EECS 16B Designing Information Devices and Systems II
Spring 2019 UC Berkeley

Homework 10

This homework is due on Wednesday, April 17, 2019, at 11:59PM.
Self-grades are due on Saturday, April 20, 2019, at 11:59PM.

Homework 10 is ***mostly optional****. You* ***must*** *do problem 1 (the brain-machine interface problem), but* ***all other questions are optional*** *to reduce pressure ahead of the exam.*

1. **Brain-machine interface**

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. This exercise will help make sure you are prepared for the next two labs. Make sure you complete this problem before your lab next week.

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.

Luckily for us, it is highly unlikely for two neurons next to each other to fire at precisely the same time, so in practice the "average" potential waveform looks like a sequence of waveforms from different neurons. Additionally, 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. We provide both training and test sets with two and 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) A waveform occurs when the potential passes a certain threshold in a neuron. In our data set, each recorded waveform contains 'N=32' samples taken at a sampling frequency of 40kHz. We can represent each vector exactly with any 32-dimensional basis, as long as the vectors making up the basis are linearly independent. However, we want to compute a minimal set of basis vectors that brings out the important features of the data using Principal Component Analysis (PCA). This idea is called "dimensionality-reduction" and is a way to both make things more tractable computationally as well as to focus on the important parts of signals.

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). **What are the dimensions of the U and V matrices returned from taking the SVD? Which of these matrices can we use to construct our desired basis?**

(e) **Complete the functions in the .ipynb to implement PCA on the training data and projecting waveform traces down into their reduced-dimensionality coordinates.** Don't forget that you will need to offset the data by the mean (We do this because we expect that there is something common to all of the neural waveforms. We are going to use these vectors to classify later and do not want the shared mean to have an effect since it does not help us distinguish different neurons.)!

(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 principle 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) Now, **project 'two_neurons_test' onto the 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 in the last part.**

Task 2: Three Neuron Sorting

(i) In the previous part we projected 2 neurons onto two 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.**

(j) 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.

(k) **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 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 "read minds" using 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).

(l) **Complete the code to classify each data point, given 2 centroids, using the method described above**

(m) 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 technique to find centroids in the corresponding section of the .ipynb file. Then use 'which_neuron' to the two_classified data set and count the number of times each neuron fired**

(n) Oftentimes when we are doing such problems in practice we do 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 Yelp Maps project). We can use this information to create a rough estimate for the average firing rate of each neuron over the time spanned by the given data.

Scipy just happens to have a built in function to compute k-means given clustered data, which we have conveniently formatted formated in the above parts.

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

2. PCA (OPTIONAL)

We collect temperature (°F) and humidity (%) data each day for four days. We can organize our data into 4×2 matrix A, where the each column of A is a type of data (that is, temperature and humidity) and each row is a particular day's measurement.

$$A = \begin{bmatrix} 64 & 39 \\ 66 & 45 \\ 70 & 41 \\ 64 & 39 \end{bmatrix}$$

We're going to work through the principal component analysis of A to illustrate how we can analyze the data that A represents.

(a) **Find the covariance matrix S of A.** The diagonal terms of S represent the variance of each measurement type, while the off-diagonal terms represent the covariance between each different measurement type. Remember that you must subtract the mean of each column of A in order to mean-center the data before finding S.

Since S is a symmetric matrix, we can eigendecompose it into the form $S = P\Lambda P^T$, where P has columns that contain the eigenvectors of S, and Λ is a diagonal matrix with the eigenvalues of S along the diagonal. The eigenvectors of S are equivalent to the principal components of A, for which S is the covariance matrix.

These eigenvalues along the diagonal of Λ are equivalent to the squared weights of the corresponding principal components of A in the columns of P.

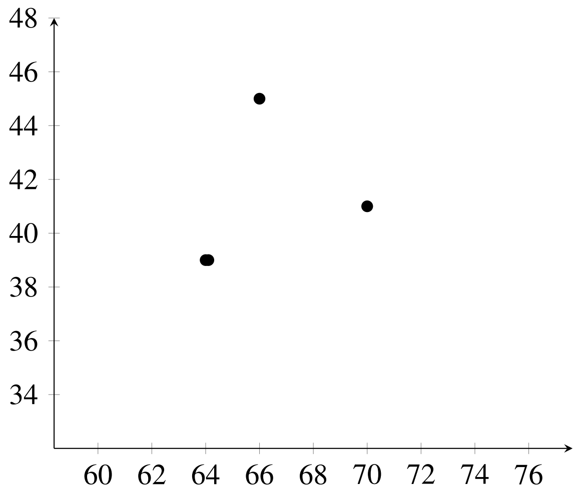
(b) **Find the eigenvalues of S and order them from largest to smallest, $\lambda_1 > \lambda_2$.**

(c) **Find the orthonormal eigenvectors $\vec{p}_i$ of S.** All eigenvectors are mutually orthogonal, and can be normalized to have unit length.

(d) Given what you have found above, **what are the principal components of the matrix A? What is the principal component matrix, P?**

(e) **What are the weights of each principal component?** These weights relate to the variance of the data in A along each principal component and indicate the relative strength of each component as a representation of the data.

(f) We have plotted the data points from A in the graph below, with column 1 data on the x axis and column 2 data on the y axis. **Plot the two principal components scaled by their weights on the following graph.** Remember that we subtracted the column means from each column before doing other computations. Think about what that means for how those two principal components should be depicted. Also, in case you were wondering, there visually look like there are three points because two of them are the same — so they landed on top of each other.



3. MIMO wireless signals (OPTIONAL)

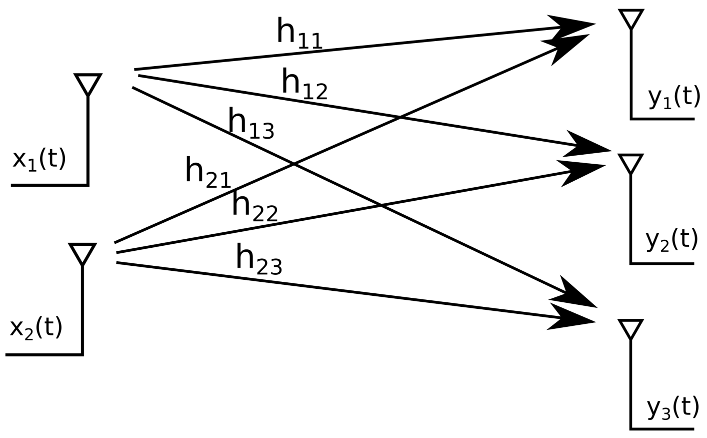
Ever wonder why newer wifi routers and cellular base stations have 4 or sometimes even more antennas on them? New wireless technologies actually use multiple antennas that each send their own signal on the same frequency band. The key here is not only do we encode signals in frequency bands, but also in spatial ones using a technique known as "Spatial Multiplexing".

We call this idea "MIMO" wireless, which stands for "multiple input multiple output". This technique is used in many standards including 802.11n/ac, 4G LTE, and WiMAX.

In this problem, we will explore how signals are decoded on the output end.

Consider the following:

We have 2 transmit antennas and 3 receive antennas, each receive antenna gets some signal from each of the transmit antennas. We can model the input output relation of the system as follows:



$$\begin{bmatrix} h_{11} & h_{21} \\ h_{12} & h_{22} \\ h_{13} & h_{23} \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}$$

or

$$H\vec{x}(t) = \vec{y}(t)$$

Here, H is the spatial-response matrix and it acts on the signals instantaneously at each time. For the purpose of this problem, we are going to pretend that there are no echoes across time.

(a) With our new MIMO wireless system, we want to recover the original $\vec{x}(t)$ signal after receiving the $\vec{y}(t)$. But we have three observations and only want to recover two inputs. In order to do this, we will left multiply $\vec{y}(t)$ by some matrix A; ideally we should then exactly recover $\vec{x}(t)$ (i.e. $A\vec{y}(t) = \vec{x}(t)$) or there is noise, get close. **Using the SVD to decompose $H = U\Sigma V^T$, analytically write down what this matrix A should be if you want to find the least-squares solution for $\vec{x}$. Simplify the matrix as much as possible.**

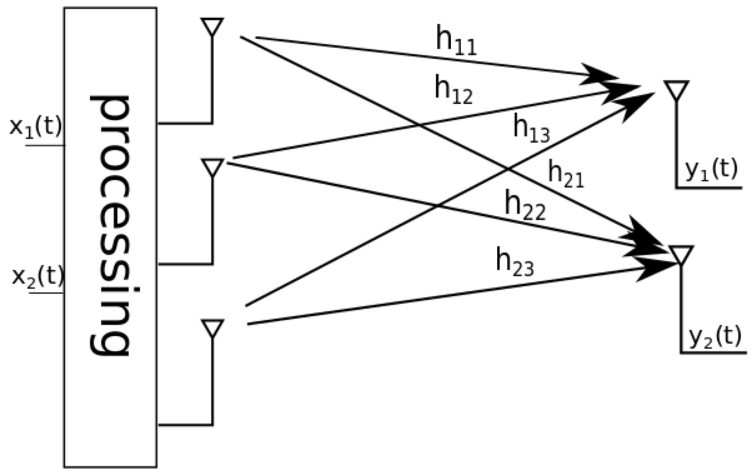
(b) **Comment on the form of the solution you found above in relation to the form of the minimum-norm solution you found using the Moore-Penrose Psuedoinverse in the previous homework.**

(c) What we just did is referred to as "post-processing" or "post-coding", and involves the receive end having more antennas than the send side. Many times this is not the case (eg. a wireless cell tower having many more antennas than a phone). What if we wanted to send 2 streams on 3 antennas and receive precisely those 2 streams back on the other end?

The channel is very similar to the one we had made in part (a). In fact, the original channel modelled with spatial response matrix H is precisely the transpose of this channel! Thus, we can say the spatial response matrix for this channel, let's call it H', is simply the following:

$$H' = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix} = H^T$$

Using the SVD of H and its relation to H', **show how you can pre-process $x_1(t)$ and $x_2(t)$ to get three numbers $w_1(t), w_2(t), w_3(t)$ that are actually transmitted over the three antennas.** They need to



be such that you recover them precisely after they have been transmitted across the channel without doing any further processing. To be more explicit, after the processing and transmission has been done $y_1(t) = x_1(t), y_2(t) = x_2(t)$. **By what matrix should you multiply $\vec{x}$ to get $\vec{w}$?**

(d) (Optional) **Why do you think that we are using the SVD here? Is there a unique solution of what to transmit that will achieve the desired goal in the previous part? Why choose this approach?**

4. Write Your Own Question And Provide a Thorough Solution.

Writing your own problems is a very important way to really learn material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top level. We rarely ask you any homework questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself (e.g. making flashcards). But we don't want the same to be true about the highest level. As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams. Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't ever happen.

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

(c) **How did you work on this homework?** (For example, *I first worked by myself for 2 hours, but got stuck on problem 3, so I went to office hours. Then I went to homework party for a few hours, where I finished the homework.*)

(d) **Roughly how many total hours did you work on this homework?**

Contributors:

- Nikhil Shinde.
- Gireeja Ranade.
- 16 Founders.
- Kyle Tanghe.
- Regina Eckert.
- Saavan Patel.