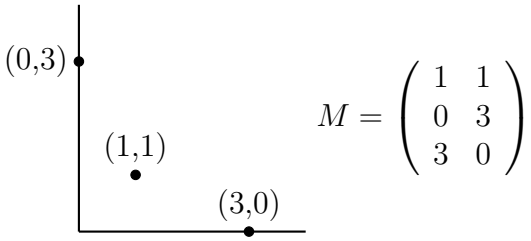


Figure 3.6 a

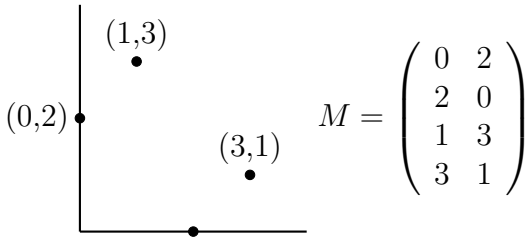


Figure 3.6 b

Figure 3.6: SVD problem

1. *Run the power method starting from $x = \binom{1}{1}$ for $k = 3$ steps. What does this give as an estimate of v_1?*

2. *What actually are the v_i's, σ_i's, and u_i's? It may be easiest to do this by computing the eigenvectors of $B = A^T A$.*

3. *Suppose matrix A is a database of restaurant ratings: each row is a person, each column is a restaurant, and a_{ij} represents how much person i likes restaurant j. What might v_1 represent? What about u_1? How about the gap $\sigma_1 - \sigma_2$?*

Exercise 3.7 *Let A be a square $n \times n$ matrix whose rows are orthonormal. Prove that the columns of A are orthonormal.*

Exercise 3.8 *Suppose A is a $n \times n$ matrix with block diagonal structure with k equal size blocks where all entries of the i^{th} block are a_i with $a_1 > a_2 > \cdots > a_k > 0$. Show that A has exactly k nonzero singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ where $\mathbf{v_i}$ has the value $(\frac{k}{n})^{1/2}$ in the coordinates corresponding to the i^{th} block and 0 elsewhere. In other words, the singular vectors exactly identify the blocks of the diagonal. What happens if $a_1 = a_2 = \cdots = a_k$? In the case where the a_i are equal, what is the structure of the set of all possible singular vectors?*
Hint: *By symmetry, the top singular vector's components must be constant in each block.*

Exercise 3.9 *Interpret the first right and left-singular vectors for the document term matrix.*

Exercise 3.10 *Verify that the sum of r-rank one matrices $\sum_{i=1}^{r} c_i \mathbf{x_i} \mathbf{y_i}^T$ can be written as XCY^T, where the $\mathbf{x_i}$ are the columns of X, the $\mathbf{y_i}$ are the columns of Y, and C is a diagonal matrix with the constants c_i on the diagonal.*

Exercise 3.11 *Let $\sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ be the SVD of A. Show that $\left|\mathbf{u_1}^T A\right| = \sigma_1$ and that $\left|\mathbf{u_1}^T A\right| = \max_{|\mathbf{u}|=1} \left|\mathbf{u^T} A\right|$.*

Exercise 3.12 *If $\sigma_1, \sigma_2, \ldots, \sigma_r$ are the singular values of A and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ are the corresponding right-singular vectors, show that*

1. $A^T A = \sum_{i=1}^{r} \sigma_i^2 \mathbf{v_i} \mathbf{v_i}^T$
2. $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_r$ *are eigenvectors of* $A^T A$.
3. *Assuming that the eigenvectors of $A^T A$ are unique up to multiplicative constants, conclude that the singular vectors of A (which by definition must be unit length) are unique up to sign.*

Exercise 3.13 *Let $\sum_i \sigma_i u_i v_i^T$ be the singular value decomposition of a rank r matrix A. Let $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ be a rank k approximation to A for some $k < r$. Express the following quantities in terms of the singular values $\{\sigma_i, 1 \leq i \leq r\}$.*

1. $||A_k||_F^2$
2. $||A_k||_2^2$
3. $||A - A_k||_F^2$
4. $||A - A_k||_2^2$

Exercise 3.14 *If A is a symmetric matrix with distinct singular values, show that the left and right singular vectors are the same and that $A = VDV^T$.*

Exercise 3.15 *Let A be a matrix. How would you compute*

$$\mathbf{v_1} = \arg\max_{|\mathbf{v}|=1} |A\mathbf{v}|?$$

How would you use or modify your algorithm for finding $\mathbf{v_1}$ to compute the first few singular vectors of A.

Exercise 3.16 *Use the power method to compute the singular value decomposition of the matrix*

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Exercise 3.17 1. *Write a program to implement the power method for computing the first singular vector of a matrix. Apply your program to the matrix*

$$A = \begin{pmatrix} 1 & 2 & 3 & \cdots & 9 & 10 \\ 2 & 3 & 4 & \cdots & 10 & 0 \\ \vdots & \vdots & \vdots & & & \vdots \\ 9 & 10 & 0 & \cdots & 0 & 0 \\ 10 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

2. *Modify the power method to find the first four singular vectors of a matrix A as follows. Randomly select four vectors and find an orthonormal basis for the space spanned by the four vectors. Then multiply each of the basis vectors times A and find a new orthonormal basis for the space spanned by the resulting four vectors. Apply your method to find the first four singular vectors of matrix A from part 1. In Matlab the command orth finds an orthonormal basis for the space spanned by a set of vectors.*

Exercise 3.18 *A matrix A is positive semi-definite if for all $\mathbf{x}$, $\mathbf{x}^T A\mathbf{x} \geq 0$.*

1. *Let A be a real valued matrix. Prove that $B = AA^T$ is positive semi-definite.*

2. *Let A be the adjacency matrix of a graph. The Laplacian of A is $L = D - A$ where D is a diagonal matrix whose diagonal entries are the row sums of A. Prove that L is positive semi definite by showing that $L = B^T B$ where B is an m-by-n matrix with a row for each edge in the graph, a column for each vertex, and we define*

$$b_{ei} = \begin{cases} -1 & \text{if } i \text{ is the endpoint of } e \text{ with lesser index} \\ 1 & \text{if } i \text{ is the endpoint of } e \text{ with greater index} \\ 0 & \text{if } i \text{ is not an endpoint of } e \end{cases}$$

Exercise 3.19 *Prove that the eigenvalues of a symmetric real valued matrix are real.*

Exercise 3.20 *Suppose A is a square invertible matrix and the SVD of A is $A = \sum_i \sigma_i u_i v_i^T$. Prove that the inverse of A is $\sum_i \frac{1}{\sigma_i} v_i u_i^T$.*

Exercise 3.21 *Suppose A is square, but not necessarily invertible and has SVD $A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$. Let $B = \sum_{i=1}^{r} \frac{1}{\sigma_i} v_i u_i^T$. Show that $BA\mathbf{x} = \mathbf{x}$ for all $\mathbf{x}$ in the span of the right-singular vectors of A. For this reason B is sometimes called the pseudo inverse of A and can play the role of A^{-1} in many applications.*

Exercise 3.22

1. *For any matrix A, show that $\sigma_k \leq \frac{||A||_F}{\sqrt{k}}$.*

2. *Prove that there exists a matrix B of rank at most k such that $||A - B||_2 \leq \frac{||A||_F}{\sqrt{k}}$.*

3. *Can the 2-norm on the left hand side in (2) be replaced by Frobenius norm?*

Exercise 3.23 *Suppose an $n \times d$ matrix A is given and you are allowed to preprocess A. Then you are given a number of d-dimensional vectors $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}$ and for each of these vectors you must find the vector $A\mathbf{x_j}$ approximately, in the sense that you must find a vector $\mathbf{y_j}$ satisfying $|\mathbf{y_j} - A\mathbf{x_j}| \leq \varepsilon ||A||_F |\mathbf{x_j}|$. Here $\varepsilon > 0$ is a given error bound. Describe an algorithm that accomplishes this in time $O\left(\frac{d+n}{\varepsilon^2}\right)$ per $\mathbf{x_j}$ not counting the preprocessing time. Hint: use Exercise 3.22.*

Exercise 3.24 *Find the values of c_i to maximize $\sum_{i=1}^{r} c_i^2 \sigma_i^2$ where $\sigma_1 \geq \sigma_2 \geq \ldots$ and $\sum_{i=1}^{r} c_i^2 = 1$.*

Exercise 3.25 (Document-Term Matrices): *Suppose we have an $m \times n$ document-term matrix A where each row corresponds to a document and has been normalized to length one. Define the "similarity" between two such documents by their dot product.*

1. *Consider a "synthetic" document whose sum of squared similarities with all documents in the matrix is as high as possible. What is this synthetic document and how would you find it?*

2. *How does the synthetic document in (1) differ from the center of gravity?*

3. *Building on (1), given a positive integer k, find a set of k synthetic documents such that the sum of squares of the mk similarities between each document in the matrix and each synthetic document is maximized. To avoid the trivial solution of selecting k copies of the document in (1), require the k synthetic documents to be orthogonal to each other. Relate these synthetic documents to singular vectors.*

4. *Suppose that the documents can be partitioned into k subsets (often called clusters), where documents in the same cluster are similar and documents in different clusters are not very similar. Consider the computational problem of isolating the clusters. This is a hard problem in general. But assume that the terms can also be partitioned into k clusters so that for $i \neq j$, no term in the i^{th} cluster occurs in a document in the j^{th} cluster. If we knew the clusters and arranged the rows and columns in them to be contiguous, then the matrix would be a block-diagonal matrix. Of course the clusters are not known. By a "block" of the document-term matrix, we mean a submatrix with rows corresponding to the i^{th} cluster of documents and columns corresponding to the i^{th} cluster of terms . We can also partition any n vector into blocks. Show that any right-singular vector of the matrix must have the property that each of its blocks is a right-singular vector of the corresponding block of the document-term matrix.*

5. *Suppose now that the k singular values are all distinct. Show how to solve the clustering problem.*

Hint: *(4) Use the fact that the right-singular vectors must be eigenvectors of $A^T A$. Show that $A^T A$ is also block-diagonal and use properties of eigenvectors.*

Exercise 3.26 *Let $\mathbf{u}$ be a fixed vector. Show that maximizing $\mathbf{x}^T \mathbf{u}\mathbf{u}^T(\mathbf{1} - \mathbf{x})$ subject to $x_i \in \{0, 1\}$ is equivalent to partitioning the coordinates of $\mathbf{u}$ into two subsets where the sum of the elements in both subsets are as equal as possible.*

Exercise 3.27 *Read in a photo and convert to a matrix. Perform a singular value decomposition of the matrix. Reconstruct the photo using only 5%, 10%, 25%, 50% of the singular values.*

1. *Print the reconstructed photo. How good is the quality of the reconstructed photo?*

2. *What percent of the Forbenius norm is captured in each case?*

Hint: *If you use Matlab, the command to read a photo is imread. The types of files that can be read are given by imformats. To print the file use imwrite. Print using jpeg format. To access the file afterwards you may need to add the file extension .jpg. The command imread will read the file in uint8 and you will need to convert to double for the SVD code. Afterwards you will need to convert back to uint8 to write the file. If the photo is a color photo you will get three matrices for the three colors used.*

Exercise 3.28

1. *Create a* 100×100 *matrix of random numbers between 0 and 1 such that each entry is highly correlated with the adjacent entries. Find the SVD of A. What fraction of the Frobenius norm of A is captured by the top 10 singular vectors? How many singular vectors are required to capture 95% of the Frobenius norm?*

2. *Repeat (1) with a* 100×100 *matrix of statistically independent random numbers between 0 and 1.*

Exercise 3.29 *Show that the running time for the maximum cut algorithm in Section 3.9.5 can be carried out in time* $O(n^3 + poly(n)k^k)$*, where poly is some polynomial.*

Exercise 3.30 *Let* $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ *be n points in d-dimensional space and let X be the* $n \times d$ *matrix whose rows are the n points. Suppose we know only the matrix D of pairwise distances between points and not the coordinates of the points themselves. The set of points* $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ *giving rise to the distance matrix D is not unique since any translation, rotation, or reflection of the coordinate system leaves the distances invariant. Fix the origin of the coordinate system so that the centroid of the set of points is at the origin. That is,* $\sum_{i=1}^{n} \mathbf{x_i} = 0$.

1. *Show that the elements of* XX^T *are given by*

$$\mathbf{x_i x_j^T} = -\frac{1}{2}\left[d_{ij}^2 - \frac{1}{n}\sum_{k=1}^{n} d_{ik}^2 - \frac{1}{n}\sum_{k=1}^{n} d_{kj}^2 + \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} d_{kl}^2\right].$$

2. *Describe an algorithm for determining the matrix X whose rows are the* $\mathbf{x_i}$.

Exercise 3.31

1. *Consider the pairwise distance matrix for twenty US cities given below. Use the algorithm of Exercise 3.30 to place the cities on a map of the US. The algorithm is called classical multidimensional scaling, cmdscale, in Matlab. Alternatively use the pairwise distance matrix of 12 Chinese cities to place the cities on a map of China.*

   Note: Any rotation or a mirror image of the map will have the same pairwise distances.

2. *Suppose you had airline distances for 50 cities around the world. Could you use these distances to construct a 3-dimensional world model?*

■

	BOS	BUF	CHI	DAL	DEN	HOU	LA	MEM	MIA	MIM
Boston	-	400	851	1551	1769	1605	2596	1137	1255	1123
Buffalo	400	-	454	1198	1370	1286	2198	803	1181	731
Chicago	851	454	-	803	920	940	1745	482	1188	355
Dallas	1551	1198	803	-	663	225	1240	420	1111	862
Denver	1769	1370	920	663	-	879	831	879	1726	700
Houston	1605	1286	940	225	879	-	1374	484	968	1056
Los Angeles	2596	2198	1745	1240	831	1374	-	1603	2339	1524
Memphis	1137	803	482	420	879	484	1603	-	872	699
Miami	1255	1181	1188	1111	1726	968	2339	872	-	1511
Minneapolis	1123	731	355	862	700	1056	1524	699	1511	-
New York	188	292	713	1374	1631	1420	2451	957	1092	1018
Omaha	1282	883	432	586	488	794	1315	529	1397	290
Philadelphia	271	279	666	1299	1579	1341	2394	881	1019	985
Phoenix	2300	1906	1453	887	586	1017	357	1263	1982	1280
Pittsburgh	483	178	410	1070	1320	1137	2136	660	1010	743
Saint Louis	1038	662	262	547	796	679	1589	240	1061	466
Salt Lake City	2099	1699	1260	999	371	1200	579	1250	2089	987
San Francisco	2699	2300	1858	1483	949	1645	347	1802	2594	1584
Seattle	2493	2117	1737	1681	1021	1891	959	1867	2734	1395
Washington D.C.	393	292	597	1185	1494	1220	2300	765	923	934

	NY	OMA	PHI	PHO	PIT	StL	SLC	SF	SEA	DC
Boston	188	1282	271	2300	483	1038	2099	2699	2493	393
Buffalo	292	883	279	1906	178	662	1699	2300	2117	292
Chicago	713	432	666	1453	410	262	1260	1858	1737	597
Dallas	1374	586	1299	887	1070	547	999	1483	1681	1185
Denver	1631	488	1579	586	1320	796	371	949	1021	1494
Houston	1420	794	1341	1017	1137	679	1200	1645	1891	1220
Los Angeles	2451	1315	2394	357	2136	1589	579	347	959	2300
Memphis	957	529	881	1263	660	240	1250	1802	1867	765
Miami	1092	1397	1019	1982	1010	1061	2089	2594	2734	923
Minneapolis	1018	290	985	1280	743	466	987	1584	1395	934
New York	-	1144	83	2145	317	875	1972	2571	2408	230
Omaha	1144	-	1094	1036	836	354	833	1429	1369	1014
Philadelphia	83	1094	-	2083	259	811	1925	2523	2380	123
Phoenix	2145	1036	2083	-	1828	1272	504	653	1114	1973
Pittsburgh	317	836	259	1828	-	559	1668	2264	2138	192
Saint Louis	875	354	811	1272	559	-	1162	1744	1724	712
Salt Lake City	1972	833	1925	504	1668	1162	-	600	701	1848
San Francisco	2571	1429	2523	653	2264	1744	600	-	678	2442
Seattle	2408	1369	2380	1114	2138	1724	701	678	-	2329
Washington D.C.	230	1014	123	1973	192	712	1848	2442	2329	-

City	Beijing	Tianjin	Shanghai	Chongqing	Hohhot	Urumqi	Lhasa	Yinchuan	Nanning	Harbin	Changchun	Shenyang
Beijing	0	125	1239	3026	480	3300	3736	1192	2373	1230	979	684
Tianjin	125	0	1150	1954	604	3330	3740	1316	2389	1207	955	661
Shanghai	1239	1150	0	1945	1717	3929	4157	2092	1892	2342	2090	1796
Chongqing	3026	1954	1945	0	1847	3202	2457	1570	993	3156	2905	2610
Hohhot	480	604	1717	1847	0	2825	3260	716	2657	1710	1458	1164
Urumqi	3300	3330	3929	3202	2825	0	2668	2111	4279	4531	4279	3985
Lhasa	3736	3740	4157	2457	3260	2668	0	2547	3431	4967	4715	4421
Yinchuan	1192	1316	2092	1570	716	2111	2547	0	2673	2422	2170	1876
Nanning	2373	2389	1892	993	2657	4279	3431	2673	0	3592	3340	3046
Harbin	1230	1207	2342	3156	1710	4531	4967	2422	3592	0	256	546
Changchun	979	955	2090	2905	1458	4279	4715	2170	3340	256	0	294
Shenyang	684	661	1796	2610	1164	3985	4421	1876	3046	546	294	0

Exercise 3.32 *One's data in a high dimensional space may lie on a lower dimensional sheath. To test for this one might for each data point find the set of closest data points and calculate the vector distance from the data point to each of the close points. If the set of these distance vectors is a lower dimensional space than the number of distance points, then it is likely that the data is on a low dimensional sheath. To test the dimension of the space of the distance vectors one might use the singular value decomposition to find the singular values. The dimension of the space is the number of large singular values. The low singular values correspond to noise or slight curvature of the sheath. To test this concept generate a data set of points that lie on a one dimensional curve in three space. For each point find maybe ten nearest points, form the matrix of distance, and do a singular value decomposition on the matrix. Report what happens.*

Using code such as the following to create the data.

```
function [ data, distance ] = create_sheath( n )
```

```
%creates n data points on a one dimensional sheath in three dimensional
%space
%
if nargin==0
    n=100;
end
data=zeros(3,n);
for i=1:n
    x=sin((pi/100)*i);
    y=sqrt(1-x^2);
    z=0.003*i;
    data(:,i)=[x;y;z];
end
%subtract adjacent vertices
distance=zeros(3,10);
for i=1:5
    distance(:,i)=data(:,i)-data(:,6);
    distance(:,i+5)=data(:,i+6)-data(:,6);
end
end
```