Data 5100 Assignment 6

Hrishabh Kulkarni

Soln. 1:

(a) To show : $I-P \rightarrow$ Projection Matrix, if P is a proj. Matrix.

Matrix $P \in \mathbb{R}^{n \times n} \rightarrow$ Proj. Matrix if: $P = P^T$ and $P = P^2$

$Q = I - P$

Check if Q has both same properties as proj. matrix,

check if, $Q = Q^T$ and $Q \cdot Q = Q$.

$\rightarrow$ ① $Q^T = (I-P)^T$

$= I^T - P^T$

$= I - P$

$= Q$

$\therefore$ Confirming that $Q = Q^T$

So, Q is Symmetric.

$\rightarrow$ ② $Q^2 = (I-P)(I-P)$

$= I - 2P + P^2$

As $P \times P = P$, substituting in P^2

$\therefore Q^2 = I - 2P + P$

$= I - P$

$= Q$

$\therefore$ Confirming that $Q^2 = Q$.

$Q = I - P$ is Symmetric and satisfies $Q^2 = Q$ as well.

So, Q is a Projection Matrix.

(b) To Show: If $U \in \mathbb{R}^{n \times *}$ is Orthogonal mat., then UU^T is a projection matrix.

Matrix $U \rightarrow$ Orthogonal mat., So, $U^TU = I \rightarrow$ each col. of U is unit length and orthogonal to others.

$\rightarrow$ To check $UU^T = (UU^T)^T$

$(UU^T)^T = U(U^T)^T$

$= UU^T$

Confirming $UU^T = (UU^T)^T$, so, it's Symmetric.

→ check $UU^T \cdot UU^T = UU^T$;

$$(UU^T)(UU^T) = U(U^TU)U^T$$
$$= U \cdot I \cdot U^T$$
$$= UU^T$$

Confirming the above.

Since, UU^T is Symmetric and satisfies $(UU^T)(UU^T) = UU^T$,
It is a Projection Matrix.

(c) To Show: $A \in \mathbb{R}^{n \times k}$ is full Rank with $k \leq n$,
then $A(A^TA)^{-1}A^T$ → is a projection Matrix.

$$P = A(A^TA)^{-1}A^T$$

→ check if $P = P^T$ and $P^2 = P$

① $P^T = (A(A^TA)^{-1}A^T)^T$
$= A((A^TA)^{-1})^T A^T$
$= A(A^TA)^{-1}A^T$
$= P$

So, $P^2 = P$

② $P \cdot P = P$
$P^2 = (A(A^TA)^{-1}A^T)(A(A^TA)^{-1}A^T)$
$P^2 = A(A^TA)^{-1}(A^TA)(A^TA)^{-1}A^T$
$= A(A^TA)^{-1}A^T$
$= P$

we can say that

So, finally, $A(A^TA)^{-1}A^T$ is a projection matrix.

Sol 2: (A)

To show: When are two different incoming light spectra, $\vec{p}$ and $\vec{p_o}$, visually distinguishable from each other?

Two light spectra $\vec{p}$ and $\vec{p_o}$ are visually indistinguishable if they produce the same response in all three types of cones (L, M, S).

Each spectrum produces a response vector (Lcone, Mcone, Scone)

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

p_i: Power of light in the i^{th} wavelength.

l_i, m_i, s_i: Special Spectral response constants for L, M, S cones.

Matrix Equation :→ $\vec{C} \cdot \vec{p} = \vec{C} \cdot \vec{p_o}$ where $\vec{C}$ is a matrix with rows $(l_i), (m_i), (s_i)$ and $\vec{p}, \vec{p_o}$ are vectors representing the spectra.

$\vec{p}$ and $\vec{p_o}$ are indistinguishable if $\vec{C} \cdot (\vec{p} - \vec{p_o}) = 0$.

Example: If two spectra $\vec{p}$ and $\vec{p_o}$ have the same cone response values for L, M, and S, they will appear as the same color to the human eye, even if they are different in terms of the actual wavelengths.

Example: (B) $n = 3$, $\vec{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$

Constants for cones are given by:

$$\vec{C} = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ s_1 & s_2 & s_3 \end{bmatrix} = \begin{bmatrix} 0.8 & 0.1 & 0.3 \\ 0.2 & 0.9 & 0.4 \\ 0.3 & 0.2 & 0.7 \end{bmatrix}$$

The cone responses for spectrum $\vec{p}$ are:

$$\begin{bmatrix} L_{cone} \\ M_{cone} \\ S_{cone} \end{bmatrix} = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ s_1 & s_2 & s_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Assuming $\vec{p} = \begin{bmatrix} 10 \\ 5 \\ 2 \end{bmatrix}$ and $\vec{p_0} = \begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$

$\vec{c} \cdot \vec{p}$:

$$\begin{bmatrix} L_{cone} \\ M_{cone} \\ S_{cone} \end{bmatrix} = \begin{bmatrix} 0.8 & 0.1 & 0.3 \\ 0.2 & 0.9 & 0.4 \\ 0.3 & 0.2 & 0.7 \end{bmatrix} \begin{bmatrix} 10 \\ 5 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.8 \times 10 + 0.1 \times 5 + 0.3 \times 2 \\ 0.2 \times 10 + 0.9 \times 5 + 0.4 \times 2 \\ 0.3 \times 10 + 0.2 \times 5 + 0.7 \times 2 \end{bmatrix}$$

8+0.5+0.6

2+4.5+0.8

3+1+1

$$= \begin{bmatrix} 9.1 \\ 7.3 \\ 5.4 \end{bmatrix}$$

$$\vec{c} \cdot \vec{p_0} = \begin{bmatrix} 0.8 & 0.1 & 0.3 \\ 0.2 & 0.9 & 0.4 \\ 0.3 & 0.2 & 0.7 \end{bmatrix} \begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} 0.8 \times 8 + 0.1 \times 6 + 0.3 \times 4 \\ 0.2 \times 8 + 0.9 \times 6 + 0.4 \times 4 \\ 0.3 \times 8 + 0.2 \times 6 + 0.7 \times 4 \end{bmatrix}$$

6.4+0.6+1.2

1.6+5.4+1.6

2.4+1.2+2.8

Conclusion:

∴ They are different in wavelength but appears to be same.

$$= \begin{bmatrix} 8.2 \\ 8.6 \\ 6.4 \end{bmatrix}$$

(B) To show; In color matching problem, can an observer always adjust the intensities of 3 primary lights to match a test light.

→ observer must find a combination of intensities
a_1, a_2, a_3 → for 3 primary lights
with spectra → $\vec{u}, \vec{v}, \vec{w}$
to match a test light spectrum $\vec{p}$ test:
→ $\vec{p}$ match $= a_1\vec{u} + a_2\vec{v} + a_3\vec{w}$.

→ This is a linear combination of primary light spectra to produce a new spectrum $\vec{p}$ match, that matches the test light.

→ A match can always be found if cone response vector → (L, M, S)
for primary lights → $\vec{u}, \vec{v}, \vec{w}$
span the same space
as the test light's cones response.

→ Primary lights spectra must be linearly independent for them to span all possible cone responses.
If they are linearly independent, the observer can adjust a_1, a_2, a_3 to match any test light.

Example:
If primary light → $\vec{u}, \vec{v}, \vec{w}$ donot span of cone responses, they it might be impossible to match certain test lights, as there would be cone responses that cannot be reached throughout any combination of $\vec{u}, \vec{v}, \vec{w}$.

Example:

$$\vec{u} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \vec{w} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

$$\vec{p}_{test} = \begin{bmatrix} 3 \\ 2 \\ 3 \end{bmatrix}$$

The goal is to find scalars a_1, a_2, a_3 :

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

Part c → python code done based on above both explanation and understandings.

Soln 3:

(a) Solving using Least-Square:

$$y_i = \vec{a}_i^T \vec{x} + \alpha + \beta i T + w_i$$ [Formulate Model with Bias and Drift]

where,

$\vec{a}_i$ → observation vector at i^{th} measurement.

$\vec{x}$ → vector of unknown temperature values.

α → constant bias (offset).

$\beta i T$ → drift term, which is dependent on time $t = iT$.

w_i → random noise

Reformulate as a Linear System: Rewrite the model in matrix form to isolate $\vec{x}, \alpha, \beta$ as:

$$y = A\vec{x} + \alpha \cdot 1 + \beta \cdot T \cdot [0, 1, 2, \ldots, m-1]^T + w$$

Now, will define an augmented matrix that includes $\vec{x}$, α, and β as unknown. Matrix equation can be written as:

$$y = [A \; 1 \; T \cdot [0, 1, 2, \cdots, m-1]^T] \begin{bmatrix} \vec{x} \\ \alpha \\ \beta \end{bmatrix} + w$$

Call augmented matrix B and vector of unknowns $\vec{z}$:

$$y = B\vec{z} + w$$

The least Square Solution:

We want to find $\vec{z} = [\vec{x} \; \alpha \; \beta]^T$ that minimize the error. we use least square approach: $\vec{z} = (B^T B)^{-1} B^T y$.

Applying Conditions: For least-squares solution to work, $B^T B$ must be invertible.

This requires that:

→ $m \geq n+2$

→ B must be full rank (rank of B should be $n+2$).

Example of Conditions met:

If $m = 5$, $n = 3$ to consider.

$m \geq n+2$

and if A is designed with linearly independent rows and columns, then B will be full rank.

(5) Covered in the py code from my understanding.

Soln. 4:

(a) To show: The weighted cost function $\Rightarrow \|D(A\vec{x}-\vec{b})\|^2$

→ Weighted least squares cost function is:

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

Expressing the above in matrix form where,

$\|D(A\vec{x}-\vec{b})\|^2$, where D is a diagonal matrix containing square roots of weights.

1. Cost function can be rewritten as:

$$\sum_{i=1}^{m} w_i(r_i)^2 \quad \text{where } r_i = \vec{a}_i^T \vec{x} - b_i$$

2. This sum can be expressed as a norm squared:

$$= \sum_{i=1}^{m} (\sqrt{w_i} r_i)^2 = \|\sqrt{W}(A\vec{x}-\vec{b})\|^2$$

3. Let $D = \sqrt{W} = \text{diag}(\sqrt{w_1}, \sqrt{w_2}, \cdots, \sqrt{w_m})$,

So, the weighted cost function becomes:

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

where D is the diagonal matrix formed from the square roots of the weights w_i.

Example: If $A \rightarrow 3 \times 2$ matrix

and $\vec{b} \rightarrow$ vector

size 3

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

and $w_1 = 2$, $w_2 = 3$, $w_3 = 1$,

Then the diagonal matrix D would be ↙

$$D = \text{diag}(\sqrt{2}, \sqrt{3}, \sqrt{1})$$

(b) To Show: if A has linearly independent columns, then $B = DA$ also has linearly independent columns.

Explanations: If A has linearly independent columns, it means that there is no non-trivial linear combinations of columns of A
⇒ which results in zero vector.

Now, consider matrix $B = DA$, where D → Diagonal Matrix with non-zero elements (since all $w_i > 0$).

Proof:

1. Suppose columns of B are linearly dependent. Then there exists a non-zero vector $\vec{c}$ such that:
$$B\vec{c} = 0 \quad \text{or equivalently} \quad DA\vec{c} = 0$$

2. Since D is diagonal with non-zero entries, multiplying by D^{-1} (its inverse)
$$A\vec{c} = 0$$

3. But, since A has linearly independent columns, this implies that $\vec{c} = 0$, which is a contradiction.

Thus, the columns of B are also linearly independent.

(c) Find formula for solution to weighted least square problem

Standard Least Squares Solutions:

The Soln. to least square problem $\longrightarrow \hat{\vec{x}} = (A^T A)^{-1} A^T \vec{b}$

Weighted Least Squares Soln:

For weighted least squares problem, the cost function to minimize is:

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

The corresponding normal equation becomes:

$$(A^T W A)\hat{\vec{x}} = A^T W \vec{b}$$

where,

$$W = \text{diag}(w_1, w_2, \ldots, w_m)$$

Thus, the solution is:

$$\hat{\vec{x}} = (A^T W A)^{-1} A^T W \vec{b}$$

Example:

Using the previous matrix A and vector $\vec{b}$, and Weight matrix $\longrightarrow W$, the weighted least squares solution can be computed by substituting into the formula above. (I have read the example for reference) (but not written here).

Soln. 5: Problem : You are fitting a least squares regression model.

So,
Least Squares Regression minimizes the sum of squared errors:

$$\min_{\vec{x}} \| A\vec{x} - \vec{b} \|^2$$

where, $A \rightarrow$ Feature Matrix
$\vec{x} \rightarrow$ Vector of fit coefficient
$\vec{b} \rightarrow$ target vector.

New Feature

$$\vec{a}_{n+1} = \frac{1}{n} \sum_{i=1}^{n} \vec{a}_i$$

This Means that new feature is just average of existing features $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n$.

Issues with Adding $\vec{a}_{n+1}$:

- Collinearity :
  $\vec{a}_{n+1}$ is a linear combination of the other features, so that we can say it is pearfectly collinear.

- Singular Matrix:
  Collinearity makes $A^T A$ singular or poor with conditions, which can lead to numerical instabalness.

- Redudancy / Repetedness :
  The new feature adds no new information only redudancy, which does not improve the model.

Conclusion:

In this the answer should be No.

It is not a good idea because adding $\rightarrow \vec{a}_{n+1}$ introduces collinearity, making the matrix $A^T A \rightarrow$ singular.

This leads $\rightarrow$ numerical issues

This can reduce the performance of the model.

The new feature does not ~~include~~ include/provide any new information $\rightarrow$ which may lead to $\rightarrow$ redundancy of in the model.

Example:

Two feature vectors in A: $\vec{a}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, $\vec{a}_2 = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$

The new new feature would be:

$$\vec{a}_{n+1} = \frac{1}{2}(\vec{a}_1 + \vec{a}_2) = \frac{1}{2}\begin{bmatrix} 1+4 \\ 2+5 \\ 3+6 \end{bmatrix}$$

$$= \begin{bmatrix} 2.5 \\ 3.5 \\ 4.5 \end{bmatrix}.$$

Adding this feature to Matrix A: will look like?

$$A = \begin{bmatrix} 1 & 4 & 2.5 \\ 2 & 5 & 3.5 \\ 3 & 6 & 4.5 \end{bmatrix}$$

So, we can't solve the least square problem

A has cols that are linearly dependent, as 3rd col. is avg.. So, A^TA is singular.

Assign_6_code_Hrishabh_Kulkarni

November 9, 2024

0.1 Question 2c

[3]:
```
import json
import numpy as np

with open("color_perception_data.json", "r") as file:
    color_data = json.load(file)

wavelength = np.array(color_data["wavelength"]["data"])
R = np.array(color_data["R_phosphor"]["data"])
G = np.array(color_data["G_phosphor"]["data"])
B = np.array(color_data["B_phosphor"]["data"])
L = np.array(color_data["L_coefficients"]["data"])
M = np.array(color_data["M_coefficients"]["data"])
S = np.array(color_data["S_coefficients"]["data"])
test_light = np.array(color_data["test_light"]["data"])

LR = np.dot(L, R)
MR = np.dot(M, R)
SR = np.dot(S, R)

MG = np.dot(M, G)
LG = np.dot(L, G)
SG = np.dot(S, G)

LB = np.dot(L, B)
MB = np.dot(M, B)
SB = np.dot(S, B)

mat = np.array([                                # Matrix of the above dot product
        [LR, LG, LB],
        [MR, MG, MB],
        [SR, SG, SB]
])

L_test_light = np.dot(L, test_light)
M_test_light = np.dot(M, test_light)
S_test_light = np.dot(S, test_light)
```

```
compared = np.array([L_test_light ,M_test_light, S_test_light])            #␣
 ↪Stored response to test light

weights = np.linalg.solve(mat, compared)
R_value = weights[0]
G_value = weights[1]
B_value = weights[2]

print("RGB weights are: \n")
print(f"R : {R_value}")
print(f"G : {G_value}")
print(f"B : {B_value}")
```

```
RGB weights are:

R : 0.42259299388355054
G : 0.0987425637870611
B : 0.5285525473174244
```

0.2 Question 3b

[6]:
```
import numpy as np

data = np.load("sensor_data.npy", allow_pickle=True).item()
A = data['A']
m, n = A.shape
y = data['y']
Intervals = 1

allones = np.ones((m, 1))
time = np.arange(m).reshape(-1, 1) * Intervals
B = np.hstack((A, allones, time))

# z = [x, alpha, beta]
z, residuals, rank, s = np.linalg.lstsq(B, y, rcond=None) # used -> least␣
 ↪square method wrt -> z

x = z[:n]
alpha = z[-2]
beta = z[-1]

print("Temperature Vector x:", x)
print("Bias Constant:", alpha)
print("Drift Constant:", beta)
```

```
Temperature Vector x: [24.90063639 21.61770814 33.8996902 ]
Bias Constant: 2.1937036086718855
```

```
Drift Constant: 0.18979612476283153
```