1. Orthonormality and Least Squares

Recall that, if $U \in \mathbb{R}^{m \times n}$ is a tall matrix (i.e. $m \geq n$) with orthonormal columns, then

$$U^T U = I_{n \times n} \quad (1)$$

However, it is not necessarily true that $UU^T = I_{m \times m}$. In this discussion, we will deal with “orthonormal” matrices, where the term “orthonormal” refers to a matrix that is square with orthonormal columns and rows. Furthermore, for an orthonormal matrix $U$,

$$U^T U = UU^T = I_{n \times n} \implies U^{-1} = U^T \quad (2)$$

This discussion will cover some useful properties that make orthonormal matrices favorable, and we will see a “nice” matrix factorization that leverages orthonormal matrices and helps us speed up least squares.

(a) Suppose you have a real, square, $n \times n$ orthonormal matrix $U$. You also have real vectors $\vec{x}_1, \vec{x}_2$, $y_1, y_2$ such that

$$y_1 = U\vec{x}_1 \quad (3)$$
$$y_2 = U\vec{x}_2 \quad (4)$$

This is analogous to a change of basis. Show that, in this new basis, the inner products are preserved.

Calculate $\langle y_1, y_2 \rangle = \vec{y}_2^T \vec{y}_1$ in terms of $\langle \vec{x}_1, \vec{x}_2 \rangle = \vec{x}_2^T \vec{x}_1 = \vec{x}_1^T \vec{x}_2$.

**Solution:** Since we have defined the $y$ vectors, we can substitute their expressions into $\vec{y}_2^T \vec{y}_1$:

$$\langle y_1, y_2 \rangle = \vec{y}_2^T \vec{y}_1$$
$$= (U\vec{x}_2)^T U\vec{x}_1$$
$$= \vec{x}_2^T U^T U \vec{x}_1$$
$$= \vec{x}_2^T \vec{x}_1$$
$$= (\vec{x}_1, \vec{x}_2) \quad (8)$$

Note that in going from eq. (7) to eq. (8), we used eq. (2).

(b) Using the change of basis defined in part 1.a, show that, in the new basis, the norms are preserved. Express $\|y_1\|^2$ and $\|y_2\|^2$ in terms of $\|\vec{x}_1\|^2$ and $\|\vec{x}_2\|^2$.

**Solution:** Recall that we can write the norm squared as

$$\|\vec{v}\|^2 = \vec{v}^T \vec{v} = \langle \vec{v}, \vec{v} \rangle$$

We can directly use the method from part 1.a to show that

$$\|y_i\|^2 = \langle y_i, y_i \rangle \quad (11)$$
\[
\begin{align*}
\vec{y}_i \cdot \vec{y}_i &= \sum_{i=1}^{m} \vec{x}_i \cdot U^\top U \vec{x}_i \\
\vec{x}_i \cdot \vec{x}_i &= \sum_{i=1}^{m} ||\vec{x}_i||^2
\end{align*}
\]
for \(i \in \{1, 2\} \).

(c) Suppose you observe data coming from the model \(y_i = \vec{a}^\top \vec{x}_i\), and you want to find the linear scale-parameters (each \(a_i\)). We are trying to learn the model \(\vec{a}\). You have \(m\) data points \((\vec{x}_i, y_i)\), with each \(\vec{x}_i \in \mathbb{R}^n\). Each \(\vec{x}_i\) is a different input vector that you take the inner product of with \(\vec{a}\), giving a scalar \(y_i\).

Set up a matrix-vector equation of the form \(X\vec{a} = \vec{y}\) for some \(X\) and \(\vec{y}\), and propose a way to estimate \(\vec{a}\).

**Solution:** Since \(y = \vec{a}^\top \vec{x}\) means that \(y = \vec{x}^\top \vec{a}\), we can stack the equations with the following definitions:

\[
X := \begin{bmatrix} \vec{x}_1^\top \\
\vec{x}_2^\top \\
\vdots \\
\vec{x}_m^\top \end{bmatrix}, \quad \vec{y} := \begin{bmatrix} y_1 \\
y_2 \\
\vdots \\
y_m \end{bmatrix}
\]

Then, we have \(\vec{y} = X\vec{a}\). Note that \(X \in \mathbb{R}^{m \times n}\), and \(\vec{y} \in \mathbb{R}^m\). We can estimate \(\vec{a}\) using least squares. Applying the standard least squares formula, we can find our estimate \(\hat{\vec{a}}\) by computing

\[
\hat{\vec{a}} = (X^\top X)^{-1} X^\top \vec{y}.
\]

(d) Let’s suppose that we can write our \(X\) matrix from part 1.c as

\[
X = MV^\top
\]

for some matrix \(M \in \mathbb{R}^{m \times n}\) and some orthonormal matrix \(V \in \mathbb{R}^{n \times n}\). Find an expression for \(\hat{\vec{a}}\) from the previous part, in terms of \(M\) and \(V^\top\).

Note: take this form as a given. We will go over how to find such a \(V\) and \(M\) later.

**Solution:** From the previous part, we have

\[
\hat{\vec{a}} = (X^\top X)^{-1} X^\top \vec{y}.
\]

Plugging in \(X = MV^\top\), we have

\[
\hat{\vec{a}} = \left( (MV^\top)^\top (MV^\top) \right)^{-1} (MV^\top)^\top \vec{y}
\]

\[
= (VM^\top MV^\top)^{-1} VM^\top \vec{y}
\]

\[
= (V^\top)^{-1} (M^\top M)^{-1} V^\top \vec{y}
\]

\[
= V (M^\top M)^{-1} \vec{y}
\]
(e) Now suppose that we have the matrix

\[
\begin{bmatrix}
\vec{x}_1^T \\
\vec{x}_2^T \\
\vdots \\
\vec{x}_m^T
\end{bmatrix} := X = U\Sigma V^T.
\] (24)

where \( U \in \mathbb{R}^{m \times m} \) is an orthonormal matrix, and \( V \in \mathbb{R}^{n \times n} \) is an orthonormal matrix. Here,

\[
\Sigma = \begin{bmatrix}
\sigma_1 & 0 & \ldots & 0 \\
0 & \sigma_2 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \sigma_n
\end{bmatrix}.
\]

Here we assume that we have more data points than the dimension of our space (that is, \( m > n \)). Also, the transformation \( V \) in part e) is the same \( V \) in this factorized representation.

**Set up a least squares formulation for estimating \( \hat{a} \) and find the solution to the least squares.**

Why might this factorization help us compute \( \hat{a} \) faster?

Note: again, take this factorization as a given. We will go over how to find \( U, \Sigma, \) and \( V \) later.

**Solution:** From the previous part, we know

\[
\hat{a} = V \left( M^T M \right)^{-1} M^T \vec{y}
\] (25)

Here, \( M = U\Sigma \) by pattern matching terms. Plugging this in,

\[
\hat{a} = V \left( (U\Sigma)^T (U\Sigma) \right)^{-1} (U\Sigma)^T \vec{y}
\] (26)

\[
= V \left( \Sigma^T U^T U \Sigma \right)^{-1} \Sigma^T U^T \vec{y}
\] (27)

\[
= V \left( \Sigma^T \Sigma \right)^{-1} \Sigma^T U^T \vec{y}
\] (28)

\[
\begin{bmatrix}
\sigma_1 & 0 & \ldots & 0 \\
0 & \sigma_2 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \sigma_n
\end{bmatrix}^{-1}
\]

\[
\begin{bmatrix}
\sigma_1 & 0 & \ldots & 0 \\
0 & \sigma_2 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \sigma_n
\end{bmatrix}
\]

\[
\Sigma^T U^T \vec{y}
\]

\[
\begin{bmatrix}
\sigma_1 & 0 & \ldots & 0 \\
0 & \sigma_2 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \sigma_n
\end{bmatrix}^{-1}
\]

\[
\Sigma^T U^T \vec{y}
\]

\[
\begin{bmatrix}
\sigma_1 & 0 & \ldots & 0 \\
0 & \sigma^2 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \sigma^2_n
\end{bmatrix}
\]

\[
\Sigma^T U^T \vec{y}
\] (29)

\[
\begin{bmatrix}
\sigma_1 & 0 & \ldots & 0 \\
0 & \sigma_2 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \sigma_n
\end{bmatrix}^{-1}
\]

\[
\Sigma^T U^T \vec{y}
\] (30)
\[
= V \begin{bmatrix}
\frac{1}{\sigma_1} & 0 & \cdots & 0 \\
0 & \frac{1}{\sigma_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{1}{\sigma_n}
\end{bmatrix}
\begin{bmatrix}
\sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma_n & 0 & \cdots & 0
\end{bmatrix}
U^\top \vec{y}
\] (31)

\[
= V \begin{bmatrix}
\frac{1}{\sigma_1} & 0 & \cdots & 0 \\
0 & \frac{1}{\sigma_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{1}{\sigma_n}
\end{bmatrix}
\begin{bmatrix}
0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{bmatrix}
U^\top \vec{y}
\] (32)

The nice part about this matrix factorization is that we can compute our least squares estimate really quickly (owing to the diagonal nature of \(\Sigma^\top \Sigma\)), since inverting an arbitrarily large matrix is computationally expensive. In particular, we only need to take the reciprocal of the diagonal elements of \(\Sigma^\top \Sigma\) when computing the matrix inverse. Multiplying this with \(\Sigma^\top\) adds the extra \(\vec{0}\) columns.

**Contributors:**
- Neelesh Ramachandran.
- Kuan-Yun Lee.
- Anant Sahai.
- Kumar Krishna Agrawal.