Stable Reconstruction with the Truncated SVD

We have seen that if $\boldsymbol{A}$ has very small singular values and we apply the pseudo-inverse in the presence of noise, the results can be disastrous. But it doesn't have to be this way. There are several ways to stabilize the pseudo-inverse. We start be discussing the simplest one, where we simply "cut out" the part of the reconstruction which is causing the problems.

As before, we are given noisy indirect observations of a vector $\boldsymbol{x}$ through a $M \times N$ matrix $\boldsymbol{A}$:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}. \tag{1}$$

The matrix $\boldsymbol{A}$ has SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, and pseudo-inverse $\boldsymbol{A}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\mathrm{T}}$.

At this point it is useful to recall that we could write the matrix $\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$ as a sum of outer products of the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$:

$$\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}} = \sum_{r=1}^{R} \boldsymbol{u}_r \boldsymbol{v}_r^{\mathrm{T}},$$

where R is the rank of $\boldsymbol{A}$, and $\boldsymbol{u}_r \in \mathbb{R}^M$ and $\boldsymbol{v}_r \in \mathbb{R}^N$ are columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. See the addendum at the end of these notes on matrix multiplication if this way of writing a matrix product seems unfamiliar.

Since $\boldsymbol{\Sigma}$ is diagonal, we can think of $\boldsymbol{U}\boldsymbol{\Sigma}$ as just rescaling the columns of $\boldsymbol{U}$, so that we can also rewrite $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ as a sum:

$$\boldsymbol{A} = \sum_{r=1}^{R} \sigma_r \boldsymbol{u}_r \boldsymbol{v}_r^{\mathrm{T}},$$

where the σ_r are the singular values. Similarly, we can write the pseudo-inverse as

$$\boldsymbol{A}^\dagger = \sum_{r=1}^{R} \frac{1}{\sigma_r} \boldsymbol{v}_r \boldsymbol{u}_r^{\mathrm{T}}.$$

Given $\boldsymbol{y}$ as above, we can write the least-squares estimate of $\boldsymbol{x}$ from the noisy measurements as

$$\widehat{\boldsymbol{x}}_{\mathrm{ls}} = \boldsymbol{A}^\dagger \boldsymbol{y} = \sum_{r=1}^{R} \frac{1}{\sigma_r} \langle \boldsymbol{y}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r. \tag{2}$$

As we can see (and have seen before) if any one of the σ_r are very small, the least squares reconstruction can be a disaster.

A simple way to avoid this is to simply truncate the sum (2), leaving out the terms where σ_r is too small ($1/\sigma_r$ is too big). Exactly how many terms to keep depends a great deal on the application, as there are competing interests. On the one hand, we want to ensure that each of the σ_r we include has an inverse of reasonable size, on the other, we want the reconstruction to be accurate (i.e., not to deviate from the noiseless least squares solution by too much).

We form an approximation $\boldsymbol{A}'$ to $\boldsymbol{A}$ by taking

$$\boldsymbol{A}' = \sum_{r=1}^{R'} \sigma_r \boldsymbol{u}_r \boldsymbol{v}_r^{\mathrm{T}},$$

for some $R' < R$. Again, our final answer will depend on which R' we use, and choosing R' is often times something of an art. It is clear that the approximation $\boldsymbol{A}'$ has rank R'. Note that the pseudo-inverse of $\boldsymbol{A}'$ is also a truncated sum

$$\boldsymbol{A}'^\dagger = \sum_{r=1}^{R'} \frac{1}{\sigma_r} \boldsymbol{v}_r \boldsymbol{u}_r^{\mathrm{T}}.$$

Given noisy data $\boldsymbol{y}$ as in (1), we reconstruct $\boldsymbol{x}$ by applying the truncated pseudo-inverse to $\boldsymbol{y}$:

$$\widehat{\boldsymbol{x}}_{\text{trunc}} = \boldsymbol{A}'^{\dagger}\boldsymbol{y} = \sum_{r=1}^{R'} \frac{1}{\sigma_r} \langle \boldsymbol{y}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r.$$

How good is this reconstruction? To answer this question, we will again compare it to the least squares reconstruction corresponding to "noiseless" measurements $\widehat{\boldsymbol{x}}_{\text{clean}} = \boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{x}$. The difference between these two is the reconstruction error (relative to $\widehat{\boldsymbol{x}}_{\text{clean}}$) as

$$\begin{aligned}
\widehat{\boldsymbol{x}}_{\text{trunc}} - \widehat{\boldsymbol{x}}_{\text{clean}} &= \boldsymbol{A}'^{\dagger}\boldsymbol{y} - \boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{x} \\
&= \boldsymbol{A}'^{\dagger}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}'^{\dagger}\boldsymbol{e} - \boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{x} \\
&= (\boldsymbol{A}'^{\dagger} - \boldsymbol{A}^{\dagger})\boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}'^{\dagger}\boldsymbol{e}.
\end{aligned}$$

Proceeding further, we can write the matrix $\boldsymbol{A}'^{\dagger} - \boldsymbol{A}^{\dagger}$ as

$$\boldsymbol{A}'^{\dagger} - \boldsymbol{A}^{\dagger} = \sum_{r=R'+1}^{R} -\frac{1}{\sigma_r} \boldsymbol{v}_r \boldsymbol{u}_r^{\mathrm{T}},$$

and so the first term in the reconstruction error can be written as

$$\begin{aligned}
(\boldsymbol{A}'^{\dagger} - \boldsymbol{A}^{\dagger})\boldsymbol{A}\boldsymbol{x} &= \sum_{r=R'+1}^{R} -\frac{1}{\sigma_r} \langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r \\
&= \sum_{r=R'+1}^{R} -\frac{1}{\sigma_r} \left\langle \sum_{j=1}^{R} \sigma_j \langle \boldsymbol{x}, \boldsymbol{v}_j \rangle \boldsymbol{u}_j, \boldsymbol{u}_r \right\rangle \boldsymbol{v}_r \\
&= \sum_{r=R'+1}^{R} -\frac{1}{\sigma_r} \sum_{j=1}^{R} \sigma_j \langle \boldsymbol{x}, \boldsymbol{v}_j \rangle \langle \boldsymbol{u}_j, \boldsymbol{u}_r \rangle \boldsymbol{v}_r \\
&= \sum_{r=R'+1}^{R} -\langle \boldsymbol{x}, \boldsymbol{v}_r \rangle \boldsymbol{v}_r \quad (\text{since } \langle \boldsymbol{u}_r, \boldsymbol{u}_j \rangle = 0 \text{ unless } j = r).
\end{aligned}$$

The second term in the reconstruction error can also be expanded against the $\boldsymbol{v}_r$:

$$\boldsymbol{A}'^{\dagger}\boldsymbol{e} = \sum_{r=1}^{R'} \frac{1}{\sigma_r} \langle \boldsymbol{e}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r.$$

Combining these expressions, the reconstruction error can be written

$$\begin{aligned} \widehat{\boldsymbol{x}}_{\text{trunc}} - \widehat{\boldsymbol{x}}_{\text{clean}} &= \underbrace{\sum_{r=1}^{R'} \frac{1}{\sigma_r} \langle \boldsymbol{e}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r}_{} + \underbrace{\sum_{r=R'+1}^{R} -\langle \boldsymbol{x}, \boldsymbol{v}_r \rangle \boldsymbol{v}_r}_{} \\ &= \text{Noise error} + \text{Approximation error.} \end{aligned}$$

Since the $\boldsymbol{v}_r$ are mutually orthogonal, and the two sums run over disjoint index sets, the noise error and the approximation error will be orthogonal. Also

$$\begin{aligned} \|\widehat{\boldsymbol{x}}_{\text{trunc}} - \widehat{\boldsymbol{x}}_{\text{clean}}\|_2^2 &= \|\text{Noise error}\|_2^2 + \|\text{Approximation error}\|_2^2 \\ &= \sum_{r=1}^{R'} \frac{1}{\sigma_r^2} |\langle \boldsymbol{e}, \boldsymbol{u}_r \rangle|^2 + \sum_{r=R'+1}^{R} |\langle \boldsymbol{x}, \boldsymbol{v}_r \rangle|^2. \end{aligned}$$

The reconstruction error, then, is signal dependent and will depend on how much of the vector $\boldsymbol{x}$ is concentrated in the subspace spanned by $\boldsymbol{v}_{R'+1}, \ldots, \boldsymbol{v}_R$. We will lose everything in this subspace. On the other hand, if it contains a significant part of $\boldsymbol{x}$, then there is not much least squares can do for you.

The worst-case noise error occurs when $\boldsymbol{e}$ is aligned with $\boldsymbol{u}_{R'}$:

$$\|\text{Noise error}\|_2^2 = \sum_{r=1}^{R'} \frac{1}{\sigma_r^2} |\langle \boldsymbol{e}, \boldsymbol{u}_r \rangle|^2 \leq \frac{1}{\sigma_{R'}^2} \cdot \|\boldsymbol{e}\|_2^2.$$

Stable Reconstruction using Tikhonov Regularization

Tikhonov[1] regularization is another way to stabilize the least squares recovery. It has the nice features that: 1) it can be interpreted using optimization, and 2) it can be computed without direct knowledge of the SVD of $\boldsymbol{A}$.

Recall that we motivated the pseudo-inverse by showing that $\widehat{\boldsymbol{x}}_{\text{ls}} = \boldsymbol{A}^{\dagger}\boldsymbol{y}$ is a solution to

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \ \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2. \tag{3}$$

When $\boldsymbol{A}$ has full column rank, $\widehat{\boldsymbol{x}}_{\text{ls}}$ is the unique solution, otherwise it is the solution with smallest energy. When $\boldsymbol{A}$ has full column rank but has singular values which are very small, huge variations in $\boldsymbol{x}$ (in directions of the singular vectors $\boldsymbol{v}_r$ corresponding to the tiny σ_r) can have very little effect on the residual $\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$. As such, the solution to (3) can have wildly inaccurate components in the presence of even mild noise.

One way to counteract this problem is to modify (3) with a **regularization** term that penalizes the size of the solution $\|\boldsymbol{x}\|_2^2$ as well as the residual error $\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$:

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \ \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\delta\|\boldsymbol{x}\|_2^2. \tag{4}$$

The parameter $\delta > 0$ gives us a trade-off between accuracy and regularization; we want to choose δ small enough so that the residual for the solution of (4) is close to that of (3), and large enough so that the problem is well-conditioned (i.e., stable in the presence of noise).

[1] Andrey Tikhonov (1906-1993) was a 20$^{\text{th}}$ century Russian mathematician.

Just as with (3), which is solved by applying the pseudo-inverse to $\boldsymbol{y}$, we can write the solution to (4) in closed form.

We have two ways of writing this solution. In terms of the SVD of $\boldsymbol{A}$, the minimizer of (4) is

$$\begin{aligned}\widehat{\boldsymbol{x}}_{\text{tik}} &= \boldsymbol{V}\widehat{\boldsymbol{\alpha}}_{\text{tik}} \\ &= \boldsymbol{V}(\boldsymbol{\Sigma}^2 + \delta\mathbf{I})^{-1}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{y}. \end{aligned} \tag{5}$$

Alternatively, we can write the solution in terms of $\boldsymbol{A}$ as

$$\widehat{\boldsymbol{x}}_{\text{tik}} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \delta\mathbf{I})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}. \tag{6}$$

Note that even if $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ is not invertible $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}+\delta\mathbf{I}$ always is (if $\delta > 0$). Thus, the expression (6) holds for all M, N, and R. It is also the case that

$$\widehat{\boldsymbol{x}}_{\text{tik}} = \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} + \delta\mathbf{I})^{-1}\boldsymbol{y}.$$

You will show that all of these expressions are equivalent on the next homework.

Tikhonov regularization is in some sense very similar to the truncated SVD, but with one significant advantage: because of the second way of writing the solution, we do not need to explicitly calculate the SVD to solve (4). The importance of not needing to explicitly compute the SVD is significant when we are solving large problems. When $\boldsymbol{A}$ is large ($M, N > 10^5$, say) it may be expensive or even impossible to construct the SVD and compute with it explicitly. However, if it has special structure (if it is sparse, for example), then it may take many fewer than MN operations to compute a matrix vector product $\boldsymbol{A}\boldsymbol{x}$.

In these situations, a **matrix free** iterative algorithm can be used to perform the inverse required in (6). A prominent example of such

an algorithm is **gradient descent** and its many variants, which we will see very soon.

We can get a better feel for what Tikhonov regularization is doing by comparing it directly to the pseudo-inverse. Recall that the least squares reconstruction $\widehat{\boldsymbol{x}}_{\text{ls}}$ can be written as

$$\widehat{\boldsymbol{x}}_{\text{ls}} = \sum_{r=1}^{R} \frac{1}{\sigma_r} \langle \boldsymbol{y}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r.$$

The Tikhonov reconstruction $\widehat{\boldsymbol{x}}_{\text{tik}}$ derived above is

$$\widehat{\boldsymbol{x}}_{\text{tik}} = \sum_{r=1}^{R} \frac{\sigma_r}{\sigma_r^2 + \delta} \langle \boldsymbol{y}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r. \tag{7}$$

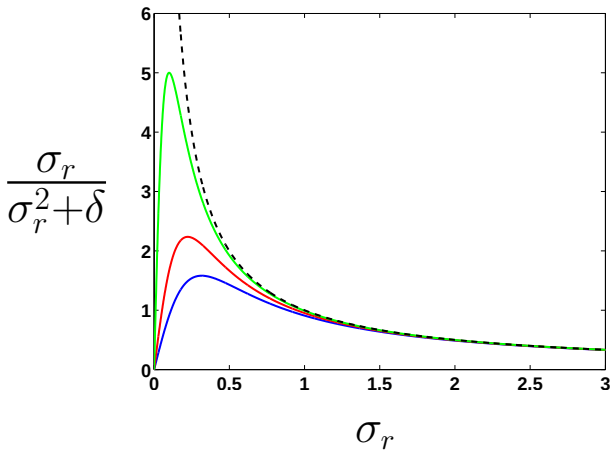
Notice that when σ_r is much larger than δ,

$$\frac{\sigma_r}{\sigma_r^2 + \delta} \approx \frac{1}{\sigma_r}, \quad \sigma_r \gg \delta,$$

but when σ_r is small

$$\frac{\sigma_r}{\sigma_r^2 + \delta} \approx 0, \quad \sigma_r \ll \delta.$$

Thus the Tikhonov reconstruction modifies the important parts (components where the σ_r are large) of the pseudo-inverse very little, while ensuring that the unimportant parts (components where the σ_r are small) affect the solution only by a very small amount. This **damping** of the singular values, is illustrated below.



Above, we see the damped multipliers $\sigma_r/(\sigma_r^2 + \delta)$ versus σ_r for $\delta = 0.1$ (blue), $\delta = 0.05$ (red), and $\delta = 0.01$ (green). The black dotted line is $1/\sigma_r$, the least squares multiplier. Notice that for large σ_r ($\sigma_r > 2\sqrt{\delta}$, say), the damping has almost no effect.

This damping makes the Tikhonov reconstruction exceptionally stable; large multipliers never appear in the reconstruction (7). In fact it is easy to check that

$$\frac{\sigma_r}{\sigma_r^2 + \delta} \leq \frac{1}{2\sqrt{\delta}}$$

no matter the value of σ_r.

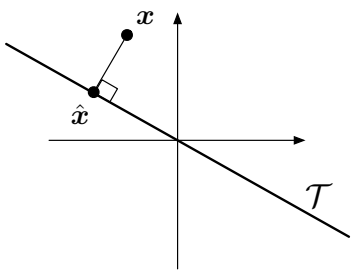
You can perform a very similar kind of noise analysis for Tikhonov reconstruction as we just did for the truncated SVD, but we will leave you to do this at home.

A geometric perspective on least squares

Before we finally move on to talk a bit more about practical algorithms, lets see one more way in which the SVD can help us to think about what least squares is doing.

Specifically, another way to think about least squares involves a simple geometric problem. Suppose we are given a vector $\boldsymbol{x} \in \mathbb{R}^M$ and a subspace[2] $\mathcal{T} = \text{span}(\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N\})$ (where $N < M$).

If $\boldsymbol{x}$ does not already live in the span of $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N$, we might ask "what is the closest point $\widehat{\boldsymbol{x}} \in \mathcal{T}$ to $\boldsymbol{x}$? This is illustrated below:



Mathematically, we want to find the $\widehat{\boldsymbol{x}} \in \mathcal{T}$ that minimizes $\|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|_2$, i.e., given $\boldsymbol{x}$, we want to solve the optimization program

$$\underset{\boldsymbol{y} \in \mathbb{R}^M}{\text{minimize}} \ \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \qquad \text{subject to} \quad \boldsymbol{y} \in \mathcal{T}.$$

This might initially seem different than what we've been considering so far – it looks like a least squares problem, but we have the constraint that $\boldsymbol{y}$ lives in a subspace. However, think for a moment about what the constraint $\boldsymbol{y} \in \mathcal{T}$ actually means.

[2]Recall that a subspace of $\mathbb{R}^M$ is just a set of vectors that can be thought of as a vector space in its own right, meaning that any linear combination of vectors in the subspace produces another vector in that same subspace. The span of $N < M$ vectors in $\mathbb{R}^M$, such as a 2D plane in 3 dimensions, is the canonical example of a subspace.

If $\boldsymbol{y} \in \mathcal{T}$, then $\boldsymbol{y}$ can be written as a linear combination of the vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N$. If $\boldsymbol{A}$ is the $M \times N$ matrix with columns given by $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N$, this is equivalent to saying that we can write $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathbb{R}^N$. Thus, instead of optimizing over $\boldsymbol{y} \in \mathcal{T}$ we can simply optimize over $\boldsymbol{\alpha}$ in the equivalent problem

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^N}{\text{minimize}} \ \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\alpha}\|_2^2.$$

Note that this is simply the standard least squares problem we have been studying for a while now, with solution $\widehat{\boldsymbol{\alpha}} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}$. This gives us a solution for $\widehat{\boldsymbol{x}}$ of our original problem of

$$\begin{aligned}
\widehat{\boldsymbol{x}} &= \boldsymbol{A}\widehat{\boldsymbol{\alpha}} \\
&= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} \\
&= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} \\
&= \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}.
\end{aligned}$$

Note that in the picture on the previous page, I indicated that the vector $\widehat{\boldsymbol{x}} - \boldsymbol{x}$ was orthogonal to the line that represents the subspace $\mathcal{T}$. This is a general fact for least squares solutions: at the optimal $\widehat{\boldsymbol{x}}$, we always have that the error $\widehat{\boldsymbol{x}} - \boldsymbol{x}$ is orthogonal to $\mathcal{T} = \mathcal{R}(\boldsymbol{A})$.

This is easy to see using the SVD. Since for any $\boldsymbol{y} \in \mathcal{R}(\boldsymbol{A})$ we can write $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\alpha}$, we have

$$\begin{aligned}
\langle \widehat{\boldsymbol{x}} - \boldsymbol{x}, \boldsymbol{y} \rangle &= \langle \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{x}, \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\alpha} \rangle \\
&= \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} \\
&= \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} \\
&= 0,
\end{aligned}$$

for any $\boldsymbol{\alpha} \in \mathbb{R}^N$.

The fact that the optimal approximation to $\boldsymbol{x}$ in $\mathcal{T}$, $\widehat{\boldsymbol{x}}$ results in an error $\widehat{\boldsymbol{x}} - \boldsymbol{x}$ that is orthogonal to $\mathcal{T}$ is often called the **orthogonality principle**. In fact, you can actually start a discussion of least squares by first showing that the orthogonality principle must be true, and then using this fact to derive the solution to the least squares problem.

Another piece of terminology you may encounter is that $\widehat{\boldsymbol{x}}$ is the **orthogonal projection** of $\boldsymbol{x}$ onto $\mathcal{T}$. In general, when people talk about an orthogonal projection, this is simply shorthand for taking a vector and finding the closest point in some subspace $\mathcal{T}$ to $\boldsymbol{x}$. The fact that you do this by finding a point in $\mathcal{T}$ from which the direction to $\boldsymbol{x}$ is orthogonal to $\mathcal{T}$ is why it gets the name "orthogonal".

A note on matrix multiplication

Although we have encountered some of these concepts already, now is a good time to review some basic notions involving how we think about matrix multiplication. When dealing with simple matrix-vector products, these ideas are straightforward, but as we get to increasingly complex matrix factorizations, it can be harder to break down what is happening.

Matrix-vector multiplication

Let's begin by considering simple matrix-vector multiplication $\boldsymbol{A}\boldsymbol{x}$. There are really two ways to think about this. First, suppose that $\boldsymbol{A}$ is an $M \times N$ matrix. We can think of $\boldsymbol{A}$ as the concatenation of N columns, denoted by $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N$:

$$\boldsymbol{A} = \begin{bmatrix} | & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_N \\ | & | & & | \end{bmatrix}.$$

The first way of thinking of a matrix-vector multiplication is that $\boldsymbol{A}\boldsymbol{x}$ is a weighted combination of the columns of $\boldsymbol{A}$:

$$\begin{bmatrix} | & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_N \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2 + \cdots + x_N\boldsymbol{a}_N.$$

To describe the second way of thinking of a matrix-vector multiplication, we are going to slightly abuse our notation. Even though we just said $\boldsymbol{a}_n$ was a column of $\boldsymbol{A}$, which is what we will do for the rest

of the course, note that we can also break up a matrix by along its rows, which we will (just for now) denote by $\boldsymbol{a}_1^{\mathrm{T}}, \boldsymbol{a}_2^{\mathrm{T}}, \ldots, \boldsymbol{a}_M^{\mathrm{T}}$:

$$\boldsymbol{A} = \begin{bmatrix} — & \boldsymbol{a}_1^{\mathrm{T}} & — \\ — & \boldsymbol{a}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \boldsymbol{a}_M^{\mathrm{T}} & — \end{bmatrix}.$$

We use a transpose when referring to the rows since normally $\boldsymbol{a}_m$ always refers to a column vector. This way of writing $\boldsymbol{A}$ suggests an alternative way to think about matrix-vector multiplication. That is, each entry of $\boldsymbol{Ax}$ is the inner product between the rows of $\boldsymbol{A}$ and the vector $\boldsymbol{x}$:

$$\begin{bmatrix} — & \boldsymbol{a}_1^{\mathrm{T}} & — \\ — & \boldsymbol{a}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \boldsymbol{v}_M^{\mathrm{T}} & — \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{x} \\ | \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{x} \\ \boldsymbol{a}_2^{\mathrm{T}}\boldsymbol{x} \\ \vdots \\ \boldsymbol{a}_N^{\mathrm{T}}\boldsymbol{x} \end{bmatrix}.$$

Matrix-matrix multiplication

Likewise, the product of an $M \times N$ matrix $\boldsymbol{A}$ and a $N \times P$ matrix $\boldsymbol{B}$ can be thought of in two different ways. The traditional way (how you would compute the elements of $\boldsymbol{AB}$ one entry at a time) is as a collection of the inner products between all of the rows of $\boldsymbol{A}$ and all of the columns of $\boldsymbol{B}$. If we continue to be a little loose with our notation, we can visualize this as:

$$\begin{bmatrix} — & \boldsymbol{a}_1^{\mathrm{T}} & — \\ — & \boldsymbol{a}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \boldsymbol{a}_M^{\mathrm{T}} & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_P \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{b}_1 & \boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{b}_P \\ \boldsymbol{a}_2^{\mathrm{T}}\boldsymbol{b}_1 & \boldsymbol{a}_2^{\mathrm{T}}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_2^{\mathrm{T}}\boldsymbol{b}_P \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a}_M^{\mathrm{T}}\boldsymbol{b}_1 & \boldsymbol{a}_M^{\mathrm{T}}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_M^{\mathrm{T}}\boldsymbol{b}_P \end{bmatrix}.$$

However, just as with matrix-vector multiplication, there are alternative perspectives. We think of each column of $\boldsymbol{AB}$ as being the result of a matrix-vector product that combines the columns of $\boldsymbol{A}$:

$$\boldsymbol{A}\begin{bmatrix} | & | & & | \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_P \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \boldsymbol{A}\boldsymbol{b}_1 & \boldsymbol{A}\boldsymbol{b}_2 & \cdots & \boldsymbol{A}\boldsymbol{b}_P \\ | & | & & | \end{bmatrix}.$$

Similarly, we can also think of each row of $\boldsymbol{AB}$ as a linear combination of the rows of $\boldsymbol{B}$:

$$\begin{bmatrix} — & \boldsymbol{a}_1^{\mathrm{T}} & — \\ — & \boldsymbol{a}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \boldsymbol{a}_M^{\mathrm{T}} & — \end{bmatrix}\boldsymbol{B} = \begin{bmatrix} — & \boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{B} & — \\ — & \boldsymbol{a}_2^{\mathrm{T}}\boldsymbol{B} & — \\ & \vdots & \\ — & \boldsymbol{a}_M^{\mathrm{T}}\boldsymbol{B} & — \end{bmatrix}.$$

Finally, there is a fourth way to think of matrix-matrix multiplication, as a sum of the rank 1 matrices formed by taking the outer product of the columns of $\boldsymbol{A}$ with the rows of $\boldsymbol{B}$:

$$\begin{bmatrix} | & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_N \\ | & | & & | \end{bmatrix}\begin{bmatrix} — & \boldsymbol{b}_1^{\mathrm{T}} & — \\ — & \boldsymbol{b}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \boldsymbol{b}_N^{\mathrm{T}} & — \end{bmatrix} = \sum_{n=1}^{N} \boldsymbol{a}_n \boldsymbol{b}_n^{\mathrm{T}}.$$

To see this, consider just $\boldsymbol{a}_1$ and what it contributes to $\boldsymbol{AB}$. Each column of $\boldsymbol{AB}$ will potentially have some contribution from $\boldsymbol{a}_1$. Just how much? Well, it's the first row of $\boldsymbol{B}$ that will determine this for each column. The same argument follows for each of the N columns of $\boldsymbol{A}$ and rows of $\boldsymbol{B}$.