EECS 16B Designing Information Systems and Devices II UC Berkeley Spring 2022

Homework 11

This homework is due on Sunday, April 17, 2022 at 11:59PM. Self-grades and HW Resubmissions are due the following Sunday, April 24, 2022 at 11:59PM.

***NOTE*: The last two problems on this homework discuss concepts that will be covered in the lecture on Tuesday, April 12. As such, we are extending the deadline for this homework to give you extra time to work on those problems. All future homeworks will run on a "Sunday due date, following Sunday resubmission" schedule as well.**

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 15 Note 16 Note 17

   (a) If the outer product form of the SVD of A is $\sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top$, **what is the best rank k approximiation for A, which we'll call X, that minimizes the Frobenius norm $\|A - X\|_F$?** Assume that $k \leq r$.

   Solution: From the theorem proved in the notes, the best rank k approximiation is

   $$X = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^\top \tag{1}$$

   so just the first k terms of the outer product form of A, which correspond to the k largest singular values of A.

   (b) We have a data matrix X and its SVD $U\Sigma V^T$. When performing PCA, **what are the principal components if the columns of X are our data points? What are the principal components if the rows of X are our data points?**

   Solution: The principal components that best explain the columns of X will be the columns of U, i.e. the left singular vectors. The principal components that best explain the rows of X will be the columns of V, i.e. the right singular vectors.

2. Rank 1 Decomposition and Image Compression With SVD

In this problem, we will introduce an important application of the singular value decomposition (SVD) called low-rank approximation in the context of image compression. Given a matrix $A \in \mathbb{R}^{m \times n}$ of rank $r \leq \min\{m, n\}$, we saw in Note 16 that we can write A using the outer-product form of the SVD:

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top. \tag{2}$$

If our matrix is high-rank, i.e., $r \approx \min\{m, n\}$, then almost all the σ_i will be nonzero and non-negligible. However, if the data has some linear, low-rank essential structure, as is usually the case with real data such as images, most of our singular values will be very small (but usually nonzero due to noise or disturbances). If, say, the data has intrinsic linear rank ℓ, then the first ℓ singular values are large, and the remaining $r - \ell$ are small:

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top \tag{3}$$

$$= \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top + \sum_{i=\ell+1}^{r} \underbrace{\sigma_i}_{\approx 0} \vec{u}_i \vec{v}_i^\top \tag{4}$$

$$\approx \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top. \tag{5}$$

This motivates approximating the data as

$$A_\ell := \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top \tag{6}$$

and using this compressed data for further analysis. As images as readily represented as matrices, in this problem, we will see how to make use of this low rank approximation technique to compress images without severely deteriorating the image quality.

Throughout this problem we will work in the `image_compression.ipynb` Jupyter Notebook. Part 1 of the notebook relates to finding rank 1 decompositions of images, while Parts 2 and 3 focus on image compression. Make sure to read through all the sections of the notebook to understand how we will use what we know about the SVD in order to compress images! You should write all your answers in the written submission and you don't need to submit the notebook.

We will start by decomposing a few images into linear combinations of rank 1 matrices. Remember that outer product of two vectors $\vec{s}\vec{g}^\top$ gives a rank 1 matrix. It has rank 1 because the column span is one-dimensional — multiples of $\vec{s}$ only — and the row span is also one dimensional — multiples of $\vec{g}^\top$ only. More generally, this exercise should provide intuition into how the outer product form of the SVD decomposes the underlying matrix into a sum of constituent rank 1 matrices with different weightings given by the singular values.

(a) Read and run through the cells in Part 1. Here we first consider a standard 4×4 checkerboard shown in Figure 1. Assume that black colors represent -1 and that white colors represent 1.

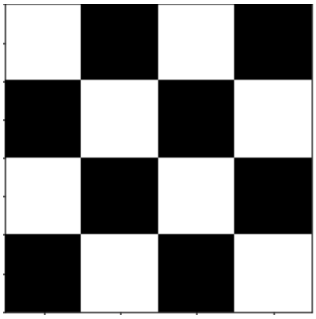


Figure 1: 4×4 checkerboard.

In particular, the checkerboard is given by the following 4×4 matrix L:

$$L = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \tag{7}$$

Express L as a linear combination of outer products.

(HINT: In order to determine how many rank 1 matrices you need to combine to represent the matrix, find the rank of the matrix you are trying to represent.)

Solution: The matrix L only has rank 1, since column vectors 1, 3 are the same, and column vectors 2, 4 are multiples of the first or third column. This means that we can express L by multiplying the first column vector $\begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}^\top$ by the multiples required to generate the other columns, which are $1, -1, 1, -1$. As a result, we get the following outer product form:

$$L = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}^\top \tag{8}$$

(b) We next consider the same checkerboard given by the following 4×4 matrix C where black colors now represent 0 and that white colors represent 1.

$$C = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{9}$$

Express C as a linear combination of outer products. *(HINT: Note that C is a rank 2 matrix and therefore you need 2 outer products to represent it. To find these individual outer products consider the columns of C.)*

Solution: We first note that C is a rank 2 matrix and thus will be a summation of 2 outer products. There are infinitely many outer products that we can choose, but we will just consider one linear combination. To find this, we note that C has two distinct columns. The first and third columns are given by $\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, while the second and fourth columns are $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$. Therefore

$$C = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}. \tag{10}$$

These rank 1 matrices are shown as images in the Jupyter notebook solutions.

(c) Now consider the Swiss flag shown in Figure 2. Assume that red colors represent 0 and that white colors represent 1.

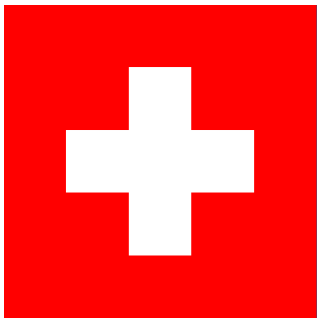


Figure 2: Swiss flag.

Assume that the Swiss flag is given by the following 5×5 matrix S:

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

Express S as a linear combination of outer products.

Solution: We again note that S is a rank 2 matrix and will be a summation of 2 outer products. There are again infinite solutions, but we will provide a couple. To see the first of these solutions, we note that we can reconstruct all of the columns of S using linear combinations of $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$. For example, to obtain the third column of S we can simply scale the aforementioned vectors by 1 and then sum their contributions. Therefore, we can write S as an outer product expansion using these vectors:

$$S = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{12}$$

(13)

Similarly, the columns of S can be written as linear combinations of $\begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$. This gives rise

to the outer product summation

$$S = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{14}$$

These rank 1 matrices are shown as images in the Jupyter notebook solutions.

(d) Run all the cells in Part 2. 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?**

Solution: We require storage of $k(m+n+1)$ real numbers.

We can actually do a bit better than this, but we'll save that for the solutions to the next HW where we ask a similar question.

(e) Run the cells in Part 3 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?**

Solution: We expect that you notice very little to no change between the images.

(f) 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.

Solution: Playing around with the slider, around $k = 70$ was the lowest we could go without noticing too much image loss. Only about 15% of the data is needed. You just have to be in the ballpark for credit.

(g) 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?**

Solution: The singular values have a much sharper cutoff. This means that we can compress this image much better than the kitten.

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

Solution: Here we can get away with $k = 35$ resulting in only using 9% of the data! This is much better than the kitten! This is true because this image is much more structured than the kitten.

3. Frobenius Norm

In this problem we will investigate the basic properties of the Frobenius norm.

Similar to how the norm of vector $\vec{x} \in \mathbb{R}^n$ is defined as $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$, the Frobenius norm of a matrix $A \in \mathbb{R}^{m \times n}$ is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |A_{ij}|^2}. \tag{15}$$

A_{ij} is the entry in the i^{th} row and the j^{th} column. This is basically the norm that comes from treating a matrix like a big vector filled with numbers.

(a) With the above definitions, **show that for a 2×2 matrix A:**

$$\|A\|_F = \sqrt{\text{tr}(A^\top A)}. \tag{16}$$

Note: The trace of a matrix is the sum of its diagonal entries. For example, let $A \in \mathbb{R}^{m \times n}$, then,

$$\text{tr}(A) = \sum_{i=1}^{\min(n,m)} A_{ii} \tag{17}$$

Think about how/whether this expression eq. (16) generalizes to general $m \times n$ matrices.

Solution: This proof is for the general case of $m \times n$ matrices. You should give yourself full credit if you did this calculation only on the 2×2 case.

$$\begin{aligned}
\text{tr}\left(A^\top A\right) &= \sum_{i=1}^{n} (A^\top A)_{ii} && (18)\\
&= \sum_{i=1}^{n} \left(\sum_{j=1}^{m} (A^\top)_{ij} A_{ji} \right) && (19)\\
&= \sum_{i=1}^{n} \left(\sum_{j=1}^{m} A_{ji} A_{ji} \right) && (20)\\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ji}^2) && (21)\\
&= \|A\|_F^2 && (22)
\end{aligned}$$

In the above solution, step eq. (18) writes out the trace definition, step eq. (19) expands the matrix multiplication on the diagonal indices (i.e. index (i,i) is the real inner product of row i and column i), step eq. (20) applies the definition of matrix transpose, and the last two steps collects the result into the definition of Frobenius norm.

(b) **Show that if U and V are square orthonormal matrices, then**

$$\|UA\|_F = \|AV\|_F = \|A\|_F . \tag{23}$$

(HINT: Use the trace interpretation from part (a).)

Solution: The direct path is just to compute using the trace formula:

$$\|UA\|_F = \sqrt{\text{tr}\left((UA)^\top (UA)\right)} = \sqrt{\text{tr}\left(A^\top U^\top UA\right)} = \sqrt{\text{tr}\left(A^\top A\right)} = \|A\|_F \tag{24}$$

Another path is to note that the Frobenius norm squared of a matrix is the sum of squared Euclidean norms of the columns of the matrix. Matrix multiplication UA proceeds to act on each

column of A independently. None of those norms change since U is orthonormal, and so the Frobenius norm also doesn't change.
To show the second equality, we must first note that $\|A^\top\|_F = \|A\|_F$, because we are just summing over the same numbers, just in a different order. Hence:

$$\|AV\|_F = \left\|(AV)^\top\right\|_F = \left\|V^\top A^\top\right\|_F \tag{25}$$

But the transpose of a square orthonormal matrix is also orthonormal, hence this case reduces to the previous case, implying

$$\left\|V^\top A^\top\right\|_F = \left\|A^\top\right\|_F = \|A\|_F \tag{26}$$

(c) **Use the SVD decomposition to show that $\|A\|_F = \sqrt{\sum_{i=1}^n \sigma_i^2}$, where $\sigma_1, \ldots, \sigma_n$ are the singular values of A.**
(HINT: The previous part might be quite useful.)
Solution:

$$\|A\|_F = \left\|U\Sigma V^\top\right\|_F = \left\|\Sigma V^\top\right\|_F = \|\Sigma\|_F \tag{27}$$

$$= \sqrt{\sum_{i=1}^n \sigma_i^2} \tag{28}$$

4. 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.

$$R = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ R_{21} & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ R_{n1} & R_{n2} & \cdots & R_{nm} \end{bmatrix} \tag{29}$$

(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 (for action) and c_j (for comedy) that define it. At the same time, every person i has a pair of sensitivities f_i and g_i that defines that person's preferences for action vs. comedy movies respectively. A person i will rate the movie j as $R_{i,j} = f_i a_j + g_i c_j$.

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

Solution:

$$R = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ R_{21} & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ R_{n1} & R_{n2} & \cdots & R_{nm} \end{bmatrix} \tag{30}$$

$$= \begin{bmatrix} f_1 a_1 + g_1 c_1 & f_1 a_2 + g_1 c_2 & \cdots & f_1 a_m + g_1 c_m \\ f_2 a_1 + g_2 c_1 & f_2 a_2 + g_2 c_2 & \cdots & f_2 a_m + g_2 c_m \\ \vdots & \vdots & \cdots & \vdots \\ f_n a_1 + g_n c_1 & f_n a_2 + g_n c_2 & \cdots & f_n a_m + g_n c_m \end{bmatrix} \tag{31}$$

$$= \begin{bmatrix} f_1 a_1 & f_1 a_2 & \cdots & f_1 a_m \\ f_2 a_1 & f_2 a_2 & \cdots & f_2 a_m \\ \vdots & \vdots & \cdots & \vdots \\ f_n a_1 & f_n a_2 & \cdots & f_n a_m \end{bmatrix} + \begin{bmatrix} g_1 c_1 & g_1 c_2 & \cdots & g_1 c_m \\ g_2 c_1 & g_2 c_2 & \cdots & g_2 c_m \\ \vdots & \vdots & \cdots & \vdots \\ g_n c_1 & g_n c_2 & \cdots & g_n c_m \end{bmatrix} \tag{32}$$

$$= \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} + \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \tag{33}$$

$$= \vec{f}\vec{a}^\top + \vec{g}\vec{c}^\top \tag{34}$$

(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:

$$R = \begin{bmatrix} | & | & | & | \\ -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} \\ | & | & | & | \end{bmatrix} \tag{35}$$

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!)

Solution: Principal component analysis is in general about understanding how best to approximate our (potentially) high-dimensional data (like recordings from a microphone, or in this case, a movie's ratings by lots of people) with its lower-dimensional essence. The first principal component is about seeing which one-dimensional line best approximates the data points — i.e. which is the line for which projecting the data points onto it results in "estimates" that are as close as possible to the data points.

In the case of this problem, every point is explicitly given as a multiple of a single vector $\vec{w}$ and so the data already lies on such a straight line going through the origin. So, the first principal component is just along the direction of $\vec{w}$. Because a principal component represents a direction, it is conventional to normalize the vector to have unit length. In this case, we are told that the vector $\vec{w}$ has length 7, and so the answer is $\frac{\vec{w}}{7}$.

(Because the line is all that matters, you could also have used the negative of this $-\frac{\vec{w}}{7}$.)

We can also do this using the SVD.

The singular value decomposition of a matrix R is a way of decomposing R into a sum of rank 1 matrices. In this sum the i^{th} rank 1 matrix is formed from taking the outer product of normalized column vectors $\vec{u}_i$ and normalized row vectors $\vec{w}_i^\top$, scaled by their respective singular values σ_i.

Looking at our given R, we can see that the matrix itself is rank 1 as the columns are all multiples of the same vector: $\vec{w}$. Seeing this we realize we can rewrite the matrix R as the following outer product:

$$R = \begin{bmatrix} | \\ \vec{w} \\ | \end{bmatrix} \begin{bmatrix} -1 & -2 & 2 & 4 \end{bmatrix} \tag{36}$$

However the SVD requires we normalize the vectors $\vec{u_1}$ and $\vec{v_1}^\top$. In order to reconstruct A properly we must scale back with the norms that we divided out to normalize.

$\|[-1,-2,2,4]\| = \sqrt{25} = 5$ and $\|\vec{w}\| = 7$. Consequently, when we pull that out, we get $\sigma_1 = 35$ as the singular value that corresponds to the first (and only) principal component.

Thus we can write the SVD of R as:

$$R = \begin{bmatrix} | \\ \frac{\vec{w}}{7} \\ | \end{bmatrix} 35 \begin{bmatrix} \frac{-1}{5} & \frac{-2}{5} & \frac{2}{5} & \frac{4}{5} \end{bmatrix} \tag{37}$$

Now we just have to pick which normalized vector to deem the principal component. Since our data (the movie ratings) are collected as columns we choose $\frac{\vec{w}}{7}$ as the principal component.

Alternate Solution:
To find the first principal component along the columns, we can use Σ and U. This is because our data is stored in the columns of R. We know that

$$R = U\Sigma V^T \tag{38}$$
$$\implies RR^T = U\Sigma\Sigma^T U^T \tag{39}$$

where $\Sigma\Sigma^T$ represents the square diagonal matrix with $min(m, n)$ singular values squared on the diagonal. Plugging in for R gives

$$RR^T = \begin{bmatrix} -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} \end{bmatrix} \begin{bmatrix} -\vec{w} \\ -2\vec{w} \\ 2\vec{w} \\ 4\vec{w} \end{bmatrix} \tag{40}$$
$$RR^T = \vec{w}\vec{w}^T + 4\vec{w}\vec{w}^T + 4\vec{w}\vec{w}^T + 16\vec{w}\vec{w}^T \tag{41}$$
$$RR^T = 25\vec{w}\vec{w}^T \tag{42}$$

By the Spectral Theorem, we know that the eigenvectors of RR^T correspond to the columns of U and the eigenvalues of RR^T correspond to the singular values squared. By inspection, we can see that the eigenvector is $\vec{w}$. Solving for the eigenvalue and singular value:

$$RR^T\vec{w} = 25\vec{w}\vec{w}^T\vec{w} \tag{43}$$
$$RR^T\vec{w} = (25\vec{w}^T\vec{w})\vec{w} \tag{44}$$
$$\implies \lambda = 25\vec{w}^T\vec{w} = 5^2 7^2 \tag{45}$$
$$\implies \sigma = \sqrt{\lambda} = 35 \tag{46}$$

For PCA, we require normalized vectors, so for that reason our first principal component is $\frac{\vec{w}}{\|\vec{w}\|} = \frac{\vec{w}}{7}$ with a corresponding singular value of $\sigma = 35$.

(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:

$$R = \begin{bmatrix} | & | & | & | & | & | \\ -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} & -3\vec{p} & 3\vec{p} \\ | & | & | & | & | & | \end{bmatrix} \tag{47}$$

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.)*

Solution: The solution to the previous part tells you what we need to do. We need to find the best two-dimensional subspace that best represents our data.

We start by taking the SVD of R.

The columns of R are all multiples of two vectors: $\vec{w}$ and $\vec{p}$. Each of these can be used to create a rank 1 matrix, and these can be summed together to form R.

Since $\vec{w}$ and $\vec{p}$ are orthogonal to one another, our life is easier. This problem's R matrix is made especially nice by seeing that a data point is either purely in the $\vec{w}$ direction, or the $\vec{p}$ direction.

Using this knowledge we rewrite R as:

$$R = \begin{bmatrix} | \\ \vec{w} \\ | \end{bmatrix} \begin{bmatrix} -1 & -2 & 2 & 4 & 0 & 0 \end{bmatrix} + \begin{bmatrix} | \\ \vec{p} \\ | \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & -3 & 3 \end{bmatrix}. \tag{48}$$

The orthogonality relationships demanded by the SVD are clearly satisfied since the row-vectors involved above have disjoint support (i.e. when one is nonzero, the other is zero) and the columns are orthogonal since we've been told so.

However for the SVD the vectors: $\vec{u_1}$, $\vec{v_1}^\top$, $\vec{u_2}$ and $\vec{v_2}^\top$ must be normalized and each rank 1 matrix must be scaled by the appropriate σ_i to allow the sum to properly reconstruct R. We also need to figure out which σ_i is bigger so we can order them properly. In the previous part, we have already done the calculations for $\vec{w}$'s part in this story. So what remains is the $\vec{p}$ part. Clearly the norm of the relevant row is $3\sqrt{2}$ which the norm of the relevant column is 7. So the singular value in question is $21\sqrt{2}$.

Using this we can rewrite R as:

$$= \begin{bmatrix} | \\ \frac{\vec{w}}{7} \\ | \end{bmatrix} 35 \begin{bmatrix} \frac{-1}{5} & \frac{-2}{5} & \frac{2}{5} & \frac{4}{5} & 0 & 0 \end{bmatrix} + \begin{bmatrix} | \\ \frac{\vec{p}}{7} \\ | \end{bmatrix} 21\sqrt{2} \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{-3}{3\sqrt{2}} & \frac{3}{3\sqrt{2}} \end{bmatrix}. \tag{49}$$

Since $35 > 21\sqrt{2}$, we arrange them in descending order such that our singular values are $\sigma_1 = 35$ and $\sigma_2 = 21\sqrt{2}$. Thus $\frac{\vec{w}}{7}$ which corresponds to σ_1 is still the first principal component vector and $\frac{\vec{p}}{7}$ which corresponds to σ_2 is the second principal component vector.

Alternate Solution:

Again, to find the first principal component along the columns, we can use Σ and U. This is because our data is stored in the columns of R. We know that

$$R = U\Sigma V^T \tag{50}$$

$$\implies RR^T = U\Sigma\Sigma^T U^T \tag{51}$$

where $\Sigma\Sigma^T$ represents the square diagonal matrix with $min(m, n)$ singular values squared on the diagonal. Plugging in for R gives

$$RR^T = \begin{bmatrix} -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} & -3\vec{p} & 3\vec{p} \end{bmatrix} \begin{bmatrix} -\vec{w} \\ -2\vec{w} \\ 2\vec{w} \\ 4\vec{w} \\ -3\vec{p} \\ 3\vec{p} \end{bmatrix} \tag{52}$$

$$RR^T = \vec{w}\vec{w}^T + 4\vec{w}\vec{w}^T + 4\vec{w}\vec{w}^T + 16\vec{w}\vec{w}^T + 9\vec{p}\vec{p}^T + 9\vec{p}\vec{p}^T \tag{53}$$

$$RR^T = 25\vec{w}\vec{w}^T + 18\vec{p}\vec{p}^T \tag{54}$$

By the spectral decomposition, we know that the eigenvectors of RR^T correspond to the columns of U and the eigenvalues of RR^T correspond to the singular values squared. By inspection, we can see that the first eigenvector is $\vec{w}$ and the second one is $\vec{p}$. Starting with the first:

$$RR^T\vec{w} = 25\vec{w}\vec{w}^T\vec{w} + 18\vec{p}(\vec{p}^T\vec{w}) \tag{55}$$

$$RR^T\vec{w} = (25\vec{w}^T\vec{w})\vec{w} + 0 \tag{56}$$

$$\implies \lambda_1 = 25\vec{w}^T\vec{w} = 5^2 7^2 \tag{57}$$

$$\implies \sigma_1 = \sqrt{\lambda_1} = 35 \tag{58}$$

When computing the first eigenvalue, we used the fact that $\vec{p}^T\vec{w} = 0$. We can repeat the same process for the second eigenvalue:

$$RR^T\vec{p} = 25\vec{w}(\vec{w}^T\vec{p}) + 18\vec{p}(\vec{p}^T\vec{w}) \tag{59}$$

$$RR^T\vec{p} = 0 + 18(\vec{p}^T\vec{p})\vec{p} \tag{60}$$

$$\implies \lambda_2 = 18\vec{p}^T\vec{p} = 3^2 7^2 2 \tag{61}$$

$$\implies \sigma_2 = \sqrt{\lambda_2} = 21\sqrt{2} \tag{62}$$

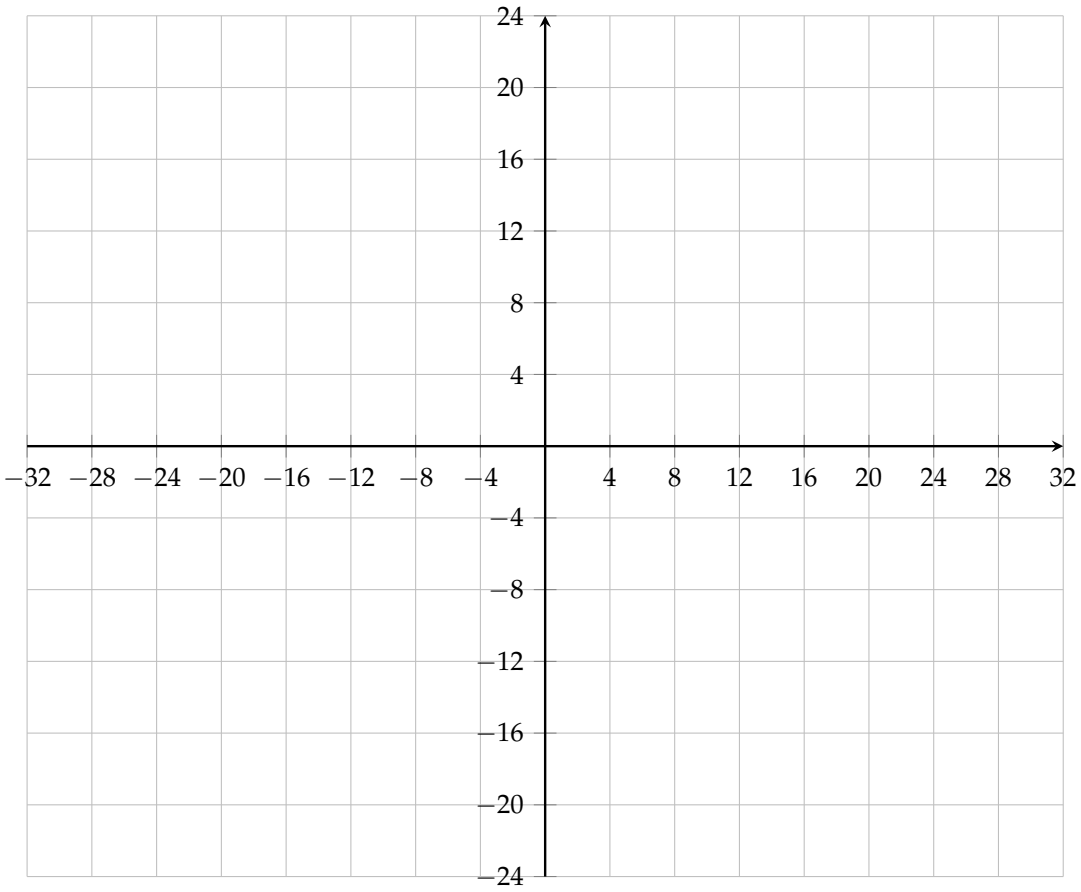
For PCA, we require normalized vectors, so for that reason our first principal component is $\frac{\vec{w}}{\|\vec{w}\|} = \frac{\vec{w}}{7}$ with a corresponding singular value of $\sigma_1 = 35$. Our second principal component is $\frac{\vec{p}}{\|\vec{p}\|} = \frac{\vec{p}}{7}$ with a corresponding singular value of $\sigma_2 = 21\sqrt{2}$. As mentioned in the solution above, we choose $\sigma_1 = 35$, $\sigma_2 = 21\sqrt{2}$ in this order because $35 > 21\sqrt{2}$.

(d) In the previous part, you had

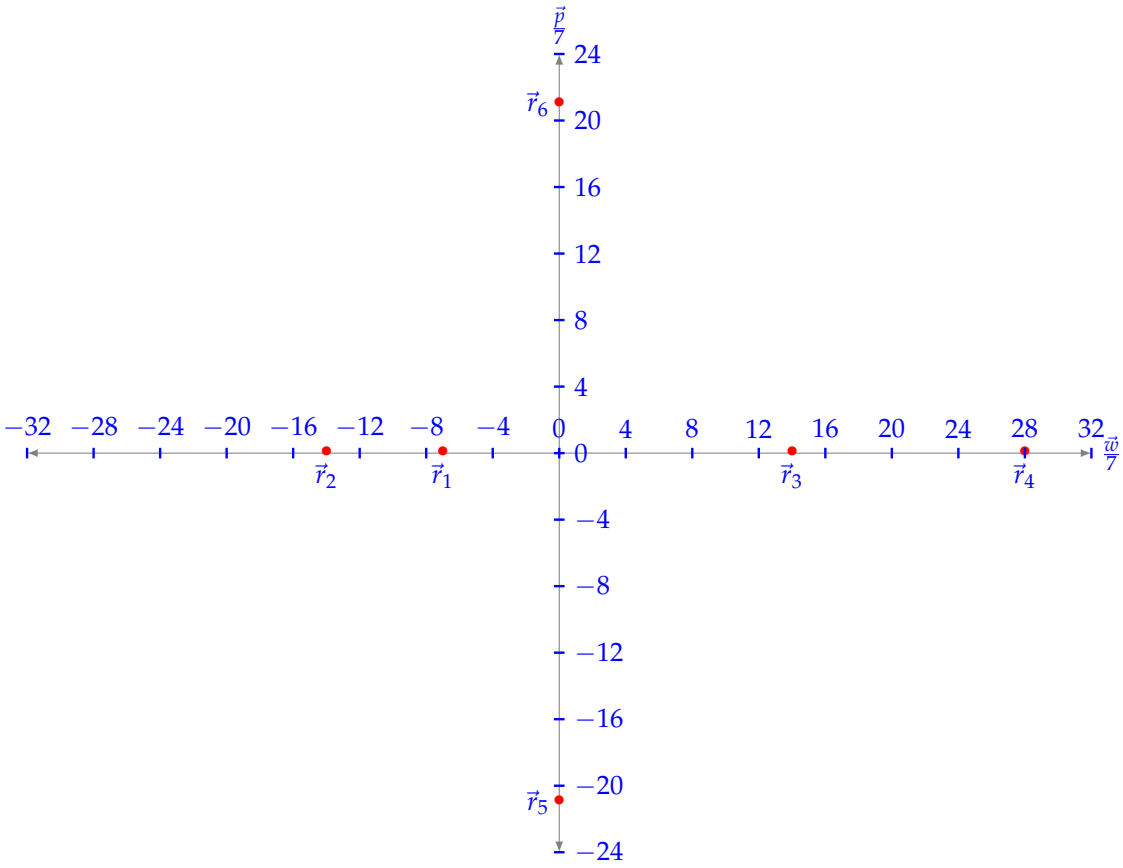
$$R = \begin{bmatrix} | & | & | & | & | & | \\ -\vec{w} & -2\vec{w} & 2\vec{w} & 4\vec{w} & -3\vec{p} & 3\vec{p} \\ | & | & | & | & | & | \end{bmatrix}$$

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 $\vec{r}_i$ (for all 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**.



Solution: Once we know what the principal components are, we know that the first four data points are just multiples of the first principal component and the last two data points are just multiples of the second principal component. What multiples? For the first four, the multiples are clearly $-7, -14, 14, 28$ since the norm of $\vec{w}$ is 7. For the final two, the multiples are clearly $-21, +21$ since the norm of $\vec{p}$ is also 7. Plotting:



5. Brain-Machine Interface and Neuron Sorting

The iPython notebook `pca_BMI.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 rows of 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`**?**

Solution: Part a) of the notebook `pca_brain_machine_interface_sol.ipynb` contains solutions to this exercise. `three_neurons_training.shape = (4398, 32)`

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

Solution:

Each row of the training data matrix corresponds to a captured waveform of a (possibly different) neuron. The columns of the training data set correspond to the times at which the neuron's waveform was sampled. Here, the times are all relative to the "start" of each waveform.

In the real world, this data would have come from a segmentation (i.e., cutting into pieces) of a long recording of a trace from an electrode embedded in the brain. We have not taught you how to do such segmentation in this course. However, even from 16A, you might have a rough idea of how to do it. You could simply look for time-periods in which there is some activity and then cut those out. In the real-world, the problem of segmentation is done iteratively with understanding what the signals of interest look like. This is because we would ideally like to align all the waveforms to start at the right starting point. (This should remind you a bit of the positioning lab that you did in 16A.) For this knowing the waveforms is useful. These kinds of "chicken-egg" problems are solved iteratively using the same philosophy that is done in k-means clustering (which some of you might have seen in 61A's yelp-like project, although it hasn't run recently.).

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`**?**

Solution: Our training data `three_neurons_training` was of shape (4398, 32). When we take the SVD our three matrices U, Σ and V will be of shapes (4398 x 4398), (4398 x 32), and (32 x 32), respectively.

(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?**

Solution: Each of our waveforms is a time series of length 32 and thus can be represented as a 32 x 1 vector. Since our data is along the rows of our data matrix, vectors in matrix V corresponds to our basis.

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

Solution: In .ipynb

(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.

Solution: In .ipynb

(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?

Solution: We can see that since there is a larger overlap between the presorted points this random basis does not create a good separation.

(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.**

Solution: In .ipynb. The projections created using PCA have more separation.

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

Solution: In .ipynb. It is not necessary to use 3 principal components since the 3-D plots show us that there are similar clusters in comparison to 2-D.

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.**

Solution: In .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.

Solution: In .ipynb

(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.)*

Solution: 2 - we can see that the clusters are relatively similar at each Z location, so we only need to worry about their location in the x-y plane.

(m) 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).

Since we already have some presorted data, we can use this information to aid in classification. We can calculate our centroids by taking the empirical mean of the presorted data corresponding to a particular neuron firing. **Use this method to find centroids in the corresponding section of the .ipynb file. Then use** `which_neuron` **on the** `two_classified` **data set and count the number of times each neuron fired.**

Solution: In the .ipynb file

6. **(OPTIONAL) Make Your Own Problem.**

   Write your own problem about content covered in the course thus far, and provide a thorough solution to it.

   NOTE: This can be a totally new problem, a modification on an existing problem, or a Jupyter part for a problem that previously didn't have one. Please cite all sources for anything (including course material) that you used as inspiration.

   NOTE: High-quality problems may be used as inspiration for the problems we choose to put on future homeworks or exams.

7. **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.)

   (c) **Roughly how many total hours did you work on this homework? Write it down here where you'll need to remember it for the self-grade form.**

Contributors:

- Daniel Abraham.
- Ashwin Vangipuram.
- Tanmay Gautam.
- Aditya Arun.
- Kourosh Hakhamaneshi.
- Antroy Roy Chowdhury.
- Nikhil Shinde.
- Nathan Lambert.
- Anant Sahai.
- Wahid Rahman.