EECS 16B Designing Information Devices and Systems II
Spring 2021 UC Berkeley Homework 12

This homework is due on Friday, April 16, 2021, at 11:00PM. Self-grades and HW Resubmission are due on Tuesday, April 20, 2021, at 11:00PM.

1. **Reading Lecture Notes**

   Staying up to date with lectures is an important part of the learning process in this course. Here are links to the notes that you need to read for this week: Note 13, Note 14

   Consider a matrix $A \in \mathbb{R}^{m\times n}$ with $rank(A) = r$ and $r \leq m < n$ (wide matrix with full row rank). We will consider the representations of the full, compact, and outer product forms of the SVD of A:

$$A = \begin{bmatrix} U_r & U_{m-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} \begin{bmatrix} V_r^T \\ V_{n-r}^T \end{bmatrix}$$

$$A = U_r \Sigma_r V_r^T$$

$$A = \sum_{i=1}^{r} \vec{u_i}\sigma_i\vec{v_i}^T$$

   Where $\vec{u_i}$ and $\vec{v_i}$ are the i^{th} column vectors of U and V respectively. **Explain what spaces the columns of $U_r, U_{m-r}, V_r,$ and V_{n-r} span. What is the rank of the matrix $\vec{u_i}\sigma_i\vec{v_i}^T$?**

2. SVD

(a) Given the matrix

$$A = \frac{1}{\sqrt{50}} \begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} + \frac{3}{\sqrt{50}} \begin{bmatrix} -4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix},$$

write out a singular value decomposition of matrix A in the form $U\Sigma V^{\top}$. Note the ordering of the singular values in Σ should be from the largest to smallest.

HINT: You don't have to compute any eigenvalues for this. Some useful observations are that

$$\begin{bmatrix} 3, 4 \end{bmatrix} \begin{bmatrix} -4 \\ 3 \end{bmatrix} = 0, \quad \begin{bmatrix} 1, -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0, \quad \left\| \begin{bmatrix} 3 \\ 4 \end{bmatrix} \right\| = \left\| \begin{bmatrix} -4 \\ 3 \end{bmatrix} \right\| = 5, \quad \left\| \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\| = \left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\| = \sqrt{2}.$$

(b) Define the matrix

$$A = \begin{bmatrix} -1 & 4 \\ 1 & 4 \end{bmatrix}.$$

Find the SVD of A. Then, find the eigenvectors and eigenvalues of A. Is there a relationship between the eigenvalues or eigenvectors of A with the SVD of A?

3. More SVD Proofs (PRACTICE)

Let $A \in \mathbb{R}^{m\times n}$ be an arbitrary matrix. Let the SVD of $A = U\Sigma V^T$. Let $r = \text{rank}(A)$. Let the columns of U be $\vec{u}_1, \ldots, \vec{u}_m$, the columns of V be $\vec{v}_1, \ldots, \vec{v}_n$, and the singular values of A be $\sigma_1, \ldots, \sigma_r, 0, \ldots 0$ since the remaining $\min\{m, n\} - r$ singular values are 0.

(a) **Show that $A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T$.**

HINT: It might be helpful to use the outer product version of matrix multiplication, i.e. $XY = \sum_{i=1}^{k} \vec{x}_i \vec{y}_i^T$ *where* $\vec{x}_i, \vec{y}_i$ *are the columns of* X *and* Y *respectively.*

(b) **Prove that if $\vec{x}$ is a linear combination of $\vec{v}_{r+1}, \ldots, \vec{v}_n$, then $A\vec{x} = 0$ so $\vec{x} \in Null(A)$.**

(c) **Now we prove the opposite of the previous part, so if $\vec{x} \in Null(A)$, then $\vec{x}$ is a linear combination of the vectors $\vec{v}_{r+1}, \ldots, \vec{v}_n$.**

The last two parts together prove that $Null(A) = \text{span}\{\vec{v}_{r+1}, \ldots, \vec{v}_n\}$. Moreover, since the $\vec{v}_i$ are orthonormal (which implies linear independence), the vectors $\vec{v}_{r+1}, \ldots, \vec{v}_n$ form an orthonormal basis for $Null(A)$.

(d) **Show that $U_r = \begin{bmatrix} \vec{u}_1 & \ldots & \vec{u}_r \end{bmatrix}$ is a basis for the column space of A.**

4. Generalizing Least Squares using the Pseudoinverse

This question will show you how to generalize the orthogonal projection from normal least squares to work for any set of vectors. On the way, we will show some powerful properties of the Moore-Penrose pseudoinverse.

(a) Suppose we have a matrix $A \in \mathbb{R}^{m \times n}$. Now consider a vector $\vec{y} \in \mathbb{R}^m$. We are interested in projecting $\vec{y}$ onto the column space of A, i.e. finding an $\vec{x}$ such that $||A\vec{x} - \vec{y}||^2$ is minimized. For that $\vec{x}$, $A\vec{x}$ is known as the orthogonal projection of $\vec{y}$ onto the column space of A (this was actually how we derived $\vec{\hat{x}}$ in 16A). **If $m \geq n$ and A has linearly independent columns, find $\vec{x}$ and then the projection $A\vec{x}$.**

HINT: Remember what least squares does.

(b) The previous part only worked when restricting A to have linearly independent columns so that $A^T A$ is invertible. Thus it does not work for a wide A with more columns than rows. (Try to think why a wide matrix can't have linearly independent columns on your own.)

We want to remove these restrictions so now we consider an arbitrary A. In this case, A can have a nontrivial nullspace, and so there can be infinite such solutions for $\vec{x}$ that minimize the objective. Thus, we want to impose an extra condition that out of all the minimizing vectors $\vec{x}$, we want to find the one with the minimum squared norm $||\vec{x}||^2$. This is called the **minimum-norm least squares problem**:

$$\underset{\vec{x}}{\operatorname{argmin}}\, ||\vec{x}||^2 \quad \text{s.t.} \quad ||A\vec{x} - \vec{y}||^2 = \min_{\vec{s}} ||A\vec{s} - \vec{y}||^2 \tag{1}$$

(Note that if A doesn't have a nullspace, then the least squares problem has a unique solution, so it is automatically the minimum norm solution.)

To solve this problem, we will use the SVD and the Moore-Penrose pseudoinverse. Let the SVD of $A = U\Sigma V^T$. We define the Moore-Penrose pseudoinverse $A^\dagger = V\Sigma^\dagger U^T$ from the last homework where

$$\Sigma^\dagger = \left[\begin{array}{ccc|c} \frac{1}{\sigma_1} & & & \\ & \ddots & & 0_{r \times (m-r)} \\ & & \frac{1}{\sigma_r} & \\ \hline & 0_{(n-r) \times r} & & 0_{(n-r) \times (m-r)} \end{array}\right].$$

First show that $||A\vec{x} - \vec{y}||^2 = ||\Sigma V^T \vec{x} - U^T \vec{y}||^2$.

HINT: Remember that orthonormal transformations don't affect norms of vectors (discussion 11A).

(c) We now let $\vec{z} = V^T\vec{x}$ and $\vec{w} = U^T\vec{y}$. Since V^T is orthonormal, we know that $||\vec{x}||^2 = ||V^T\vec{x}||^2 = ||\vec{z}||^2$. Additionally since V^T is invertible, we can convert between $\vec{x}$ and $\vec{z}$, which means we can equivalently optimize over $\vec{z}$ instead of $\vec{x}$ in (1). Combining this with the result from part (b), our minimum-norm least squares problem simplifies to

$$\underset{\vec{z}}{\operatorname{argmin}}\, ||\vec{z}||^2 \quad \text{s.t.} \quad ||\Sigma\vec{z} - \vec{w}||^2 = \min_{\vec{s}} ||\Sigma\vec{s} - \vec{w}||^2. \tag{2}$$

Using the fact that Σ has a simple form with singular values on its diagonal, **find the minimum-norm least squares solution for (2). Write your answer in terms of $\vec{w}$ and $\Sigma^\dagger$ (not Σ).**

HINT: Sbow that

$$\Sigma\vec{z} - \vec{w} = \begin{bmatrix} \sigma_1 z_1 - w_1 \\ \vdots \\ \sigma_r z_r - w_r \\ -w_{r+1} \\ \vdots \\ -w_m \end{bmatrix}$$

where $r = rank(A)$, and see which entries you can minimize by choosing $\vec{z}$.

(d) Since we found the $\vec{z}$ that solved (2), we can convert this solution back to $\vec{x}$ and it should solve (1). **Do this and show that the minimum-norm least squares solution for (1) is $\vec{x} = A^\dagger\vec{y}$.**

(e) Thus, we just proved that applying the pseudoinverse exactly gives the minimum-norm least squares solution which works for all A matrices!

Now, we can finally go back to part (a) and address the restrictions on A. Using your new-found tool, **determine the projection of $\vec{y}$ onto the column space of A, where now A can be any arbitrary matrix.**

(f) Let $r = \text{rank}(A)$. We will define $U_r = \begin{bmatrix} \vec{u}_1 & \dots & \vec{u}_r \end{bmatrix}$ which is the first r columns of U. **Show that the expression for the projection from part (e) can be simplified to $U_r U_r^T \vec{y}$ by plugging in the SVD and pseudoinverse.**

HINT: Write out both Σ and $\Sigma^\dagger$ and multiply them out.

This simplification of the projected vector is very interesting, and there actually is an alternative reasoning to arrive at it. We know that U_r is a basis for the column space of A, which was proved in lecture. Then projecting onto the column space of A should be the same as projecting onto U_r. But now, note that U_r has linearly independent columns, so we can just use part (a) replacing A with U_r. Thus, the projected vector is $U_r(U_r^T U_r)^{-1} U_r^T \vec{y} = U_r I^{-1} U_r^T \vec{y} = U_r U_r^T \vec{y}$, where $U_r^T U_r = I$ since U_r has orthonormal columns.

5. **Brain-machine interface and neuron sorting**

The iPython notebook pca_brain_machine_interface.ipynb will guide you through the process of analyzing brain machine interface data leveraging the SVD. This will help you to prepare for the project.

For SIXT33N (your robot car), you will need to use the SVD and PCA to classify your sound inputs so that your car does what you tell it to do.

Brain-Machine interfaces (BMIs) are a way for a brain to directly communicate to an external device. They're often used for research, mapping, assisting or repairing human cognitive or sensory-motor function.

Data was collected that shows waveform traces of (simulated) brain waves of a subject whose arm is pointing in certain directions over time. These waveform traces are gathered from electrodes inserted into the brain, but the electrodes are generally recording more than 1 neuron at the same time — the brain is crowded after all. To make predictions based on neuron firing rates, we need to be able to distinguish waveforms that come from different neurons.

Each neuron has its own waveform "signature" shape unique to that neuron. This is due to the physical characteristics of neurons, such as their physical shape and structure. It is impossible to know beforehand how many neurons an electrode measures or what each neuron's waveform looks like, so we must first separate the waveforms from the different neurons near the electrode.

The goal of this problem is to see how we can use the SVD (in its PCA manifestation) and clustering to decide which neuron fired. We will first look at a case where there are only two neurons, then a case where there are three neurons. The neurons have also been presorted using a professional software package, so we can check our model against presorted data.

Please complete the notebook by following the instructions given.

Task 1: Two Neuron Waveform Sorting

(a) The data you have are traces of length 32 timesteps. The data is arranged along the rwos fo the matrix, each row is one trace. Import the data sets and see the average waveform for each presorted neuron by running the corresponding cells in the .ipynb. **What is the shape of the training data matrix three_neurons_training?**

(b) **What do the columns and rows of the training data matrix: three_neurons_training represent?**

We can approximate each waveform in a lower dimensional space using PCA. In your implementation of PCA you will decide how to choose these components.

(c) You will be using the SVD in your PCA function. **What matrices does the SVD return? What are the dimensions of these matrices for the SVD of three_neurons_training?**

(d) To represent each waveform in a lower dimensional space we want a basis for the time signals of the waveforms of the neurons. To construct this basis we start by taking the SVD (of three_neurons_training). **Which of the U and V matrices can we use to construct our desired basis?**

(e) **Read through PCA_train and impliment PCA_project**. We will use these functions throughout the rest of the notebook, so make sure you understand them.

(f) **Call PCA_train on two_neurons_training and plot the 2 principal components.** Note that since the dataset is randomized, you might get different plots every time you run the second code cell of this notebook (the _make_training_set function). Note that these components have no real 'physical' relationship to the actual shape of the neuron plots. They are a part of a basis, not the representation of exemplar traces.

(g) Before we project onto our principal components, let us project our data onto 2 random 32 length vectors. **Run the corresponding part in the .ipynb file and comment on the separation of the 2 neuron's distinctive trace shapes in this projected basis.** Do you think that it would be easy to tell them apart just by looking at their projection into these two random dimensions?

(h) **Project two_neurons_test onto the two principal components found using the SVD and produce a 2D scatter plot in the new basis.** We will also try projecting the presorted data containing 2 neurons so we can see how the model behaves on the 2 neurons. **Comment on the difference between the projections here and part (g) where you projected onto random directions.**

(i) Now we will repeat this process for three principal components. **Do you see a significant improvement with 3 principal components?**

Task 2: Three Neuron Sorting

(j) In the previous part we sorted 2 neurons using two or three principal components. In this part we are now dealing with 3 neurons. **Replicate what we did in the previous parts by finding and projecting our data on the first two principal components of this three neuron dataset in the .ipynb.**

(k) Now **call PCA_train on three_neurons_training and plot the 3 principal components. Do the same classification process as the 2 neuron data, but now with the 3 neuron data. Compare your model's behavior with that of the presorted data.** The plot_3D function will be useful to view the results.

(l) **How many principal components do you actually need to cluster the 3 neurons? (Hint: Think about whether there was an increase in separability between the last two parts)**

Task 3: Determining Neurons (OPTIONAL) Now that we know how to project our data onto a basis where it is clearly separable we can classify our points and determine which neuron fired. (This is what we need to know if we want to use the firings of different neurons to build a BMI.)

To classify our points we use the following algorithm:

i. Find centroids: points that represent the average waveform of a particular neuron firing, in the basis of principal components.

ii. Classify each incoming point by assigning it the value of the centroid closest to it (i.e. declare that the neuron waveform that a point represents is the same as the neuron waveform of the centroid the point is closest to).

(m) **OPTIONAL** When we are doing such problems in practice, we will not have a presorted data set. In such cases we can use algorithms like k-means clustering to determine our centroids (You should have seen this algorithm in CS61A during the Maps project). As a refresher, k-means will do the same thing as in (l)i. but with a couple modifications. A step is added at the beginning to randomly assign each data point to some centroid and a step is added to the end such that if the classifications remained the same as last time, exit. Else, go back to step 2.

Scipy happens to have a built in function to compute k-means given clustered data, which we have conveniently formatted for you.

Run the code that we have provided to do k-means for you and find the centroids. **Then use 'which_neuron' on the two_classified data set and count the number of times each neuron fired. Are these values the same as the previous part ?**

In later courses, you will build on the basic techniques taught here. But in reality, the ideas that you have learned in 16AB so far already form a conceptual core for machine learning. You can go very far with just the ideas you have already learned.

6. Movie Ratings and PCA

Recall from the lecture on PCA that we can think of movie ratings as a structured set of data. For every person i and movie j, we have that person's rating $R_{i,j}$ (thought of as a real number).

Suppose that there are m movies and n people. Let's think about arranging this data into a big $n \times m$ matrix R with people corresponding to rows and movies corresponding to columns. So the entry in the i-th row and j-th column should be $R_{i,j}$ above. Note that this is organized differently from how it was in lecture. Each row corresponds to a unique person and each column to a unique movie.

$$R = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ R_{21} & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ R_{n1} & R_{n2} & \cdots & R_{nm} \end{bmatrix}$$

(a) Suppose we believe that there is actually an underlying pattern to this rating data and that a separate study has revealed that every movie is characterized by a set of characteristics: say action and comedy. This means that every movie j has a pair of numbers $a[j], c[j]$ that define it. At the same time, every person i has a pair of sensitivities $s_a[i], s_c[i]$ that essentially define that person's preferences. A person i will rate the movie j as $R_{i,j} = s_a[i]a[j] + s_c[i]c[j]$.

If we arrange the sensitivities into a pair of n-dimensional vectors $\vec{s_a}, \vec{s_c}$, and the movie characteristics into a pair of m-dimensional vectors $\vec{a}, \vec{c}$, **use outer products to express the rating matrix R in terms of these vectors $\vec{s_a}, \vec{s_c}, \vec{a}, \vec{c}$.**

(b) Now suppose that we want to discover the underlying nature of movies from the data R itself.

Suppose for this part, that you have four observed rating data vectors (corresponding to four different movies being rated by six individuals).

All of the movie data vectors just happened to be multiples of the following 6-dimensional vector $\vec{w} = \begin{bmatrix} 2 \\ -2 \\ 3 \\ -4 \\ 4 \\ 0 \end{bmatrix}$. (For your convenience, note that $\|\vec{w}\| = 7$.)

You arrange the movie data vectors as the columns of a matrix R given by:

$$R = \begin{bmatrix} | & | & | & | \\ -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} \\ | & | & | & | \end{bmatrix} \tag{3}$$

You want to perform PCA (for movies) using the SVD of the matrix R to better understand the pattern in your data.

The first "principal component vector" is a *unit vector* that tells which direction we would want to project the columns of R onto to get the best rank-1 approximation for R.

Find this first principal component vector of the columns of R to explain the nature of your movie data points.

(HINT: You don't need to actually compute any SVDs in this simple case. Also, be sure to think about what size vector you want as the answer. Don't forget that you want a unit vector!)

(c) Suppose that now, we have two more data points (corresponding to two more movies being rated by the same set of six people, i.e. we added two columns to our matrix) that are multiples of a different vector $\vec{p}$ where:

$\vec{p} = \begin{bmatrix} 6 \\ 3 \\ -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. (For your convenience, note that $\|\vec{p}\| = 7$ and that $\vec{p}^\top \vec{w} = 0$.)

We augment our ratings data matrix with these two new data points to get:

$$R = \begin{bmatrix} | & | & | & | & | & | \\ -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} & -3\vec{p} & 3\vec{p} \\ | & | & | & | & | & | \end{bmatrix} \tag{4}$$

Find the first two principal components corresponding to the nonzero singular values of R. This is what we would use to best project the movie data points onto a two-dimensional subspace.

What is the first principal component vector? What is the second principal component vector? Justify your answer.

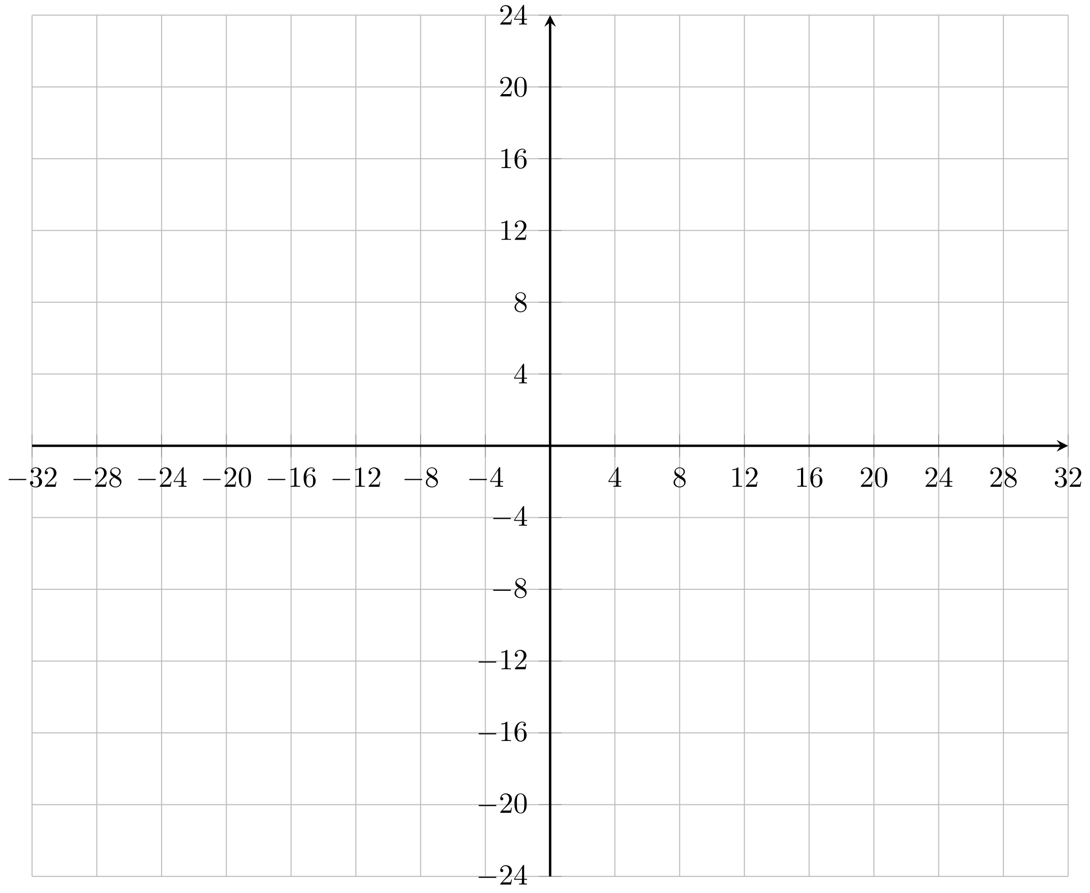
(Hint: Think about the inner product of $\vec{w}$ and $\vec{p}$ and what that implies for being able to appropriately decompose R. Again, very little computation is required here.)

(d) In the previous part, you had

$$R = \begin{bmatrix} | & | & | & | & | & | \\ -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} & -3\vec{p} & 3\vec{p} \\ | & | & | & | & | & | \end{bmatrix}$$

with $\|\vec{w}\| = 7$ and $\|\vec{p}\| = 7$, satisfying $\vec{p}^\top \vec{w} = 0$.

If we use $\vec{r_i}$ to denote the i-th column of R, **plot the movie data points $\vec{r_i}$ projected onto the first and second principal component vectors along the columns of R.** The coordinate along the first principal component should be represented by horizontal axis and the coordinate along the second principal component should be the vertical axis. **Label each point, and the axes. Remember that principal component vectors are normalized**.


24
20
16
12
8
4
−32 −28 −24 −20 −16 −12 −8 −4 4 8 12 16 20 24 28 32
−4
−8
−12
−16
−20
−24

7. Image Compression With SVD

Open the **image_compression.ipynb** Jupyter Notebook. Make sure to read through the theory section to understand how we will use what we know about PCA and SVD in order to compress images! Once you have read through the math and are comfortable with it, go ahead and start running the coding cells and answer the questions below.

(a) Run all cells up to and throughout Part 1. Thoroughly read through the 'rank_k_approx' function. **Since we are now only using the top k singular values, in terms of $k, m,$ and n, how many real numbers do we require to describe the rank k approximation of A?**

(b) Run the cells in Part 2 of the notebook. Notice there are two images: One compressed, one uncompressed. **Can you easily visually detect any difference between the two images even though we are using only about 40% of the data?**

(c) Run the cells that plot the singular values. **Given the plots above, why did the image look so good even with only about 40% of the data?**

(d) Play around with the slider for the kitten picture. **What is the lowest acceptable value of k that you can go without losing too much image quality? What were the memory savings?**
Note: Use your best judgment since this depends on eyesight, screen resolution, etc.

(e) Plot the singular values for the image of the US flag. **What do you notice about the singular values here in comparison to the kitten's singular values? What does this mean for our low rank compression?**

(f) Play around with the interactive slider for the US flag. **What is the lowest acceptable value of k here? What is the memory saving?**

8. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

Contributors:

- Elena Jia.
- Neelesh Ramachandran.
- Aditya Arun.
- Daniel Abraham.
- Ashwin Vangipuram.
- Nikhil Shinde.
- Gaoyue Zhou.
- Nathan Lambert.
- Anant Sahai.