Introduction to Machine Learning

Homework Solution - 6

Instructor: Prof Chinmay Hegde *TA:* Devansh Bisla, Maryam Majzoubi

Problem 1
Assume that you have 4 samples each with dimension 3, described in the data matrix X,

$$X = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 5 \\ 1 & 2 & 3 \\ 0 & 2 & 5 \end{bmatrix}$$

For the problems below, you may do the calculations in python (or R or Matlab). Explain your calculations in each step.

(a) Find the sample mean.

(b) Zero-center the samples, and find the eigenvalues and eigenvectors of the data covariance matrix Q.

(c) Find the PCA coefficients corresponding to each of the samples in X.

(d) Reconstruct the original samples from the top two principal components, and report the reconstruction error for each of the samples.

(Solution)

1.

$$\mu = \begin{bmatrix} 1.5 \\ 2.5 \\ 3.5 \end{bmatrix} \tag{0.1}$$

2.

$$X_n = X - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \mu^T = \begin{bmatrix} 1.5 & -0.5 & 2.5 \\ 0.5 & 1.5 & 1.5 \\ -0.5 & -0.5 & -0.5 \\ -1.5 & -0.5 & 1.5 \end{bmatrix}$$

$$Q = X_n^T X_n / 4$$

$$\lambda_1 = 3.5617; \quad v_1 = [-0.4506, 0.1925, 0.8717]^T$$

$$\lambda_2 = 1.1734; \quad v_2 = [0.6668, 0.7219, 0.1852]^T$$

$$\lambda_3 = 0.015; \quad v_3 = [-0.5936, 0.6647, -0.4536]^T$$

3.

$$C = X_n.[v_1, v_2, v_3] = \begin{bmatrix} -2.9515 & 0.1761 & -0.0888 \\ 1.3710 & 1.6941 & 0.0199 \\ -0.3068 & -0.7869 & 0.1913 \\ 1.8872 & -1.0832 & -0.1223 \end{bmatrix}$$

4.

$$R = X_n.[v_1, v_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 1.4473 & -0.4409 & -2.5403 \\ 0.5118 & 1.4868 & 1.5090 \\ -0.3865 & -0.6271 & -0.4133 \\ -1.5726 & -0.4187 & 1.4445 \end{bmatrix}$$

The absolute error is:

$$E = \begin{bmatrix} 0.1521 \\ 0.0340 \\ 0.3274 \\ 0.2094 \end{bmatrix}$$

Problem 2
In class, we analyzed the per-iteration complexity of k-means. Here, we will prove that the k-means algorithm will terminate in a finite number of iterations. Consider a data set $X = \{x_1, \dots, x_n\} \in \mathbb{R}^d$.

(a) Show that the k-means loss function can be re-written in the form:

$$F(\eta, \mu) = \sum_{i=1}^{n} \sum_{j=1}^{k} \eta_{ij} \|x_i - \mu_j\|^2$$

where $\eta = (\eta_{ij})$ is a suitable binary matrix (with 0/1 entries). Provide a precise interpretation of η.

(b) Show that each iteration of Lloyd's algorithm can only decrease the value of F.

(c) Conclude that the algorithm will terminate in no more than T iterations, where T is some finite number. Give an upper bound on T in terms of the number of points n.

(Solution)

(a) With the following choice of η the function F would be the same as k-means clustering loss function:

$$\eta_{ij} = \begin{cases} 1 & j = argmin(||x_i - \mu_j||^2) \\ 0 & else \end{cases} \tag{0.2}$$

(b) In the estimation step, we find a better η, so if a new cluster matches a sample better, we assign it to that cluster and this will decreases F.
In the maximization step, we find a better μ, and we can prove that updating the mean to the average of samples in each cluster reduces the loss function as well. Thus the algorithm can only decrease the value of F.

(c) There are maximum k^n possible configurations for the cluster centers. As mentioned above, the value of F can only decrease at each iteration, therefore we cannot revisit the same configuration of clusters. This result in the upper bound of k^n iterations for the algorithm.

Problem 3
Using the Senate Votes dataset demo'ed in Lecture 11, perform k-means clustering with $k = 2$ and show that you can learn (most of) the Senators' parties *in a completely unsupervised manner*. Which Senators did your algorithm make a mistake on, and why?

(Solution) We thank Dimitrios Chaikalis (dc4204) for some parts of the solution.

Hw6_Q3

April 30, 2020

[1]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
```

[15]:
```
# Loading data
url_votes = 'https://raw.githubusercontent.com/exemplary-citizen/
 ↪PCA-and-Senate-Voting-Data/master/senator_pca_problem/senator_data_pca/'

senator_df =  pd.read_csv(url_votes + 'data_matrix.csv',error_bad_lines=False)
af = pd.read_csv(url_votes + 'politician_labels.txt', header=None)
af["affiliations"] = af[0].str.split().str[-1]

X = np.array(senator_df.values[:, 3:].T, dtype='float64')
X_mean = np.mean(X, axis = 0)
X_original = X.copy()
X = X - np.mean(X, axis = 0)

labels = af["affiliations"]
labels = labels.replace("Red",1)
labels = labels.replace("Blue",0)
```

[3]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2).fit(X)
```

[21]:
```
# compute error
err = np.mean(labels != kmeans.labels_)
print(f"{err*100}%")
```

```
4.0%
```

[30]:
```
print('The misclassified senators were:')
af[labels != kmeans.labels_]
```

```
The misclassified senators were:
```

```
[30]:                      0 affiliations
      21       Nelson Blue         Blue
      34   Jeffords Yellow       Yellow
      58        Chafee Red          Red
      62     Dayton Yellow       Yellow
```

We can ignore the candidates with yellow affiliations Hence out error percentage is 2%. Other than that senators Nelson (democrat) and Chafee (republican), voted with the other party enough times, to have the k-Means algorithm include them in the cluster for the opposite side.

```
[ ]:
```

Problem 4
The *Places Rated Almanac*, written by Boyer and Savageau, rates the livability of several US cities according to nine factors: climate, housing, healthcare, crime, transportation, education, arts, recreation, and economic welfare. The ratings are available in tabular form, available as a supplemental text file. Except for housing and crime, higher ratings indicate better quality of life. Let us use PCA to interpret this data better.

(a) Load the data and construct a table with 9 columns containing the numerical ratings. (Ignore the last 5 columns – they consist auxiliary information such as longitude/latitude, state, etc.)

(b) Replace each value in the matrix by its base-10 logarithm. (This pre-processing is done for convenience since the numerical range of the ratings is large.) You should now have a data matrix X whose rows are 9-dimensional vectors representing the different cities.

(c) Perform PCA on the data. Remember to center the data points first by computing the mean data vector μ and subtracting it from every point. With the centered data matrix, do an SVD and compute the principal components.

(d) Write down the first two principal components v_1 and v_2. Provide a qualitative interpretation of the components. Which among the nine factors do they appear to correlate the most with?

(e) Project the data points onto the first two principal components. (That is, compute the highest 2 scores of each of the data points.) Plot the scores as a 2D scatter plot. Which cities correspond to outliers in this scatter plot?

(f) Repeat Steps 2-5, but with a slightly different data matrix – instead of computing the base-10 logarithm, use the z-scores. (The z-score is calculated by computing the mean μ and standard deviation σ for each feature, and normalizing each entry x by $\frac{x-\mu}{\sigma}$). How do your answers change?

(Solution) We thank Dimitrios Chaikalis (dc4204) for some parts of the solution.

HW6_Q4

April 30, 2020

Ques 4 The *Places Rated Almanac*, written by Boyer and Savageau, rates the livability of several US cities according to nine factors: climate, housing, healthcare, crime, transportation, education, arts, recreation, and economic welfare. The ratings are available in tabular form, available as a supplemental text file. Except for housing and crime, higher ratings indicate better quality of life. Let us use PCA to interpret this data better.

(a) Load the data and construct a table with 9 columns containing the numerical ratings. (Ignore the last 5 columns – they consist auxiliary information such as longitude/latitude, state, etc.)

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd

     data = np.loadtxt('places.txt',skiprows=1,usecols=(1,2,3,4,5,6,7,8,9))
     names = np.loadtxt('places.txt',dtype=str,skiprows=1,usecols=0)
```

```
[2]: data, names
```

```
[2]: (array([[ 521., 6200.,  237., …,  996., 1405., 7633.],
             [ 575., 8138., 1656., …, 5564., 2632., 4350.],
             [ 468., 7339.,  618., …,  237.,  859., 5250.],
             …,
             [ 540., 8371.,  713., …, 1022.,  842., 4946.],
             [ 570., 7021., 1097., …, 2797., 1327., 3894.],
             [ 608., 7875.,  212., …,  122.,  918., 4694.]]),
      array(['Abilene,TX', 'Akron,OH', 'Albany,GA',
             'Albany-Schenectady-Troy,NY', 'Albuquerque,NM', 'Alexandria,LA',
             'Allentown,Bethlehem,PA-NJ', 'Alton,Granite-City,IL', 'Altoona,PA',
             'Amarillo,TX', 'Anaheim-Santa-Ana,CA', 'Anchorage,AK',
             'Anderson,IN', 'Anderson,SC', 'Ann-Arbor,MI', 'Anniston,AL',
             'Appleton-Oshkosh-Neenah,WI', 'Asheville,NC', 'Athens,GA',
             'Atlanta,GA', 'Atlantic-City,NJ', 'Augusta,GA-SC',
             'Aurora-Elgin,IL', 'Austin,TX', 'Bakersfield,CA', 'Baltimore,MD',
             'Bangor,ME', 'Baton-Rouge,LA', 'Battle-Creek,MI',
             'Beaumont-Port-Arthur,TX', 'Beaver-County,PA', 'Bellingham,WA',
             'Benton-Harbor,MI', 'Bergen-Passaic,NJ', 'Billings,MT',
             'Biloxi-Gulfport,MS', 'Binghampton,NY', 'Birmingham,AL',
             'Bismarck,ND', 'Bloomington,IN', 'Bloomington-Normal,IL',
        'West-Palm-Beach-Boca-Raton-Delray-Beach,FL', 'Wheeling,WV-OH',
        'Wichita,KS', 'Wichita-Falls,TX', 'Williamsport,PA',
        'Wilmington,DE-NJ-MD', 'Wilmington,NC', 'Worcester,MA',
        'Yakima,WA', 'York,PA', 'Youngstown-Warren,OH', 'Yuba-City,CA'],
       dtype='<U42'))
```

(b) Replace each value in the matrix by its base-10 logarithm. (This pre-processing is done for convenience since the numerical range of the ratings is large.) You should now have a data matrix X whose rows are 9-dimensional vectors representing the different cities.

[3]:
```python
X = np.log10(data)
```

(c) Perform PCA on the data. Remember to center the data points first by computing the mean data vector μ and subtracting it from every point. With the centered data matrix, do an SVD and compute the principal components.

[4]:
```python
# centering the data
X = X - np.mean(X, axis = 0)

# Computing covariance matrix
n = len(X)
Q = (1/n) * X.T @ X

# computing eigen vector and eigen values
[l,v] = np.linalg.eig(Q)
```

(d) Write down the first two principal components v_1 and v_2. Provide a qualitative interpretation of the components. Which among the nine factors do they appear to correlate the most with?

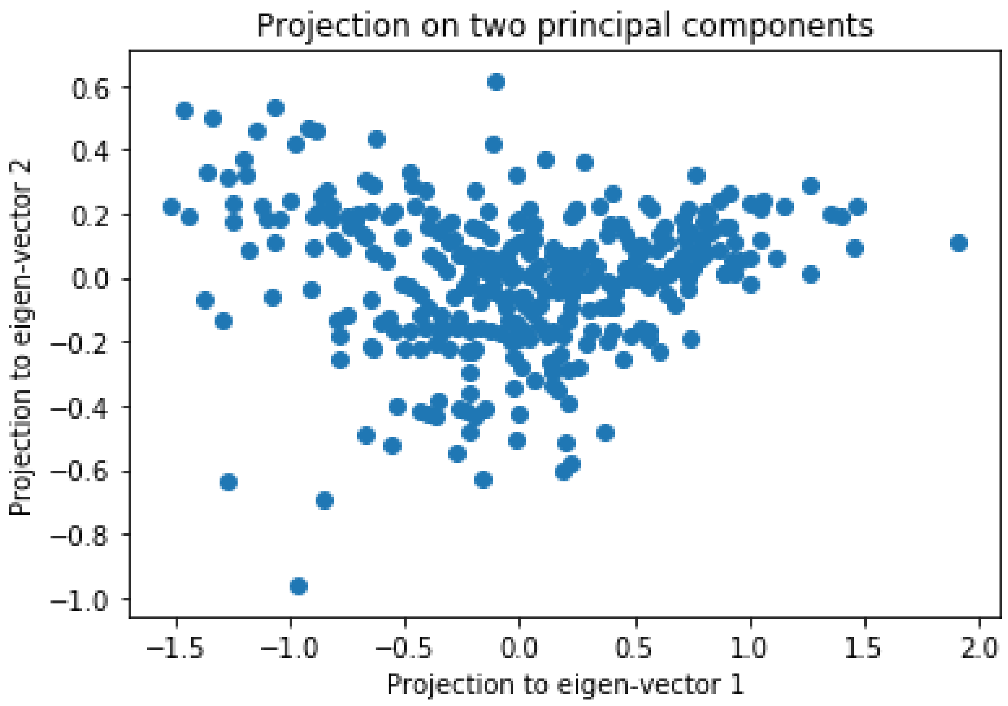
[9]:
```python
# computing first two eigen vectors
v1 = v[:,0]
v2 = v[:,1]
np.concatenate([v1.reshape(-1,1),v2.reshape(-1,1)],1)
```

[9]:
```
array([[ 0.03507288, -0.0088782 ],
       [ 0.09335159, -0.00923057],
       [ 0.40776448,  0.85853187],
       [ 0.10044536, -0.22042372],
       [ 0.15009714, -0.05920111],
       [ 0.03215319,  0.06058858],
       [ 0.87434057, -0.30380632],
       [ 0.15899622, -0.33399255],
       [ 0.01949418, -0.0561011 ]])
```

We look at the absolute value of the eigen vectors. It is observed that the first principal component weighs the 7th quality (i.e. the Arts) while the second eigen-vector weighs the 3rd quality (Healthcare).

(e) Project the data points onto the first two principal components. (That is, compute the highest 2 scores of each of the data points.) Plot the scores as a 2D scatter plot. Which cities correspond to outliers in this scatter plot?

[11]:
```
proj_v1 = np.matmul(X,v1)
proj_v2 = np.matmul(X,v2)
plt.scatter(proj_v1, proj_v2)
plt.xlabel('Projection to eigen-vector 1')
plt.ylabel('Projection to eigen-vector 2')
plt.title('Projection on two principal components')
plt.show()
```



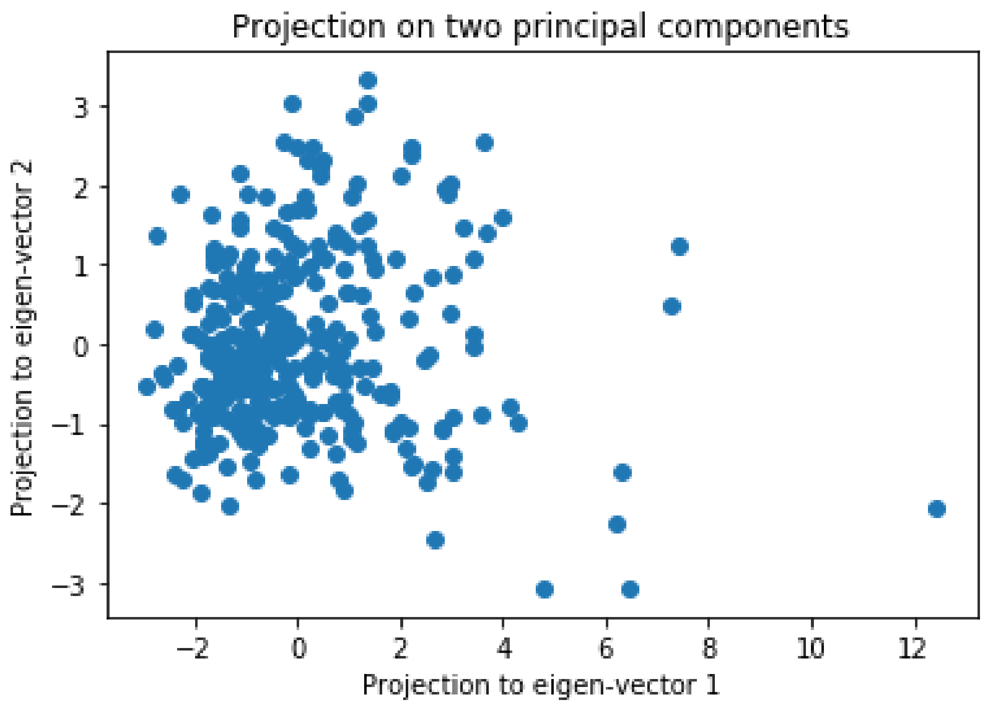
In the above graph, the most clear outliers are, the city on the right end and the city on the bottom end.

[13]:
```
# computing the name of the cities
i = np.argmax(proj_v1)
j = np.argmin(proj_v2)
names[[i,j]]
```

[13]:
```
array(['New-York,NY', 'Glens-Falls,NY'], dtype='<U42')
```

(f) Repeat Steps 2-5, but with a slightly different data matrix – instead of computing the base-10 logarithm, use the z-scores. (The z-score is calculated by computing the mean μ and standard deviation σ for each feature, and normalizing each entry x by $\frac{x-\mu}{\sigma}$). How do your answers change?

[14]:
```
Xn = data
m = np.mean(Xn,axis=0)
s = np.std(Xn,axis=0)
Xn = (Xn - m)/s
Q = np.matmul(np.transpose(Xn),Xn)/n
[l,v] = np.linalg.eig(Q)
v1 = v[:,0]
v2 = v[:,1]
proj_v1 = np.matmul(Xn,v1)
proj_v2 = np.matmul(Xn,v2)
plt.scatter(proj_v1, proj_v2)
plt.xlabel('Projection to eigen-vector 1')
plt.ylabel('Projection to eigen-vector 2')
plt.title('Projection on two principal components')
plt.show()
```



[19]:
```
ind = np.argsort(proj_v1)[-5:]
print(names[ind])

ind = np.argsort(proj_v2)[0:3]
print(names[ind])
```

```
['Boston,MA' 'Chicago,IL' 'Los-Angeles,Long-Beach,CA' 'San-Francisco,CA'
 'New-York,NY']
```