Homework Set 3, CPSC 8420, Spring 2022

Song, Zhiyuan

April 8, 2022

Problem 1

Data points were centered when they were dealt with PCA and LDA. Thep projection lines crossed the center of the points. If centered data points were scatter-plotted, the projection lines should have crossed the origin.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 21 15:12:32 2022

@author: Henry Song
"""

import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import svd
from numpy.linalg import inv
data_raw = np.array([[1,3],[2,5],[3,4],[4,3],[5,2],[5,1]])
data = data_raw - np.mean(data_raw,axis=0)
center = np.mean(data_raw,axis=0)
u,s,_ = svd(data.T.dot(data))
phi11 = u[0,0];phi21 = u[1,0]
line_x = np.linspace(np.min(data_raw[:,0]),np.max(data_raw[:,0]),10)#+center[0]
line_y = (line_x-center[0])*phi21/phi11+center[1]
plt.scatter(data_raw[:,0],data_raw[:,1])

cluster = {0:data_raw[:4],1:data_raw[4:]}
centroids = [np.mean(cluster[i],axis=0)[...,None] for i in range(2)]
Sb = (centroids[0]-centroids[1]).dot((centroids[0]-centroids[1]).T)
Sw = 0
for i in range(2):
    for j in range(len(cluster[i])):
        diff = cluster[i][j][...,None]-centroids[i]
        Sw = Sw + diff.dot(diff.T)
u2,s2,v2 = svd(inv(Sw).dot(Sb))
w = u2[:,0]
line_y1 = (line_x-center[0])*w[1]/w[0]+center[1]
plt.plot(line_x,line_y,label='PCA')
plt.plot(line_x,line_y1,label='LDA')
plt.xlabel('X coordinate')
```

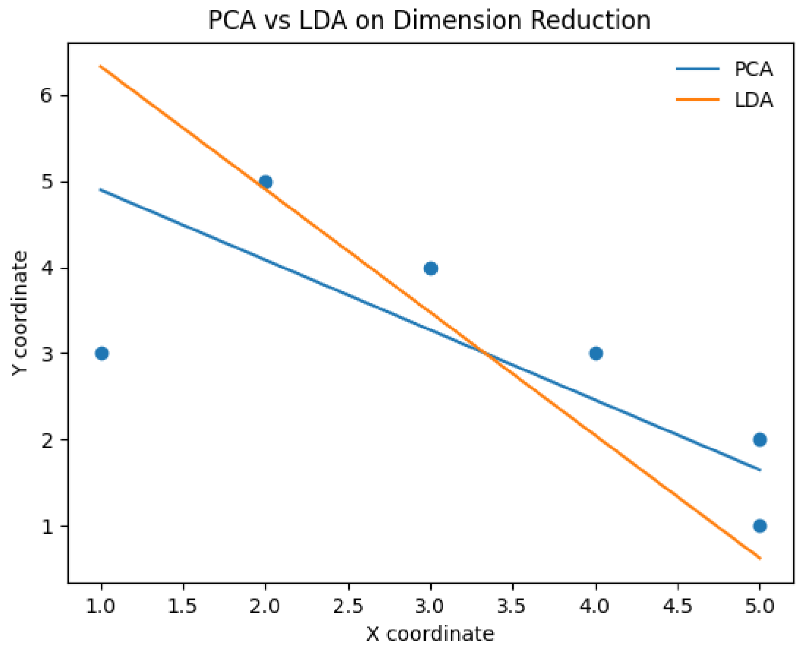

Figure 1: **Dimension Reduction using PCA and LDA**

```
plt.ylabel('Y coordinate')
plt.legend(frameon=False)
plt.title('PCA vs LDA on Dimension Reduction')
plt.savefig('prob1.png')
```

Problem 2

Given positive data-set $\{\{1,1\},\{2,2\},\{2,3\}\}$, as well as negative data-set $\{\{3,2\},\{3,3\},\{4,4\}\}$, please determine the decision boundary when leveraging k-NN where $k = 1$ and $k = 3$ respectively. Boundaries were determined using different k.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 26 18:05:06 2022

@author: Henry Song
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors
cmap_light = ListedColormap(['#FFAAAA', '#AAAAFF'])
data_set = np.array([[1,1],[2,2],[2,3],[3,2],[3,3],[4,4]])
y = np.array([0,0,0,1,1,1])
```

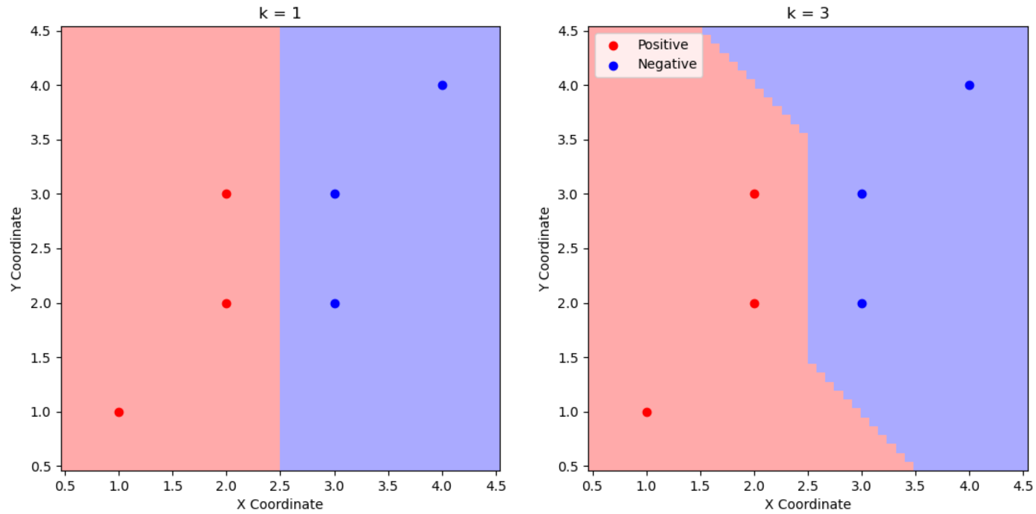

Figure 2: $K-NN$ **Bundary Determination with Different** k

```
num = 50
x_range = np.linspace(np.min(data_set[:,0])-0.5, np.max(data_set[:,0])+0.5,num)
y_range = np.linspace(np.min(data_set[:,1])-0.5, np.max(data_set[:,1])+0.5,num)
X,Y = np.meshgrid(x_range,y_range)
K = [1,3]
labels = ['Positive','Negative']
fig, axes = plt.subplots(figsize=(13,6),nrows=1,ncols=2)
for k,ax in zip(K,axes):
    knn = neighbors.KNeighborsClassifier(k)
    knn.fit(data_set,y)
    Z = knn.predict(np.c_[X.ravel(),Y.ravel()])
    Z = Z.reshape(X.shape)
    ax.pcolormesh(X,Y,Z,cmap=cmap_light)
    ax.scatter(data_set[:3,0],data_set[:3,1],c='#FF0000',label='Positive')
    ax.scatter(data_set[3:,0],data_set[3:,1],c='#0000FF',label='Negative')
    ax.set_xlabel('X Coordinate')
    ax.set_ylabel('Y Coordinate')
    ax.set_title('k = '+str(k))
axes[1].legend(loc='upper left')
plt.savefig('prob2.png')
```

Problem 3

Given SPD matrices X, Y, Z, now please follow the idea/method used in LDA/PCA to find the best solution to:

$$\begin{gathered}\underbrace{arg\ max}_{a,b}\ \ a^TZb \\ s.t.\ \ a^TXa = 1,\ b^TYb = 1\end{gathered} \tag{1}$$

Applying Lagrange multiplier to this optimization problem, Eq.1 is equivalent to

$$\min_{a,b,\lambda,\beta} \quad -a^T Zb + \frac{1}{2}\lambda\left(a^T Xa - 1\right) + \frac{1}{2}\beta\left(b^T Yb - 1\right) \tag{2}$$

Let $f = -a^T Zb + \lambda\left(a^T Xa - 1\right) + \beta\left(b^T Yb - 1\right)$, we can get condition-constrained equations by taking partial derivative and make differential eqatuions equal to zero, respected to a, b, λ, β, respectively.

$$\frac{\partial f}{\partial a} = -Zb + \lambda S_X a = -Zb + \lambda Xa = 0 \tag{3}$$

$$\frac{\partial f}{\partial b} = -Z^T a + \beta S_Y b = -Z^T a + \beta Yb = 0 \tag{4}$$

Where S_X and S_Y are the symetric part of the X and Y, respectively. In general, $S_A = \frac{1}{2}\left(A + A^T\right)$. In the case of a SPD matrix A,$S_A = A$. Besides, other two derivatives will lead to constrains, as shown below.

$$\frac{\partial f}{\partial \lambda} = a^T Xa - 1 = 0$$
$$\frac{\partial f}{\partial \beta} = b^T Yb - 1 = 0$$

If we manipulate Eq.3 with left multiply a^T, we will get $a^T Zb = \lambda a^T Xa = \lambda$. Similarly, we can also obtain $b^T Za = \beta$ from Eq.4. Therefore, λ and β are equivalent. In order to get $\max a^T Zb$, we need to maximize either λ or β (λ used in the later context). We can obtain the expression of b in terms of a using Eq.3, $b = \lambda Z^{-1} Xa$, and plug it into Eq.4. Consequently, we can get

$$Za = \lambda\beta Y Z^{-1} Xa$$

$$X^{-1} Z Y^{-1} Za = \lambda^2 a \tag{5}$$

Similarly, we can express a in terms of b from Eq.4 and plug it into Eq.4 to get the equation for b.

$$Y^{-1} Z X^{-1} Zb = \lambda^2 b \tag{6}$$

Eq.5 and 6 are nothing but eigenvalue problems for matrices $\mathbf{A} = X^{-1} Z Y^{-1} Z$ and $\mathbf{B} = Y^{-1} Z X^{-1} Z$, respectively.
Besides, we can notice that matrices $\mathbf{A}$ and $\mathbf{B}$ can be expressed as

$$\mathbf{A} = \mathbf{UM}$$
$$\mathbf{B} = \mathbf{MU}$$

where $\mathbf{U} = X^{-1} Z$ and $\mathbf{M} = Y^{-1} Z$. In the eigenvalue decomposition on A and B, we know that UM and MU have the same eigenvalues. Therefore, Eq.5 and Eq.6 can result in the same eigenvalues for λ^2 from $[W_A, V_A] = \mathrm{eig}(A)$ and $[W_B, V_B] = \mathrm{eig}(B)$. Meanwhilee, matrices X, Y and Z are all SPD, leading to the eigenvalues for either A or B are non-negative real numbers. We can simply sort the eigenvalues and select the eigenvectors corresponding to largest eigenvalue for a and b from W_A amd W_B, respectively. As a result, $a^T Zb$ can be maximized with the constrains $a^T Xa = 1$ and $b^T Yb = 1$.