Mathematic Topics for Computer Science

Kon Yi

July 8, 2025

Abstract

A note for some cool topics, and I learn it by reading resource in the Internet by myself.

Contents

Chapter 1

Singular Value Decomposition (SVD)

1.1 Introduciton

Singular Value Decomposition (SVD) is a type of factorization of matrix. It says that a matrix A can be written as a product of three matrices $A = UDV^t$ where the columns of U and V are orthonormal and the matrix D is idagonal. SVD can be used to find a low rank matrix to be a good approximation of the original matrix A.

Unlike Spectral Decomposition Theorem, SVD is defined for all matrices (rectangular or square), and since U and V are orthogonal matrices, so we know

$$A^{-1} = VD^{-1}U^t$$

To gain insight into the SVD, treat the rows of an $n \times d$ matrix A as n points in a d-dimensional space and consider the problem of finding the best k-dimensional subspace w.r.t. the set of points. Here best means minimize the sum of the squares of the perpendicular distances of the points to the subspace. We begin with a special case where the subspace is 1-dimensional, a line through the origin. We will see that the best-fitting k-dimensional subspace can be found by k applicaitons of the best fitting line algorithm.

Suppose we have a set of points $\{\mathbf{x}_i \mid 1 \leq i \leq n\}$ in the plane. Consider projecting a point $\mathbf{x}_i$ onto a line through the origin. Then

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

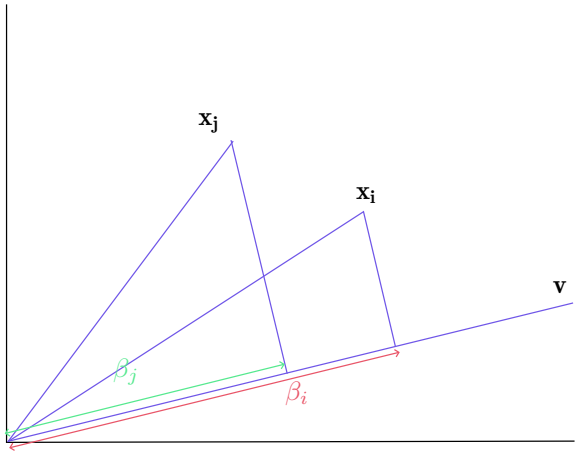

Figure 1.1: The projection of the point $\mathbf{x}_i$ onto the line through the origin in the direction of $\mathbf{v}$

See Figure 1.1. Thus,

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

and notice that $\sum_{i=1}^{n} \left(x_{i1}^2 + x_{i2}^2 + \cdots + x_{id}^2\right)$ is a constant (not dependent on the line), so minimizing the sum of the squares of the distances is equivalent to maximizing the sum of the squares of the lengths of the projections onto the line.

The reason to choose "square" is because the square has many beautiful properties, such as Pythagoras theorem. By this, we can deduce the equivalence of evaluating the maximum of the sum of squared projection. In addition, we will see that we can use Greedy Algorithm to find the best-fit k-dimensional subspaces (which we will define soon) and for this too, the square is important. Just like the least-square method in Calculus, there is a reason behind choosing square.

1.2 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 length squared of the projections is $|A\mathbf{v}|^2$. The best fit line is the one that 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, which is a column vector, as the best fit line through the origin for n points in $d-$space that are the rows of A. Thus

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

The value $\sigma_1(A) = |A\mathbf{v_1}|$ is called the *first singular value* of A. Note that σ_1^2 is the sum of the squares of the projections of the points to the line determined by $\mathbf{v_1}$.

The greedy approach to find the best $2-$dimensional subspce for a matrix A, takes $\mathbf{v_1}$ as the first basis vector for the $2-$dimensional subspace and finds the best $2-$dimensional subspace containing $\mathbf{v_1}$. The fact that we are using the sum of squared distances will again help. 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 $\mathbf{v_1}$ in the subspace plus the sum of squared projections along a vector perpendicular to $\mathbf{v_1}$ in the subspace. See Figure 1.2.

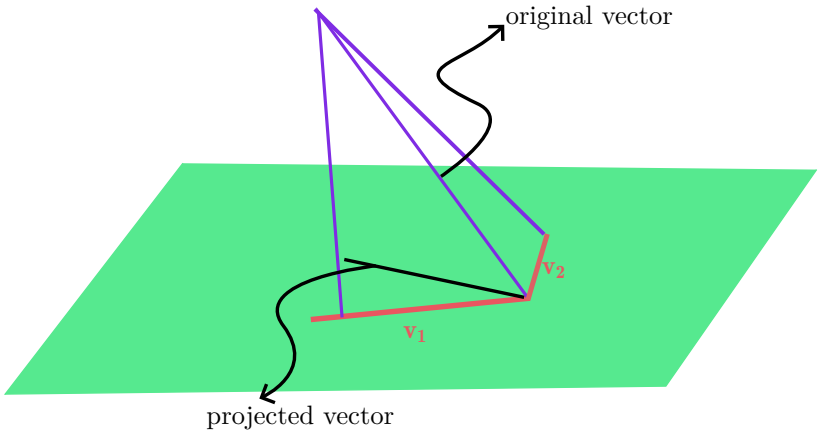

Figure 1.2: The squared length of projection from original vector to a subspace spanned by $\mathbf{v_1}$ and a unit vector perpendicular to $\mathbf{v_1}$, $\mathbf{v_2}$, is the squared length of projection on $\mathbf{v_1}$ plus the squared length of projection on $\mathbf{v_2}$.

Thus, instead of looking for the best 2-dimensional subspace containing $\mathbf{v_1}$, look for a unit vector perpendicular to $\mathbf{v_1}$, which is the $\mathbf{v_2}$ above, 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

similar manner, and we may guess this strategy is correct. We will see that the greedy strategy is in fact true for every dimeension.

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

$$\mathbf{v_2} = \underset{v \perp v_1, |\mathbf{v}|=1}{\arg\max} |A\mathbf{v}|.$$

The value $\sigma_2(A) = |A\mathbf{v_2}|$ is called the *second singular value* of A. The *third singular vector* $\mathbf{v_3}$ is defined similarly by

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

and so on. The process stops when we have found

$$\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$$

as singular vectors and

$$\underset{v \perp \mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}, |v|=1}{\arg\max} |A\mathbf{v}| = 0.$$

Note. Here we know r is rank A since if we extend $S = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}\}$ to an orthogonal basis by Gram-Schmidt process, say $B = S \cup T$, then we know $\forall t \in T$, $t \in \ker A$ by above arguments, so $T \subseteq \ker A$. Conversely, we can show that $\ker A \subseteq T$ by some simple proof. Hence, T spans $\ker A$, and by rank and nullity theorem we have $|S| = r = \operatorname{rank} A$.

Here we show that the greedy algorithm indeed find the best subspaces of every dimension. That is, it is impossible that we find a better subspace by picking some $\mathbf{v_i}$ that does not maximize $|A\mathbf{v}|$ in the greedy process.

Theorem 1.2.1. Let A be an $n \times d$ matrix where $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ are the singular vectors defined above. For $1 \leq k \leq r$, let V_k be the subspace spanned by $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$. Then 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 of A. For any 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 on W. Now choose a 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, then we can pick any unit vector of W to be $\mathbf{w_2}$. If not, choose $\mathbf{w_2}$ to be the unit vector in W perpendicular to the projection of $\mathbf{v_1}$ onto W (such $\mathbf{w_2}$ is perpendicular to $\mathbf{v_1}$ by Theorem of Three Perpendiculars). Now since $\mathbf{v_1}$ was chosen to maximize $|A\mathbf{v_1}|^2$, it follows that $|A\mathbf{w_1}|^2 \leq |A\mathbf{v_1}|^2$. Since $\mathbf{v_2}$ was chosen to maximize $|A\mathbf{v_2}|^2$ over all unit $\mathbf{v}$ perpendicular to $\mathbf{v_1}$, we have $|A\mathbf{w_2}|^2 \leq |A\mathbf{v_2}|^2$. 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.

Now 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 a basis $\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_k}$ of W so that $\mathbf{w_k}$ is perpendicular to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k-1}}$[a]. Then

$$\sum_{i=1}^{k} |A\mathbf{w_i}|^2 \leq \sum_{i=1}^{k-1} |A\mathbf{v_i}|^2 + |A\mathbf{w_k}|^2$$

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

$$\sum_{i=1}^{k} |A\mathbf{w_i}|^2 \leq \sum_{i=1}^{k} |A\mathbf{v_i}|^2,$$

proving that V_k is at least as good as W and hence is optimnal. ■

[a] We prove in Appendix A.1 that such basis exists

Note. Note that the n vectors $A\mathbf{v_i}$ is really 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 intepretation 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 the matrix A". This is indeed the case and is the matrix analogy of decomposing a vector into its components along orthognal 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 $\mathbf{a_j}$, we have

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

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 entries of A. Thus, the sum of squares of the singular values of A is indeef the square of "whole contentcof 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 1.2.1. For any matrix A, the sum of squares of the singular values equals the Frobenius norm. That is, $\sum \sigma_i^2(A) = \|A\|_F^2$.

Proof. By the preceding discussion. ■

A matrix A can be described fully by how it transforms the vectors v_i. Every vector v can be written as a linear combination of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ and a vector perpendicular to all the $\mathbf{v_i}$. Now $A\mathbf{v}$ is the same linear combination of $A\mathbf{v_1}, A\mathbf{v_2}, \ldots, A\mathbf{v_r}$ as $\mathbf{v}$ is of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$. So the $A\mathbf{v_1}, A\mathbf{v_2}, \ldots, A\mathbf{v_r}$ form a fundamental set of vector associated with A, We normalize them to length one by

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

The vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}$ are called the *left singular vectors* of A. The $\mathbf{v_i}$ are called the *right singular vectors*. The SVD theorem will fully explain the reason for these terms.

Clearly, the right singular vectors are orthogonal by definition. We now show that the left singular vectors are also orthogonal and that $A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^t$.

Theorem 1.2.2. Let A be a rank r matrix. The left singular vectors of A, $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}$, are orthogonal.

Proof. The proof is by induction on r. For $r = 1$, there is only one $\mathbf{u_i}$ so the theorem is trivially true. For the inductive part consider the matrix

$$B = A - \sigma_1 \mathbf{u_1} \mathbf{v_1}^t.$$

The implied algorithm in the definition of singular value decomposition applied to B is identical to a run of the algorithm on A for its second and later singular vectors and singular values. To see this, first observe that $B\mathbf{v_1} = A\mathbf{v_1} - \sigma_1 \mathbf{u_1} \mathbf{v_1}^t \mathbf{v_1} = 0$. since $\sigma_1 \mathbf{u_1} = A\mathbf{v_1}$ and $\mathbf{v_1}^t \mathbf{v_1} = \langle \mathbf{v_1}, \mathbf{v_1} \rangle = 1$. It then follows that the first right singular vector, call it $\mathbf{z}$, of B will be perpendicular to $\mathbf{v_1}$ since if it

had a component $\mathbf{z_1}$ along $\mathbf{v_1}$, then

$$\left| B\frac{\mathbf{z}-\mathbf{z_1}}{|\mathbf{z}-\mathbf{z_1}|} \right| = \frac{|B\mathbf{z}|}{|\mathbf{z}-\mathbf{z_1}|} > |B\mathbf{z}|,$$

contradicting the arg max definition of z. But for any $\mathbf{v}$ perpendicular to $\mathbf{v_1}$, $B\mathbf{v} = A\mathbf{v}$. Thus, the top singular vector of B is indeed a second singular vector of A. Repeating this argument shows that a run of the algorithm on B is the same as a run on A for its second and later singular vectors [a]. Thus, there is a run of the algorithm that finds that B has right singular vectors $\mathbf{v_2}, \mathbf{v_3}, \ldots \mathbf{v_r}$ and corresponding left singular vectors $\mathbf{u_2}, \mathbf{u_3}, \ldots, \mathbf{u_r}$. By the induction hypothesis, $\mathbf{u_2}, \mathbf{u_3}, \ldots, \mathbf{u_r}$ is orthogonal.

It remains to show that $\mathbf{u_1}$ is orthogonal to the other $\mathbf{u_i}$. Suppose not and for some $i \geq 2$, $\mathbf{u_1}^t\mathbf{u_i} \neq 0$. Without loss of generality assume that $\mathbf{u_1}^t\mathbf{u_i} > 0$. The proof is symmetric for the case where $\mathbf{u_1}^t\mathbf{u_i} < 0$. Now, for infinitesimally small $\varepsilon > 0$, the vector

$$A\left(\frac{\mathbf{v_1}+\varepsilon\mathbf{v_i}}{|\mathbf{v_1}+\varepsilon\mathbf{v_i}|}\right) = \frac{\sigma_1\mathbf{u_1}+\varepsilon\sigma_i\mathbf{u_i}}{\sqrt{1+\varepsilon^2}}$$

has length at least as large as its component along $\mathbf{u_1}$ which is

$$\mathbf{u_1}^t\left(\frac{\sigma_1\mathbf{u_1}+\varepsilon\sigma_i\mathbf{u_i}}{\sqrt{1+\varepsilon^2}}\right) = \left(\sigma_1+\varepsilon\sigma_i\mathbf{u_1}^t\mathbf{u_i}\right)\left(1-\frac{\varepsilon^2}{2}+O\left(\varepsilon^4\right)\right) = \sigma_1+\varepsilon\sigma_i\mathbf{u_1}^t\mathbf{u_i}-O\left(\varepsilon^2\right) > \sigma_1$$

a contradiction. Thus, $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}$ are orthogonal. ■

[a] Notice that every singular vector of B must be perpendicular to $\mathbf{v_1}$ by the above argument about if it has a component $\mathbf{z_1}$ along $\mathbf{v_1}$.

1.3 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 $\sigma_1, \sigma_2, \ldots, \sigma_r$. Then, $\mathbf{u_i} = \frac{1}{\sigma_i}A\mathbf{v_i}$, for all $i = 1, 2, \ldots, r$, are the left singular vectors and by Theorem 1.3.1, A can be decomposed into a sum of rank one matrices as

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

We first prove a simple lemma stating that two matrices A and B are identical if and only if $Av = Bv$ for all vectors v.

Lemma 1.3.1. Two matrices A and B are identical if and only if $Av = Bv$ for all vectors v.

Proof. We only prove the "if" direction. Notice that $a_{ij} = e_i^t A e_j = e_i^t B e_j = b_{ij}$ for all i, j, so this is true. ■

Note. The lemma statas that in the abstract, a matrix A can be viewed as a transformation that maps vector v onto Av.

Theorem 1.3.1 (SVD theorem). 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. For each singular vector $\mathbf{v_j}$, we know

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

Since any vector $\mathbf{v}$ can be expressed as a linear combination of the singular vectors plus a vector perpendicular to the $\mathbf{v_i}$'s. Hence, we know $A\mathbf{v} = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^t \mathbf{v}$ and by Lemma 1.3.1 we know $A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^t$. ■

Definition 1.3.1. The decomposition in Theorem 1.3.1 is called the *singular value decomposition*, SVD, of A. In matrix notation $A = UDV^t$ where the columns of U and V consist of the left and right singular vectors, respectively, and D is a diagonal matrix whose diagonal entries are the singular values of A.

$$
\begin{aligned}
UDV^t &= \begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \cdots & \mathbf{u_r} \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{pmatrix} \begin{pmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_r} \end{pmatrix} \\
&= \begin{pmatrix} \sigma_1 \mathbf{u_1} & \sigma_2 \mathbf{u_2} & \ldots & \sigma_r \mathbf{u_r} \end{pmatrix} \begin{pmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_r} \end{pmatrix} \\
&= \sigma_1 \mathbf{u_1} \mathbf{v_1}^t + \sigma_2 \mathbf{u_2} \mathbf{v_2}^t + \cdots + \sigma_r \mathbf{u_r} \mathbf{v_r}^t = A.
\end{aligned}
$$

Note. The above U, D, V^t forms the reduced (compact) SVD, since $U \in M_{n \times r}(F)$, $D \in M_{r \times r}(F)$, and $V^t \in M_{r \times d}(F)$. Actually the full SVD is to fill 0s in U, D, V^t to make these 3 matrices square. We prefer to use reduced SVD since it can save less entries than full SVD.

Note. 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, too. 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.

1.4 Best Rank k Approximations

There are two important matrix norms, the Frobenius norm denoted $\|A\|_F$ and the 2-norm denoted $\|A\|_2$. The 2-norm of the matrix A is given by

$$\max_{|\mathbf{v}|=1} |A\mathbf{v}|$$

and thus equals to the largest singular value of the matrix.

Let A be an $n \times d$ matrix and think of the rows of A as n points in d-dimensional space. The Frobenius norm of A is the square root of the sum of squared distnace of the points to the origin. The 2-norm is the square root of the sum of squared distances to the origin along the direction that maximizes this quantity. More specifically, suppose $r_1, r_2, \ldots, r_n$ are the n rows of A, then

$$\|A\|_F = \sqrt{\|r_1\|^2 + \|r_2\|^2 + \cdots + \|r_n\|^2} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} |a_{ij}|^2},$$

and since $|r_i \cdot \mathbf{v}|$ is the length of the projection from r_i to the line in the direction of $\mathbf{v}$, so $|A|_2$ is like finding the maximum of the sum of lenghts of all projections.

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 termns. It is clear that A_k has rank k[1]. Furthermore, A_k is the best rank k approximation to A when the error is measured in either 2-norm or the Frobenius norm.

Lemma 1.4.1. 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 of A. Since the $\mathbf{v_i}$ are orthonormal, the projection of the vector $\mathbf{a}$ onto V_k is given by

$$\sum_{i=1}^{k} (\mathbf{a} \cdot \mathbf{v_i}) \mathbf{v_i}^t.$$ [a]

Thus, the matrix whose rows are the projections of the rows of A onto V_k is given by $\sum_{i=1}^{k} A\mathbf{v_i}\mathbf{v_i}^t$. The last expression simplifies to

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

■

[a] Since $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\}$ is an orthonormal basis of V_k and $\mathbf{a} \cdot \mathbf{v_i} = \langle \mathbf{a}, \mathbf{v_i} \rangle$.

The matrix A_k is the best rank k approximation to A in both the Frobenius and the 2-norm. First we show that the matrix A_k is the best approximation to A in the Frobenius norm.

Theorem 1.4.1. For any matrix B of rank at most k

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

Proof. First notice that the meaning of $\|A - B\|_F^2$ is that, if we regard the rows of A, B as some vectors, then $\|A - B\|_F^2$ is the sum of squared distance of $a_i - b_i$ for all i, where a_i, b_i are the i-th rows of A and B.

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 projeciton of the corresponding row of A onto V, otherwise replacing the rows of B with the projection of the corresponding row of A onto V does not make the rank of new B exceeding k since all the new rows of B are in V and $\dim V \leq k$. However, this would reduce $\|A - B\|_F^2$ (think about the meaning of $\|A - B\|_F^2$).

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 distance of rows of A to V.

Now if we pick V as V_k, the space spanned by the first k left singular vectors, then by Theorem 1.2.1, we know this is the best-fit k-dimensional space of A, and by Lemma 1.4.1, we know

$$\|A - A_k\|_F^2 \leq \|A - B\|_F^2$$

for all matrix B of rank at most k. ■

Next we tackle the 2-norm. 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.

Lemma 1.4.2. $\|A - A_k\|_2^2 = \sigma_{k+1}^2$.

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 $A - A_k$. Let $\mathbf{v}$ be the top singular vector of $A - A_k$. Express $\mathbf{v}$ as a linear combination of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ [a] . That is, write $\mathbf{v} = \sum_{i=1}^{r} \alpha_i \mathbf{v_i}$.

[1] See Appendix A.2

Then

$$|(A-A_k)\mathbf{v}| = \left|\sum_{i=k+2}^{r} \sigma_i \mathbf{u_i} {\mathbf{v_i}}^t \sum_{j=1}^{r} \alpha_i \mathbf{v_j}\right| = \left|\sum_{i=1}^{k+1} \alpha_i \sigma_i \mathbf{u_i} {\mathbf{v_i}}^t \mathbf{v_i}\right|$$
$$= \left|\sum_{i=k+1}^{r} \alpha_i \sigma_i \mathbf{u_i}\right| = \sqrt{\sum_{i=k+1}^{r} \alpha_i^2 \sigma_i^2}.$$

The $\mathbf{v}$ maximizing this last quantity, subject to the constraint that $|\mathbf{v}| = \sum_{i=1}^{r} \alpha_i^2 = 1$, occurs when $\alpha_{k+1} = 1$ and the rest of the α_i's are 0. Thus,

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

proving the lemma. ■

[a] Since we can extend $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ to an orthonormal basis of $\mathbb{R}^d$ (Suppose $A \in M_{n\times d}(\mathbb{R})$) , and since the part of this basis except $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ is a basis of $\ker A$, so express $\mathbf{v}$ in this basis is equivalent to express $\mathbf{v}$ in $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$

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

Theorem 1.4.2. If A is a $n \times d$ 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 $\|A - A_k\|_2 = 0$. Thus assume that A is of rank greater than k. By Lemma 1.4.2, $\|A - A_k\|_2^2 = \sigma_{k+1}^2$. Now suppose there is some matrix B of rank at most k such that B is a better 2-norm approximation to A than A_k. That is, $\|A - B\|_2 < \sigma_{k+1}$ [a]. Note that $\ker B$ is 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

$$\ker B \cap \text{span}\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}\}.$$

Scale z so that $|\mathbf{z}| = 1$. We now show that for this vector $\mathbf{z}$, which lies in the space of the first $k+1$ singular vectors of A, that $(A - B)\mathbf{z} \geq \sigma_{k+1}$. Hence, the 2-norm of $A - B$ is at least σ_{k+1} contradicting the assumption that $\|A - B\|_2 < \sigma_{k+1}$. First

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

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

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

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

$$|A\mathbf{z}|^2 = \left|\sum_{i=1}^{n} \sigma_i \mathbf{u_i} \mathbf{v_i} 6 t \mathbf{z}\right|^2 = \sum_{i=1}^{n} \sigma_i^2 \left({\mathbf{v_i}}^t \mathbf{z}\right)^2 = \sum_{i=1}^{k+1} \sigma_i^2 \left({\mathbf{v_i}}^t \mathbf{z}\right)^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} \left({\mathbf{v_i}}^t \mathbf{z}\right)^2 = {}^{b}\sigma_{k+1}^2.$$

It follows that

$$\|A - B\|_2^2 \geq \sigma_{k+1}^2$$

contradicting the assumption that $\|A - B\|_2 < \sigma_{k+1}$. This proves the theorem. ■

[a] Later we will see that $\sigma_i > 0$ for all i

[b] Since $\mathbf{z}$ is in span$\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}\}$, so $\mathbf{z} = \sum_{i=1}^{k+1} \left({\mathbf{v_i}}^t \mathbf{z}\right)^2 v_i$ and since $|\mathbf{z}| = 1$, so we know $\sum_{i=1}^{k+1} \left({\mathbf{v_i}}^t \mathbf{z}\right)^2 = 1$.

1.5 Computing Singular Values

As previously seen. We know $A\mathbf{v_j} = \sigma_i \mathbf{u_j}$ for all j.

Now if we write down the transpose of A, then we know

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

and thus we have

$$A^t \mathbf{u_i} = \sigma_i \mathbf{v_i}.$$

Now we can observe that

$$A^t A \mathbf{v_i} = A^t \sigma_i \mathbf{u_i} = \sigma_i^2 \mathbf{v_i}$$

and

$$A A^t \mathbf{u_i} = A \sigma_i \mathbf{v_i} = \sigma_i^2 \mathbf{u_i},$$

so it seems that the singular values of A are identical to the eigenvalues of $A^t A, AA^t$, and we'll prove that in fact this is true.

We first introduce a concept called positive semi-definite matrix.

Definition 1.5.1. A $n \times n$ square matrix A is called *positive semi-definite* if

- A is symmetric.
- $x^t A x \geq 0$ for all $x \in \mathsf{F}^n$.

We now show some equivalent conditions of a matrix A to be positive semi-definite.

Lemma 1.5.1. For a symmetric $n \times n$ matrix A, the followings are equivalent.

1. $v^t A v \geq 0$ for all $v \in \mathsf{F}^n$.
2. All the eigenvalues of A are non negative.
3. There exists some matrix B such that $A = B^t B$.
4. A is a gram matrix, that is, $A_{ij} = u_i^t u_j$ for every i, j, and $u_1, u_2, \ldots, u_n \in U$, which is a vector space.

proof of 1. $\to$ 2. For any eigenvector v of A, suppose its corresponding eigenvalue is λ, then we know

$$v^t A v = v^t \lambda v = \lambda v^t v \geq 0$$

and since $v^t v \geq 0$, so $\lambda \geq 0$. ■

proof of 2. $\to$ 3. Since A is symmetric, so it is self-adjoint, so there exists an orthonormal basis $\mathcal{B} = \{v_1, v_2, \ldots, v_n\}$ such that $[A]_{\mathcal{B}}$ is diagonal , and suppose

$$[A]_{\mathcal{B}} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix},$$

now if we define $S(v_i) = \sqrt{\lambda_i} v_i$ for all i in $1, 2, \ldots, n$ and extend it to a linear transformation, then we know

$$[S]_{\mathcal{B}} = \begin{pmatrix} \sqrt{\lambda_1} & & & \\ & \sqrt{\lambda_2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_n} \end{pmatrix}.$$

Besides, notice that $[S]_{\mathcal{B}}$ is self-adjoint, that is, $[S]_{\mathcal{B}} = S_{\mathcal{B}^t}$, and also

$$[S]_{\mathcal{B}}^t[S]_{\mathcal{B}} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} = [A]_{\mathcal{B}}.$$

■

proof of 3. → 4. . Suppose $A = B^tB$, and if $B = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}$, where b_i is the i-th column of B , then we know $(B^tB)_{ij} = b_i^t b_j$. ■

proof of 4. → 1. . Suppose $A_{ij} = u_i^t u_j$, then for any vector $v = \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix}^t$, we know

$$\begin{aligned} v^tAv &= \sum_{i=1}^n \sum_{j=1}^n v_i A_{ij} v_j = \sum_{i=1}^n \sum_{j=1}^n v_i u_i^t u_j v_j \\ &= \left(\sum_{i=1}^n u_i v_i\right)^t \left(\sum_{j=1}^n u_j v_j\right) = \left\|\sum_{i=1}^n u_i v_i\right\|^2 \geq 0. \end{aligned}$$

■

Now go back to see A^tA, suppose $A = UDV^t$, then we can see that

$$A^tA = VD^tU^tUDV^t = VD^tDV^t,$$

where

$$D^tD = \begin{pmatrix} \sigma_1^2 & & & & & & \\ & \sigma_2^2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r^2 & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix},$$

and since V is an orthogonal matrix, so $A^tA \sim D^tD$, which means the eigenvalues of A^tA are exactly σ_i^2. Now we have another problem, is $\sigma_i > 0$ or $\sigma_i < 0$? Notice that $(A^tA)^t = A^tA$, and thus it is symmetric. Also, for any v satisfies the size of A^tA,

$$v^tA^tAv = (Av)^t(Av) \geq 0,$$

so all of the eigenvalues of A^tA are positive by Lemma 1.5.1.

Hence, if we want to find all singular values of A, we can just find the eigenvalues of A^tA.

Note. Observe the matrix D^tD, we can found that if some of the eigenvalues of A^tA repeat, then some of the σ_i^2 repeat, and thus the sequence, $\sigma_1^2, \sigma_2^2, \dots, \sigma_r^2$ is unique.

1.6 Power Method for Computing the Singular Value Decomposition

Conputing the singular value decomposition is an important branch of numerical analysis in which there have been many sophiscated developments over a long period of time. Here we present an "in-principle" method to establish that the approximation of SVD of a matrix A can be computed in polynomial time. The method we present is called the Power Method, is simple and in fact the conceptual starting point for many algorithms. It is easiest to describe first in the case when A is square symmetric and has the same rignt and left sngular vectors [2], namely,

[2] See Appendix A.3, which shows the left and right singular vectors of a square symmetric matrix must be identical.

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

In this case, we have

$$A^2 = \left(\sum_{i=1}^{r} \sigma_i \mathbf{v_i} \mathbf{v_i}^t \right) \left(\sum_{j=1}^{r} \sigma_j \mathbf{v_j} \mathbf{v_j}^t \right) = \sum_{i,j=1}^{r} \sigma_i \sigma_j \mathbf{v_i} \mathbf{v_i}^t \mathbf{v_j} \mathbf{v_j}^t = \sum_{i=1}^{r} \sigma_i^2 \mathbf{v_i} \mathbf{v_i}^t.$$

Note. The outer product $\mathbf{v_i} \mathbf{v_j}^t$ is a matrix rather than a value and is not zero even for $i \neq j$.

Similarly, if we take the k-th power of A, again all the cross terms are zero and we will get

$$A^k = \sum_{i=1}^{r} \sigma_i^k \mathbf{v_i} \mathbf{v_i}^t.$$

If we had $\sigma_1 > \sigma_2$, we would have

$$\frac{1}{\sigma_1^k} A^k \to \mathbf{v_1} \mathbf{v_1}^t.$$

Now we do not know σ_1 beforehand and cannot find this limit, but if we just take A^k and divide by $\left\|A^k\right\|_F$ so that the Frobenius norm is normalized to 1 now, that matrix will converge to the rank 1 [3] matrix $\mathbf{v_1} \mathbf{v_1}^t$[4] from which $\mathbf{v_1}$ may be computed.

Note. This is still an intuitive description, which we will make precise shortly.

First, we cannot make the assumption that A is square and has the same right and left singular vectors. But $B = AA^t$ satisfies both these conditions. If again, the SVD of A is $\sum_i \sigma_i \mathbf{u_i} \mathbf{v_i}^t$, then by direct computation we know

$$B = AA^t = \left(\sum_i \sigma_i \mathbf{u_i} \mathbf{v_i}^t \right) \left(\sum_j \sigma_j \mathbf{v_j} \mathbf{u_j}^t \right) = \sum_i \sigma_i^2 \mathbf{u_i} \mathbf{u_i}^t.$$

This is the spectral decomposition of B [5]. Using same kind of calculation above,

$$B^k = \sum_i \sigma_i^{2k} \mathbf{u_i} \mathbf{u_i}^t.$$

As k increases, for $i > 1$, $\sigma_i^{2k} / \sigma_1^{2k}$ goes to 0 and B^k is approximately equal to $\sigma_1^{2k} \mathbf{u_1} \mathbf{u_1}^t$ provided that for each $i > 1$, $\sigma_i(A) < \sigma_1(A)$. This is because

$$B^k = \sigma_1^{2k} \left(\mathbf{u_1} \mathbf{u_1}^t + \sum_{i \neq 1} \frac{\sigma_i^2}{\sigma_1^2} \mathbf{u_i} \mathbf{u_i}^t \right) \to \sigma_1^2 \mathbf{u_1} \mathbf{u_1}^t.$$

This suggests a way of finding σ_1 and $\mathbf{u_1}$, by successively powering B. But there are two issues. First, if there is a significant gap between the first and second singular values of a matrix, then the above argument applies and the power method will quickly converge to the first left singular vector. Suppose there is no significant gap. In the extreme case, there may be ties for the top singular values. Then the above argument does not work. There are cubersome eays of overcoming this by assuming a "gap" between σ_1 and σ_2; such proofs do have the advantage that with a greater gap, better results can be proved, but at the cost of some mess. Here, instead, we will adopt a clean solution in Theorem 1.6.1 below which states that even with ties, the power method converges to some vector in the span of thosse singular vectors corresponding to the "nearly highest" singular values.

[3] See Appendix A.4, which proves it is rank one.
[4] See Appendix A.5, which shows it converges.
[5] See Appendix B.1 for detail.

A second issue is that computing B^k costs k matrix multiplications when done in a straight-foward manner or $O(\log k)$ when done by successive squaring. Instead we compute

$$B^k\mathbf{x}$$

where $\mathbf{x}$ is a random unit length vector, the idea being that the component of $\mathbf{x}$ in the direction of $\mathbf{u_1}$ would get multiplied by σ_1^2 each time, while the component of $\mathbf{x}$ along other $\mathbf{u_i}$ would be multiplied only by σ_i^2. Of course, if the component of $\mathbf{x}$ along $\mathbf{u_1}$ is zero to start with, this would not help at all - it will always remain 0[6]. But, this problem is fixed by picking $\mathbf{x}$ to be random as we show in Lemma 1.6.1.

Each increase in k requires multiplying B by the vector $B^{k-1}\mathbf{x}$, which can further break up into

$$B^k\mathbf{x} = A\left(A^t\left(B^{k-1}\mathbf{x}\right)\right).$$

This requires two matrix-vector products, involving the matrices A^t and A. In many applications, data matrices are sparse - many entries are 0[7]. Sparse matrices are often represented by giving just a linked list of the non-zero entries and their values. If A is represented in this sparsed manner, then we can calculate a matrix vector product in time proportional to the number of nonzero entries in A. Since $B^k\mathbf{x} \approx \sigma_1^{2k}\mathbf{u_1}\left(\mathbf{u_1}^t \cdot \mathbf{x}\right)$ is a scalar multiple of $\mathbf{u_1}$, $\mathbf{u_1}$ can be recovered from $B^k\mathbf{x}$ by normalization (since $\|\mathbf{u_1}\| = 1$).

We start with a technical lemma needed in the proof of the theorem.

Lemma 1.6.1. Let $(x_1, x_2, \ldots, x_d)$ be a unit d-dimensional vector picked at random from the set $\{\mathbf{x} : |\mathbf{x}| \leq 1\}$. The probability that $|\mathbf{x_1}| \geq \frac{1}{20\sqrt{d}}$ is at least $\frac{9}{10}$.

Proof. We first show that for a vector $\mathbf{v}$ picked at random with $\mathbf{v} \leq 1$, the probability that $v_1 \geq \frac{1}{20\sqrt{d}}$ is at least $\frac{9}{10}$. Then we let $\mathbf{x} = \frac{\mathbf{v}}{|\mathbf{v}|}$. This can only increase the value of v_1, so the result follows.

Let $\alpha = \frac{1}{20\sqrt{d}}$. The probability that $|v_1| \geq \alpha$ equals one minus the probability that $|v_1| \leq \alpha$. The probability that $|v_1| \leq \alpha$ is equal to the fraction of the volume of the unit sphere with $|v_1| \leq \alpha$. To get an upper bound on the volume of the sphere with $|v_1| \leq \alpha$, consider twice the volume of the unit radius cylinder of height α. The volume of the portion of the sphere with $|v_1| \leq \alpha$ is less than or equal to $2\alpha V(d-1)$[a], and

$$\Pr(|v_1| \leq \alpha) \leq \frac{2\alpha V(d-1)}{V(d)}.$$

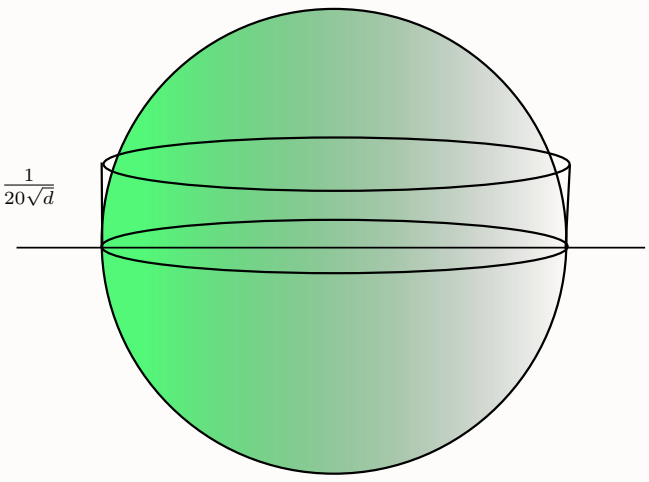

Figure 1.3: A d-dimensional sphere with a cylinder of radius one and height $\frac{1}{20\sqrt{d}}$.

[6] The core of power method is to use $\sigma_1^{2k}\mathbf{u_1}\left(\mathbf{u_1}^t \cdot \mathbf{x}\right)$ to approximate $\mathbf{u_1}$, but if $\mathbf{x}$ has no component in the direction of $\mathbf{u_1}$, then this approximation would be invalid since the inner product in the preceding equation would be zero.

[7] A leading example is the matrix of hypertext links in the web. There are more than 10^{10} web pages and the matrix would be 10^{10} by 10^{10}. But on the average only about 10 entries per row are non-zero; so only about 10^{11} of the possible 10^{20} entries are non-zero.

Now the volume of the unit radius sphere is at least twice the volume of the cylinder of height $\frac{1}{\sqrt{d-1}}$ and radius $\sqrt{1-\frac{1}{d-1}}$ or

$$V(d) \geq \frac{2}{\sqrt{d-1}} V(d-1)\left(1-\frac{1}{d-1}\right)^{\frac{d-1}{2}}.$$

Using $(1-x)^a \geq 1-ax$[b],

$$V(d) \geq \frac{2}{\sqrt{d-1}} V(d-1)\left(1-\frac{d-1}{2}\frac{1}{d-1}\right) \geq \frac{V(d-1)}{\sqrt{d-1}}$$

and

$$\Pr(|v_1| \leq \alpha) \leq \frac{2\alpha V(d-1)}{\frac{1}{\sqrt{d-1}} V(d-1)} \leq \frac{\sqrt{d-1}}{10\sqrt{d}} \leq \frac{1}{10}.$$

Thus the probability that $v_1 \geq \frac{1}{20\sqrt{d}}$ is at least $\frac{9}{10}$. ■

[a] $V(n)$ is the volume of a unit sphere in n-dimensional space.
[b] See https://en.wikipedia.org/wiki/Bernoulli%27s_inequality.

Note. This lemma tells us if we pick any random unit vector, then with high probability, its component along any direcion is big enough ($\geq \frac{1}{20\sqrt{d}}$). Hence, this can prevent the case the component in the $\mathbf{u_1}$ is too small or even 0.

Theorem 1.6.1. Let A be an $n \times d$ matrix and $\mathbf{x}$ a random unit length vector. Let V be the vector space spanned by the left singular vectors of A corresponding to singular value greater than $(1-\varepsilon)\sigma_1$. Let k be $\Omega\left(\frac{\ln(d/\varepsilon)}{\varepsilon}\right)$. Let $\mathbf{w}$ be the unit vector after k iterations of the power method, namely,

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

The probability that $\mathbf{w}$ has a component of at least ε perpendicular to V is at most $\frac{1}{10}$.

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 complete $\{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}\}$ into a basis $\{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_d}\}$ of d-space. Write $\mathbf{x}$ in the basis of the $\mathbf{u_i}$'s as

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

Since $\left(AA^t\right)^k = \sum_{i=1}^{d} \sigma_i^{2k} \mathbf{u_i} \mathbf{u_i}^t$, it follows that $\left(AA^t\right)^k \mathbf{x} = \sum_{i=1}^{d} \sigma_i^{2k} c_i \mathbf{u_i}$. For a random unit length vector $\mathbf{x}$ picked independent of A, the $\mathbf{u_i}$ are fixed vectors and picking $\mathbf{x}$ at random is equivalent to picking random c_i. From Lemma 1.6.1, $|c_1| \geq \frac{1}{20\sqrt{d}}$ with probability at least $\frac{9}{10}$.

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 $\sigma_{m+1}, \ldots, \sigma_n$ are the singular values that are less than $(1-\varepsilon)\sigma_i$. Now

$$\left|\left(AA^t\right)^k \mathbf{x}\right|^2 = \left|\sum_{i=1}^{d} \sigma_i^{2k} c_i \mathbf{u_i}\right|^2 = \sum_{i=1}^{d} \sigma_i^{4k} c_i^2 \geq \sigma_1^{4k} \geq \frac{1}{400d} \sigma_1^{4k},$$

with probability at least $\frac{9}{10}$. [a]

Now we want to calculate the upper bound of the length of the component of $\left(AA^t\right)^k \mathbf{x}$ which is

perpendicular to V. Notice that this part of component is

$$\sum_{i=m+1}^{d} \sigma_i^{2k} c_i \mathbf{u_i},$$

and also we know the upper bound of the squared length of this component 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 is at most

$$\frac{(1-\varepsilon)^{2k}\sigma_1^{2k}}{\frac{1}{20\sqrt{d}}\sigma_1^{2k}} = O\left(\sqrt{d}(1-\varepsilon)^{2k}\right) = O\left(\sqrt{d}e^{-2\varepsilon k}\right) = O\left(\sqrt{d}e^{-\Omega(\ln(d/\varepsilon))}\right) = O(\varepsilon)$$

as desired.

Note. Now we prove that the probability of $\mathbf{w}$ has a component of at most ε perpendicular to V is greater than $\frac{9}{10}$ since we need the condition that

$$\left|\left(AA^t\right)^k \mathbf{x}\right|^2 \geq \frac{1}{400d}\sigma_1^{4k}.$$

Hence, the probability that $\mathbf{w}$ has a component of at least ε perpendicular to V is at most $\frac{1}{10}$.

[a] If we did not choose $\mathbf{x}$ at random, then c_1 could be 0 and this argument won't work.

■

Note. After enough times of iteration ($\Omega\left(\frac{\ln(d/\varepsilon)}{\varepsilon}\right)$), if we pick ε small enough, we can make $\mathbf{w}$ almost have only the component in the direction of $\mathbf{u_1}$, or at least the component in the wrong direction would not be too large (probability of length exceeding $\varepsilon \leq \frac{1}{10}$).

1.7 Application of Singular Value Decomposition

1.7.1 Principal Component Analysis

The traditional use of SVD is in Principal Component Analysis (PCA). PCA is illustrated by an example - customer- product data where there are n customers buying d products. Let matrix A with elements a_{ij} represent the probability of customer i purchasing product j (or the amount or utility of project j to customer i).

Now A may be very huge, so we want to save the matrix with less information, here we first talk about we can save the whole A, but we still want to store it with less information, and then we will talk about what if A is too big so that we cannot even know what evey entry of A is, and in this case we have to predict the entries of A with small portion of the entry of A and then store the prediction result with less information.

But how can we save A with less information? One hypothesis is that there are really only k underlying basic factors like age, income, family size, etc, that determine a customer's purchase behavior. An individual customer's behavior is determined by some weighted combination of these undelying factors. That is, a customer's purchase behavior can be characterized by a k-dimensional vector where k is much smaller than n and d. The components of the vector are weights for each of the basic factors. Associated with each basic factor is a vector of probabilities, each component of which is the probability of purchasing a given product by someone whose behavior depends only on that factor. More abstractly, A is an $n \times d$ matrix that can be expressed as the product of two matrices U and V, where U is an $n \times k$ matrix expressing the factor weights for each customer and V is a $k \times d$ matrix expressing the purchase

probabilities of products that correspond to that factor. One twist is that A may not be equal to UV, but close to it since there may be noise or random perturbations.

Taking the best rank k approximation A_k from SVD (recall Best rank k approximation) gives us such U, V since we can do SVD on A_k and this gives (UD) and V^t this two rank k matrices.

In this traditional setting, one assumed that A was available fully and we wished to find U, V to identify the basic factors or in some applications to "denoise" A (if we think $A - UV$ as noise). Now imagine that n and d are very large, on the order of thousands or even millions, there is probably little one could do to estimate or even store A. In this setting, we may assume that we are given just a few elements of A and wish to estimate (predict) A. If A was an arbitrary matrix of size $n \times d$, this would requires $\Omega(nd)$ pieces of information and cannot be done with a few entries. But again hypothesis 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 cn be used to estimate the whole of A and gives U and V to approximate A. The area is called collaborative filtering. It uses machine learning to imitate SVD and find the solution to minimize the loss function

$$\min_{U,V} \sum_{(i,j)\in\Omega} (A_{ij} - \langle u_i, v_j \rangle)^2 + \lambda \left(\|u_i\|^2 + \|v_i\|^2 \right).$$

where Ω is the set of observed entries in A (the entries we know).

1.7.2 Clustering a mixture of Spherical Gaussians

1.7.3 An Application of SVD to a Discrete Optimization Problem

In the last example, 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 have an optimization problem. Again applying dimension reduction to the data 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 node set V of a directed graph into two subsets S and $\overline{S}$ so that the number of edges from S to $\overline{S}$ is maximized. Let $A = (a_{ij})_{n\times n}$ 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 $i \in \overline{S}$. The vector $\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}$ is unknown and we are trying to find it (or equivalently the cut) , so as to maximize the number of edges accross 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,

$$\sum_{i,j} x_i(1 - x_j)a_{ij} = \mathbf{x}^t A(\mathbf{1} - \mathbf{x}), \tag{1.1}$$

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

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

The SVD is used to solve this problem approximately 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 Equation 1.1 to get,

$$\text{Maximize } \mathbf{x}^t A_k(\mathbf{1} - \mathbf{x}) \quad \text{subject to } x_i \in \{0, 1\}. \tag{1.2}$$

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

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 Equation 1.1 and Equation 1.2 differ by at most this amount.

2. A near optimal $\mathbf{x}$ for Equation 1.2 can be found by exploiting the low rank of A_k, which by Item 1 is near optimal for Equation 1.1 where near optimal means with additive error at most $\frac{n^2}{\sqrt{k+1}}$.

proof of 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 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 1.4.2, we know $\|A-A_k\|_2 = \sigma_{k+1}(A)$. 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 1. ■

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 required to be identical is already NP-hard to solve exactly because it subsumes the problem of whether for a set of n integers, $\{a_1,a_2,\ldots,a_n\}$ [8], there is a partition into two subsets whose sums are equal. So look for algorithms that solve the MAximum Cut Problem Approximately.

proof of 2. 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 coordinateds of the vector $\mathbf{u_i}$ corrsponding to elements in the set S and also for $\mathbf{v_i}$. That is, $\mathbf{u_i}(S)=\sum_{j\in S} u_{ij}$. We will maximize $\sum_{i=1}^k \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(\overline{S})$ by dynamic programming.

> **Note.** Notice that for a 0-1 vector $\mathbf{x}$, it is corresponding to a $S\subseteq\{1,2,\ldots,n\}$ (if $\mathbf{x_i}=1$, then $i\in S$). Hence, $\mathbf{x}^t\mathbf{u_i}=\sum_{j\in S}u_{ij}=\mathbf{u_i}(S)$ and $\mathbf{v_i}^t(\mathbf{1}-\mathbf{x})=\sum_{j\notin S}v_{ij}=\mathbf{v_i}(\overline{S})$, so $\sum_{i=1}^k \sigma_i\,(\mathbf{x}^t\mathbf{u_i})\,(\mathbf{v_i}^t(\mathbf{1}-\mathbf{x}))=\sum_{i=1}^k\sigma_i\mathbf{u_i}(S)\mathbf{v_i}(\overline{S})$.

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

$$\mathbf{w}(S)=\left(\mathbf{u_1}(S),\mathbf{v_1}(\overline{S}),\mathbf{u2}(S),\mathbf{v_2}(\overline{S}),\ldots\right).$$

If we had the list of all such vectors (for all possible S), we could find $\sum_{i=1}^k\sigma_i\mathbf{u_i}(S)\mathbf{v_i}(\overline{S})$ for each of them and take the maximum. There are 2^n subsets of S, but several S could have the same $\mathbf{w}(S)$ and in that case it sufficies to list just one of them. Round each coordinate of each $\mathbf{u_i}$ to the nearest integer multiple of $\frac{1}{nk^2}$.

> **Note.** That is, suppose one coordinate of $\mathbf{u_i}$ is 0.041 and $\frac{1}{nk^2}$ is 0.015, then we set this component to 0.045 since this is the closest number to original value and is a multiple of $\frac{1}{nk^2}$.

Call the rounded vector $\widetilde{\mathbf{u_i}}$. Similarly obtain $\widetilde{\mathbf{v_i}}$. Let $\widetilde{\mathbf{w}}$ denote the vector

$$\widetilde{\mathbf{u_1}}(S),\widetilde{\mathbf{v_1}}(S),\widetilde{\mathbf{u_2}}(S),\widetilde{\mathbf{v_2}}(S),\ldots,\widetilde{\mathbf{u_k}}(S),\widetilde{\mathbf{v_k}}(S).$$

We will construct a list of all possible values of the vectors $\widetilde{\mathbf{w}}(S)$.

> **Note.** Again, if several different S's lead to the same vector $\widetilde{\mathbf{w}}(S)$, we will keep only one copy of the vector $\widetilde{\mathbf{w}}(S)$.

The list will be constructed by Dynamic Programming. For the recursive step of Dynamic Programming, 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 $S'\subseteq$

[8] See Appendix A.6 for detail.

$\{1, 2, \ldots, i+1\}$, namely, S and $S \cup \{i+1\}$. In the first case, the vector

$$\widetilde{\mathbf{w}}(S') = \left(\widetilde{\mathbf{u_1}}(S), \widetilde{\mathbf{v_1}}(\overline{S}) + \widetilde{v}_{1,i+1}, \widetilde{\mathbf{u_2}}(S), \widetilde{\mathbf{v_2}}(\overline{S}) + \widetilde{v}_{2,i+1}, \ldots\right).$$

In the seconde case, the vector

$$\widetilde{\mathbf{w}}(S') = \left(\widetilde{\mathbf{u_1}}(S) + \widetilde{u}_{1,i+1}, \widetilde{\mathbf{v_1}}(\overline{S}), \widetilde{\mathbf{u_2}}(S) + \widetilde{u}_{2,i+1}, \widetilde{\mathbf{v_2}}(\overline{S}), \ldots\right).$$

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 a constant. Now we show the error is at most $\frac{n^2}{\sqrt{k+1}}$ as claimed. Since $\mathbf{u_i}, \mathbf{v_i}$ are unit vectors, $|\mathbf{u_i}(S)|, \left|\mathbf{v_i}(\overline{S})\right| \leq \sqrt{n}$ (By Cauchy's Inequality). Also $|\widetilde{\mathbf{u_i}}(S) - \mathbf{u_i}(S)| \leq \frac{n}{nk^2} = \frac{1}{k^2}$ (By measuring the error in every coordinate after rounding) and similarly for $\mathbf{v_i}$. To bound the error, we use an elementary fact: if a, 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

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

Using this, we get that

$$\left|\sum_{i=1}^{k} \sigma_i \widetilde{\mathbf{u_i}}(S)\widetilde{\mathbf{v_i}}(\overline{S}) - \sum_{i=1}^{k} \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(\overline{S})\right| \leq \frac{3k\sigma_1\sqrt{n}}{k^2} \leq \frac{3n^{\frac{3}{2}}}{k},$$

and this meets the claimed error bound.

Note. By measuring every

$$\left|\widetilde{\mathbf{u_i}}(S)\widetilde{\mathbf{v_i}}(\overline{S}) - \mathbf{u_i}(S)\mathbf{v_i}(\overline{S})\right|,$$

we know it is $\leq 3 \cdot \sqrt{n} \cdot \frac{1}{k^2}$, and adding every term for each i we can get the desired inequality.

In the last $\leq$ sign, we need $\sigma_1 \leq n$, this can be proved with Lemma 1.2.1. By this, we know

$$\sigma_1 \leq \|A\|_F,$$

and also

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2} \leq \sqrt{n^2} = n$$

since A is an adjacency matrix with n^2 entries.

Next, we show that the running time is polynomially bound. $|\widetilde{\mathbf{u_i}}(S)|, |\widetilde{\mathbf{v_i}}(S)| \leq 2\sqrt{n}$. Since $\widetilde{\mathbf{u_i}}(S), \widetilde{\mathbf{v_i}}(\overline{S})$ are all integer multiples of $\frac{1}{nk^2}$, there are at most $4n\sqrt{n}k^2$ possible values[a] of $\widetilde{\mathbf{u_i}}(S), \mathbf{v_i}(\overline{S})$ from which it follows that the list of $\widetilde{\mathbf{w}}(S)$ never gets larger than $\left(4n\sqrt{n}k^2\right)^{2k} = 4^{2k}n^{3k}k^{4k}$ which for fixed k is polynomially bounded. ■

[a] Since it may be positive or negative and thus between $-2\sqrt{n}$ and $2\sqrt{n}$.

We summarized what we have accomplished.

Theorem 1.7.1. 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 polynomial time of n for any fixed k.

It could be quite a surprise to have an algorithm that actually achieves the same accuracy in time polunomial in n and k because it would give an exact max cut in polynomial time (when k grows large, the error will approach 0).

1.7.4 SVD as a Compression Algorithm

Suppose A is the pixel intensity matrix of a large image. The entry a_{ij} gives the intensity of the ij-th pixel. If A is $n \times n$, the transmission of A requires transmitting $O\left(n^2\right)$ real numbers. Instead, one could send A_k, that is, the top k singular values $\sigma_1, \sigma_2, \ldots, \sigma_k$ along with the left and right singular vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$ and $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$. This would require sending $O(kn)$ real numbers instead of $O(n^2)$ real

numbers. If k is much smaller than n, this results in savings. For many images, a k much smaller than n can be used to reconstruct the image procided that a very low resolution version of the image is sufficient. Thus, one could use SVD as a compression method.

It turns out that in a more sophiscated approach, for certain classes of pictures one could use a fixed basis so that the top hundred singular vectors are sufficient to represent any picture approximately . This means that the space spanned by the top hundred singular vectors is not too different from the space spanned by the top two hundred singular vectors of a given matrix in that class. For example, Human face pictures are all similar, so the top first hundred singular vectors may not differ too much for different human face pictures and thus we can pick these singular vectors as a standard basis. Compressing these matrices by this standard basis can save substantially since the standard basis is transmitted only once and a matrix is transmitted by sending the top several hundred singular values for the standard basis (no need to send singular vectors again).

Figure 1.4: Rank = 1

Figure 1.5: Rank = 48

Figure 1.6: Rank = 100

Figure 1.7: Rank = 723 (Original)

Figure 1.8: SVD compressed image

We can see that low rank pictures have more noise.

Listing 1.1: Source Code of SVD Compression

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import ipywidgets as widgets
from IPython.display import display, clear_output

def load_color_image(path):
    img_bgr = cv2.imread(path)
    img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
    return img_rgb.astype(float) / 255.0

def svd_compress_channel(channel, k):
    U, S, VT = np.linalg.svd(channel, full_matrices=False)
    S_k = np.diag(S[:k])
    U_k = U[:, :k]
    VT_k = VT[:k, :]
    compressed = U_k @ S_k @ VT_k
    return np.clip(compressed, 0, 1)

def compress_color_image(img_rgb, k):
    channels = [svd_compress_channel(img_rgb[:, :, i], k) for i in range(3)]
    return np.stack(channels, axis=2)

```

```python
def plot_two_images(original, compressed, rank):
    fig, axes = plt.subplots(1, 2, figsize=(12, 6))
    axes[0].imshow(original)
    axes[0].set_title('Original Image')
    axes[0].axis('off')

    axes[1].imshow(compressed)
    axes[1].set_title(f'Compressed Image (rank={rank})')
    axes[1].axis('off')

    plt.tight_layout()
    # 不用 plt.show()，return fig
    return fig

def save_image(img_rgb, rank):
    filename = f"rank_{rank:03d}.jpg"
    img_uint8 = (np.clip(img_rgb, 0, 1) * 255).astype(np.uint8)
    img_bgr = cv2.cvtColor(img_uint8, cv2.COLOR_RGB2BGR)
    cv2.imwrite(filename, img_bgr)
    print(f"Saved compressed image as {filename}")

def interactive_svd_compression(image_path):
    img_rgb = load_color_image(image_path)
    max_rank = min(img_rgb.shape[0], img_rgb.shape[1])

    slider = widgets.IntSlider(value=10, min=1, max=max_rank, step=1, description='Rank:')
    out = widgets.Output()
    save_button = widgets.Button(description="Save Compressed Image")

    def update(change):
        k = change['new']
        compressed_img = compress_color_image(img_rgb, k)
        with out:
            clear_output(wait=True)
            fig = plot_two_images(img_rgb, compressed_img, k)
            display(fig)
        save_button.compressed_img = compressed_img
        save_button.current_rank = k

    def on_save_clicked(b):
        if hasattr(save_button, 'compressed_img') and hasattr(save_button, 'current_rank'):
            save_image(save_button.compressed_img, save_button.current_rank)
        else:
            print("No compressed image to save yet.")

    slider.observe(update, names='value')
    save_button.on_click(on_save_clicked)

    display(slider, save_button, out)

    # 預先觸發一次
    slider.value = slider.value

# 使用你的圖片檔案名
interactive_svd_compression("argue.jpg")
```

1.7.5 Singular Vectors and ranking Documents

Chapter 2

Learning and VC-dimension

2.1 Learning

Learning algorithms are general purpose tools that solve problems often without detailed domain-specific knowledge (領域専業知識). They have proven to be very effective in a large number of contexts. We start with an example. Suppose one wants an algorithm to distinguish among different types of motor vehicles such as cars, trucks, and tractors. Using domain knowledge one can create a set of features. Some examples of featurs are number of wheels, the power of the engine, the number of doors, and the length of vehicle. If there are d features, each object can be represented as a d-dimensional vector, calledf the feature vector, with each component of vector giving the value of one feature such as engine power, the number of doors, etc. Using domain knowledge one develops a set of rules to distinguish among different types of objects. A learning algorithm uses the feature vector representation of an object, which is assumed to be given by the domain "expert". Instead of having the domain expert develop a set of rules, the algorithm only asks the expert, called the teacher, to label each vector. In this example, the labels would be "car", "truck", and "tractor". The algorithm's task is to develop a set of rules that applied to the given vectors gives the same labels as the teacher. Each feature vector is called an example and the given set of labeled examples is called the *training set*. The task of the learner is to output a set of rules that correctly labels all training examples. Of course, for this limited task, one could output the rule "for each training example, use the teacher's label". But we insist on Occam's razor principle that states that the rules output by the algorithm must be more succint than the table of all training examples. This is akin to developing a specific theory to explain extensive observations. The theory must be more succinct than just a list of observations.

However, the general task of interest is not to be correct just on the training example, but have learnt rules correctly predict the labels of future examples. Intuitively, if the classifier is trained on sufficiently many training example, then it seems likely that it would work well on the space of all examples. We will see later that the theory of Vapnik-Chervonenkis dimension (VC-dimension) confirms this intuition. In general, optimization technieques such as linear and convex programming, play an important role in learning algorithms. Throughout this chapter, we assume all the labels are binary. It is not difficult to see that the general problem of classifying into one of several types can be reduced to binary classification. Classifying into car or non-car, tractor or non-tractor, etc. will pin down the type of vehicle. So the teacher's labels are assumed to be $+1$ or -1. For an illustration, see Figure 2.1, where, examples are in 2-dimensions (there are two features) where those labeled -1 are unfilled circles and those labeled $+1$ are filled circles. The right hand picture illustrates a rule that the algorithm could come up with: the examples above the thick line are -1 and those below are $+1$.

The simplest rule in d-dimensional space is generalization of a line in the plane, namely, a half space. Does a weighted sum of feature values exceed a threshold? Such a rule may be thought of as being implemented by a threshold gate that tackes feature values as inputs, computes their weighted sum and output yes or no depending whether or not the sum is greater than the threshold. One could also look at a network of interconnected thershold gates called a neural net. Threshold gates are sometimes called perceptrons (感知器) since one model of human perception is that it is done by a neural net in the brain.

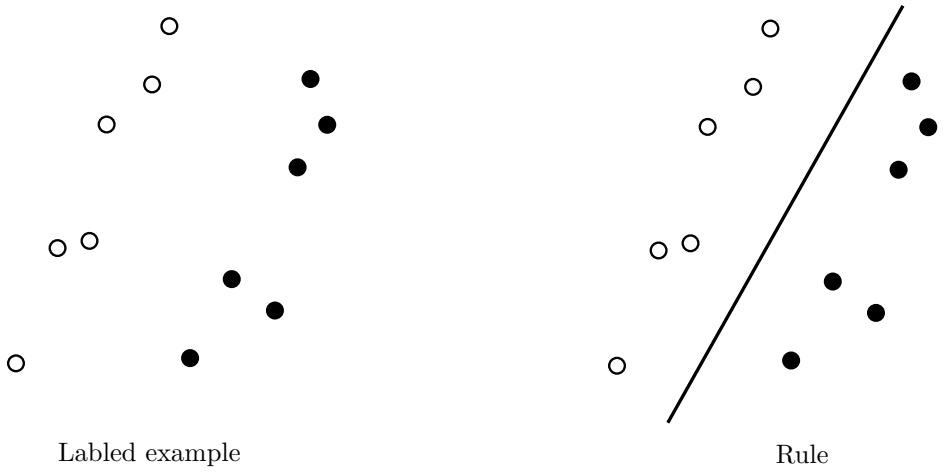

Figure 2.1: Training set and the rule that is learnt

2.2 Linear Separators, the Perception Algorithm, and Margins

The problem of learning a half-space or a linear seperator consists of n labeled examples $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}$ in d-dimensional space. The task is to find a d-dimensional vector $\mathbf{w}$, if one exists, and a threshold b such that

$$\begin{aligned} \mathbf{w} \cdot \mathbf{a_i} &> b \text{ for each } \mathbf{a_i} \text{ labelled } +1, \\ \mathbf{w} \cdot \mathbf{a_i} &< b \text{ for each } \mathbf{a_i} \text{ labelled } -1. \end{aligned} \tag{2.1}$$

Definition 2.2.1 (linear separator)**.** A vector-threshold pair, $(\mathbf{w}, b)$, satisfying the Equation 2.1 is called a *linear separator*.

The above formulation is a linear program (LP) [1] in the unknowns $\mathbf{w}$ and b that can be solved by a general purpose LP algorithm. Linear programming is solvable in polynomial time but a simpler algorithm called the *perceptron learning algorithm* can be much faster when there is a feasible solution $\mathbf{w}$ with a lot of wiggle room or margin [2], though it is not polynomial time bounded in general.

We begin by adding an extra coordinate to each $\mathbf{a_i}$ and $\mathbf{w}$, writing $\hat{\mathbf{a}}_\mathbf{i} = (\mathbf{a_i}, 1)$ and $\hat{\mathbf{w}} = (\mathbf{w}, -b)$. Suppose l_i is the ± 1 label on $\hat{\mathbf{a}}_\mathbf{i}$. Then, the inequalities in Equation 2.1 can be rewritten as

$$(\hat{\mathbf{w}} \cdot \hat{\mathbf{a}}_\mathbf{i})\, l_i > 0 \quad 1 \le i \le n.$$

Since the right hand side is zero, we may scale $\hat{\mathbf{a}}_\mathbf{i}$ so that $|\hat{\mathbf{a}}_\mathbf{i}| = 1$. Adding the extra coordinate increased the dimension by one but now the separator contains the origin. For simplicity of notation, in the rest of this section, we drop the hats and let $\mathbf{a_i}$ and $\mathbf{w}$ stand for the corresponding $\hat{\mathbf{a}}_\mathbf{i}$ and $\hat{\mathbf{w}}$.

Note. Actually we can more specifically give another definition of linear separator as a hyperplane:

[1] See Appendix B.2.
[2] That is, datas in different categories are not too close and thus can be separated easier.

Definition 2.2.2. A linear separator is a hyperplane

$$\mathbf{w} \cdot \mathbf{x} = b,$$

and it is equivalent to what we describe it above, as a tuple $(\mathbf{w}, b)$.

and thus we say the new linear separator contains the origin after adding a new dimension since new hyperpplane is $\mathbf{w} \cdot \mathbf{x} = 0$.

2.2.1 The perceptron learning algorithm

The perceptron learning algorithm is simple and elegant. We wish to find a solution $\mathbf{w}$ to

$$(\mathbf{w} \cdot \mathbf{a_i})\, l_i > 0 \quad 1 \leq i \leq n \tag{2.2}$$

where $|\mathbf{a_i}| = 1$. Starting with $\mathbf{w} = l_1 \mathbf{a_1}$, pick any $\mathbf{a_i}$ with $(\mathbf{w} \cdot \mathbf{a_i})\, l_i \leq 0$, and replace $\mathbf{w}$ by $\mathbf{w} + l_i \mathbf{a_i}$. Repeat until $(\mathbf{w} \cdot \mathbf{a_i}) l_i > 0$ for all i.

The intuition behind the algorithm is that correcting $\mathbf{w}$ by adding $\mathbf{a_i} l_i$ causes the new $(\mathbf{w} \cdot \mathbf{a_i})\, l_i$ to be higher by $\mathbf{a_i} \cdot \mathbf{a_i} l_i^2 = |\mathbf{a_i}|^2$. This is good for this $\mathbf{a_i}$. But this change may be bad for other $\mathbf{a_j}$. The proof below show that this is very simple process quickly yields a solution $\mathbf{w}$ provided there exists a solution with a good margin.

Definition 2.2.3 (margin). For a solution $\mathbf{w}$ to Equation 2.2, where $|\mathbf{a_i}| = 1$ for all examples, the margin is defined to be the minimum distance of the hyperplane $\{\mathbf{x} \mid \mathbf{w} \cdot \mathbf{x} = 0\}$ to any $\mathbf{a_i}$, namely, $\min_i \frac{(\mathbf{w} \cdot \mathbf{a_i}) l_i}{|\mathbf{w}|}$ [a].

[a] See Appendix A.7

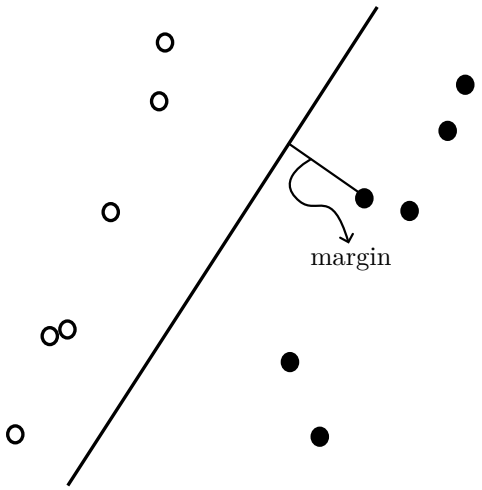

Figure 2.2: Margin of a linear separator

If we did not require that all $|\mathbf{a_i}| = 1$ in Equation 2.2, then one could artificially increase the margin by scailing up the $\mathbf{a_i}$. If we did not divide by $|\mathbf{w}|$ in the definition of margin, then again, one could artificially increase the margin by scailing $\mathbf{w}$ up. The interesting thing is that the number of steps of the algorithm depends only upon the best margin any solution can achieve, not upon n and d. In practice, the perception learning algorithm works well.

Note. We require $|\mathbf{a_i}| = 1$ and dividing $|\mathbf{w}|$ is because the algorithm relies on the margin of the datas, so we need to prevent some meaningless zooming on the margin, which do not help to identify the pattern. That is, we need the stable geometric meaning and thus it represents the

speed of convergence of this algortihm.

Theorem 2.2.1. Suppose there is a solution $\mathbf{w}^*$ to Equation 2.2 with margin $\delta > 0$. Then, the perceptron learning algorithm finds some solution $\mathbf{w}$ with $(\mathbf{w} \cdot \mathbf{a_i})\, l_i > 0$ for all i in at most $\frac{1}{\delta^2} - 1$ iterations.

Proof. Without loss of generality by scaling $\mathbf{w}^*$ so that $|\mathbf{w}^*| = 1$. Consider the cosine of the angle between the current vector $\mathbf{w}$ and $\mathbf{w}^*$, that is, that is, $\frac{\mathbf{w}^t \mathbf{w}^*}{|\mathbf{w}|}$. In each step of the algorithm, the numerator of this fraction increase by at least δ because

$$(\mathbf{w} + \mathbf{a_i} l_i) \cdot \mathbf{w}^* = \mathbf{w} \cdot \mathbf{w}^* + l_i \mathbf{a_i} \cdot \mathbf{w}^* \geq \mathbf{w}^t \mathbf{w}^* + \delta.$$

On the other hand, the square of the denominator increases by at most one since

$$|\mathbf{w} + \mathbf{a_i} l_i|^2 = (\mathbf{w} + \mathbf{a_i} l_i)\,(\mathbf{w} + \mathbf{a_i} l_i) = |\mathbf{w}|^2 + 2\,(\mathbf{w} \cdot \mathbf{a_i})\, l_i + |\mathbf{a_i}|^2 l_i^2 \leq |\mathbf{w}|^2 + 1$$

where, since $\mathbf{w}^t \mathbf{a_i} l_i \leq 0$, the cross term is non-positive.

Note. We only modifty $\mathbf{w}$ by adding $\mathbf{a_i} l_i$ when $(\mathbf{w} \cdot \mathbf{a_i}) l_i \leq 0$, so the cross term is non-positive.

After t iterations, $\mathbf{w} \cdot \mathbf{w}^* \geq (t+1)\delta$ since at the start $\mathbf{w} \cdot \mathbf{w}^* = l_1\,(\mathbf{a_1} \cdot \mathbf{w}^*) \geq \delta$ and at each iteration $\mathbf{w}^t \mathbf{w}^*$ increases by at least δ. Similarly after t iterations $|\mathbf{w}|^2 \leq t+1$ since at the start $|\mathbf{w}| = |\mathbf{a_1}| \leq 1$ and at each iteration $|\mathbf{w}|^2$ increases by at most 1. Thus the cosine of the angle between $\mathbf{w}$ and $\mathbf{w}^*$ is at least $\frac{(t+1)\delta}{\sqrt{t+1}}$ and the cosine cannot exceed one. Therefore, the algorithm must stop before $\frac{1}{\delta^2} - 1$ iterations and at termination, $(\mathbf{w} \cdot \mathbf{a_i})\, l_i > 0$ for all i. This proves the theorem. ■

How strong is the assumption that there is a separator with margin at least δ? Suppose for the moment, the $\mathbf{a_i}$ are picked from the uniform density on the surface of the unit hypersphere. There is a result that states that for any fixed hyperplane passing through the origin, most of the mass of the unit sphere is within distance $O\left(\frac{1}{\sqrt{d}}\right)$ of the hyperplane. So, the probablity of one fixed hyperplane having a margin more than $\frac{c}{\sqrt{d}}$ is low. But this does not mean that there is no hyperplane with a larger margin. By the union bound, one can only assert that the probability of some hyperplane having a large margin is at most the probability of a specific one having a large margin times the number of hyperplanes, which is infinite. Later we will see using VC-dimension arguments that indeed the probability of some hyperplane having a large margin is low if the examples are selected at random from the hypersphere. So the assumption of large margin separators existing may not be valid for the simplest random models. But intuitively, if what is to be learnt, like whether something is a car, is not very hard, then, with enough features in the model, there will not be many "near cars" that could be confused with cars nor many "near non-cars". In a real problem such as this, uniform density is not a valid assumption. In this case, there should be a large margin separator and the theorem would work.

The question arises as to how small margins can be. Suppose the examples $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}$ were vectors with d coordinates, each coordinate a 0 or 1 and the decision rule for labeling the examples was the following.

If the first 1 coordinate of the example is odd, label the example +1.
If the first 1 coordinate of the example is even, label the example −1.

This rule can be represented by the decision rule

$$(a_{i1}, a_{i2}, \ldots, a_{in}) \left(1, -\frac{1}{2}, \frac{1}{4}, -\frac{1}{8}, \ldots\right)^t = a_{i1} - \frac{1}{2} a_{i2} + \frac{1}{4} a_{i3} - \frac{1}{8} a_{i4} + \cdots > 0.$$

However, the margin in this example can be exponentially small. Indeed, if for an example $\mathbf{a}$, the first $\frac{d}{10}$ coordinates are all 0s, then the margin is $O\left(2^{-\frac{d}{10}}\right)$.

2.2.2 Maximizing the Margin

In this section, we present an algorithm to find the maximum separator. The margin of a solution $\mathbf{w}$ to $(\mathbf{w}^t \mathbf{a_i})\, l_i > 0$, $1 \leq i \leq n$, where $|\mathbf{a_i}| = 1$ is $\delta = \min_i \frac{l_i (\mathbf{w}^t \mathbf{a_i})}{|\mathbf{w}|}$. Since this is not a concave function of $\mathbf{w}$, it is difficult to deal with computationally.

Convex optimization techniques in general can only handle the maximization of concave functions or the minimization of convex functions over convex sets. However, by modifying the weight vector, one can convert the optimization problem to one with a concave objective function. Note that

$$l_i\left(\frac{\mathbf{w}^t\mathbf{a_i}}{|\mathbf{w}|\delta}\right) \geq 1$$

for all $\mathbf{a_i}$. Let $\mathbf{v} = \frac{\mathbf{w}}{\delta|\mathbf{w}|}$ be the modified weight vector. The maximizing δ is equivalent to minimizeing $|\mathbf{v}|$. So the optimization problem is

$$\text{minimize } |\mathbf{v}| \text{ subject to } l_i\left(\mathbf{v}^t\mathbf{a_i}\right) \geq 1, \ \forall i.$$

Although $|\mathbf{v}|$ is a convex function of the coordinates of $\mathbf{v}$, a better convex function to minimize is $|\mathbf{v}|^2$ since $|\mathbf{v}|^2$ is differentiable. So we reformulate the problem as:

Maximum Margin Problem

$$\text{minimize } |\mathbf{v}|^2 \text{ subject to } l_i\left(\mathbf{v}^t\mathbf{a_i}\right) \geq 1, \ \forall i.$$

Note. Notice that if we find such $\mathbf{v}$, then there must be some $\mathbf{a_i}$ such that $l_i\left(\mathbf{v}^t\mathbf{a_i}\right) = 1$, otherwise, if for all i we have $l_i\left(\mathbf{v}^t\mathbf{a_i}\right) > 1$, then we can scale down $\mathbf{v}$ and still satisfy all the inequalities, but this means we find a smaller $\mathbf{v}$, which is a contradiction. Now if we know $l_i\left(\mathbf{v}^t\mathbf{a_i}\right) = 1$, then we know

$$\frac{l_i\left(\mathbf{v}^t\mathbf{a_i}\right)}{|\mathbf{v}|} = \frac{1}{|\mathbf{v}|},$$

which means the margin must be $\frac{1}{|v|}$ since for all j we have $l_j\left(\mathbf{v}^t\mathbf{a_j}\right) \geq 1$.

This convex optimization problem has been much studied and algorithms that use the special structure of this problem solve it more efficiently than general convex optimization methods. We do not discuss these improvents here. An optimal solution $\mathbf{v}$ to this problem has the follwing property. Let V be the space spanned by the examples $\mathbf{a_i}$ for which there is equality, namely for which $l_i\left(\mathbf{v}^t\mathbf{a_i}\right) = 1$. We claim that $\mathbf{v}$ lies in V. If not, $\mathbf{v}$ has a component orthogonal to V. Reducing this component infinitesimally does not violate any inequality, since, we are moving orthogonally to the exactly satisfied constraints; but it does decrease $|\mathbf{v}|$ contradicting the optimality. If V is full dimensional, then there are d ($\mathbf{v}, \mathbf{a_i}$ are d-dimensional vectors.) independent examples for which inequality holds. These d equations then have a unique solution $\mathbf{v}$ must be that solution. These examples are then called the *suppoer vectors*. The d support vectors determine uniquely the maximum margin separator.

Note. If V is full dimensional, then we have d equations like

$$l_i\left(\mathbf{v}^t\mathbf{a_i}\right) = 1,$$

and since $\mathbf{v}$ is in the space spanned by these d vectors, and thus $\mathbf{v} = \sum_{j=1}^{d} c_j\mathbf{a_j}$. Hence, we have

$$l_i\left(\sum_{j=1}^{d} c_j{\mathbf{a_j}}^t\mathbf{a_i}\right) = 1 \quad \forall 1 \leq i \leq d.$$

Now if we deinfe a $d \times d$ matrix M by $M_{ij} = l_i\left({\mathbf{a_j}}^t\mathbf{a_i}\right)$ and $\mathbf{c} = \begin{pmatrix}\mathbf{c_1} & \mathbf{c_2} & \dots & \mathbf{c_d}\end{pmatrix}^t$, then we have

$$M\mathbf{c} = \mathbf{1},$$

where $\mathbf{1}$ is a column vector whose every component is 1. Now we show that M is invertible and thus $\mathbf{c}$ is unique, which means $\mathbf{v}$ is unique. Since $M = DG$, where D is a diagonal matrix where $\operatorname{diag} D = \begin{pmatrix}l_1 & l_2 & \dots & l_d\end{pmatrix}$ and G is a gram matrix where $G_{ij} = {\mathbf{a_j}}^t\mathbf{a_i}$. Now we show that D, G are both invertible, and thus M is invertible. First, since D is diagonal, and thus it is invertible. Now

notice that G is positive defnite since G is a gram matrix, and thus if $\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_d \end{pmatrix}^t$, then

$$\mathbf{x}^t G \mathbf{x} = \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j \mathbf{a_j}^t \mathbf{a_i} = \left\| \sum_{i=1}^{d} x_i \mathbf{a_i} \right\|^2$$

and if $\mathbf{x} \neq \mathbf{0}$, then the above equation is > 0 since $\{\mathbf{a_i}\}_{i=1}^{d}$ is linearly independent. Now we have a claim:

Claim. If G is a positive definite matrix, then it is invertible.

Proof. Suppose G is not invertible, then there exists $\mathbf{v} \neq \mathbf{0}$ such that $G\mathbf{v} = 0$ (If invertible, then $\ker G = \{0\}$). However, this means $\mathbf{v}^t G \mathbf{v} = 0$ for some $\mathbf{v} \neq 0$, which means G is not positive definite, a contradiction. By this, we know G is invertible, and we're done. ⊛

Definition 2.2.4. Suppose we have a training data set $\{(\mathbf{a_i}, l_i)\}_{i=1}^{n}$, where $\mathbf{a_i}$ is the characteristic vector and $|\mathbf{a_i}| = 1$, and $l_i \in \{+1, -1\}$ is the corresponding label to $\mathbf{a_i}$, then if we want to solve

$$\min_{\mathbf{v}} |\mathbf{v}|^2 \text{ subject to } l_i \left(\mathbf{v}^t \mathbf{a_i}\right) \geq 1, \ \forall i,$$

and suppose $\mathbf{v}$ is the best separator (the one with maximal margin), then *support vectors* are

$$\left\{ \mathbf{a_i} \mid l_i \left(\mathbf{v}^t \mathbf{a_i}\right) = 1 \right\},$$

which are the vectors most close to the separator hyperplane.

2.2.3 Linear Separators that classify most examples correctly

It may happen that there are linear separators for which almost all but a small fraction of examples are on the correct side. Going back to Equation 2.2 ask if there is a $\mathbf{w}$ for which at least $(1 - \varepsilon)n$ of the n inequalities in Equation 2.2 are satisfied. Unfortunately, such problems are NP-hard and there are no good algorithms to solve them. A good way to think about this is that we suffer a "loss" of one for each misclassified point and would like to minimize the loss. But this loss function is discontinuous, it goes from 0 to 1 abruptly. However, with a nicer loss function it is possible to solve the problem. One possibility is to introduce slack variables y_i, $i = 1, 2, \dots, n$, where y_i measures how badly the example $\mathbf{a_i}$ is classified. We then include the slack variables in the objective function to be minimized:

$$\begin{aligned} &\text{minimize } |\mathbf{v}|^2 + c \sum_{i=1}^{n} y_i \\ &\text{subject to } \left.\begin{matrix} (\mathbf{v} \cdot \mathbf{a_i}) \, l_i \geq 1 - y_i \\ y_i \geq 0 \end{matrix}\right\} i = 1, 2, \dots, n \end{aligned} \tag{2.3}$$

Note. $\{y_i\}_{i=1}^{n}$ is a set of parameters that will be computed by the algorithm, and we needs $y_i \geq 0$ since we want $1 - y_i < 1$, which makes the classification not too rigorous. We do this because sometimes the training datas cannot be fully classified by a linear separators.

If for some i, $l_i \left(\mathbf{v} \cdot \mathbf{a_i}\right) \leq 1$, then set y_i to its lowest value, namely 0, since each y_i has positive coefficient in the coset function. If, however, $l_i \left(\mathbf{v} \cdot \mathbf{a_i}\right) < 1$, then set $y_i = 1 - l_i \left(\mathbf{v} \cdot \mathbf{a_i}\right)$, so y_i is just the amount of violation of this inequality. Thus, the objective function is trying to minimize a combination of the total violation as well as 1/margin (the squared length of $\mathbf{v}$). It is easy to see that this is the same as minimizing

$$|\mathbf{v}|^2 + c \sum_{i} (1 - l_i \left(\mathbf{v} \cdot \mathbf{a_i}\right))^{+}{}^{3},$$

subject to the constraints. The second term is the sum of the violations.

[3] $x^+ = \begin{cases} 0, & \text{if } x \leq 0; \\ x, & \text{otherwise.} \end{cases}$

2.3 Nonlinear Separators, Support Vector Machines, and Kernels

Appendix

Appendix A

Additional Proofs

A.1 A claim in the proof of Theorem 1.2.1

Claim. If V is a finite-dimensional vector space, and $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k-1}}$ are all vectors in V. If W is a k-dimensional subspace of V, then there exists some vectors v in W such that v is perpendicular to $\mathbf{v_i}$ for all $1 \leq i \leq k-1$.

Proof. Suppose $U = \operatorname{span}\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k-1}}\}$, then define $U^\perp$ as

$$U^\perp = \{v \in V \mid \langle v, u\rangle = 0 \ \forall u \in U\}.$$

Now we want to show $W \cap U^\perp \neq \{0\}$. We can notice that $\dim W = k$ and $\dim U^\perp = \dim V - \dim U = \dim V - (k-1)$ since orthogonal set is linearly independent. Hence,

$$\dim\left(W \cap U^\perp\right) \geq \dim W + \dim U^\perp - \dim V = k + (\dim V - (k-1)) - \dim V = 1.$$

⊛

A.2 A_k has rank k

Claim. If A is a $n \times d$ matrix, and

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

where $\mathbf{u_i}, \mathbf{v_i}$ are the left, right singular vectors, then A_k has rank k.

Proof. First notice that

$$A_k = \begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \cdots & \mathbf{u_k} \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{pmatrix} \begin{pmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_k} \end{pmatrix}.$$

Since $\mathbf{u_i}$'s and $\mathbf{v_i}$'s are orthogonal and hence linearly independent, so we have

$$\operatorname{rank}\begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \cdots & \mathbf{u_k} \end{pmatrix} = \operatorname{rank}\begin{pmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_k} \end{pmatrix} = k.$$

Besides, since

$$\begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{pmatrix}$$

is diagonal, and hence it also has rank k.

Now we need two claims to prove that A_k is of rank k.

Claim. If X is an $n \times k$ matrix and $\operatorname{rank} X = k$ (full column rank), then $\operatorname{rank}(XY) = \operatorname{rank} Y$ for all Y.

Proof. Notice that $\ker X = \{0\}$ by rank and nullity theorem. Besides, if $v \in \ker XY$, then

$$X(Yv) = 0.$$

However, this means $Yv = 0$ due to the above argument, so $v \in \ker Y$, which means $\ker XY \subseteq \ker Y$. Also, if $w \in \ker Y$, then $XYw = X0 = 0$, so $\ker Y \subseteq \ker XY$. Thus, $\ker XY = \ker Y$. Now by rank and nullity theorem we know $\operatorname{rank} XY = \operatorname{rank} Y$. ⊛

Claim. If Z is a $k \times n$ matrix and $\operatorname{rank} Z = k$ (full row rank), then $\operatorname{rank} YZ = \operatorname{rank} Y$ for all Y.

Proof. Notice that

$$\operatorname{rank} YZ = \operatorname{rank} Z^t Y^t = \operatorname{rank} Y^t = \operatorname{rank} Y.$$

by the first claim, and so we're done. ⊛

Now by these two claims, it is easy to see that $\operatorname{rank} A_k$ is equal to the rank of the middle diagonal matrix, which is of rank k, so we're done. ⊛

A.3 Symmetric matrices have same left and right singular vectors.

Claim. Suppose A is a square symmetric matrix, then A's left, right singular vectors are identical.

Proof. First notice that all right singular vectors are A^tA's eigenvectors, and all left singular vectors are AA^t's eigenvectors. Now since A is symmetric, so $AA^t = A^2 = A^tA$. Notice that

$$VD^2V^t = VDU^tUDV^t = A^tA = A^2 = AA^t = UDV^tVDU^t = UD^2U^t.$$

Hence,

$$(U - V)D^2(U - V)^t = 0,$$

which means

$$\sum_{i=1}^{r} \sigma_i^2 (u_i - v_i)(u_i - v_i)^t = 0.$$

However,

$$(u_i - v_i)(u_i - v_i)^t$$

is positive semi-definite since for any x we have

$$x^t(u_i - v_i)(u_i - v_i)^t x = \left\|(u_i - v_i)^t x\right\|^2 \geq 0$$

and by this, we know $u_i = v_i$ for every i. ⊛

A.4 Outer product on same vector gives rank one matrix

Definition A.4.1. For two vectors u and v, we define the matrix uv^t as the *outer product* of u and v.

Claim. For any column vector $v \in \mathbb{R}^n$, the matrix $vv^t \in \mathbb{R}^{n\times n}$ is symmetric and of rank one (unless $v = 0$, in which case its rank is zero).

Proof. Since $(vv^t)^t = vv^t$, so it is symmetric. For any $x \in \mathbb{R}^n$, we compute:

$$vv^t x = v(v^t x) = (v^t x)v \in \text{span}(v)$$

Hence, the image of the linear transformation defined by vv^t is contained in $\text{span}(v)$, which is a one-dimensional subspace (unless $v = 0$). Thus, vv^t is at most rank one. Since $vv^t \neq 0$ when $v \neq 0$, its rank is exactly one. ⊛

A.5 Dividing Frobenius norm

Claim. If we have $\sigma_1 > \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_r$, and supose A is square and symmetric (and hence has same left and right singular vectors) , then

$$\frac{A^k}{\|A^k\|_F}$$

will converge to $\mathbf{v_1}\mathbf{v_1}^t$ as k growing large.

Proof. First notice that

$$A^k = \sum_{i=1}^{r} \sigma_i^k \mathbf{v_i}\mathbf{v_i}^t.$$

Thus,

$$\|A^k\|_F^2 = \left\|\sum_{i=1}^{r} \sigma_i^k \mathbf{v_i}\mathbf{v_i}^t\right\|_F^2 = \sum_{i=1}^{r} \sigma_i^{2k},$$

and we have

$$\frac{A^k}{\|A^k\|_F} = \frac{\sum_{i=1}^{r} \sigma_i^k \mathbf{v_i}\mathbf{v_i}^t}{\left(\sum_{i=1}^{r} \sigma_i^{2k}\right)^{\frac{1}{2}}} = \sum_{i=1}^{r} \frac{\sigma_i^k}{\left(\sum_{i=1}^{r} \sigma_i^{2k}\right)^{\frac{1}{2}}} \mathbf{v_i}\mathbf{v_i}^t.$$

Since we know

$$\begin{cases} \dfrac{\sigma_i^k}{\left(\sum_{i=1}^{r} \sigma_i^{2k}\right)^{\frac{1}{2}}} \to 1, & \text{if } i = 1; \\ \dfrac{\sigma_i^k}{\left(\sum_{i=1}^{r} \sigma_i^{2k}\right)^{\frac{1}{2}}} \to 0, & \text{otherwise.} \end{cases}$$

so it converges to $\mathbf{v_1}\mathbf{v_1}^t$. ⊛

A.6 Why optimization problem subsumes set partition problem?

We talk about the case that A is rank one and left, right singular vectors are same. Hence, we can suppose $A = \mathbf{a}\mathbf{a}^t$ for some column vector $\mathbf{a}$ (Use spectral decomposition and since A is positive semi-definite so the eigenvalue is positive and thus we can merge the scalar into $\mathbf{a}$).

By this, we know

$$\begin{aligned}\mathbf{x}^t A(\mathbf{1}-\mathbf{x}) &= \mathbf{x}^t\mathbf{a}\mathbf{a}^t(\mathbf{1}-\mathbf{x}) = \left(\mathbf{a}^t\mathbf{x}\right)\cdot\left(\mathbf{a}^t(\mathbf{1}-\mathbf{x})\right)\\ &= \left(\sum_{i=1}^{n} a_i x_i\right)\left(\sum_{j=1}^{n} a_j(1-x_j)\right) = \left(\sum_{i:x_i=1} a_i\right)\left(\sum_{j:x_j=0} a_j\right).\end{aligned}$$

Now we want to maximize this, and this occurs when

$$\left(\sum_{i=1}^{n} a_i x_i\right) \quad \text{is most close to} \quad \left(\sum_{j:x_j=0} a_j\right),$$

so it is a partition problem in disguise and thus a NP-hard problem.

A.7 Why we deine margin like this?

Here we talk about why the formula of the margin is

$$\min_i \frac{(\mathbf{w}\cdot\mathbf{a_i})\, l_i}{|\mathbf{w}|}.$$

Here, we need to know a theorem first.

Theorem A.7.1. In a hyperplane $\mathbf{w}\cdot\mathbf{x}=0$, the distance from a point $\mathbf{a_i}$ to this hyperplane is

$$\frac{|\mathbf{w}\cdot\mathbf{a_i}|}{|\mathbf{w}|}.$$

We can think of $\mathbf{w}$ is the normal vector of this hyperplane, and thus we can use $\mathbf{w}\cdot\mathbf{a_i}$ to find the length of the component of $\mathbf{a_i}$ along the direction of $\mathbf{w}$ multiplied by the length of $\mathbf{w}$.

In the context above, we use l_i to ensure $(\mathbf{w}\cdot\mathbf{a_i})$ is positive and thus we can drop the absolute value.

Corollary A.7.1. In a hyperplane $\mathbf{w}\cdot\mathbf{x}=b$, the distance from a point $\mathbf{a_i}$ to this hyperplane is

$$\frac{|\mathbf{w}\cdot\mathbf{a_i}-b|}{|\mathbf{w}|}.$$

Proof. Notice that moving the plane $\mathbf{w}\cdot\mathbf{x}=0$ to $\mathbf{w}\cdot\mathbf{x}=b$ is to move every point on the former hyperplane along the $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ direction by $\frac{b}{\|\mathbf{w}\|}$ units. This is because suppose $\mathbf{w}\cdot\mathbf{x_0}=0$, then

$$\mathbf{w}\cdot\left(\mathbf{x_0}+\frac{b}{\|\mathbf{w}\|}\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) = 0+b=b.$$

Hence, we can imagine this formula is correct. ■

Appendix B

Other Topics

B.1 Spectral Decomposition

Spectral decomposition is a way to express a real symmetric (or Hermitian, in complex case) matrix as a sum of projections onto its eigenvectors. It's a special case of diagonalization.

Here we just talk about when the field is $\mathbb{R}$ and for self-adjoint matrices, but for the complex case and Hermitian matrices the definition is similar.

Theorem B.1.1. If $A \in M_{n\times n}(\mathbb{R})$ and A is self-adjoint , then

$$A = Q\Lambda Q^t$$

for some orthogonal Q and diagonal Λ, and this is called the *spectral decomposition* or the *eigendecomposition* of A.

Now we can briefly talk about projections.

Definition B.1.1. If we say P is a projection on V, then it means $\operatorname{Im} P = V$.

Definition B.1.2. An operator P is called a *projection* if $P^2 = P$.

From geometric view, this means if V is the target space projected onto, then for every vector in V, its projection is itself. Since suppose $v \in V = \operatorname{Im} P$, then

$$Pv = P(Px) = P^2x = Px = v.$$

Definition B.1.3. An operator p is called an *orthogonal projection* if $P^2 = P$ and $P^t = P$.

The reason that P^t has to be equal to P is that if we see this in the geometric view, orthogonal projection means projecting a vector direcly onto the target space V , so if Px is the orthogonal projection from x to V , we want the property

$$x - Px \perp V.$$

Now let $x \in \mathbb{R}^n$, and $v = Px \in \operatorname{Im} P$, and then for any $y \in \operatorname{Im} P$ (and thus $py = y$), we know

$$\langle x - Px, y\rangle = \langle x, y\rangle - \langle v, y\rangle,$$

and since we know

$$\langle x, y\rangle = x^t y = x^t Py = \left(P^t x\right)^t y = \langle P^t x, y\rangle = \langle Px, y\rangle = \langle v, y\rangle$$

, so we can have $\langle x, y\rangle - \langle v, y\rangle = 0$, and thus $x - Px \perp V$.

Here we introduce another equivalent form of spectral decomposition.

Theorem B.1.2. Suppose A is real symmetric, then A can be written as

$$A = \sum_{i=1}^{n} \lambda_i \mathbf{q_i} \mathbf{q_i}^t$$

where $\{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_n}\}$ is an orthonal basis consisting of eigenvectors of A and $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A.

Proof. By Theorem B.1.1, if we pick $\{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_n}\}$ as the column vectors of Q, where $\mathbf{q_i}$ means the i-th column of Q, then we know

$$Q\Lambda Q^t = \begin{bmatrix} \lambda_1 \mathbf{q_1} & \lambda_2 \mathbf{q_2} & \ldots & \lambda_n \mathbf{q_n} \end{bmatrix} Q^t = \sum_{i=1}^{n} \lambda_i \mathbf{q_i} \mathbf{q_i}^t$$

since $\operatorname{diag} \Lambda = \begin{pmatrix} \lambda_1 & \lambda_2 & \ldots & \lambda_n \end{pmatrix}$. ■

Corollary B.1.1. For a real symmetric A, we know

$$A = \sum_{i=1}^{n} \lambda_i T_i$$

for some orthogonal projections T_i, and T_i projects vectors on its component on the direction of $\mathbf{v_i}$, where $\mathbf{v_i}$ is an eigenvector of A and $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$ forms an orthonormal basis of $\mathbb{R}^n$ consisiting of eigenvectors of A.

B.2 Linear Programming

Definition B.2.1. Linear Programming is a method to maximize or minimize target linear function subject to some linear conditions.

For example,

$$\text{maximize } c_1 x_1 + c_2 x_2 + \cdots + c_d x_d$$

subject to

$$\begin{cases} a_{11} x_1 + a_{12} x_2 + \cdots + a_{1d} x_d \leq b_1; \\ a_{21} x_1 + a_{22} x_2 + \cdots + a_{2d} x_d \leq b_2; \\ \vdots \\ a_{n1} x_1 + a_{n2} x_2 + \cdots + a_{nd} x_d \leq b_n; \end{cases}$$

is a LP problem.