Instructions

The following questions should be answered by hand, and you must show all your work. Questions 2c and 3b require using Python; please turn in your code and any requested outputs along with your handwritten work. Your solutions should be submitted to Canvas as a single PDF.

Assignment

1. **Projection Matrices**
   As we learned in class, a *projection matrix* $P \in \mathbb{R}^{n \times n}$ is a matrix that satisfies $P = P^\top$ and $P = P^2$.
   (a) Show that if P is a projection matrix then so is $I - P$
   (b) Suppose that $U \in \mathbb{R}^{n \times k}$ is an orthogonal matrix (in other words, its columns are orthonormal). Show that $UU^\top$ is a projection matrix.
   (c) Suppose that $A \in \mathbb{R}^{n \times k}$ is full rank, with $k \leq n$. Show that $A(A^\top A)^{-1}A^\top$ is a projection matrix.

2. **Color Perception**
   In humans, color perception is based on the responses of three types of color light receptors, called *cones*. The three types of cones respond in a different way to the *light spectrum* $\vec{p} \in \mathbb{R}^n$, which defines the power of incoming light at n different wavelenghts. The cones are called L, M, and S because they respond mainly to long, medium, and short wavelength light, respectively. Here, we will divide the visible spectrum of light into $n = 20$ wavelength bands and model the cones' responses with the following:

$$L_{\text{cone}} = \sum_{i=1}^{20} l_i p_i, \quad M_{\text{cone}} = \sum_{i=1}^{20} m_i p_i, \quad S_{\text{cone}} = \sum_{i=1}^{20} s_i p_i,$$

   where p_i is the incoming power in the i^{th} wavelength band, and l_i, m_i, and s_i are nonnegative constants that describe the spectral responses of the different cones. The color that we perceive is a complex function of the vector $(L_{\text{cone}}, M_{\text{cone}}, S_{\text{cone}})$, with different cone response vectors perceived as different colors. (In reality, color perception is more complicated, but the basic idea is correct).

(a) When are two different incoming light spectra, $\vec{p}$ and $\vec{p}_0$, visually indistinguishable from each other? Your answer should include a matrix/vector equation and the language of linear algebra.

(b) In a color matching problem, an observer is shown a test light and is asked to change the intensities of three primary lights until the sum of the three primary lights looks like the test light. In other words, the observer is asked to find a spectrum $\vec{p}_{\text{match}}$ that is visually indistinguishable from the test light spectrum $\vec{p}_{\text{test}}$. The matching spectrum has the form

$$\vec{p}_{\text{match}} = a_1\vec{u} + a_2\vec{v} + a_3\vec{w},$$

where $\vec{u}, \vec{v}, \vec{w}$ are the spectra of the primary lights, and a_i are the intensities to be found. Can this always be done? Discuss.

(c) A computer monitor has three primary lights (or *phosphors*), R, G, and B. We want to adjust the intensity of the phosphors to create a color that looks like a reference test light. Find the weights that achieve the match or explain why no such weights exist. The data for this problem is in `color_perception.json`, which contains the vectors `wavelength`, `B_phosphor`, G_phosphor, R_phosphor, L_coefficients,M_coefficients, S_coefficients, and test_light. You can open the data file as a python dictionary with the following code segment:

```
import json
with open("color_perception.json", "r") as f:
    data = json.load(f)
```

3. **State estimation with bias and drift** Many digital sensors work by measuring an electric current or voltage that is *correlated* to the variable that you are actually trying to measure. For example, a temperature sensor might measure electric current in a circuit, but then solve an

equation to provide the user with an estimate of the circuit temperature rather than current. Here we consider the system:

$$y_i = \vec{a}_i^\top \vec{x} + v_i, \text{ for } i = 1, \ldots, m,$$

where we define

- y_i is the i^{th} measurement of the electric current
- $\vec{x} \in \mathbb{R}^n$ is the vector of temperature values that we want to estimate from the measurements. This is often called the *state* that we are trying to estimate.
- v_i is the error in measurement i

We will assume that all measurements are taken starting at time T, and at equal time intervals T thereafter, such that the i^{th} measurement occurs at time $t = iT$. The row vectors $\vec{a}_i^\top$ can be placed into an *observation matrix* defined as

$$A = \begin{bmatrix} a_1^\top \\ a_2^\top \\ \vdots \\ a_m^\top \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

You can assume that $m \geq n$ and that A is full rank.

Usually, we assume that the error term v_i is just random noise from a Gaussian distribution, which we will call w_i. But in this case, we will assume that the error also has a constant *bias*, or offset, denoted α, and a *drift* term βiT that causes the measurements to slowly change over time. Therefore, the total error in measurement i is given by

$$v_i = \alpha + \beta iT + w_i.$$

Finally, your goal:

(a) Show how you can solve for the temperature vector $\vec{x}$, the bias constant α, and the drift constant β using least-squares. If there are conditions that must be met (e.g., conditions for the observation matrix A) in order for the method to work, then explain those conditions and give an example of where the conditions are not met.

(b) Apply your method to the data provided in `sensor_data.npy`. Please print out your solution for the temperature vector $\vec{x}$, the bias constant α, and the drift constant β. The file contains an observation matrix A and a measurement vector y. You can load the data with the following code:

```
import numpy as np
data = np.load(
    "sensor_data.npy",
    allow_pickle=True
).item()
```

4. **Least squares with weighted residuals**
In class, we learned about least-squares as a way to project a target vector $\vec{b} \in \mathbb{R}^m$ onto the subspace spanned by the columns of a feature matrix $A \in \mathbb{R}^{m\times n}$ by finding an optimal parameter vector $\vec{x} \in \mathbb{R}^n$. Another way to think about least-squares is that you are trying to find the vector $\vec{x}$ that minimizes the squared length of the *residual* vector $\vec{r} = A\vec{x} - \vec{b}$. In optimization terminology, this corresponds to minimizing a *cost function*:

$$\|A\vec{x} - \vec{b}\|^2 = \sum_{i=1}^{m} \left(\vec{a}_i^\top \vec{x} - b_i\right)^2,$$

where $\vec{a}_i^\top$ are the columns of $A^\top$ (which is the same as the rows of A).

In a variation of least squares called *weighted least squares*, we instead seek a vector $\vec{x}$ that minimizes the modified cost function

$$\sum_{i=1}^{m} w_i \left(\vec{a}_i^\top \vec{x} - b_i\right)^2,$$

where each of the w_i are known positive weights. The purpose of the weights is to allow us to assign more importance to certain components of the residual vector. This might be useful if, for example, you know that some elements of the target vector $\vec{b}$ were measured more accurately, or are more representative of future situations that you will apply your model to.

(a) Show that the weighted cost function can be written as

$$\|D(A\vec{x} - \vec{b})\|^2$$

for a particular diagonal matrix D. What are the elements of D? *Note*: This allows us to solve the weighted problem by minimizing $\|B\vec{x} - \vec{d}\|$, with $B = DA$ and $\vec{d} = D\vec{b}$.

(b) Show that if A has linearly independent columns, then B also has linearly independent columns.

(c) The standard least squares solution is given by $\hat{x} = (A^\top A)^{-1} A^\top \vec{b}$. Find the analogous formula for the solution to the weighted least squares problem. Your answer should be in terms of A, $\vec{b}$, and a diagonal matrix $W = \mathbf{diag}(\vec{w})$.

5. **Feature augmentation via averaging**
You are using least squares to fit a regression model to $\vec{b} \approx A\vec{x}$, where $\vec{b}$ is the target variable, $\vec{x}$ are the fit coefficients you are trying to find, and your feature matrix A is given by

$$A = \begin{bmatrix} | & & | & | \\ \vec{a}_1 & \cdots & \vec{a}_n & 1 \\ | & & | & | \end{bmatrix}.$$

The vectors $\vec{a}_i$ are the features and we have appended a column of 1s to additionally fit an intercept term.

A friend suggests adding a new feature, $\vec{a}_{n+1}$, that is the average of all the other features. He says that by adding this new feature, you may end up with a better model. Is this a good idea? Why or why not?