Dimension reduction with singular value decomposition

Huy L. Nguyễn

We recently covered the Johnson-Lindenstrauss lemma, which gives a universal bound on the number of dimensions needed to approximately preserve all the pairwise distances among n points. The good thing about the result is that it is applicable to any dataset. However, it also fails to take advantage of the specific features of any given dataset. Today, we will consider a different method for dimension reduction that optimizes for the specific dataset at hand.

1 Best fit subspaces

Suppose our data points are rows of an $n \times d$ matrix A:$a_1, \dots, a_n$ with $n \geq d$. Our goal is to find a k-dimensional subspace such that the sum of the squared distance between the data points and the subspace is minimized. Suppose that the k-dimensional subspace is spanned by k vectors $v_1, v_2, \dots, v_k$. Our goal is to find $v_1, \dots, v_k$ and nk coefficients c_{ij} that minimizes $\sum_{i=1}^{n} \|a_i - \sum_{j=1}^{k} c_{ij} v_j\|^2$.

To solve this problem, we first review the Pythagoras theorem.

Theorem 1.1. *Consider a vector v and a subspace M. Let dist be the distance between v and the subspace and proj be the length of the projection of v onto M. We have*

$$\|v\|^2 = dist^2 + proj^2$$

The theorem gives us an alternative view of the best-fit subspace problem. Since the lengths of the data points are fixed, minimizing the sum of the squared distances between the points and the subspace is equivalent to maximizing the sum of the squared lengths of the projections.

2 Singular vectors

First, let's consider the simplest case where $k = 1$ i.e. we are looking for a 1-dimensional line going through the origin such that the sum of the squared distances between the points and the line is minimized. Suppose that v is a unit vector along this line. The length of the projection of a_i on the line is $|\langle a_i, v\rangle|$. As discuss before, we are looking for v that maximizes $\sum_{i=1}^{n} |\langle a_i, v\rangle|^2 = \|Ax\|^2$

We define the first singular vector of A as

$$v_1 = \operatorname*{argmax}_{\|v\|=1} \|Av\|$$

Note that the maximizer is not unique: for any vector v, the vector $-v$ gives the same objective value. We simply pick an arbitrary maximizer and refer to it as the *first singular vector*. The value $\sigma_1 = \|Av_1\|$ is called the *first singular value* of A.

We can define the other singular vectors recursively. The second singular vector is the best fit line among those orthogonal to v_1:

$$v_2 = \operatorname*{argmax}_{\|v\|=1, v \perp v_1} \|Av\|, \ \sigma_2 = \|Av_2\|$$

and so on

$$v_i = \underset{\|v\|=1, v \perp v_1, \ldots, v_{i-1}}{\operatorname{argmax}} \|Av\|$$

We can show that for any k, the best fit k-dimensional subspace is simply the span of the first k singular vectors.

Theorem 2.1. *The span of the first k singular vectors is a best fit k dimensional subspace.*

Proof. We can prove the theorem using induction. For $k = 1$, the theorem holds by definition. Next, we assume that the theorem is true for $k = m$ and we will prove the theorem for $k = m + 1$. Suppose W is a best fit $m + 1$ dimensional subspace. We can choose a basis for W as follows. First, let w_{m+1} be a unit vector in W that is orthogonal to the projections of $v_1, \ldots, v_m$ onto W. Next, we keep choosing vectors in W that are orthogonal to previous vectors to obtain a full orthogonal basis of W: $w_1, w_2, \ldots, w_{m+1}$.

Notice that w_{m+1} is orthogonal to $v_1, \ldots, v_m$ because each v_i is the sum of two components: its projection onto W and its component that is orthogonal to W and w_{m+1} is orthogonal to both of them.

By the hypothesis, the span of $v_1, \ldots, v_m$ is a best fit d dimensional subspace so:

$$\sum_{i=1}^{m} \|Av_i\|^2 \geq \sum_{i=1}^{m} \|Aw_i\|^2$$

Because w_{m+1} is orthogonal to $v_1, \ldots, v_d$ and v_{m+1} is the best fit line among vectors orthogonal to $v_1, \ldots, v_m$:

$$\|Av_{m+1}\|^2 \geq \|Aw_{m+1}\|^2$$

Thus,

$$\sum_{i=1}^{m+1} \|Av_i\|^2 \geq \sum_{i=1}^{m+1} \|Aw_i\|^2$$

Because W is a best fit subspace, the span of $v_1, \ldots, v_{m+1}$ is also a best fit subspace. $\square$

The vectors $v_1, \ldots, v_d$ are called the right singular vectors of A. The vectors Av_i are also useful and we normalize them to length one:

$$u_i = \frac{1}{\sigma_i} A v_i$$

These u_i are called the left singular vectors of A. It can be shown that these u_i are orthogonal and in fact, u_i is the unit vector that maximizes $\|A^T u\|$ among vectors orthogonal to $u_1, \ldots, u_{i-1}$ i.e. they are the right singular vectors of A^T.

These vectors form a decomposition of A into rank 1 matrices:

$$A = \sum_i \sigma_i u_i v_i^T$$

3 Power method for computing SVD

Suppose the SVD of A is $A = U \Sigma V^T$, where the u_i, v_i are the columns of U and V, respectively. Notice that $A^T A = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T$. Thus, σ_i^2 are the eigenvalues of $A^T A$.

In your linear algebra course, you might have seen an algorithm for computing the eigenvalues of a matrix. The singular values of A are simply the square roots of the eigenvalues of $M = A^T A$ so they can be computed that way. However, that requires solving a high degree equation, a nontrivial task. We will see a iterative method for approximating the singular values. There are more advanced ones but the basic idea is similar to the one we discuss.

Suppose for now that $\sigma_1 > \sigma_2$. The basic idea, which is similar to part of the LSH algorithm we covered before, is to amplify this difference. The key observation is that for a symmetric matrix M, the eigenvalues of M^2 are the squares of the eigenvalues of M. Thus, the gap between the first two eigenvalues values in M^2 is larger than the gap for M. In general, the eigenvalues of M^k are the kth power of the eigenvalues of M.

Thus, we will use a large value of k and the eigenvalues of M^k will be very different from each others, making it easier for us to compute them. Most of the times, we are interested in only the first few eigenvalues rather than the whole spectrum. In particular, we will focus on computing only the largest one. To do so, instead of working with M^k and use matrix multiplications, a slow operation, we will use the following technique. Let x be a random vector in $\mathbb{R}^d$. Because the singular vectors v_i form a basis, we can write x as

$$x = \sum_i c_i v_i$$

where $\sum_i c_i^2 = 1$. Suppose we compute $M^k x$, which can be computed using k matrix-vector product, which takes $O(knd)$ time in total.

$$M^k x = (V\Sigma^2 V^T)^k \sum_i c_i v_i = \sum_i c_i \sigma_i^{2k} v_i$$

Since x is random, we would expect the coefficients c_i to be of similar magnitude. However, because the $2k$ power of the singular values are very different, we expect the sum to be dominated by $\sigma_1^{2k} v_1$. Thus, $M^k x \approx c_1 \sigma_1^{2k} v_1$ and we can recover an approximation of v_i simply by normalizing $M^k x$ to length one.

Theorem 3.1. *Suppose x is a vector such that $c_1 \geq \delta$. Let $w = \frac{1}{\|M^k x\|} M^k x$ with $k = \frac{\ln(1/(\varepsilon\delta))}{2\varepsilon}$. Let w' be the projection of w on the span of singular vectors with singular values smaller than $(1-\varepsilon)\sigma_1$. We have $\|w'\| \leq \varepsilon$.*

Proof. Let $v_1, \ldots, v_t$ be the singular vectors with singular values at least $(1-\varepsilon)\sigma_1$. Let $v_{t+1}, \ldots, v_d$ be the singular vectors with singular values smaller than $(1-\varepsilon)\sigma_1$. Because $v_1, \ldots, v_d$ form a basis, we can write x as

$$x = \sum_i c_i v_i$$

with $\sum_i c_i^2 = \|x\|^2 = 1$. First we look at the length of $M^k x$.

$$\|M^k x\|^2 = \|\sum_i \sigma_i^{2k} c_i v_i\|^2 = \sum_i \sigma_i^{4k} c_i^2 \geq \sigma_1^{4k} \delta^2$$

$M^k x$ consists of two components: its projection on the span of $v_1, \ldots, v_t$ and its projection on the span of $v_{t+1}, \ldots, v_d$. The length of its projection on $v_{t+1}, \ldots, v_d$ is

$$\sum_{i \geq t+1} \sigma_i^{4k} c_i^2 \leq (1-\varepsilon)^{4k} \sigma_1^{4k}$$

where we used $\sum_i c_i^2 = 1$.

Thus, for the normalized vector w with $\|w\| = 1$, we have

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

□

For random vector x, it turns out that with constant probability, we have $c_1 \geq 1/(20\sqrt{d})$.

4 Spectral clustering

There are many applications of SVD. In this section, we consider a toy example of clustering and use SVD to find the clusters.

Suppose that there is a graph (e.g. a social network) with n nodes belonging to two communities S_1, S_2. The edges are generated randomly and independently. For two nodes in the same community, there is an edge between them with probability p. For two nodes in different communities, there is an edge between them with probability $q < p$ and $q = \Omega(\log n/n)$. We are given a random graph generated using this process. The goal is to recover the communities with high probability.

To solve this problem, let's first consider the *expectation* C of the adjacency matrix. We observe that the expectation C has rank 2!

Because each node has n possible edges, each of them is present with probability at least $q = \Omega(\log n/n)$, we expect a concentration phenomenon and the realized adjacency matrix is close to the expectation. Thus, the idea is to find a rank 2 approximation of the adjacency matrix and use that to recover the communities.

The algorithm is of the following form: compute a rank 2 matrix $\tilde{M}$ that best fit the adjacency matrix M. We expect that most columns of $\tilde{M}$ are close to the columns of C but we don't know which one is close. The idea is to just take a random column of $\tilde{M}$ and then partition the rest of the columns based on their distance to the chosen column.

There is a formal proof showing that the procedure finds the partition with high probability but we will not cover it here.

One can also generalize the more than 2 clusters. This model then suggests the following approach for finding k clusters: find the best rank k approximation to the adjacency matrix and then cluster the rows/columns of the resulting matrix.