homework05

March 18, 2021

1 COMPSCI 527 Homework 5

1.0.1 Group Members: Shivam Kaul, Gin Wang, Beck Addison

1.0.2 Problem 0 (3 points)

1.1 Part 1: Linear Systems and the SVD

1.1.1 Problem 1.1 (Exam Style)

1. $A = \begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$

Clearly, b in not in the range of A.

Therefore, this is an **Incompatible** system. 2. $A = \begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

b is in the range of A, so it is compatible. Rank$(A) = 1$ and $n = 2$

Therefore, this is an **Underdetermined** system as it as infinitely many solutions. 3. $A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

b is in the range of A, so it is compatible. Rank$(A) = 2$ and $n = 2$

Therefore, this is an **Invertible** system as it is a square matrix with one solution. 4. $A = \begin{bmatrix} 2 & 1 \\ 0 & 4 \\ 6 & 3 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$

Clearly, b in not in the range of A.

Therefore, this is an **Incompatible** system. 5. $A = \begin{bmatrix} 2 & 1 \\ 0 & 4 \\ 6 & 3 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

b is in the range of A, so it is compatible. Rank$(A) = 2$ and $n = 2$

Therefore, this is a **Redundant** system as it has more equations than the variables required to find the solution. 6. $A = \begin{bmatrix} 2 & 1 \\ 4 & 2 \\ 6 & 3 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$

Clearly, b in not in the range of A.

Therefore, this is an **Incompatible** system.

1.1.2 Problem 1.2 (Exam Style)

The given matrix is:

$$A = \begin{bmatrix} 0 & 0 \\ 0 & a^2 \end{bmatrix}$$

Let P be the permutation matrix defined as follows:

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The results of pre-multiplying and post-multiplying A with P are shown below:

$$PA = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & a^2 \end{bmatrix} = \begin{bmatrix} 0 & a^2 \\ 0 & 0 \end{bmatrix}$$

Here, the rows are interchanged.

$$AP = \begin{bmatrix} 0 & 0 \\ 0 & a^2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ a^2 & 0 \end{bmatrix}$$

Here, the columns are interchanged.

Assume a diagonal matrix , where

$$= \begin{bmatrix} a^2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$PP = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ a^2 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & a^2 \end{bmatrix} = A$$

Therefore,

$$A = PP = UV^T$$

where

$$U = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} a^2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The next matrix, B is given as:

$$B = \begin{bmatrix} 0 & -a^2 \\ 2a^2 & 0 \\ 0 & 0 \end{bmatrix}$$

Let a diagonal matrix be defined:

$$= \begin{bmatrix} 2a^2 & 0 \\ 0 & -a^2 \\ 0 & 0 \end{bmatrix}$$

If permutation matrices

$$P_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

are multiplied with in the following manner, we get:

$$P_1 P_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2a^2 & 0 \\ 0 & -a^2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -a^2 \\ 2a^2 & 0 \\ 0 & 0 \end{bmatrix} = B$$

Therefore,

$$B = P_1 P_2 = UV^T$$

where

$$U = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2a^2 & 0 \\ 0 & -a^2 \\ 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

1.2 Part 2: Linear Regression

1.2.1 Problem 2.1 (Exam Style)

The general condition under which A has rank 1 would be when $x_i = x_j$ for all i, j in $[0, n-1]$ where $i \neq j$. Geometrically, this means that A is a two dimensional vector since only one row is linearly independent. Therefore, A describes only one unique point (x_0, y_0). A cannot have rank 0 because the second column of A must be all ones.

1.2.2 Problem 2.2 (Exam Style)

If A has rank 1, then for all i, j in $[0, n-1]$ where $i \neq j$, $y_i = y_j$. This indicates that all we need are x_0 and y_0 to construct the set U_0.

$$\hat{u}, \hat{v} = \text{argmin}_{u,v} L_s(u, v)$$

$$L_s(u, v) = \frac{n}{2}(ux_0 + v - y_0)^2$$

Since $L_s(u, v)$ must be positive due to squaring, it reaches 0 (the minimum) when $ux_0 + v - y_0 = 0$.

We can also see this when we set the gradient of $L_s(u, v)$ to 0:

$$\nabla L_s(u, v) = n(-y_0 + ux_0 + v)\begin{bmatrix} x_0 \\ 1 \end{bmatrix} = 0$$

Where we have $ux_0 + v - y_0 = 0$.

Geometrically speaking, the set U_0 then would be $\{(u, v)$ where $v = y_0 - ux_0\}$, describing points on the line $v = y_0 - x_0 u$.

1.2.3 Problem 2.3 (Exam Style)

$$\hat{u}_1, \hat{v}_1 = \text{argmin}_{u,v} u^2 + v^2$$

In the case where A has rank 1,

$$u^2 + v^2 = u^2 + (y_0 - ux_0)^2$$

We know $u^2 + v^2$ is a paraboloid that has one extrema the minimum, setting the gradient of this equation to 0 on u gives us $\hat{u}_1$

$$(2x_0^2 + 2)u - 2x_0 y_0 = 0$$

$$\hat{u}_1 = \frac{x_0 y_0}{x_0^2 + 1}$$

$$\hat{v}_1 = y_0 - \frac{x_0^2 y_0}{x_0^2 + 1} = \frac{y_0}{x_0^2 + 1}$$

1.2.4 Problem 2.4

[2]:
```
import urllib.request
import pickle
from os import path as osp
```

```python
pickle_file_name = 'sample_sets.pkl'
if not osp.exists(pickle_file_name):
    fmt = 'https://www2.cs.duke.edu/courses/spring21/compsci527/homework/5/{}'
    url = fmt.format(pickle_file_name)
    urllib.request.urlretrieve(url, pickle_file_name)
with open(pickle_file_name, 'rb') as file:
    sample_sets = pickle.load(file)
```

[122]:
```python
import numpy as np

def regress(x):
    A = np.hstack((x[:, 0].reshape(-1,1), np.ones((x.shape[0],1))))
    y = x[:,1].reshape(-1,1)
    return np.matmul(np.linalg.pinv(A),y)
```

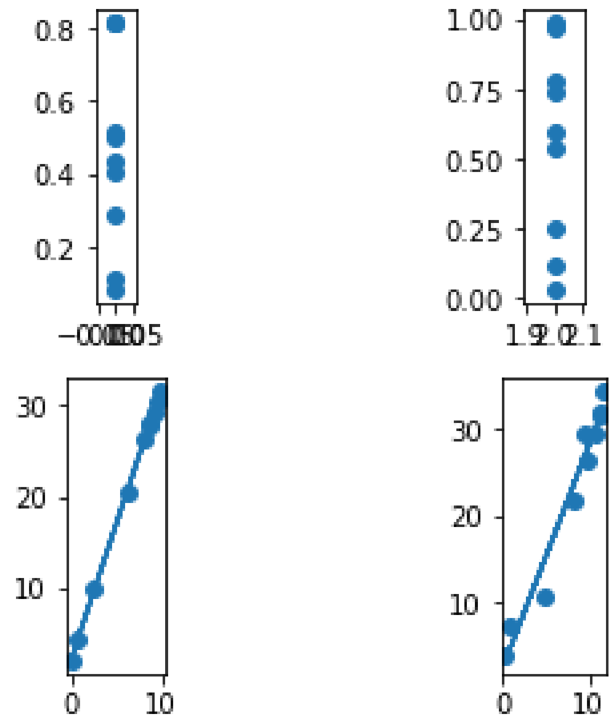
[123]:
```python
from matplotlib import pyplot as plt
%matplotlib inline

# extract point sets for x,y
point_sets = np.asarray(sample_sets)

for plot_ct, point_set in enumerate(point_sets):

    plt.subplot(2, 2, plot_ct + 1)  # generate empty plot
    x, y = point_set.T # extract, x, y values
    m, b = regress(point_set) # predict y from x
    y_hat = m*x + b           # generate regression plot
    x_min, x_max = np.min(x), np.max(x)
    y_min, y_max = np.min(y), np.max(y)
    plt.scatter(x, y)        # generate scatter for (x,y) coords
    plt.plot(x, y_hat)            # generate line for y_hat = mx + b regression
    plt.gca().set_aspect('equal')

plt.figure(figsize=(10,10))
plt.tight_layout()
plt.show()
```



1.3 Part 3: Principal Component Analysis

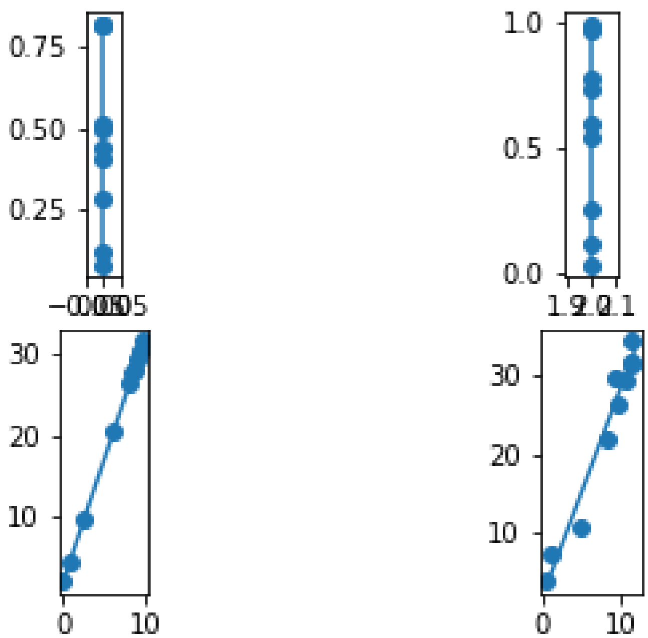
1.3.1 Problem 3.1

[117]:
```
def pca2(x):
    centroid_m = np.mean(x, axis = 0)                #mean/centroid = mean along␣
 ↪each column
    matrix_B = x - centroid_m                        #B = x - mean
    u, sigma, v = np.linalg.svd(matrix_B)            #get SVD of matrix_B to␣
 ↪solve for n
    coefficients_n = v[-1]                           #n = the last vector in v
    constant_c = centroid_m.T @ coefficients_n       #c = m.Tn
    return coefficients_n, constant_c
```

[124]:
```
for plot_ct, point_set in enumerate(point_sets):
     plt.subplot(2, 2, plot_ct + 1)                           # generate empty plot
     x, y = point_set.T
     n, c = pca2(point_set)                                   # run pca in 2 dims to␣
 ↪get n and c
     t = np.array([[0, -1],[1,0]]) @ n                        # getting t as [-n_y, n_x]
     project_coeffs = np.dot(point_set, t)/np.dot(t,t) #projecting the points␣
 ↪onto t
```

```
    start_a = np.min(project_coeffs)*t + c*n          # setting the start point␣
↪as the min proj coefficient in direction of t plus the constant with n's␣
↪components
    end_b = np.max(project_coeffs)*t + c*n            # setting the end point␣
↪as the max proj coefficient in direction of t plus the constant with n's␣
↪components

    new_x, new_y = np.array([start_a, end_b]).T       # refactoring for plt.
↪plot arguments
    plt.scatter(x, y)
    plt.plot(new_x, new_y)
    plt.gca().set_aspect('equal')                     # setting aspect to equal␣
↪- is this right?? These plots look awful
plt.figure(figsize=(10,10))
plt.tight_layout()
plt.show()
```




```
<Figure size 720x720 with 0 Axes>
```

The major difference between traditional LR and PCA is clearly the ability of PCA to plot the line of best fit (of least squares error) without being constrained by a relationship of a dependent and independent variable. As a result, we see vertical lines in the upper two subplots where before that was not possible as any function of $y = x$ must maintain uniqueness (that is, only one value of y corresponds to any value in x).