Total Least-Squares

Our main approach thus far to "solving" $\boldsymbol{y} \approx \boldsymbol{A}\boldsymbol{x}$ is to optimize

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2.$$

Thought of another way, if we can't find a $\boldsymbol{x}$ such that $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ exactly, we are looking for the smallest possible perturbation we could add to $\boldsymbol{y}$ so that there is an exact solution. Mathematically, the standard least-squares program above is equivalent to solving

$$\underset{\Delta\boldsymbol{y},\boldsymbol{x}}{\text{minimize}} \ \|\Delta\boldsymbol{y}\|_2^2 \quad \text{subject to} \quad (\boldsymbol{y} + \Delta\boldsymbol{y}) = \boldsymbol{A}\boldsymbol{x}.$$

This reformulation makes it clear that least-squares implicitly assumes that all of the error (i.e. all of the reasons we can't find an exact solution) lies in the measured data $\boldsymbol{y}$.

But what if the entries of $\boldsymbol{A}$ are also subject to error? That is, how can we account for *modeling error* as well as measurement error? *Total least-squares* (TLS) is a framework for doing exactly this in a principled manner. TLS finds the smallest perturbations $\Delta\boldsymbol{y}, \Delta\boldsymbol{A}$ such that

$$(\boldsymbol{y} + \Delta\boldsymbol{y}) = (\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{x}$$

has an exact solution. It does this by solving

$$\underset{\Delta\boldsymbol{A},\Delta\boldsymbol{y},\boldsymbol{x}}{\text{minimize}} \ \|\Delta\boldsymbol{A}\|_F^2 + \|\Delta\boldsymbol{y}\|_2^2 \quad \text{subject to} \quad (\boldsymbol{y} + \Delta\boldsymbol{y}) = (\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{x},$$

where $\|\Delta\boldsymbol{A}\|_F^2$ is the **Frobenius norm** of $\Delta\boldsymbol{A}$ and is given by

$$\|\Delta\boldsymbol{A}\|_F^2 = \sum_{m=1}^{M}\sum_{n=1}^{N} |\Delta A[m,n]|^2.$$

Example: 1D linear regression

Say we are given a set of points

$$(a_1, y_1), \quad (a_2, y_2), \cdots, (a_M, y_M)$$

Suppose that the goal is to find the "best" line that fits these points. (For simplicity, we will only consider lines that pass through the origin.) That is, we are looking for the slope x such that the $a_m x$ are as close to the y_m as possible.

The standard least-squares framework models this problem as follows. We observe

$$y_m = a_m x + \text{noise},$$

or in matrix form,

$$\boldsymbol{y} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} x + \text{noise}.$$

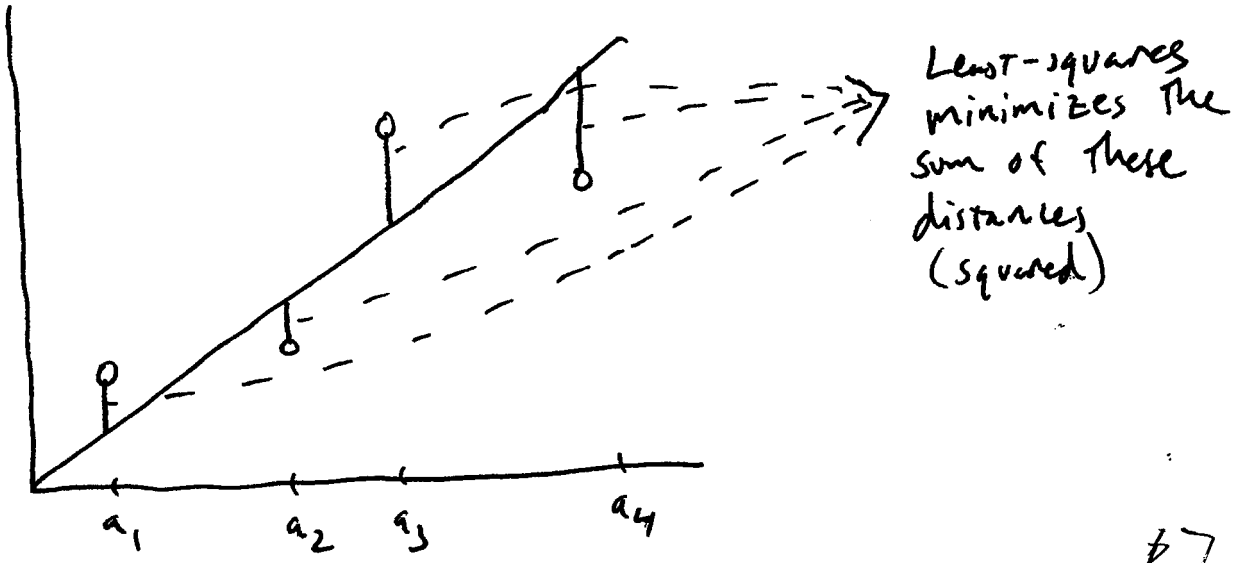
The solution is of course

$$\widehat{x} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} = \frac{\sum_{m=1}^{M} a_m y_m}{\sum_{m=1}^{M} a_m^2}.$$

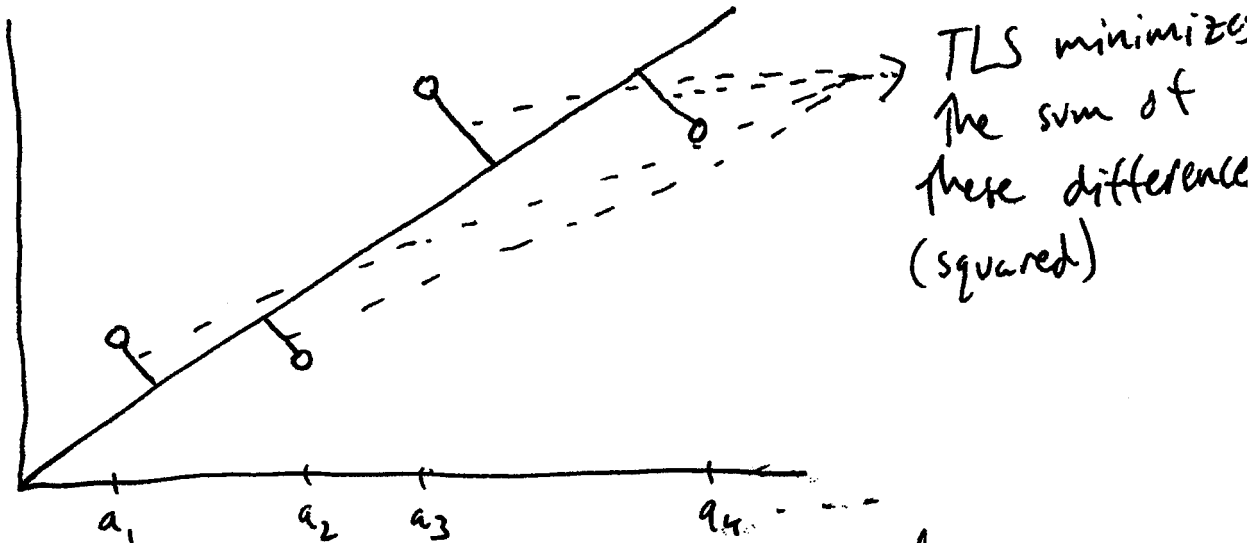
This solution minimizes the size of the residual

$$\|\boldsymbol{r}\|_2^2 = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \sum_{m=1}^{M} |y_m - a_m x|^2.$$

Geometrically, we are choosing the slope that minimizes the sum of the squares of the *vertical distances* of the points to the line we choose to approximate them:



In contrast, the TLS estimate (which we will see how to compute below) minimizes the *distance in the plane* of the points to the line we choose:



This distance includes changes in both the a_m and y_m.

Solving TLS – Part I

We will assume that $\boldsymbol{A}$ is an $M \times N$ matrix, with $M > N$, and $\text{rank}(\boldsymbol{A}) = N$ (i.e. $\boldsymbol{A}$ is overdetermined with full column rank). The problem only really makes sense if $\text{rank}(\boldsymbol{A}) < M$, otherwise there is always an exact solution. By being careful with the details, the method we present here can also be extended to the case where $\text{rank}(\boldsymbol{A}) < N < M$, but we will leave it to you to fill in those gaps.

We want to find $\Delta\boldsymbol{A}, \Delta\boldsymbol{y}, \boldsymbol{x}$ such that

$$(\boldsymbol{y} + \Delta\boldsymbol{y}) = (\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{x},$$

for $\Delta\boldsymbol{y}, \Delta\boldsymbol{A}$ of minimal size. Rewrite this as

$$\begin{aligned}
(\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{x} - (\boldsymbol{y} + \Delta\boldsymbol{y}) &= \boldsymbol{0} \\
\Rightarrow \begin{bmatrix} \boldsymbol{A} + \Delta\boldsymbol{A} & \boldsymbol{y} + \Delta\boldsymbol{y} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ -1 \end{bmatrix} &= \boldsymbol{0} \\
\Rightarrow (\boldsymbol{C} + \boldsymbol{\Delta}) \begin{bmatrix} \boldsymbol{x} \\ -1 \end{bmatrix} &= \boldsymbol{0}
\end{aligned}$$

where

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{y} \end{bmatrix}, \quad \boldsymbol{\Delta} = \begin{bmatrix} \Delta\boldsymbol{A} & \Delta\boldsymbol{y} \end{bmatrix}.$$

Note that both $\boldsymbol{C}$ and $\boldsymbol{\Delta}$ are $M \times (N+1)$ matrices.

The result of the progression of equations above says that we are looking for a $\boldsymbol{\Delta}$ (of minimal size) such that there is a vector $\begin{bmatrix} \boldsymbol{x} \\ -1 \end{bmatrix}$ in the nullspace of $\boldsymbol{C} + \boldsymbol{\Delta}$. Since

$$\boldsymbol{v} \in \text{Null}(\boldsymbol{C} + \boldsymbol{\Delta}) \Leftrightarrow \alpha\boldsymbol{v} \in \text{Null}(\boldsymbol{C} + \boldsymbol{\Delta}) \quad \text{for all} \quad \alpha \in \mathbb{R},$$

and $\boldsymbol{x}$ in arbitrary, we are really just asking that $\boldsymbol{C} + \boldsymbol{\Delta}$ has a nullspace; as long as there is at least one vector in the nullspace

whose last entry is nonzero, we can find a vector of the required form just by normalizing. In short, this means that our task is to find $\boldsymbol{\Delta}$ such that the $M \times (N+1)$ matrix $\boldsymbol{C} + \boldsymbol{\Delta}$ is **rank deficient**, that is $\text{rank}(\boldsymbol{C} + \boldsymbol{\Delta}) < N + 1$.

Put another way, we want to solve the optimization program

$$\underset{\boldsymbol{\Delta}}{\text{minimize}} \ \|\boldsymbol{\Delta}\|_F^2 \quad \text{subject to} \quad \text{rank}(\boldsymbol{C} + \boldsymbol{\Delta}) = N.$$

Making the substitution $\boldsymbol{X} = \boldsymbol{C} + \boldsymbol{\Delta}$, this is equivalent to solving

$$\underset{\boldsymbol{X}}{\text{minimize}} \ \|\boldsymbol{C} - \boldsymbol{X}\|_F^2 \quad \text{subject to} \quad \text{rank}(\boldsymbol{X}) = N,$$

and then taking $\widehat{\boldsymbol{\Delta}} = \widehat{\boldsymbol{X}} - \boldsymbol{C}$.

This is a low-rank approximation problem.[1] We now consider how to solve problems of this form (which arise in many other important contexts).

The SVD and Matrix Approximation

Let $\boldsymbol{A}$ be an $M \times N$ matrix with rank R. We are often interested in determining the closest matrix to $\boldsymbol{A}$ that has rank r[2]? More precisely, we would like to solve

$$\underset{\boldsymbol{X}}{\text{minimize}} \ \|\boldsymbol{A} - \boldsymbol{X}\|_F^2 \quad \text{subject to} \quad \text{rank}(\boldsymbol{X}) = r. \tag{1}$$

The functional above is standard least-squares, but the constraint set (the set of rank-r matrices) is a complicated entity. Nevertheless, as with many things in this class, the SVD reveals the solution immediately.

[1] Or at least a "lower rank" approximation problem.

[2] We will assume that $r < R$, as for $r = R$ the answer is easy, and for $R < r \leq \min(M, N)$ the question is not well-posed.

Low-rank approximation.
Let $\boldsymbol{A}$ be a matrix with SVD

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = \sum_{p=1}^{R} \sigma_p \boldsymbol{u}_p \boldsymbol{v}_p^{\mathrm{T}}.$$

Then (1) is solved simply by truncating the SVD:

$$\widehat{\boldsymbol{X}} = \sum_{p=1}^{r} \sigma_p \boldsymbol{u}_p \boldsymbol{v}_p^{\mathrm{T}} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^{\mathrm{T}},$$

where $\boldsymbol{U}_r$ contains the first r columns of $\boldsymbol{U}$, $\boldsymbol{V}_r$ contains the first r columns of $\boldsymbol{V}$, and $\boldsymbol{\Sigma}_r$ is the first r columns and r rows of $\boldsymbol{\Sigma}$.

The framed result above, known as the Eckart-Young theorem, is an immediate consequence of the following lemma.

Subspace Approximation Lemma. For fixed $\boldsymbol{A}$ with SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, the optimization program

$$\underset{\substack{\boldsymbol{Q}:M\times r \\ \boldsymbol{\Theta}:r\times N}}{\text{minimize}} \ \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{\Theta}\|_F^2 \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}, \tag{2}$$

has solution

$$\widehat{\boldsymbol{Q}} = \boldsymbol{U}_r,$$
$$\widehat{\boldsymbol{\Theta}} = \boldsymbol{U}_r^{\mathrm{T}}\boldsymbol{A},$$

where $\boldsymbol{U}_r = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_r \end{bmatrix}$ contains the first r columns of $\boldsymbol{U}$.

We prove this lemma in the technical details section at the end of the notes. To see how it implies the Eckart-Young theorem, we can

interpret the search over $M \times r$ matrices $\boldsymbol{Q}$ with orthonormal columns as a search over all possible *column spaces* of dimension r. Then the search over $\boldsymbol{\Theta}$ finds the best linear combinations in a column spaces to approximate the columns of $\boldsymbol{A}$. Since any rank-r matrix can be represented this way, the optimization program (2) is equivalent to (1); if $\widehat{\boldsymbol{Q}}, \widehat{\boldsymbol{\Theta}}$ solve (2), then $\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{Q}}\widehat{\boldsymbol{\Theta}}$ solves (1).

Also note that

$$\widehat{\boldsymbol{\Theta}} = \boldsymbol{U}_r^{\mathrm{T}} \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^{\mathrm{T}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \boldsymbol{\Sigma} \boldsymbol{V}^{\mathrm{T}},$$

where $\mathbf{I}$ is the $r \times r$ identity matrix, and $\mathbf{0}$ is a $r \times (R - r)$ matrix of zeros. This matrix of all zeros has the same effect as removing all but the first r terms along the diagonal of $\boldsymbol{\Sigma}$ and all but the first r rows of $\boldsymbol{V}^{\mathrm{T}}$. Thus

$$\widehat{\boldsymbol{Q}}\widehat{\boldsymbol{\Theta}} = \boldsymbol{U}_r \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \boldsymbol{\Sigma} \boldsymbol{V}^{\mathrm{T}} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^{\mathrm{T}}.$$

What is the error between $\boldsymbol{A}$ and its best rank-r approximation $\widehat{\boldsymbol{A}}$? Well,

$$\boldsymbol{A} - \widehat{\boldsymbol{A}} = \sum_{p=r+1}^{R} \sigma_p \boldsymbol{u}_p \boldsymbol{v}_p^{\mathrm{T}},$$

and so the error matrix has singular values $\sigma_{r+1}, \ldots, \sigma_R$. Since the Frobenius norm (squared) can be calculated by summing the squares of the singular values,

$$\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F^2 = \sum_{p=r+1}^{R} \sigma_p^2.$$

Solving TLS – Part II

Recall that the TLS problem reduced to solving a problem of the form

$$\underset{\boldsymbol{X}}{\text{minimize}} \ \|\boldsymbol{C}-\boldsymbol{X}\|_F^2 \quad \text{subject to} \quad \text{rank}(\boldsymbol{X}) = N,$$

and then taking $\widehat{\boldsymbol{\Delta}} = \widehat{\boldsymbol{X}} - \boldsymbol{C}$. In light of our discussion above, we take the SVD of $\boldsymbol{C}$,

$$\boldsymbol{C} = \boldsymbol{W}\boldsymbol{\Gamma}\boldsymbol{Z}^{\mathrm{T}} = \sum_{n=1}^{N+1} \gamma_n \boldsymbol{w}_n \boldsymbol{z}_n^{\mathrm{T}},$$

and create $\widehat{\boldsymbol{X}}$ by leaving out the last term in the sum above[3]:

$$\widehat{\boldsymbol{X}} = \sum_{n=1}^{N} \gamma_n \boldsymbol{w}_n \boldsymbol{z}_n^{\mathrm{T}}.$$

Then

$$\widehat{\boldsymbol{\Delta}} = \widehat{\boldsymbol{X}} - \boldsymbol{C} = -\gamma_{N+1}\boldsymbol{w}_{N+1}\boldsymbol{z}_{N+1}^{\mathrm{T}}.$$

Now we are ready to construct the actual estimate $\widehat{\boldsymbol{x}}$. Recall that we want a vector such that

$$(\boldsymbol{C}+\widehat{\boldsymbol{\Delta}})\begin{bmatrix}\boldsymbol{x}\\ -1\end{bmatrix} = \boldsymbol{0}, \quad \text{meaning} \quad \widehat{\boldsymbol{X}}\begin{bmatrix}\boldsymbol{x}\\ -1\end{bmatrix} = \boldsymbol{0}.$$

The null space of $\widehat{\boldsymbol{X}}$ is (by construction) simply the span of $\boldsymbol{z}_{N+1}$, meaning we need to find a scalar α such that

$$\begin{bmatrix}\boldsymbol{x}\\ -1\end{bmatrix} = \alpha\, \boldsymbol{z}_{N+1}.$$

[3] If $\boldsymbol{C}$ has fewer than $N+1$ non-zero singular values, then it is already rank deficient, and we can take $\widehat{\boldsymbol{X}} = \boldsymbol{C} \Rightarrow \widehat{\boldsymbol{\Delta}} = \boldsymbol{0}$.

Thus we can take

$$\widehat{\boldsymbol{x}}_{\text{TLS}} = \frac{-1}{z_{N+1}[N+1]} \cdot \begin{bmatrix} z_{N+1}[1] \\ z_{N+1}[2] \\ \vdots \\ z_{N+1}[N] \end{bmatrix}.$$

If it happens that $z_{N+1}(N+1) = 0$, this means $\widehat{\Delta \boldsymbol{y}} = \boldsymbol{0}$, and we would need an $\boldsymbol{x}$ such that

$$(\boldsymbol{A} + \widehat{\Delta \boldsymbol{A}})\boldsymbol{x} = \boldsymbol{y}.$$

Such an $\boldsymbol{x}$ may or may not exist (and probably doesn't), so in this case there is no TLS solution.

In the special case where the smallest singular value of $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{y} \end{bmatrix}$ is not unique, i.e.

$$\gamma_1 \geq \gamma_2 \geq \gamma_q > \gamma_{q+1} = \gamma_{q+2} = \cdots = \gamma_{N+1}, \quad \text{for some } \; q < N,$$

then the TLS solution may not be unique. We take

$$\boldsymbol{Z}' = \begin{bmatrix} \boldsymbol{z}_{q+1} & \boldsymbol{z}_{q+2} & \cdots & \boldsymbol{z}_{N+1} \end{bmatrix},$$

and try to find a vector in the span that has the right form; any vector $\boldsymbol{x}$ such that

$$\begin{bmatrix} \boldsymbol{x} \\ -1 \end{bmatrix} \in \text{Span}\left(\{\boldsymbol{z}_{q+1}, \ldots, \boldsymbol{z}_{N+1}\}\right)$$

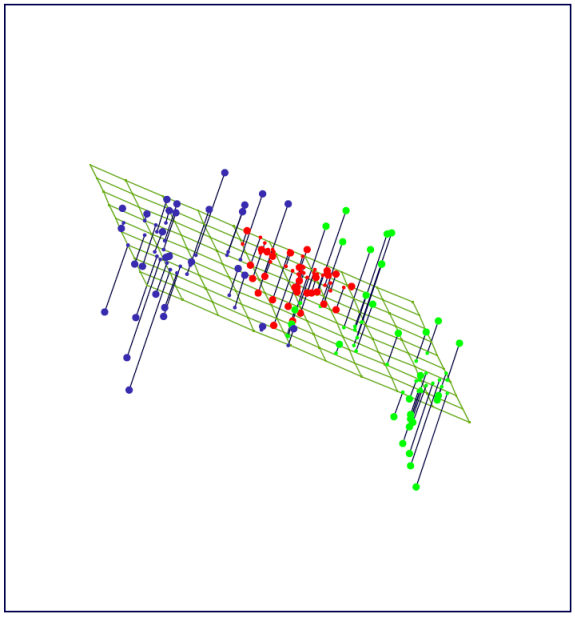
is equally good. All we need is a $\boldsymbol{\beta}$ such that the last entry of $\boldsymbol{Z}'\boldsymbol{\beta}$ is equal to -1.

Principal Components Analysis

Principal Components Analysis (PCA) is a standard technique for **dimensionality reduction** of data sets. It is a way to automatically find a **subspace** which approximates the data. It is used everywhere in signal processing, machine learning, and statistics.

There are actually two equivalent ways to think about PCA. The first is statistical: we are trying to find a transform that is carefully tuned to the (second-order) statistics of the data. The second perspective, which is what we will adopt in this course, is more geometrical: given a set of vectors, we are trying to find a subspace of a certain dimension that comes closest to containing this set.

Specifically, suppose that we have data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$, and want to find the K-dimensional affine space (subspace plus offset) that comes closest to containing them. Here is a picture[4]



[4]From Ch. 14 of Tibshirani and Hastie's *Elements of Statistical Learning*.

Our goal is to find an offset $\boldsymbol{\mu} \in \mathbb{R}^D$ and a matrix $\boldsymbol{Q}$ with orthonormal columns such that

$$\boldsymbol{x}_n \approx \boldsymbol{\mu} + \boldsymbol{Q}\boldsymbol{\theta}_n \quad \text{for all} \;\; n = 1, \ldots, N,$$

for some $\boldsymbol{\theta}_n \in \mathbb{R}^K$. We cast this as the following optimization problem. Given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, solve

$$\underset{\boldsymbol{\mu}, \boldsymbol{Q}, \{\boldsymbol{\theta}_n\}}{\text{minimize}} \;\; \sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{\mu} - \boldsymbol{Q}\boldsymbol{\theta}_n\|_2^2 \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}.$$

Note that if we fix $\boldsymbol{\mu}$ and define $\widetilde{\boldsymbol{x}}_n = \boldsymbol{x}_n - \mu$, then we can recast the optimization with respect to $\boldsymbol{Q}$ and the $\boldsymbol{\theta}_n$ as

$$\underset{\boldsymbol{Q}, \{\boldsymbol{\theta}_n\}}{\text{minimize}} \;\; \sum_{n=1}^{N} \|\widetilde{\boldsymbol{x}}_n - \boldsymbol{Q}\boldsymbol{\theta}_n\|_2^2 \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}.$$

If $\widetilde{\boldsymbol{X}}$ and $\boldsymbol{\Theta}$ denote the matrices whose columns are given by $\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_n$ and $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ respectively, then we can also write this as

$$\underset{\substack{\boldsymbol{Q}:D\times K \\ \boldsymbol{\Theta}:K\times N}}{\text{minimize}} \;\; \|\widetilde{\boldsymbol{X}} - \boldsymbol{Q}\boldsymbol{\Theta}\|_F^2 \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}.$$

This is exactly the optimization problem that we looked at previously in our Subspace Approximation Lemma! Thus the solution is given by computing the SVD of $\widetilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$ and then taking as our solution

$$\begin{aligned} \widehat{\boldsymbol{Q}} &= \boldsymbol{U}_K, \\ \widehat{\boldsymbol{\Theta}} &= \boldsymbol{U}_K^{\mathrm{T}}\widetilde{\boldsymbol{X}}, \end{aligned}$$

where $\boldsymbol{U}_K = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_K \end{bmatrix}$ contains the first K columns of $\boldsymbol{U}$.

Finally, let us return to the question of how to set $\boldsymbol{\mu}$. For any given $\boldsymbol{\mu}$, the solution for $\boldsymbol{Q}$ and $\boldsymbol{\Theta}$ is given by the Subspace Approximation Lemma. This results in setting

$$\boldsymbol{\theta}_n = \boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{x}_n - \boldsymbol{\mu}).$$

Plugging this in for $\boldsymbol{\theta}_n$ in our objective function, we have that

$$\begin{aligned}\boldsymbol{x}_n - \boldsymbol{\mu} - \boldsymbol{Q}\boldsymbol{\theta}_n &= \boldsymbol{x}_n - \boldsymbol{\mu} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{x}_n - \boldsymbol{\mu}) \\ &= (\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})(\boldsymbol{x}_n - \boldsymbol{\mu}).\end{aligned}$$

Hence, the problem of selecting $\boldsymbol{\mu}$ reduces to the optimization problem

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \ \sum_{n=1}^{N} \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})(\boldsymbol{x}_n - \boldsymbol{\mu})\|_2^2$$

The vector $\boldsymbol{\mu}$ is unconstrained; we can solve for the optimal $\boldsymbol{\mu}$ by taking a gradient and setting it equal to zero. To make this easier, note that

$$\|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})(\boldsymbol{x}_n - \boldsymbol{\mu})\|_2^2 = (\boldsymbol{x}_n - \boldsymbol{\mu})^T(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})(\boldsymbol{x}_n - \boldsymbol{\mu})$$

by simply expanding out the norm squared as an inner product and then using the fact that $\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}$ is a projector, i.e., it is symmetric and $(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})^2 = \mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}$. Thus, by taking a gradient and setting it equal to zero we have

$$\begin{aligned}\mathbf{0} &= -2\sum_{n=1}^{N}(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})(\boldsymbol{x}_n - \boldsymbol{\mu}) \\ &= -2(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})\left(\left(\sum_{n=1}^{N}\boldsymbol{x}_n\right) - N\boldsymbol{\mu}\right).\end{aligned}$$

We can satisfy this condition by taking the offset $\boldsymbol{\mu}$ to be the sample mean (average of all the observed vectors):

$$\widehat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n.$$

Note that this choice is not unique – any choice of $\boldsymbol{\mu}$ that results in $\sum(\boldsymbol{x}_n - \boldsymbol{\mu})$ living in the nullspace of $\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}$ would also suffice – but $\boldsymbol{\mu}$ is the easy and obvious choice, and also what is usually done in practice, because it makes computing the solution to the PCA problem straightforward.

Computing the PCA solution

Specifically, in practice you would typically proceed by first computing the mean $\widehat{\boldsymbol{\mu}}$ of your data as described above. Given $\widehat{\boldsymbol{\mu}}$, you can then form the matrix $\widetilde{\boldsymbol{X}}$ whose columns are given by

$$\widetilde{\boldsymbol{x}}_n = \boldsymbol{x}_n - \widehat{\boldsymbol{\mu}}.$$

Alternatively, if you know a priori that your columns of zero mean (or should have zero mean) based on the underlying process generating the data, then you can skip this step, setting $\widetilde{\boldsymbol{X}} = \boldsymbol{X}$.

In either case, once you have formed $\widetilde{\boldsymbol{X}}$, you simply compute the SVD of $\widetilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$ and then set

$$\widehat{\boldsymbol{Q}} = \boldsymbol{U}_K,$$
$$\widehat{\boldsymbol{\theta}}_n = \boldsymbol{U}_K^{\mathrm{T}}\widetilde{\boldsymbol{x}}_n,$$

where $\boldsymbol{U}_K = \begin{bmatrix}\boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_K\end{bmatrix}$ contains the first K columns of $\boldsymbol{U}$. We can think of $\widehat{\boldsymbol{\theta}}_n$ as a representation of $\boldsymbol{x}_n$ is a K-dimensional

subspace, with $\widehat{\boldsymbol{Q}}$ giving us a basis for that subspace (which is useful for projecting vectors $\boldsymbol{x} \in \mathbb{R}^N$ into the subspace).

Note that if you look up a discussion of PCA in most textbooks or online, you will typically see a slightly different presentation. Specifically, most texts describe an approach to the problem that involves forming the matrix

$$\boldsymbol{S} = \sum_{n=1}^{N} (\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}})(\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}})^{\mathrm{T}},$$

taking and eigenvalue decomposition $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}}$, and then taking

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_K \end{bmatrix},$$

where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K$ are the eigenvectors of $\boldsymbol{S}$ corresponding to the K largest eigenvalues.

This approach is **completely equivalent** to our approach above.[5] The reason that PCA is typically presented in this was is that $\boldsymbol{S}$ can be interpreted as a scaled version of an empirical estimate of the covariance matrix for the underlying distribution generating the data. While this provides a nice connection with the other (statistical) interpretation of PCA, I personally find the SVD approach more intuitive. In PCA, we are simply trying to find a low-rank approximation to our dataset, which is directly and optimally handled by computing a truncated SVD.

[5] Recall the relationship between the SVD of $\widetilde{\boldsymbol{X}}$ and the eigendecomposition of $\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^{\mathrm{T}}$.

Technical Details: Subspace Approx. Lemma

We prove the subspace approximation lemma from above. First, with $\boldsymbol{Q}$ fixed, we can break the optimization over $\boldsymbol{\Theta}$ into a series of least-squares problems. Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N$ be the columns of $\boldsymbol{A}$, and $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ be the columns of $\boldsymbol{\Theta}$. Then

$$\underset{\boldsymbol{\Theta}}{\text{minimize}} \ \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{\Theta}\|_F^2$$

is exactly the same as

$$\underset{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_N}{\text{minimize}} \sum_{n=1}^{N} \|\boldsymbol{a}_n - \boldsymbol{Q}\boldsymbol{\theta}_n\|_2^2.$$

The above is our classic closest point problem, and is optimized by taking $\boldsymbol{\theta}_n = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{a}_n$ (since the columns of $\boldsymbol{Q}$ are orthonormal). Thus we can write the original problem (2) as

$$\underset{\boldsymbol{Q}:M\times r}{\text{minimize}} \ \sum_{n=1}^{N} \|\boldsymbol{a}_n - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{a}_n\|_2^2 \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I},$$

and then take $\widehat{\boldsymbol{\Theta}} = \widehat{\boldsymbol{Q}}^{\mathrm{T}}\boldsymbol{A}$.

Expanding the functional and using the fact that $(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})^2 = (\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})$, we have

$$\begin{aligned}\sum_{n=1}^{N} \|\boldsymbol{a}_n - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{a}_n\|_2^2 &= \sum_{n=1}^{N} \boldsymbol{a}_n^{\mathrm{T}}(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})\boldsymbol{a}_n \\ &= \sum_{n=1}^{N} \|\boldsymbol{a}_n\|_2^2 - \boldsymbol{a}_n^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{a}_n.\end{aligned}$$

Since the first term does not depend on $\boldsymbol{Q}$, our optimization program is equivalent to

$$\underset{\boldsymbol{Q}:M\times r}{\text{maximize}} \ \sum_{n=1}^{N} \boldsymbol{a}_n^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{a}_n \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}.$$

Now recall that for any vector $\boldsymbol{v}$, $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = \mathrm{trace}(\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}})$. Thus

$$\begin{aligned}
\sum_{n=1}^{N} \boldsymbol{a}_n\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{a}_n &= \sum_{n=1}^{N} \mathrm{trace}(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{a}_n\boldsymbol{a}_n^{\mathrm{T}}\boldsymbol{Q}) \\
&= \mathrm{trace}\left(\boldsymbol{Q}^{\mathrm{T}}\left(\sum_{n=1}^{N}\boldsymbol{a}_n\boldsymbol{a}_n^{\mathrm{T}}\right)\boldsymbol{Q}\right) \\
&= \mathrm{trace}\left(\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})\boldsymbol{Q}\right).
\end{aligned}$$

The matrix $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ has eigenvalue decomposition

$$\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^{\mathrm{T}},$$

where $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ come from the SVD of $\boldsymbol{A}$ (we will take $\boldsymbol{U}$ to be $M \times M$, possible adding zeros down the diagonal of $\boldsymbol{\Sigma}^2$). Now

$$\begin{aligned}
\mathrm{trace}\left(\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})\boldsymbol{Q}\right) &= \mathrm{trace}\left(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^{\mathrm{T}}\boldsymbol{Q}\right) \\
&= \mathrm{trace}\left(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}^2\boldsymbol{W}\right),
\end{aligned}$$

where $\boldsymbol{W} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{Q}$. Notice that $\boldsymbol{W}$ also has orthonormal columns, as $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}$. Thus our optimization program has become

$$\underset{\boldsymbol{W}:M\times r}{\text{maximize}} \ \mathrm{trace}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}^2\boldsymbol{W}) \quad \text{subject to} \quad \boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \mathbf{I}.$$

After we solve this, we can take any $\widehat{\boldsymbol{Q}}$ such that $\widehat{\boldsymbol{W}} = \boldsymbol{U}^{\mathrm{T}}\widehat{\boldsymbol{Q}}$.

This last optimization program is equivalent to a simple linear program that is solvable by inspection. Let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_r$ be the columns of $\boldsymbol{W}$. Then

$$\begin{aligned}
\operatorname{trace}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}^2\boldsymbol{W}) &= \sum_{p=1}^{r} \boldsymbol{w}_p^{\mathrm{T}}\boldsymbol{\Sigma}^2\boldsymbol{w}_p \\
&= \sum_{p=1}^{r}\sum_{m=1}^{M} |w_p[m]|^2\sigma_m^2 \\
&= \sum_{m=1}^{M} h[m]\sigma_m^2, \quad \text{where} \quad h[m] = \sum_{p=1}^{r} |w_p[m]|^2.
\end{aligned}$$

Notice that

$$h[m] = \sum_{p=1}^{r} |W[p,m]|^2$$

is a sum of the squares of a *row* of $\boldsymbol{W}$. Since the sum of the squares of every column of $\boldsymbol{W}$ is one, the sum of the squares of every entry in $\boldsymbol{W}$ must be r, and so

$$\sum_{m=1}^{M} h[m] = r.$$

It is clear that $h[m]$ is non-negative, but it also true that $h[m] \leq 1$. Here is why: since the columns of $\boldsymbol{W}$ are orthonormal, they can be considered as part of an orthonormal basis for $\mathbb{R}^M$. That is, there is a $M \times (M-r)$ matrix $\boldsymbol{W}_0$ such that the $M \times M$ matrix $\begin{bmatrix}\boldsymbol{W} & \boldsymbol{W}_0\end{bmatrix}$ has both orthonormal columns and orthonormal rows — thus the sum of the squares of each row are equal to one. Thus the sum of the squares of the first r entries cannot be larger than this.

Thus the maximum value trace($\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Sigma}^2\boldsymbol{W}$) can take is given by the linear program

$$\underset{\boldsymbol{h}\in\mathbb{R}^M}{\text{maximize}} \ \sum_{m=1}^{M} h[m]\sigma_m^2 \quad \text{subject to} \quad \sum_{m=1}^{M} h[m] = r, \ \ 0 \le h[m] \le 1.$$

We can intuit the answer to this program. Since all of the σ_m^2 and all of the $h[m]$ are positive, we want to have as much weight as possible assigned to the largest singular values. Since the weights are constrained to be less than 1, this simply means we "max out" the first r terms; the solution to the program above is

$$\widehat{h}[m] = \begin{cases} 1, & m = 1, \ldots, r \\ 0, & m = r+1, \ldots, M. \end{cases}$$

This means that the sum of the squares of the first r rows in $\widehat{\boldsymbol{W}}$ are equal to one, while the rest are zero. There might be many such matrices that fit this bill, but one of them is

$$\widehat{\boldsymbol{W}} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix},$$

where above, $\mathbf{I}$ is the $r \times r$ identity matrix, and $\mathbf{0}$ is a $(M-r) \times r$ matrix of all zeros. It is easy to see that choosing

$$\widehat{\boldsymbol{Q}} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots \boldsymbol{u}_r \end{bmatrix}$$

satisfies

$$\boldsymbol{U}^{\mathrm{T}}\widehat{\boldsymbol{Q}} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix}.$$