

## CS 3220: HOMEWORK 2

Instructor: Anil Damle

Due: September 24, 2019

### POLICIES

You may discuss the homework problems freely with other students, but please refrain from looking at their code or writeups (or sharing your own). Ultimately, you must implement your own code and write up your own solution to be turned in. Your solution, including plots and requested output from your code should be submitted via the CMS as a pdf file. Additionally, please submit any code written for the assignment via the CMS as well.

---

This homework is a bit shorter as it is due in roughly one weeks time. It also starts to get at some of the new material in the course and also is the first HW that contains some computational questions.

### QUESTION 1:

Here, we consider a question relating to norms:

- But, first, a bit of insight that will help with the next question. While we will talk about solving linear systems and more later in this course, a key property of orthogonal matrices that we have not yet discussed much is how they act as basis transforms. In particular, consider an arbitrary vector  $x \in \mathbb{R}^n$ , while a bit formal, we can write

$$x = Ix,$$

where  $I$  is the  $n \times n$  identity matrix. Therefore, we say  $x$  is the representation of  $x$  in the “canonical” basis (*i.e.* the columns of the identity matrix). As the columns of any orthogonal matrix  $Q \in \mathbb{R}^{n \times n}$  also form a basis for all of  $\mathbb{R}^n$ , for any  $x$  there should also be a vector  $y \in \mathbb{R}^n$  such that  $x = Qy$ . We can then say  $y$  is a representation of  $x$  in the basis  $\{q_1, \dots, q_n\}$  (the columns of  $Q$ ). To make this concrete:

- (a) Prove that for any vector  $x \in \mathbb{R}^n$  and orthogonal matrix  $Q \in \mathbb{R}^{n \times n}$  there is a vector  $y \in \mathbb{R}^n$  such that  $x = Qy$ .
  - (b) Given  $x$  and  $Q$  how can one compute  $y$ ?
- We have discussed how Orthogonal matrices do not change the 2-norm of vectors. A generalization of that idea is to think about matrix norms that are invariant under orthogonal transformations. A norm  $\|\cdot\|$  is orthogonally invariant if given any matrix  $A \in \mathbb{R}^{m \times n}$  and appropriately sized orthogonal matrices  $Q_1$  and  $Q_2$

$$\|Q_1 A Q_2\| = \|A\|.$$

- (c) Prove that the induced matrix 2-norm ( $\|\cdot\|_2$ ) is orthogonally invariant.
- (d) It turns out that  $\|\cdot\|_F$  is also orthogonally invariant. Use this fact to prove that for a square matrix  $A$  with SVD  $A = U\Sigma V^T$

$$\|A\|_F = \left( \sum_{i=1}^n \sigma_i^2 \right)^{1/2}$$

## QUESTION 2:

In this question we are going to explore how to solve a problem about aligning two sets of points via an orthogonal transformation. This question will also serve as a warm up for some of the “implementation” and computational questions on the homeworks for this course. Therefore, this question necessitates writing some relatively short code and generating plots. If you are unfamiliar with some of the tools/routines in your language of choice that are appropriate for any of the following questions I encourage you to use the resources on the course website and/or others you find to help you. Office hours and Piazza are also available if you have a specific question. If you have not had some experience with this already, I strongly encourage you to spend some time working out how to make nice plots and visualizations for your work—it will be worth it.

First, since we will deal with a certain type of rotation matrix in two dimensions we will need to characterize them.

(a) Show that for any  $\theta \in [0, 2\pi)$

$$Q_\theta = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

is an orthogonal matrix.

(b) By considering how this matrix transforms the vectors

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

can you reason out what type of rotation this is? (*i.e.*, applying  $Q_\theta$  transforms the vector in what way?)

Now, let  $X$  and  $Y$  be  $2 \times n$  matrices where we consider each column of  $X$  and  $Y$  to be a data point in two dimensional space (we will assume  $n \geq 2$  and generally for this problem expect  $n \gg 2$ ). Furthermore, let us assume that the columns of  $X$  and  $Y$  are the same set of points up to a rotation (more generally an orthogonal transform, which can also include reflections). In that case we know that there exists an orthogonal matrix  $Q$  such that

$$Y = QX.$$

In other words, every column of  $Y$  is constructed by applying an orthogonal transform to the respective column of  $X$ . An interesting question is if we can determine  $Q$  simply given  $X$  and  $Y$ . We can also then ask a related question, which is if  $X$  and  $Y$  satisfy

$$Y = QX + E$$

for some matrix  $E$  where  $\|E\|_F$  is small can we estimate  $Q$ . (This is a version of the alignment problem where the columns of  $X$  and  $Y$  are almost the same up to an orthogonal transform but there is some “error”).

It turns out that this problem is well known and has a nice solution. In particular, the following is known as an Orthogonal Procrustes problem:

$$\min_{\text{orthogonal } Q} \|QX - Y\|_F. \tag{1}$$

This optimization problem is exactly what we want to solve, find an orthogonal matrix  $Q$  such that the Frobenius norm between  $QX$  and  $Y$  is minimized. The solution is computed as follows, let  $B = YX^T$  and let  $B = U\Sigma V^T$  be the SVD of  $B$ . The solution of (1) is then  $Q = UV^T$ . For the following, you may use the built in SVD computation routine in your language of choice along with built in linear algebra routines.

- (c) For  $n = 100$  generate a matrix  $X$  with random elements; you need not concern yourself with the way this is done for the moment or the specific distribution of the entries, but your language of choice should have a routine to accomplish this easily (*e.g.*,  $\mathbf{X} = \mathbf{randn}(2,100)$  in Matlab; if you have questions on how to do this please let us know). Now, pick some  $\theta \in (0, 2\pi)$  and construct  $Q_\theta$  and compute  $Y = Q_\theta X$ . Produce scatter plots of the columns of  $X$  and  $Y$ , do they align with your interpretation of the action of  $Q_\theta$  from before?
- (d) Implement solving the alignment problem from above, using your constructed  $X$  and  $Y$  solve for  $Q$  (defined as the solution to (1)). Report  $Q$  and  $Q_\theta$ , are they the same or different, what is  $\|Q - Q_\theta\|_F$ ? Also report  $\|QX - Y\|_F$ , does it align with your expectations? (You may either implement computing  $\|\cdot\|_F$  yourself or use a built in routine.)
- (e) Now, let  $E \in \mathbb{R}^{2 \times 100}$  be a matrix with random entries as above (*i.e.*,  $E$  is generated in the same manner as  $X$ ). Now, for various values of  $0 \leq \epsilon \leq 10^{-1}$  let  $Y = Q_\theta X + \epsilon E$  solve the alignment problem as above and produce plots showing how  $\|QX - Y\|_F$ ,  $\epsilon\|E\|_F$ , and  $\|Q - Q_\theta\|_F$  vary as a function of  $\epsilon$ . Does what you observe seem reasonable to you? And does this process seem reasonably robust to noise?
- (f) Finally, lets look at how this process scales with the number of data points  $n$ . Specifically we will consider how the time to compute  $Q$  scales with  $n$ . So, for various increasing values of  $n$  (say starting at  $n = 100$  and going up to at least  $n = 10,000$  in some reasonable increments) generate  $X$  and  $Y = Q_\theta X$  as before for some  $\theta$  of your choosing. Then, measure the time it takes to compute  $Q$  given  $X$  and  $Y$  (this means forming  $B$ , computing its SVD, and forming  $Q = UV^T$ ). Your language of choice should have built in routines to measure the elapsed time between two lines of code and you may use those. What relationship do you observe between  $n$  and the time taken? To illustrate the relationship include in your homework a plot  $t_{\text{elapsed}}(n)$  vs  $n$ . Since we will want to characterize the relationship as  $t_{\text{elapsed}}(n) \sim n^q$  for some  $q$  it may be more insightful in general to plot  $\log t_{\text{elapsed}}(n)$  vs.  $\log n$ ; why is this the case?