EECS 16B Designing Information Devices and Systems II
Fall 2021

Note 17: PCA

Overview

In this note, we will examine a key application of the singular value decomposition (SVD) by using *low-rank approximations* for dimensionality reduction of data. When we collect data, there are potentially lots of things that are influencing the exact values that we see. Our goal is often to figure out the most important of these factors. This allows us to compress or distill our data down to its important essence, and remove the dimensions of data that are not imporant. The key new idea here is that of approximation. How can we do this using the linear-algebraic tools that we have built?

1 Analyzing Matrix Data

Imagine we are conducting some sort of statistical study — say, we are interested in how a group of n students enjoy a set of m movies. Each student gives a numerical rating of each movie, indicating how much they enjoyed it. We place these ratings into a matrix R, where R_{ij} is the rating the i^{th} student gives to the j^{th} movie. Our goal will be to draw conclusions from this dataset that enable us to make future predictions.

For instance, if we are told how much a new student enjoys a subset of our movies, we should be able to predict how much they will enjoy the other movies in our set. Alternatively, if we produce a new movie and are told how much a subset of our students enjoy it, we should be able to predict how much the other students in our study will enjoy it.

How on earth can we come up with these predictions? Aren't we all *unique and special*, so knowing how a student enjoys some movies provides no information on how they will enjoy the rest? In such a case, then yes, nothing could be done and our problem would be unsolvable.

However, in practice, it turns out that students are not as completely unique and special as all that! Typically, there is an underlying structure that lets us predict how much a student will enjoy each movie. For instance, imagine that every j^{th} film has a certain intrinsic "goodness" g_j, and every i^{th} student has a "sensitivity" s_i. Then the i^{th} student's rating of the j^{th} movie can be computed as the product of the student's sensitivity and the film's goodness, yielding

$$R_{ij} = s_i g_j. \tag{1}$$

In such an ideal scenario, our ratings matrix would simply be

$$R = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} \begin{bmatrix} g_1 & g_2 & \cdots & g_m \end{bmatrix} = \vec{s}\vec{g}^\top. \tag{2}$$

To take a slightly more complex scenario, imagine for simplicity that movies can be described by a mix of three properties - their levels of "romance," "comedy," and "action", represented by the vectors $\vec{r}$, $\vec{c}$, and $\vec{a}$. And imagine that each student computes how much they enjoy each movie by taking a linear combination of the "romance", "comedy", and "action" of each movie, where the coefficients of this linear combination vary with each student. Let these coefficients be stored in the vectors $\vec{s}_r$, $\vec{s}_c$, and $\vec{s}_a$ respectively. We'll call these coefficients the student's "sensitivities" to each of the three genres.

In such a case, we have that

$$R_{ij} = s_{r,i}r_j + s_{c,i}c_j + s_{a,i}a_j, \tag{3}$$

so the overall ratings matrix R can be expressed as

$$R = \vec{s}_r\,\vec{r}^\top + \vec{s}_c\,\vec{c}^\top + \vec{s}_a\,\vec{a}^\top. \tag{4}$$

The point is that, despite being an $n \times m$ matrix, an idealized R according to this model can be represented as the sum of only a constant number of outer products of n- and m-dimensional vectors.

What could we do if we knew these vectors? Well, imagine that we were given a new student and their ratings on a subset of the movies, stored in the vector $\vec{p}$. What do we want to know about them? Well, we've said that their sensitivities to romance, comedy, and action fully describe their movie preferences. So if we can recover their sensitivities to these genres, then we can make predictions about their ratings of a future movie, so long as we know what genres this future movie is in.

And how can we recover these sensitivities? We know that we can write each provided rating

$$p_j = w_r r_j + w_c c_j + w_a a_j, \tag{5}$$

where w_r, w_c, and w_a are the sensitivities of our new student. Stacking, we obtain the vector equation

$$\begin{bmatrix} r_1 & c_1 & a_1 \\ r_2 & c_2 & a_2 \\ \vdots & \vdots & \vdots \\ r_m & c_m & a_m \end{bmatrix} \begin{bmatrix} w_r \\ w_c \\ w_a \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{bmatrix}. \tag{6}$$

Using least squares, we can therefore recover the sensitivities w_r, w_c, and w_a, and use them to make future predictions. In a similar manner, if we are presented with a new movie and know how a subset of our students rate it, we can set up a least squares problem to solve for its romance, comedy, and action components, and then use these inferred components to make predictions for the rest of the students.

2 Analyzing Data Using the SVD

What did we rely on when making our predictions? Fundamentally, we relied on being able to write our large ratings matrix R as a low-rank sum of outer products. In reality, however, there will always be noise and variation in our data. We need some way to construct a "low-rank" approximation of a large matrix that preserves the fundamental structure of our data.

Let's see if the SVD helps. After all, given an $m \times n$ matrix A with rank r and singular values

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = 0, \tag{7}$$

we saw earlier that the SVD lets us write it as the sum of outer products

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top . \tag{8}$$

If our matrix is high-rank, i.e., $r \approx \min\{m, n\}$, then almost all of the singular values will be nonzero. However, we claim that when we have some linear, low-rank essential structure to our data, as is usually the case with real data such as images, most of our singular values will be very small. For instance, imagine that σ_1 and σ_2 are significantly larger than the others. Then we can write A as

$$(\sigma_1 \vec{u}_1 \vec{v}_1^\top + \sigma_2 \vec{u}_2 \vec{v}_2^\top) + \sum_{i=3}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top , \tag{9}$$

where the contribution of the summation is very small in comparison to the first two terms. This motivates approximating our dataset as just

$$\widehat{A} = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \sigma_2 \vec{u}_2 \vec{v}_2^\top , \tag{10}$$

and using this model to make predictions in the manner described.

As it turns out, empirically, this approach works quite well, and is the heart of what is known as *principal component analysis*. Still, it would be nice to somehow quantifiably justify its efficacy. In particular, why is it that taking just the first few terms of the SVD provides us with the "best" low-rank approximation to our dataset?

3 Low-Rank Approximations

Let's make our intuition about the SVD more formal. Given a matrix A, we will set a goal of computing a "low-rank" approximation $\widehat{A}$ that is of rank $\leq k$, but that is still "approximately equal" to A.

What does "approximately equal" mean? Well, we wish to minimize the magnitude of the error $A - \widehat{A}$, where the squared magnitude of a matrix is defined to be the sum of the squares of all its elements. Why? Because we have no a-priori reason to care about some entries more than other entries of the matrix. It turns out that this quantity is called the *Frobenius norm*, and we define it as

$$\|A\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2 . \tag{11}$$

Why do we care about minimizing this quantity $\left\|A - \widehat{A}\right\|_F^2$? One way to think about it is that our low-rank approximation is essentially a model that we are trying to fit to our data. For instance, in the above "movies" example, we are essentially trying to fit a model of student movie preferences to observed data. So it makes sense that we should try to pick a model that minimizes the error between the predicted and observed data points.

Our conjecture is that the best such approximation of an $m \times n$ matrix A with SVD

$$A = U\Sigma V^\top = \begin{bmatrix} | & & | \\ \vec{u}_1 & \cdots & \vec{u}_m \\ | & & | \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} \begin{bmatrix} - & \vec{v}_1^\top & - \\ - & \vec{v}_2^\top & - \\ & \vdots & \\ - & \vec{v}_n^\top & - \end{bmatrix} = \sum_{i=1}^{m} \sigma_i \vec{u}_i \vec{v}_i^\top . \tag{12}$$

is

$$\widehat{A} = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^\top, \tag{13}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots$. In other words, we sum up only the largest k outer products, rather than all r of them. Note that if $k \geq r$, then since $\sigma_{r+1} = \cdots = \sigma_k = 0$, our claim is that the best approximation of A that is rank k or less is just $\widehat{A} = A$. This tracks with our intuition.

How can we prove this result? First, we should take the time to establish some properties of the Frobenius norm.

3.1 The Frobenius Norm

Consider an $m \times n$ matrix M. As stated above, the Frobenius norm of this matrix is defined to be

$$\|M\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} M_{ij}^2. \tag{14}$$

First, let's look at how the columns of M relate to this quantity. Let the columns of M be m-dimensional vectors $\vec{v}_i$ such that

$$M = \begin{bmatrix} | & & | \\ \vec{v}_1 & \cdots & \vec{v}_n \\ | & & | \end{bmatrix}. \tag{15}$$

Observe that the squared norm of a particular vector is

$$\|\vec{v}_j\|^2 = \sum_{i=1}^{m} M_{ij}^2. \tag{16}$$

In other words, it is the sum of squares of all the elements in the j^{th} column. Summing this quantity over all the columns, we obtain the sum of squares of all the elements in M, which is the squared Frobenius norm. Expressed algebraically, it is the case that

$$\|M\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} M_{ij}^2 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} M_{ij}^2 \right) = \sum_{j=1}^{n} \|\vec{v}_j\|^2. \tag{17}$$

A similar result can be derived in an exactly analogous manner for the rows of M.

Now, we will look at how the Frobenius norm of a matrix varies under an orthonormal change of basis. Let Q be a matrix with m orthonormal columns, such that $Q^\top Q = I_{m \times m}$. Observe that, using the above result,

$$\|QM\|_F^2 = \left\| \begin{bmatrix} | & & | \\ Q\vec{v}_1 & \cdots & Q\vec{v}_n \\ | & & | \end{bmatrix} \right\|_F^2 \tag{18}$$

$$= \sum_{j=1}^{n} \|Q\vec{v}_j\|^2 \quad = \|M\|_F^2 \tag{19}$$

$$= \sum_{j=1}^{n} \vec{v}_j^\top Q^\top Q \vec{v}_j \tag{20}$$

$$= \sum_{j=1}^{n} \vec{v}_j^\top \vec{v}_j \tag{21}$$

$$= \sum_{j=1}^{n} \left\| \vec{v}_j \right\|^2 \tag{22}$$

$$= \|M\|_F^2 , \tag{23}$$

so the Frobenius norm of a matrix remains unchanged under pre-multiplication by Q. One way to interpret the above algebraic result is to notice that pre-multiplying by Q doesn't change the norm of any of the columns of M, so its Frobenius norm should similarly remain unchanged. It is interesting to note that here, we don't actually need Q to be square — that never came up in the above derivation. We just need orthonormal columns.

In a very similar manner, it can be shown that post-multiplication by a matrix $Q^\top$ with orthonormal rows again does not change the Frobenius norm. A way of seeing this is to simply take transposes everywhere in the above calculation, recognizing that the Frobenius norm of a matrix does not change if we take the transpose.

Now, combining the above observations with our knowledge of the SVD, we find that, given a matrix A with SVD $A = U\Sigma V^\top$,

$$\|A\|_F = \left\| U\Sigma V^\top \right\|_F \tag{24}$$

$$= \left\| U^\top U \Sigma V^\top V \right\|_F \tag{25}$$

$$= \|\Sigma\|_F , \tag{26}$$

since U and V are both square[1] orthonormal matrices, so pre-multiplying by $U^\top$ and post-multiplying by V both do not change the Frobenius norm, as in both cases their rows and columns are all orthonormal. Writing out the Frobenius norm of Σ in terms of the singular values explicitly, we find that

$$\|A\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2}. \tag{27}$$

This seems like a useful result that we should hang on to.

3.2 Reducing to an easier case via the SVD

Now, let's get back to the question of low-rank approximations. As stated before, we wish to choose a rank at most k matrix $\widehat{A}$ that minimizes $\left\| A - \widehat{A} \right\|_F$. Looking at the SVD of A and applying simplifications very similar to those derived in the previous section, pre-multiplying by $U^\top$ and post-multiplying by V, we see that

$$\left\| A - \widehat{A} \right\|_F = \left\| U\Sigma V^\top - \widehat{A} \right\|_F \tag{28}$$

[1]Here, we are using the full form of the SVD, not the compact form.

$$= \left\| U^\top U \Sigma V^\top V - U^\top \widehat{A} V \right\|_F \tag{29}$$

$$= \left\| \Sigma - U^\top \widehat{A} V \right\|_F . \tag{30}$$

Let $X = U^\top \widehat{A} V$, for convenience. Observe that, since U and V are both invertible, given any X we can choose a corresponding $\widehat{A}$, and vice-versa. Furthermore, the rank of $\widehat{A}$ and X are the same, so X has rank no more than k. Thus, the problem reduces to choosing an X of rank no more than k that minimizes

$$\|\Sigma - X\|_F . \tag{31}$$

Essentially, the SVD tells us that all we really need to understand is the diagonal case. How can we best approximate a diagonal matrix with sorted entries down the diagonal using a rank at most k matrix?

Let's write out Σ and X explicitly, to try and get a better idea of what remains for us to prove. We wish to choose columns $\vec{x}_i$ that minimize

$$\left\| \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} - X \right\|_F^2 \tag{32}$$

$$= \left\| \begin{bmatrix} \sigma_1 \vec{e}_1 & \cdots & \sigma_r \vec{e}_r & \vec{0} & \cdots & \vec{0} \end{bmatrix} - \begin{bmatrix} | & & | & | & & | \\ \vec{x}_1 & \cdots & \vec{x}_r & \vec{x}_{r+1} & \cdots & \vec{x}_n \\ | & & | & | & & | \end{bmatrix} \right\| \tag{33}$$

We will use the notation $\vec{e}_i$ here to represent the i^{th} column of the identity matrix $I_{m\times m}$ throughout the note.

Expressing the squared Frobenius norm as the sum of the squared norms of the columns, we see that we are tasked with picking columns $\vec{x}_i$ minimize

$$\|\sigma_1 \vec{e}_1 - \vec{x}_1\|^2 + \cdots + \|\sigma_r \vec{e}_r - \vec{x}_r\|^2 + \left\|\vec{0} - \vec{x}_{r+1}\right\|^2 + \cdots + \left\|\vec{0} - \vec{x}_n\right\|^2 \tag{34}$$

$$= \sum_{i=1}^{r} \|\sigma_i \vec{e}_i - \vec{x}_i\|^2 + \sum_{i=r+1}^{n} \|\vec{x}_i\|^2 , \tag{35}$$

while keeping the rank of X less than or equal to some given constant k. First, we should pick

$$\vec{x}_{r+1} = \vec{x}_{r+2} = \cdots = \vec{x}_n = \vec{0}, \tag{36}$$

since making them nonzero would hurt us by increasing the sum of norms that we are trying to minimize.

But what about the $\vec{x}_1, \ldots, \vec{x}_r$? There are two cases here.

If $k \geq r$, then we can pick $\vec{x}_i = \sigma_i \vec{e}_i$ for $i = 1, \ldots, r$. Then X is a perfect rank $\leq k$ approximation for Σ, i.e., $X = \Sigma$ and rank $X \leq k$. This can be turned into a perfect approximation for A, i.e., $\widehat{A} = A$. This confirms our claim that the best rank $\leq k$ approximation for A is just A, in the case that $k \geq r$. We are now done for the case $k \geq r$. So from now on let us assume $k < r$.

The other case, $k < r$, is more interesting. Because of the constraint on the rank of X, we can't make all $\vec{x}_i$ equal to the corresponding columns of Σ, as then the rank of X would be r, and in particular greater than k.

One intuitive guess could be to remove the k largest singular values since they are affecting the norm the most. Then we would pick $\vec{x}_1 = \sigma_1 \vec{e}_1, \vec{x}_2 = \sigma_2 \vec{e}_2, \ldots, \vec{x}_k = \sigma_k \vec{e}_k$ and set the rest of the $\vec{x}_i = \vec{0}$ for $i > k$.

This definitely makes X have rank k, and turns out to be the correct answer. But how can we be sure that there isn't a better choice out there? This requires a proof, which we will do in the next section. If you already believe the result, you might want to jump to the end and read Section 5.

4 The Projection Perspective

In the previous section, we used the SVD to distill our problem down to the diagonal case. The problem is: how can we choose an X of rank no more than k that minimizes

$$\|\Sigma - X\|_F? \tag{37}$$

We saw that what matters here is choosing the r vectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_r$ that are matched up with the non-zero parts of Σ in the above subtraction, since the rest of X should be all zeros. We have a guess as to what the optimal choice should be, but we need to prove that it is indeed optimal.

The projection perspective approaches this problem by stepping back a bit. Dealing with the constraint on the rank of X is annoying. What does this condition really mean from a geometric point of view? It means that all these r vectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_r$ must lie in a k-dimensional subspace. This is what rank means.

Let's reason about the problem in terms of a basis for this subspace. Imagine that we are already given this subspace as the span of some $m \times k$ matrix Q. What kind of matrix should we ask for? We know that orthonormal matrices are comparatively easy to work with, and any other matrix with linearly independent columns can be orthonormalized, so for convenience, let's assume that all the columns of Q have been orthonormalized, so $Q^\top Q = I_{k\times k}$. Then what should each of the $\vec{x}_i$ be, assuming they are constrained to lie in the column space of Q?

Well, once we're given Q, we simply need to choose each $\vec{x}_i$ to minimize the term it is involved in — specifically, each $\vec{x}_i$ should minimize

$$\|\sigma_i\vec{e}_i - \vec{x}_i\|^2. \tag{38}$$

And how can we choose such an $\vec{x}_i$? We know this from 16A! It is simply the projection of $\sigma_i\vec{e}_i$ onto the column space of Q, which from least squares is just

$$\vec{x}_i = Q(Q^\top Q)^{-1}Q^\top(\sigma_i\vec{e}_i) = QQ^\top(\sigma_i\vec{e}_i) = \sigma_i QQ^\top\vec{e}_i, \tag{39}$$

simplifying by recalling that $Q^\top Q = I_{k\times k}$, as we chose the columns of Q to be an orthonormal basis.

This means that finding the right orthonormal Q is enough to solve our problem. In hindsight, this is not surprising since our initial motivation was to discover a subspace that best captures the data.

4.1 Finding the Best Q

So our problem has been simplified further. Rather than trying to find an arbitrary $m \times n$ matrix X with rank $\leq k$, we can instead find an $m \times k$ orthonormal matrix Q, and then compute $\vec{x}_i$ by projecting each of the columns of Σ onto the column space of Q.

Since we're now working with Q, rather than X, it makes sense to write the quantity we're interested in minimizing in terms of Q. Substituting in our least-squares solution for each of the $\vec{x}_i$ into our earlier expression for the squared Frobenius norm, and setting all the $\vec{x}_i = \vec{0}$ for all $i > r$ (as we argued would be optimal), we find that the new quantity to minimize becomes

$$\sum_{i=1}^{r} \|\sigma_i \vec{e}_i - \vec{x}_i\|^2 + \sum_{i=r+1}^{n} \|\vec{x}_i\|^2 \tag{40}$$

$$= \sum_{i=1}^{r} \left\|\sigma_i \vec{e}_i - \sigma_i Q Q^\top \vec{e}_i\right\|^2 + \sum_{i=r+1}^{n} \left\|\vec{0}\right\|^2 \tag{41}$$

$$= \sum_{i=1}^{r} \sigma_i^2 \left\|\vec{e}_i - Q Q^\top \vec{e}_i\right\|^2. \tag{42}$$

Can we simplify this expression somewhat? Notice that $\left(\vec{e}_i - QQ^\top \vec{e}_i\right) \perp QQ^\top \vec{e}_i$, since we know from 16A that the least squares projection is orthogonal to the residual error vector. Thus, by the Pythagorean theorem, we can write

$$\|\vec{e}_i\|^2 = \left\|\vec{e}_i - QQ^\top \vec{e}_i\right\|^2 + \left\|QQ^\top \vec{e}_i\right\|^2 \implies \left\|\vec{e}_i - QQ^\top \vec{e}_i\right\|^2 = \|\vec{e}_i\|^2 - \left\|QQ^\top \vec{e}_i\right\|^2. \tag{43}$$

Making this substitution in our earlier expression, noticing that $\|\vec{e}_i\| = 1$ and rearranging, our problem reduces to finding an orthonormal Q that minimizes the quantity

$$\sum_{i=1}^{r} \sigma_i^2 \left(\|\vec{e}_i\|^2 - \left\|QQ^\top \vec{e}_i\right\|^2\right) = \sum_{i=1}^{r} \sigma_i^2 - \sum_{i=1}^{r} \sigma_i^2 \left\|QQ^\top \vec{e}_i\right\|^2 \tag{44}$$

$$= \sum_{i=1}^{r} \sigma_i^2 - \sum_{i=1}^{r} \left\|QQ^\top \sigma_i \vec{e}_i\right\|^2 \tag{45}$$

$$= \|\Sigma\|_F^2 - \left\|QQ^\top \Sigma\right\|_F^2 \tag{46}$$

Notice that we have broken the quantity we are minimizing into two summations. The first summation is simply the squared Frobenius norm of Σ, which does not depend on Q. The second is the negation of the squared Frobenius norm of $QQ^\top \Sigma$. Thus, in order to minimize the overall quantity, we should aim to choose our Q to *maximize* $\left\|QQ^\top \Sigma\right\|_F^2$, since we are subtracting this quantity from another fixed value.

4.2 Working with this even simpler problem

Cool! We've managed to simplify our problem further, by taking advantage of properties of projections and orthonormal bases. Recall from earlier that $\|QM\|_F^2 = \|M\|_F^2$ when Q has orthonormal columns. Then we see that the quantity we are trying to maximize can be slightly simplified as

$$\left\|QQ^\top \Sigma\right\|_F^2 = \left\|Q^\top \Sigma\right\|_F^2. \tag{47}$$

Be aware that, since Q is not square, $QQ^\top \neq I_{m\times m}$, as the rows are not necessarily mutually orthogonal. So we can't use this property to make any further simplifications.

How can we show that our guessed solution (choosing the first k columns of the identity for Q) is indeed the best choice for maximizing (47)? That choice would give $\left\|Q^\top \Sigma\right\|_F^2 = \sum_{j=1}^{k} \sigma_j^2$. One way to show that this is optimal is to show that we cannot possibly do any better than that.

To do this, let's just expand this product out directly. For notational convenience, let the columns of Q be $\vec{q}_i$, and so

$$Q^\top = \begin{bmatrix} - & \vec{q}_1^\top & - \\ - & \vec{q}_2^\top & - \\ & \vdots & \\ - & \vec{q}_k^\top & - \end{bmatrix}. \tag{48}$$

Remember that each of the $\vec{q}_i$ are m-dimensional orthonormal vectors.

Now, writing out our product, we obtain

$$\left\|Q^\top \Sigma\right\|_F^2 = \left\| \begin{bmatrix} - & \vec{q}_1^\top & - \\ - & \vec{q}_2^\top & - \\ & \vdots & \\ - & \vec{q}_k^\top & - \end{bmatrix} \begin{bmatrix} \sigma_1 \vec{e}_1 & \sigma_2 \vec{e}_2 & \cdots & \sigma_r \vec{e}_r & \vec{0} & \cdots & \vec{0} \end{bmatrix} \right\|_F^2 \tag{49}$$

$$= \sum_{i=1}^{k} \sum_{j=1}^{r} \left(\vec{q}_i^\top \vec{e}_j\right)^2 \sigma_j^2 \tag{50}$$

$$= \sum_{i=1}^{k} \sum_{j=1}^{r} Q_{ji}^2 \sigma_j^2 \tag{51}$$

$$= \sum_{j=1}^{r} \sigma_j^2 \sum_{i=1}^{k} Q_{ji}^2 \tag{52}$$

Thus, we have expressed our desired Frobenius norm as a linear combination of the squared singular values σ_j^2. For convenience and to focus our attention on what is going to matter, let the coefficients of our linear combination be denoted as

$$d_j = \sum_{i=1}^{k} Q_{ji}^2, \tag{53}$$

where d_j is defined to be the squared norm of the jth row of Q. Then we can express the term we want to maximize as

$$\left\|Q^\top \Sigma\right\|_F^2 = \sum_{j=1}^{r} \sigma_j^2 d_j. \tag{54}$$

Now, observe that the Frobenius norm of $Q^\top$ itself is $\|Q\|_F = k$, since Q has k unit-norm columns. Expressing this quantity in terms of the d_j, we see that

$$\sum_{j=1}^{m} d_j = \left\|Q^\top\right\|_F^2 = \|Q\|_F^2 = k. \tag{55}$$

Furthermore, notice that, since they are defined as a sum of squares in (53), each of the $d_j \geq 0$. As $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$, we could try to upper-bound the quantity we are trying to maximize by choosing $d_1 = k$ and setting the other $d_j = 0$.

That would give $k\sigma_1^2$ which is bigger than our guessed optimal solution. But is this extreme choice of d_j actually achievable by some Q? The challenge is that we know a lot about the columns of Q since they are orthonormal. But d_j is related to the j^{th} *row* of Q. How can we understand anything about the rows of Q?

If Q were square, then the fact that $Q^\top Q = I$ would tell us that $Q^\top$ is the inverse of Q and so $Q^\top$ would also be orthonormal. But Q is decidedly not square, so what can we do?

Observe that the k columns of Q can be extended using Gram-Schmidt to form a full orthonormal basis for $\mathbb{R}^m$. Let this extended basis be

$$\widehat{Q} = \begin{bmatrix} | & & | & | & & | \\ \vec{q}_1 & \cdots & \vec{q}_k & \vec{q}_{k+1} & \cdots & \vec{q}_m \\ | & & | & | & & | \end{bmatrix}. \tag{56}$$

By the properties of orthonormal square matrices, we know that in addition to the columns, the *rows* of $\widehat{Q}$ also form an orthonormal basis, and so are all of unit norm. Thus, looking at the j^{th} row, we can write

$$\sum_{i=1}^{m} \widehat{Q}_{ji}^2 = 1 \tag{57}$$

for all j.

This tells us that the rows of $\widehat{Q}$ can't be too large, and thus, truncating the sum by removing the terms corresponding to the $(k+1)^{\text{th}}$ column onwards, we see that

$$d_j = \sum_{i=1}^{k} Q_{ji}^2 \le \sum_{i=1}^{m} \widehat{Q}_{ji}^2 = 1. \tag{58}$$

So in addition to requiring that they sum to k, we have established that each individual $0 \le d_j \le 1$ for all j.

Now, we have found a simpler problem inside our original problem.

4.3 Working with the inner problem

At this point, we have turned even the simplified problem into something[2] only involving numbers. From (54), we know we want to maximize $\sum_{j=1}^{r} \sigma_j^2 d_j$ over the choice of d_j that we know must satisfy certain constraints. From (55), we know their overall sum $\sum_{j=1}^{m} d_j = k$. We also know from (58), that $0 \le d_j \le 1$ for all j.

A consequence of this is that our previous overly greedy assignment of values to d_j is not possible if $k > 1$, since we must have $d_1 \le 1 < k$. Instead, if we wish to maximize our linear combination of the σ_j^2, we should set

$$d_1 = d_2 = \cdots = d_k = 1 \tag{59}$$

and

$$d_{k+1} = d_{k+2} = \cdots = d_m = 0, \tag{60}$$

in order to place as much weight as possible on the largest squared singular values without violating the constraints on the d_j. This gives us $\sum_{j=1}^{k} \sigma_j^2$ as the best we can do.

Why is this the case? We can think of each d_j as telling us how much money we want to spend to buy product j. The store only has up to 1 liter of each product in stock. Each item costs one dollar per liter. The

[2]You will see in later courses like 127 and 170 that such problems are actually called "linear programs" and are extremely useful. They arise naturally in lots of settings. In 127 and 170, you will learn ways of thinking about them that are generally useful. In our case here, we can reason about the problem directly.

product j contains σ_j^2 grams of gold per liter. If we want to get as much gold as possible, what should we do if we have k dollars to spend? The answer is to buy out all the stock of the first k products since they all cost the same, but the first k products have more gold per liter than any of the other products.

The rigorous proof of this claim is omitted here in this note, but only because you were walked through this exact proof on the homework. (It follows from an argument that doing anything else can't be optimal since anything else could be improved by moving it closer to this solution.)

But is *this* assignment of d_j above actually achievable in our original problem of interest? It certainly satisfies all of our inequalities, but that is not sufficient to show it is actually possible — perhaps we could have derived another inequality that makes this assignment impossible. After all, the d_j are defined in (53) from the underlying Q matrix. Maybe no Q matrix exists that satisfies our desired asignments of d_j?

Consequently, the best way to show that it is in fact feasible is to present such a Q. We want the first k rows of Q to all be of unit norm, and the remaining columns to all be 0. Thus, one choice would be to write

$$Q = \begin{bmatrix} I_{k\times k} \\ 0_{(m-k)\times k} \end{bmatrix} = \begin{bmatrix} \vec{e}_1 & \dots & \vec{e}_k \end{bmatrix}. \tag{61}$$

It is straightfoward to verify that the columns of Q form an orthonormal basis of a k-dimensional subspace, so this choice of Q is valid and also shown to be optimal.

So what low-rank approximation does this optimal choice of Q correspond to? Substituting back, we see that

$$\begin{aligned} \widehat{A} &= UXV^\top \qquad (62)\\ &= UQQ^\top \Sigma V^\top. \qquad (63) \end{aligned}$$

Now, observe that

$$QQ^\top = \begin{bmatrix} I_{k\times k} \\ 0_{(m-k)\times k} \end{bmatrix} \begin{bmatrix} I_{k\times k} & 0_{k\times(m-k)} \end{bmatrix} = \begin{bmatrix} I_{k\times k} & 0_{k\times(m-k)} \\ 0_{(m-k)\times k} & 0_{(m-k)\times(m-k)} \end{bmatrix} \tag{64}$$

In words, $QQ^\top$ is a diagonal matrix where the first k entries along the diagonal are 1, and the remainder are 0. Thus, we see that

$$\begin{aligned} \widehat{A} &= UQQ^\top \Sigma V^\top \qquad (65)\\ &= U \begin{bmatrix} I_{k\times k} & 0_{k\times(m-k)} \\ 0_{(m-k)\times k} & 0_{(m-k)\times(m-k)} \end{bmatrix} \Sigma V^\top \qquad (66)\\ &= U \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} V^\top \qquad (67) \end{aligned}$$

$$= \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^\top, \tag{68}$$

as expected. This completes the proof.

The entire proof here is not something that we would expect you to be able to come up with on your own at the level of 16B. However, it is elementary enough that you should be able to follow the steps, and the goal is to help you begin to feel how each part follows naturally.

5 Applying Low-Rank Approximation to PCA

The above theorem tells us that if we want to find the best k-dimensional subspace to use to approximate a set of (real) data, we can take the data, arrange it into a matrix M by stacking the columns of data, and then taking the SVD. Looking at the SVD in outer product form $M = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top$, where the σ_i are in decreasing order, we can just choose to truncate the sum to the top k terms. We get an approximation of all the data $\widehat{M} = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^\top$. More importantly, we also get a basis for our subspace of interest. Namely, the $\vec{u}_i$ gives an orthonormal basis for the k-dimensional subspace that best approximates our data. The first such $\vec{u}_1$ is called the first principal component (for columns), the second such $\vec{u}_2$ is called the second principal component (for columns), and so on.

If we had arranged the data into rows instead, then we could do the same thing and we would use the first $\vec{v}_i$ instead as the principal components. (Or if we want to keep them in row form, $\vec{v}_i^\top$.) The choice depends nature of the data and what we are trying to do with it.

Once we have learned such a subspace from training data, we can use it for dimensionality reduction by projecting new raw data points onto the subspace. So instead of storing the entire data point, this allows us to store just the coefficients of the projections in the subspace basis we have found. For example, if the raw data points have $m = 100$ entries in them, and we decided to use $k = 3$ to choose a 3-dimensional subspace, then we have managed to distill the essential information into just 3 coefficients. These coefficients are also easier to calculate by the orthonormality of the $\vec{u}_i$. For a set of orthonormal vectors U, we know that its projection coefficients from least square simplifies to $(U^\top U)^{-1} U^\top \vec{x} = U^\top \vec{x}$ since $U^\top U = I$. So for the case of $k = 3$, we just keep

$$U^\top \vec{x} = \begin{bmatrix} \vec{u}_1^\top \\ \vec{u}_2^\top \\ \vec{u}_3^\top \end{bmatrix} \vec{x} = \begin{bmatrix} \vec{u}_1^\top \vec{x} \\ \vec{u}_2^\top \vec{x} \\ \vec{u}_3^\top \vec{x} \end{bmatrix} \tag{69}$$

instead of the whole m-dimensional raw data point $\vec{x}$. And to get the approximated data point, we just multiply the coefficients by the basis to get $UU^\top \vec{x}$. Such dimensionality reduction is often very useful in reducing the computational burden[3] for doing data analysis, classification, regression, etc.

In some problem contexts, the data that we are given has offsets from the origin. What do we mean? Consider two-dimensional data. Our approach to PCA will find the best line through the origin that goes through the data. But what if the data was actually along a line that didn't go through the origin? How

[3]For reasons that you will learn in future courses like 189, the advantage is not simply computational. Distilling the data down by dimensionality reduction can also be helpful in eliminating irrelevant noise that could confuse subsequent stages of learning. Even more subtly, it can help make problems tractable that otherwise might seem impossible. For example, if the data that is being recorded is a microphone trace that is hundreds of samples long, then many approaches to learning how to classify words might require thousands of examples of each word. By reducing the dimensionality of the incoming microphone trace, it can become possible to learn from fewer examples using those approaches.

would we find this? In that case, we would want to "center" the data first. These offsets are often estimated and removed before one does PCA analysis. Such offsets can be found by taking appropriate means of the training data, and then subtracting them from the data. However, it is useful to consider that "de-meaning" step as being something separate from the core task of subspace discovery. Instead, it should be considered an application-specific data cleaning or preprocessing step. It should only be done if you actually believe that such offsets might exist.

5.1 PCA Algorithm

We can formalize the above section with the following Principal Component Analysis (PCA) algorithm:

Inputs: a set of data points $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$
Outputs: the k principal components which best approximate your data

(a) Arrange the data into a matrix:

- We can either use data as columns: $X = \begin{bmatrix} | & | & \cdots & | \\ \vec{x}_1 & \vec{x}_2 & \cdots & \vec{x}_n \\ | & | & \cdots & | \end{bmatrix}$
- Or we can use data as rows: $X = \begin{bmatrix} - & \vec{x}_1^\top & - \\ - & \vec{x}_2^\top & - \\ \vdots & \vdots & \vdots \\ - & \vec{x}_n^\top & - \end{bmatrix}$

(b) Compute the SVD: $X = U\Sigma V^\top = \sum \sigma_i \vec{u}_i \vec{v}_i^\top$

(c) Find the first k principal components

- If our data is columns: choose $\vec{u}_1, \vec{u}_2, \ldots \vec{u}_k$
- If our data is rows: choose $\vec{v}_1, \vec{v}_2, \ldots \vec{v}_k$

(d) Project your data onto the principal components to get the underlying lower dimensional structure (which you can later use for clustering/classification).

- If our data is columns, the projection of data point $\vec{x}_i$ onto the k-kdimensional subspace has coefficients $U_k^\top \vec{x}_i$ and projection $U_k U_k^\top \vec{x}_i = \sum_{p=1}^{k} (\vec{u}_p^\top \vec{x}_i)\vec{u}_p$. Doing it for all columns is $U_k U_k^\top X$.
- If our data is rows, the projection of data point $\vec{x}_i$ onto the k-kdimensional subspace has coefficients $V_k^\top \vec{x}_i$ and projection $V_k V_k^\top \vec{x}_i = \sum_{p=1}^{k} (\vec{v}_p^\top \vec{x}_i)\vec{v}_p$. Doing it for all rows is $X V_k V_k^\top$.

Notice that this projection approach works because of the orthonormality of the vectors in U and V. If the vectors in U and V were not orthonormal, the least squares projection would not simplify so nicely.

You may be asking, how do you choose the best number of principal components? Well, there are several ways. If you have an underlying idea of what the dimensionality of the underlying structure should be, you could try to use that. Alternatively, you could observe the singular values to see at which point they drop off significantly, as then they won't affect the approximation much. Then only use the principal components/singular vectors corresponding to relatively larger singular values of the matrix.

6 PCA via Minimizing Reconstruction Error

We now show an alternative derivation of PCA which shines a new light on what PCA really does.

Remember that what we want to do is find the best low-dimensional subspace for our data, so that we can reduce the dimension of our data. If this subspace is very close to our data, the projection of the data onto the subspace has little error, but saves space by only having to store 1 coefficient for each dimension.

Another way of saying this is that we want to know the best directions to project our data points onto such that the error between the data point and the projection is minimized. This is called the **reconstruction error** and we will now formalize it with math. If we call the direction we are projecting onto $\vec{w}$, and normalize it so $\|\vec{w}\| = 1$, then the squared reprojection error of data point $\vec{x}_i$ is $\left\|\vec{x}_i - (\vec{x}_i^\top \vec{w})\vec{w}\right\|^2$. We want to minimize the total reprojection error for all our n data points so we want to solve

$$\underset{\|\vec{w}\|=1}{\operatorname{argmin}} \sum_{i=1}^{n} \left\|\vec{x}_i - (\vec{x}_i^\top \vec{w})\vec{w}\right\|^2 \tag{70}$$

Solving this optimization problem will give you the best projection direction, which we will show is exactly the 1st principal component!

We first simplify this norm:

$$\begin{aligned}
\left\|\vec{x}_i - (\vec{x}_i^\top \vec{w})\vec{w}\right\|^2 &= \left(\vec{x}_i - (\vec{x}_i^\top \vec{w})\vec{w}\right)^\top \left(\vec{x}_i - (\vec{x}_i^\top \vec{w})\vec{w}\right) && (71)\\
&= \vec{x}_i^\top \vec{x}_i + (\vec{x}_i^\top \vec{w})^2 \vec{w}^\top \vec{w} - 2(\vec{x}_i^\top \vec{w})\vec{x}_i^\top \vec{w} && (72)\\
&= \|\vec{x}_i\|^2 + (\vec{x}_i^\top \vec{w})^2 \cdot 1 - 2(\vec{x}_i^\top \vec{w})^2 && (73)\\
&= \|\vec{x}_i\|^2 - (\vec{x}_i^\top \vec{w})^2 && (74)
\end{aligned}$$

Since the first term is constant with respect to our optimization variable $\vec{w}$, and since minimizing a negative is equivalent to maximizing a positive, we can rewrite our problem as

$$\underset{\|\vec{w}\|=1}{\operatorname{argmin}} \sum_{i=1}^{n} -(\vec{x}_i^\top \vec{w})^2 = \underset{\|\vec{w}\|=1}{\operatorname{argmax}} \sum_{i=1}^{n} (\vec{x}_i^\top \vec{w})^2 \tag{75}$$

We now will manipulate the summation term by rearranging the terms in the inner product:

$$\begin{aligned}
\underset{\|\vec{w}\|=1}{\operatorname{argmax}} \sum_{i=1}^{n} (\vec{x}_i^\top \vec{w})^2 &= \underset{\|\vec{w}\|=1}{\operatorname{argmax}} \sum_{i=1}^{n} \vec{w}^\top \vec{x}_i \vec{x}_i^\top \vec{w} && (76)\\
&= \underset{\|\vec{w}\|=1}{\operatorname{argmax}}\, \vec{w}^\top \left(\sum_{i=1}^{n} \vec{x}_i \vec{x}_i^\top\right) \vec{w} && (77)\\
&= \underset{\|\vec{w}\|=1}{\operatorname{argmax}}\, \vec{w}^\top X X^\top \vec{w} && (78)
\end{aligned}$$

Here we assume that our data matrix X has the data points $\vec{x}_i$ as columns, and so we can use the outer product form of matrix multiplication. Plugging in the SVD expression $U\Sigma V^\top$ in for X,

$$\underset{\|\vec{w}\|=1}{\operatorname{argmax}}\, \vec{w}^\top X X^\top \vec{w} = \underset{\|\vec{w}\|=1}{\operatorname{argmax}}\, \vec{w}^\top U\Sigma V^\top V \Sigma^\top U^\top \vec{w} \tag{79}$$

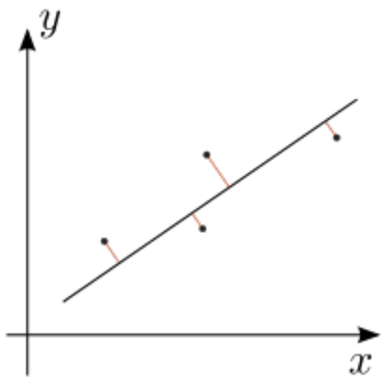
$$= \underset{\|\vec{w}\|=1}{\operatorname{argmax}} \vec{w}^\top U \Sigma \Sigma^\top U^\top \vec{w} \tag{80}$$

Let us assume that each data point $\vec{x}_i \in \mathbb{R}^m$ so X is $m \times n$, U is $m \times m$, and Σ is $m \times n$. Then multiplying Σ by $\Sigma^\top$, we should expect to get a $m \times m$ square matrix with squared singular values along the diagonal. We also perform a change of basis with $U^\top \vec{w} = \vec{\tilde{w}}$. Since orthonormal matrices don't affect norms, our norm constraint is still that $\left\|\vec{\tilde{w}}\right\| = 1$. This allows us to rewrite our maximization as

$$\underset{\left\|\vec{\tilde{w}}\right\|=1}{\operatorname{argmax}} \; \vec{\tilde{w}}^\top \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_m^2 \end{bmatrix} \vec{\tilde{w}} \tag{81}$$

Remember that σ_1 is the largest of all the singular values. Thus, since we have to pick a vector $\vec{\tilde{w}}$ of norm 1, we would maximize our objective by selecting $\vec{\tilde{w}} = \vec{e}_1$. Now, to find the actual value of $\vec{w}$, we can substitute this into $\vec{w} = U\vec{\tilde{w}}$ to find that $\vec{w} = U\vec{e}_1 = \vec{u}_1$. Thus, the direction that minimizes the reconstruction error will be the first principal component of the data matrix, $\vec{u}_1$. Similarly, the orthogonal direction with the second least reconstruction error will be the second principal component $\vec{u}_2$ and so forth.

This interpretation tells us the principal components (U or V vectors depending on column or row data) are exactly the directions that reduce the orthogonal projection error. This can be seen graphically in the following figure:



Note how this error metric is different from least squares as that minimizes the vertical error to the line of best fit, while PCA minimizes the orthogonal projection error to the line [4].

Contributors:

- Rahul Arya.
- Anant Sahai.
- Ayan Biswas.
- Druv Pai.
- Ashwin Vangipuram.
- Kamyar Salahi.

[4] This idea is also called total least squares, which is discussed more in-depth in CS 189.