IV. Matrix Approximation using Least-Squares

The SVD and Matrix Approximation

We being with the following fundamental question. Let $\boldsymbol{A}$ be an $M \times N$ matrix with rank R. What is the closest matrix to $\boldsymbol{A}$ that has rank r[1]?

As before, we will use the Frobenius norm to measure the distance between two matrices:

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

Recall that $\|\boldsymbol{X}\|_F^2$ is also equal to the sum of the squares of the singular values of $\boldsymbol{X}$.

We can now formulate our problem as

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

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

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

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

[1] 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.

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

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

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, which we will actually use again later in this set of notes.

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

$$\begin{aligned} \hat{\boldsymbol{Q}} &= \boldsymbol{U}_r, \\ \hat{\boldsymbol{\Theta}} &= \boldsymbol{U}_r^{\mathrm{T}} \boldsymbol{A}, \end{aligned}$$

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 $\hat{\boldsymbol{Q}}, \hat{\boldsymbol{\Theta}}$ solve (2), then $\hat{\boldsymbol{A}} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{\Theta}}$ solves (1).

Also note that

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

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

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

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

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

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,

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

In what follows, we use this low-rank matrix approximation result to develop two fundamental tools: total least-squares, and principal components analysis.

Total Least-Squares

Our fundamental 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

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

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,

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

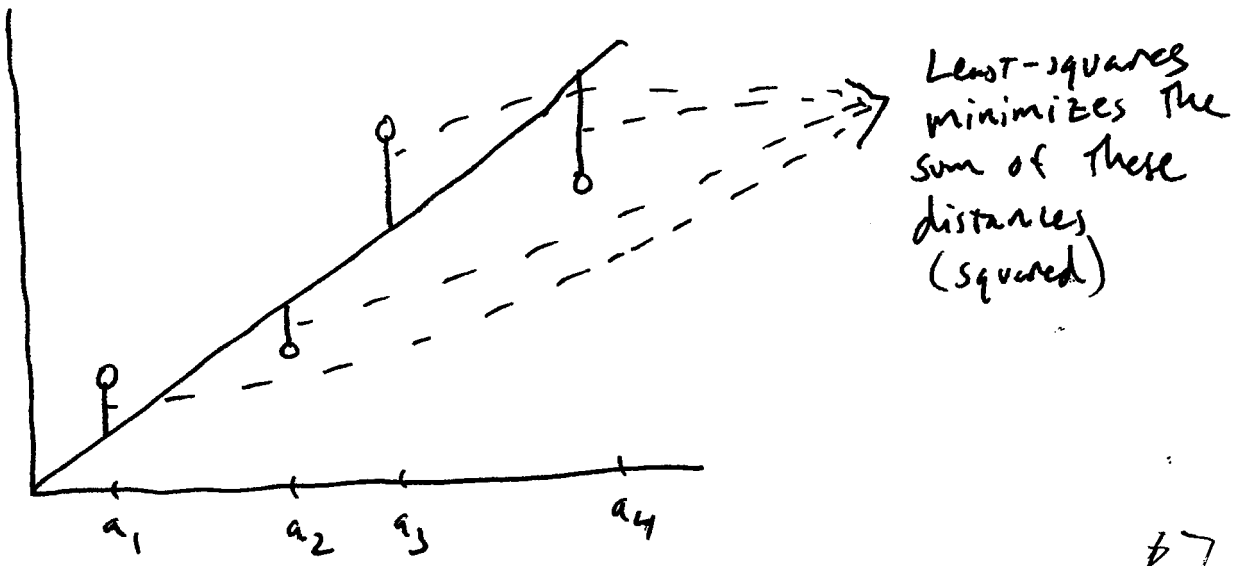
The solution is of course

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

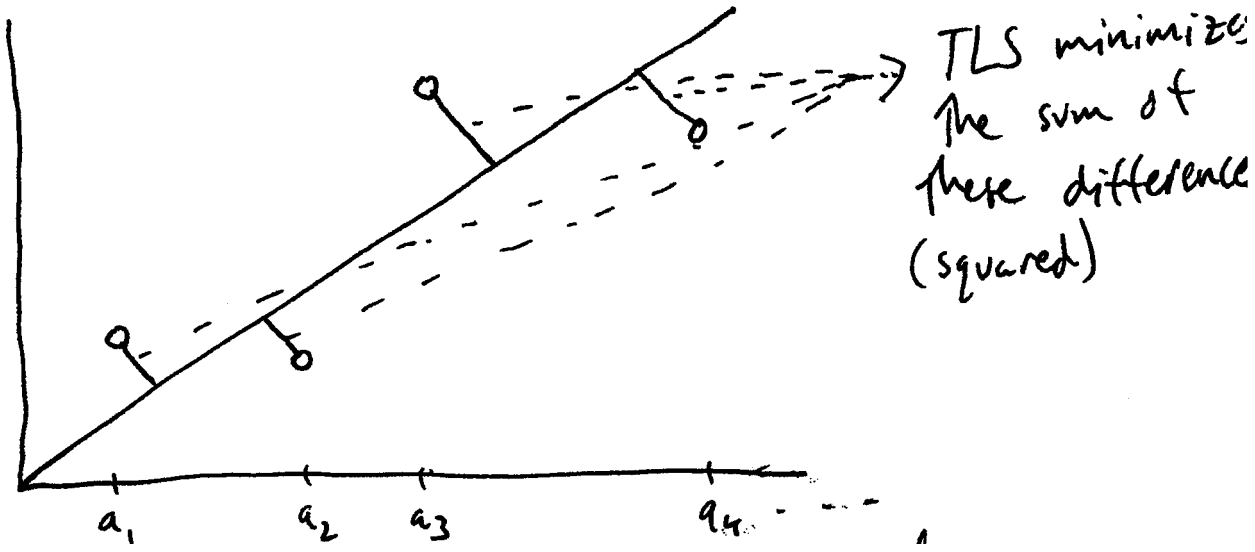
This solution minimizes the size of the residual

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

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

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 I 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 $\hat{\boldsymbol{\Delta}} = \hat{\boldsymbol{X}} - \boldsymbol{C}$.

This is a low-rank approximation problem[2], and we now know exactly how to solve it. 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 $\hat{\boldsymbol{X}}$ by leaving out the last term in the sum above[3]:

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

Then

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

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

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

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

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

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

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

Thus we can take

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

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 simplifying **linear relationships** in the data. It is used everywhere in signal processing, machine learning, and statistics, with applications including data compression, pattern recognition, and factor analysis.

There are two 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 is geometrical: given a set of vectors, we are trying to find a subspace of a certain dimension that comes closest to containing this set.

The Karhunen-Loeve Transform

The Karhunen-Loeve (KL) transform is an orthobasis that is tailored to the statistics of a class of random vectors. Suppose that $\boldsymbol{x} \in \mathbb{R}^D$

is random and has[4] mean and covariance

$$\mathrm{E}[\boldsymbol{x}] = \mathbf{0}, \quad \mathrm{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}] = \boldsymbol{R}.$$

Then the KL transform (or the KL basis) is simply the eigenvector $\boldsymbol{V}$ of $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}}$:

$$\boldsymbol{x} = \sum_{n=1}^{D} \alpha_n \boldsymbol{v}_n, \quad \alpha_n = \langle \boldsymbol{x}, \boldsymbol{v}_n \rangle.$$

This transform has the property that if we want to truncate the sum above (i.e. compress the vector by using fewer than D numbers to represent it), we get an error that is optimal in the mean-square error sense.

Let's set this problem up carefully. We want to find a subspace $\mathcal{T}$ of dimension K such that when we project $\boldsymbol{x}$ onto $\mathcal{T}$, we lose as little of $\boldsymbol{x}$ (in expectation) as possible. We want to solve

$$\underset{\mathcal{T}}{\text{minimize}}\, \mathrm{E}\left[\underset{\boldsymbol{t}\in\mathcal{T}}{\text{minimize}}\, \|\boldsymbol{x} - \boldsymbol{t}\|_2^2\right] \quad \text{subject to} \quad \dim(\mathcal{T}) = K.$$

For a fixed $\mathcal{T}$, we know how to solve the inner optimization program if we have an orthobasis, so we can re-write the above as a search of sets of K orthogonal vectors in $\mathbb{R}^D$:

$$\underset{\boldsymbol{Q}:D\times K}{\text{minimize}}\, \mathrm{E}\left[\|\boldsymbol{x} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{x}\|_2^2\right] \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \mathbf{I}.$$

[4] Modifying this discussion to vectors that are not zero-mean is straightforward.

Now notice that

$$\begin{aligned}
\mathrm{E}\left[\|\boldsymbol{x} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{x}\|_2^2\right] &= \mathrm{E}\left[\|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})\boldsymbol{x}\|_2^2\right] \\
&= \mathrm{E}[\mathrm{trace}((\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})] \\
&= \mathrm{trace}((\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})\,\mathrm{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}](\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}) \\
&= \mathrm{trace}((\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})\boldsymbol{R}(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}),
\end{aligned}$$

where in the second step above we have used the fact that for any vector $\boldsymbol{v}$, $\|\boldsymbol{v}\|_2^2 = \mathrm{trace}(\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}})$. Now notice that

$$\begin{aligned}
&\mathrm{trace}((\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})\boldsymbol{R}(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}) = \\
&\qquad \mathrm{trace}(\boldsymbol{R}) - 2\,\mathrm{trace}(\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{R}) + \mathrm{trace}(\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}).
\end{aligned}$$

We now apply three facts: $\mathrm{trace}(\boldsymbol{R})$ does not depend on $\boldsymbol{Q}$, $\mathrm{trace}(\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}) = \mathrm{trace}(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}) = K$ also does not depend on $\boldsymbol{Q}$, and

$$\mathrm{trace}(\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{R}) = \mathrm{trace}(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{Q}),$$

to transform into the equivalent program

$$\underset{\boldsymbol{W}:D\times K}{\text{maximize}} \;\; \mathrm{trace}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{W}) \quad \text{subject to} \quad \boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \mathbf{I}.$$

In the Technical Details section below, we show that this expression is maximized by taking

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

where the $\boldsymbol{v}_k$ correspond to the K eigenvectors of $\boldsymbol{R}$ corresponding to the K largest eigenvalues.

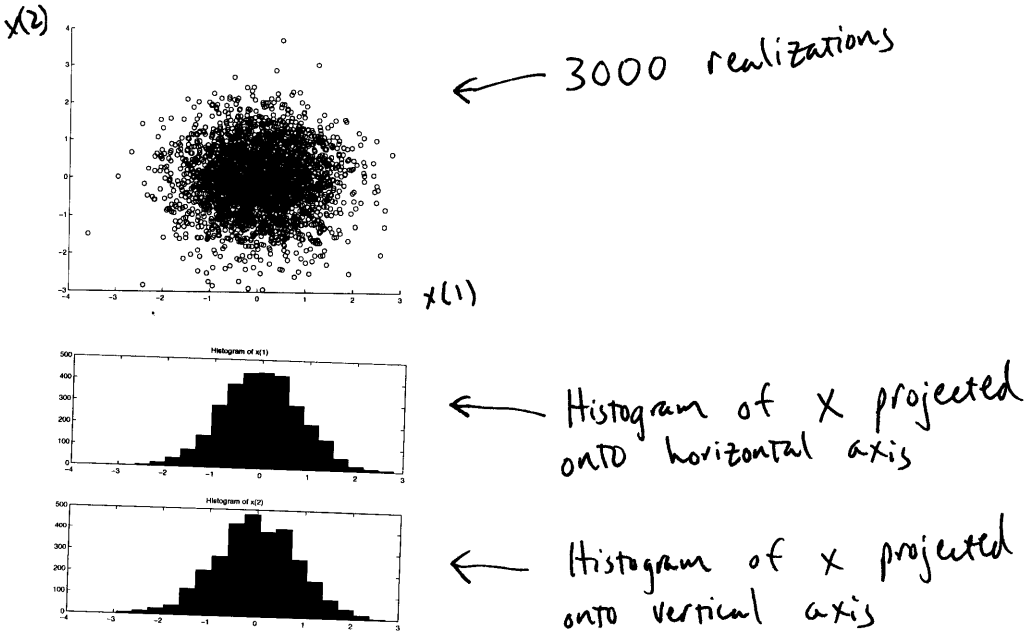
Moral: The best (in terms of mean-squared error) way to get a K term approximation of random data is to transform into the orthobasis formed by the eigenvectors of the covariance matrix, then

truncating the coefficients to K terms. This set of eigenvectors $\boldsymbol{V}$ is called the **KL transform**.

In some sense, the $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K$ are the K most important "features" of $\boldsymbol{x}$ — they are completely determined by the covariance matrix $\boldsymbol{R}$.

Examples in $\mathbb{R}^2$:

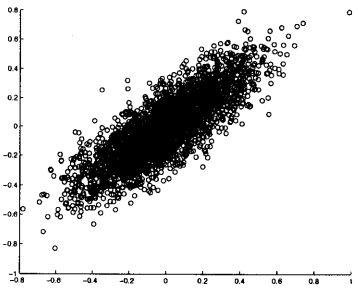
① $R = I$. A typical scatter plot will look like:

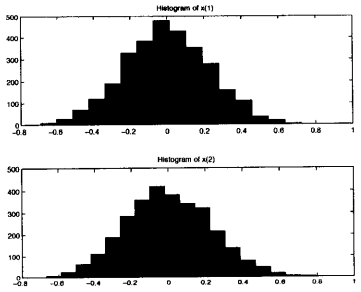


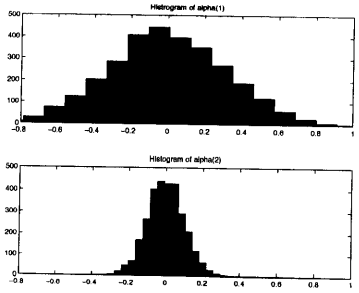
(2) $R = V \Lambda V^T$ $\quad \Lambda = \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix}$ with $\lambda_1 = 10\,\lambda_2$

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

scatter plot, 3000 realizations







← projected onto $v_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

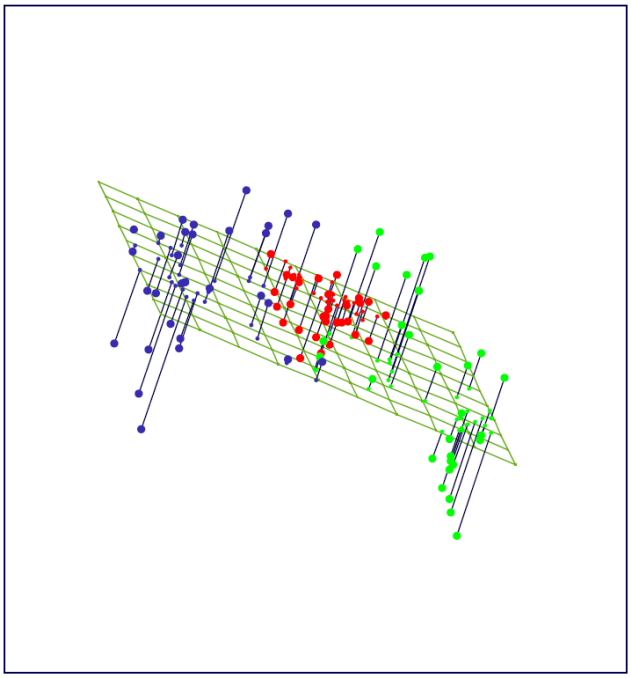
Histograms projected onto horiz. (top) and vert. (bottom) axes.

projected onto the principal axis $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

11

PCA on observed data

A very similar procedure to the above solves a common geometrical problem. Suppose that I have a bunch of data points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$, and I want to find the K-dimensional affine space (subspace plus offset) that comes closest to containing them. We don't even need to think of the data as random here; they are just points that we want to fit with a hyperplane. Here is a picture[5]



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

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

[5]This is pulled from Chapter 14 of Tibshirani and Hastie's *Elements of Statistical Learning*.

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

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

If we fix $\boldsymbol{\mu}$ and $\boldsymbol{Q}$, then by arguments very similar to those we have made before, the optimal $\boldsymbol{\theta}_n$ are given by

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

This means our objective reduces to solving

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

The offset $\boldsymbol{\mu}$ is uncontrained; if we again fix $\boldsymbol{Q}$, we can solve for the optimal $\boldsymbol{\mu}$ by taking a gradient and setting it equal to zero:

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

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

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

All that remains is solving for $\boldsymbol{Q}$. We have

$$\underset{\boldsymbol{Q}:D\times K}{\text{minimize}} \sum_{n=1}^{N} \|(\mathbf{I}-\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}})(\boldsymbol{x}_n-\hat{\boldsymbol{\mu}})\|_2^2 \quad \text{subject to} \quad \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}=\mathbf{I}.$$

Again, using an argument that perfectly parallels that in the Technical Details section below, this program is solved by forming

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

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

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

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

So even though we posed this problem as being purely geometrical, the answer parallels the statistical KL transform — we simply replace the "true" covariance matix $\boldsymbol{R}$ with the sample covariance $N^{-1}\boldsymbol{S}$.

Technical Details: Subspace Approx. Lemma

We prove the subspace approximation lemma from page 2. 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 $\hat{\boldsymbol{\Theta}} = \hat{\boldsymbol{Q}}^{\mathrm{T}}\boldsymbol{A}$.

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 $\hat{\boldsymbol{Q}}$ such that $\hat{\boldsymbol{W}} = \boldsymbol{U}^{\mathrm{T}}\hat{\boldsymbol{Q}}$.

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

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

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

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

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

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

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

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

satisfies

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