Homework 1 for CSI 431 Solutions

September 30, 2020

1 [20 pts.] Data analysis

In a statistical survey of 2000 families on ownership of cars 30 families responded that they do not own a car, 470 families owned 1, 850 families owned 2, 490 families owned 3, 150 owned 4, and there were 10 families which owned 10 cars each.

(a)[5 pts] Plot the sample-based Probability Mass Function (PMF) and Cumulative Distribution Function (CDF) of the random variable corresponding to **number of cars per family**

Solution:

The total number of cars is

$N = 2000$

$P(X = 0) = 30/N = .015 \quad P(X \leq 1) = 0.15$

$P(X = 1) = 470/N = 0.235 \quad P(X \leq 1) = 0.25$

$P(X = 2) = 850/N = 0.425 \quad P(X \leq 2) = 0.675$

$P(X = 3) = 490/N = 0.245 \quad P(X \leq 3) = 0.92$

$P(X = 4) = 150/N = 0.075 \quad P(X \leq 4) = .995$

$P(X = 10) = 10/N = 0.05 \quad P(X \leq 10) = 1$

Note the PMF is defined only at those discrete values. The CDF, however, is defined for all numbers. The CDF of a discrete distribution is a piecewise constant function open on the right side of each interval (since if $X = x, X \leq x$).

(b)[5 pts] Calculate the expected **number of cars per family** and its variance. Show the steps of your calculation.

Solution:

$E[X] = \sum_x xP(x) = 0 \times .015 + 1 \times .235 + 2 \times 0.425 + 3 \times 0.245 + 4 \times 0.075 + 10 \times .005 = 2.17$

$E[X^2] = \sum_x x^2P(x) = 0 \times .015 + 1 \times .235 + 4 \times 0.425 + 9 \times 0.245 + 16 \times 0.075 + 100 \times .005 = 5.84$

$Var(X) = E[X^2] - E[X]^2 = 5.81 - 2.17^2 = 1.1311$

(d)[5 pts]Calculate the expected value of number of cars per family if you exclude the families with 10 cars. Is the mean (as compared to part b) stable or sensitive to the removal of these data points?

What other statistical measures are there to estimate the average behavior? Are they less or more stable with regards to the outliers? Justify your answer by computing those alternative measures for the original sample.

Solution: $E[X] = \sum_x xP(x) = 0 \times .0151 + 1 \times .236 + 2 \times 0.427 + 3 \times 0.246 + 4 \times 0.076 + = 2.1307$ *This is a roughly 2% decrease from the removal of only .005% of the data points thus the mean value is sensitive to outliers. Mode and median are more stable median and mode = 2*

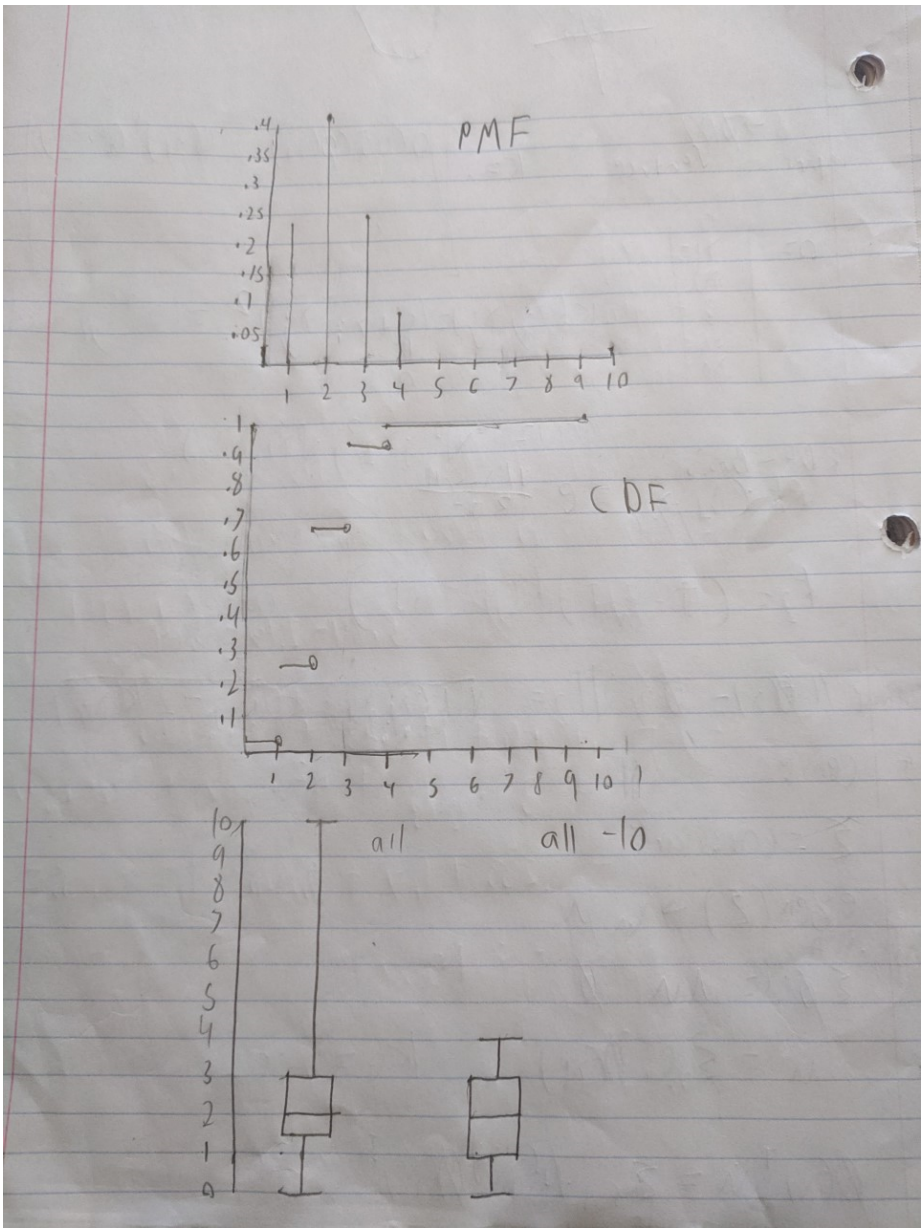
Solution:

(e)[5 pts] *Create a box plots for the* ***number of cars per family*** *for the variables* X_{all} *(cars per family including all observations) and* X_{-10} *(cars per family after excluding the 10-car families). (don't use python libraries for this - just draw it by hand in your submission). Look here:* `http://www.physics.csbsju.edu/stats/box2.html` *for examples of box plots. You should have two boxes one for* X_{all} *and one for* X_{-10} *with their specific statistics: mean, median, ranges and inter-quantile ranges (see slides on numeric attributes for details). .*

$Q1_A = 1.5, Q3_A = 3$*, and the top whisker will be at 10*

$Q1_C = 1, Q3_C = 3$*, and the top whisker is at 4*

Plots for 1a, 1e:



2 [10 pts.] Irreducible data example

In class we discussed that not all datasets' dimensionality can be successfully reduced using PCA.

(a)[2 pts] Discuss the cases when PCA will fail.

Solution:
PCA fails when variability within the dataset cannot be effectively reduced to fewer dimensions. In particular, this reduction must be done to linear, orthogonal dimensions. Datasets where intuitively sensible reduced dimensions are nonlinear (say, points around a circle where we can reduce to a radius or angle) or non-orthogonal (data along two 45 degree offset lines) are poor candidates for PCA. An additional consideration is the goal of dimensionality reduction. PCA aims only to preserve total variance. If we are attempting to separate two clusters of points, but the separating plane is along the principal component(s), dimensionality reduction via PCA will eliminate separability for these two clusters.

(b)[3 pts] How do we quantify that it fails?

Solution:
As above, PCA aims to preserve total variance. A failed PCA is one where the total variance after dimensionality reduction is too small a fraction of the original total variance. Depending on the application, "too small" can be less than 80%, 90%, 99%, etc.

(c)[5 pts] Provide a minimal example of a dataset (specify the points as vectors of numbers) in which PCA will not work well for dimensionality reduction. Explain why. *Hint: Think of 2D points and reduction to 1D.*

Solution:
Consider the points $[1,1],[1,-1],[-1,1],[-1,-1$ *- the corners of a square in 2D. There is no line onto which we can project that will yield more than half the total variance.*

3 [50 pts] Dimensionality reduction

For this question you will use the Cloud Dataset from the UCI ML repository: `https://archive.ics.uci.edu/ml/datasets/Cloud`. Read about it to get familiar with what is measured. Within the data, there are two datasets: DB #1 and DB #2. For this homework, just use the 1024 vectors in *DB #1*. Use python for all your programming. You will have to submit your code in Code.zip together with the relevant write-up in the main solution file Solution.pdf.

(a)[5 pts] Load the data into a python program and center it. Note: there should be a function called *center()* in your code that achieves this.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('cloud data DB1.tsv')

def center(data):
    centered_data = data - np.outer(np.ones(data.shape[0],),np.mean(data,axis=0))
    return centered_data
```

(b)[5 pts] Compute the covariance matrix of the data Σ. *Hint: by using the definitions of sample covariance, as a matrix product or as a sum of outer products. See book for details.* Use Numpy for linear algebra computations (https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.linalg.html).As a result you should have a function *covar()* in your code which does not use the built-in covariance functions.

Code:

```
def covar(data):
    n = data.shape[0]
    centered_data = center(data)
    return (centered_data.T.dot(centered_data)) / n
```

(c)[5 pts] Compute the eigenvectors and eigenvalues of Σ. The numpy linear algebra module referenced above has a function that can help.

Code:

```
cov = cov3(centered_data,centered_data)
eigval,eigvec = np.linalg.eig(cov)

#np.linalg.eig() sorts and returns the values, If there is need for Sorting the eigenvalues
sorted_eigval = -np.sort(-eigval)
sorted_eigvec = [eigvec[i] for i in np.argsort(eigval)]
```

(d)[10 pts] Determine the number of principal components (PCs) r that will ensure 90% retained variance? How did you compute this? Provide a function in your code that determines r based on an arbitrary percentage α of retained variance.

Solution & Code:
The first dimension itself ensures 98% *covarience*

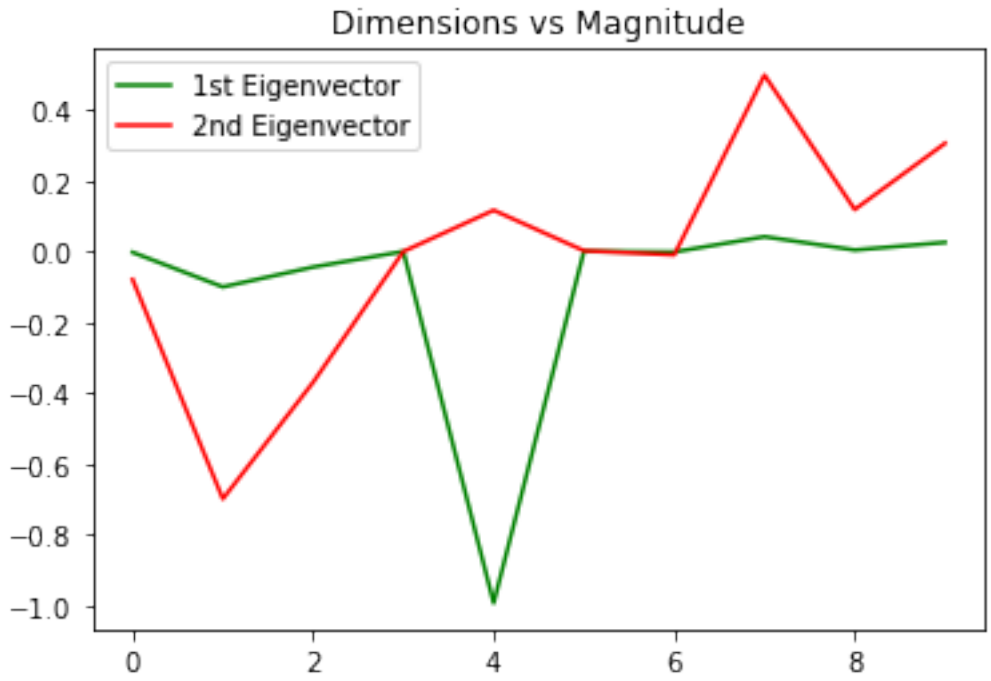
```
for i in range(1,11):
    print np.sum(eigval[:i])/np.sum(eigval)
```

(e)[10 pts] Plot the first two components in a figure with horizontal axis (x) corresponding to the dimensions and vertical axis (y) corresponding to the magnitude of the component in this dimension. There will be 2 traces with d points

in this figure. Include the figure in your Solution.pdf. Also save the top two components in a text file "Components.txt", with each component on a separated line and represented as d comma separated numbers (i.e. the file should have two lines with d numbers separated by commas). Include "Components.txt" in your Code.zip.

Solution&Code:

```
plt.plot(range(10),eigvec[:,0],c='g')
plt.plot(range(10),eigvec[:,1],c='r')
plt.title("Dimensions vs Magnitude")
plt.legend(['1st Eigenvector', '2nd Eigenvector'])
```



(f)[10 pts] Compute the reduced dimension data matrix A with two dimensions by projection on the first two PCs. Plot the points using a scatter plot (two dimensional diagram that places each sample i according to its new dimensions a_{i1}, a_{i2}). Discuss the observations. Are there clusters of close-by points? What is the retained variance for $r = 2$? Argue for or against whether these are sufficient dimensions.

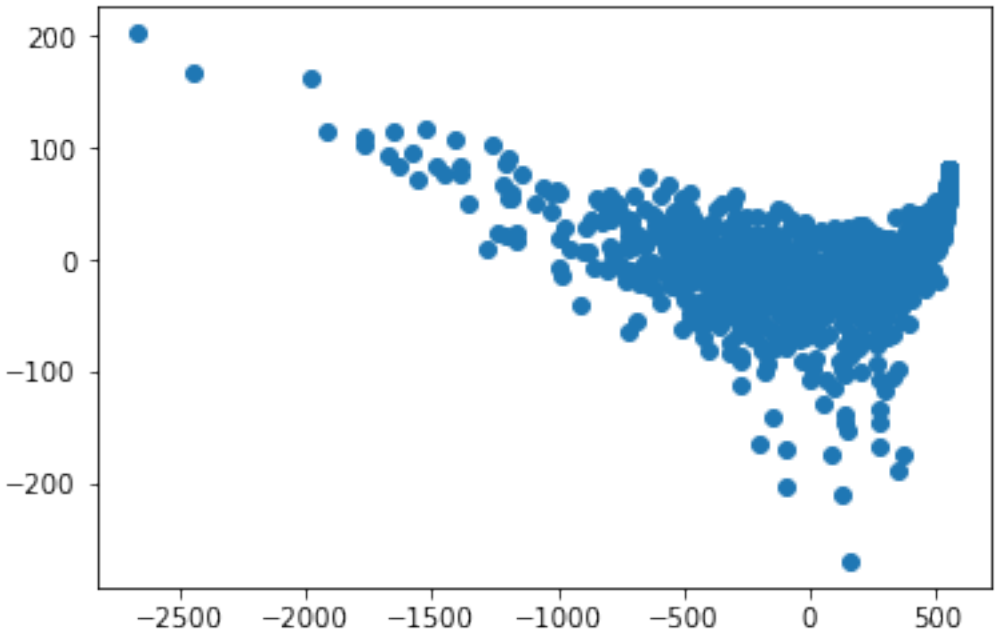
Code:
Transforming the data using the first two eigenvectors

```
transformed = centered_data.dot(eigvec[:,:2])
```

```
final = transformed.reshape(-1,2)
```

Plotting the data with new dimensions

```
x = final[:,0]
y = final[:,1]
plt.scatter(x, y)
```



4 [20 pts.] Kernel methods

Consider the problem of finding *the most dissimilar diametric pair (MDDP)*: this is a pair of data points that are dissimilar from the mean and also dissimilar from each other. Below is an algorithm that would find such a pair given a data matrix D:

Algorithm 1: MDDP(D)

Result: a, b - the most dissimilar diametric points in D
Compute the data mean $\mu = mean(D)$;
$s = +\infty$;
for i *in* $(1 \dots n)$ **do**
  for j *in* $(i+1 \dots n)$ **do**
    $temp = x_i^T \mu + x_j^T \mu + x_i^T x_j$;
    if $temp < s$ **then**
      $s = temp$;
      $a = x_i$;
      $b = x_j$;
    end
  end
end

The algorithm computes the sum of inner products $x_i^T \mu + x_j^T \mu + x_i^T x_j$ for each pair of points and returns the pair with the lowest such quantity.
(a)[5 pts] Demonstrate the execution of this algorithm on the following data matrix of 2D instances: D=$\begin{pmatrix} 0 & 1 \\ 1 & 3 \\ 5 & 0 \\ 2 & 4 \end{pmatrix}$. Show the steps and the resulting MDD pair of points.

Solution:

i	j	$temp$	s	a	b
1	*2*	*13*	*13*	*(0 1)*	*(1 3)*
1	*3*	*12*	*12*	*(0 1)*	*(5 0)*
1	*4*	*18*			
2	*3*	*23*			
2	*4*	*34*			
3	*4*	*32*			

Final output: s = 12, a = (0 1), b = (5 0)

(b)[15 pts] As we discussed in class sometimes we would like to kernelize methods to handle non-linearity in data. Provide a pseudo-code for a kernel version of the MDDP algorithm above. The goal is to kernelize the algorithm for an arbitrary kernel **Hint: Assume that you can compute the kernel matrix K, corresponding to some mapping $\phi()$ and then use the basic kernel operations we discussed in class and also in the book, to derive the steps of MDDP in terms of elements in K.**

Solution:
MDDP target function: $t = x_i^T \mu + x_j^T \mu + x_i^T x_j$

In the feature space, the above function can be rewritten as:

$$t_\phi = \phi(x_i)^T \mu_\phi + \phi(x_j)^T \mu_\phi + \phi(x_i)^T \phi(x_j)$$

By definition,

$$\mu_\phi = \frac{1}{n} \sum_{n=1}^{n} \phi(x_i) = \frac{1}{n}(\phi(x_1) + \phi(x_2) + \cdots + \phi(x_i))$$

The first term in t_ϕ can be rewritten as:

$$\begin{aligned}
\phi(x_i)^T \mu_\phi &= \phi(x_i)^T * \frac{1}{n}(\phi(x_1) + \phi(x_2) + \cdots + \phi(x_i)) \\
&= \frac{1}{n}(\phi(x_i)^T \phi(x_1) + \phi(x_i)^T \phi(x_2) + \cdots + \phi(x_i)^T \phi(x_n)) \\
&= \frac{1}{n}(K(x_i, x_1) + K(x_i, x_2) + \cdots + K(x_i, x_n))
\end{aligned}$$

*The above expression means ($\frac{1}{n}$ * the sum of row i in Kernel K)*
Similarly, the second term in t_ϕ is:
*$\phi(x_i)^T \mu_\phi$ = ($\frac{1}{n}$ * the sum of row j in Kernel K)*
The third term in t_ϕ is just $\phi(x_i)^T \phi(x_j) = K(x_i, x_j)$ by Kernel definition
So, the full expression of t_ϕ would be:

$$t_\phi = \frac{1}{n} sum(K(i,:)) + \frac{1}{n} sum(K(j,:)) + K(i,j)$$

Now, we can rewrite the MDDP algorithm using the above expression.

Algorithm 2: MDDP(D)

```
Result: a, b - the most dissimilar diametric points in D
Compute kernel matrix K;
for r in (1...n) do
    rowSum(r) = (1/n) sum(K(r,:));
end
s = +∞;
for i in (1...n) do
    for j in (i+1...n) do
        temp = rowSum(i) + rowSum(j) + K(i,j);
        if temp < s then
            s = temp;
            a = x_i;
            b = x_j;
        end
    end
end
```