Data Science for Mechanical Systems Assignment 3

SACHIN FRANCIS DSOUZA – sfd2121

Problem 1.

1. Loss Function to Minimize Average sum of the squared errors (MSE) is:

$$\mathbf{L(\beta_0, \beta_1) =} \quad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$\mathbf{L(\beta_0, \beta_1) =} \quad \frac{1}{n}\sum_{i=1}^{n} \mathbf{(y_i - \beta_0 - \beta_1 x_i)^2}$$

Where $\mathbf{y_i}$ is the actual value and $\mathbf{y^\wedge_i}$ is the predicted value

2. The gradient function for the above loss function is obtained by partially differentiating with $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$ respectively.

$$\partial L/\partial \beta_0 = (-2/n)\sum_{i=1}^{n}(\mathbf{yi - \beta 0 - \beta 1xi})$$

$$\partial L/\partial \beta_1 = (-2/n)\sum_{i=1}^{n}(\mathrm{xi})(\mathbf{yi - \beta 0 - \beta 1xi})$$

3. The Value of Loss Function: **30068.440558989987**

4. The Value of Parameters after 10 iterations with learning rate of 0.001 is

$\boldsymbol{\beta_0}$ **= 0.6855626678299438**

$\boldsymbol{\beta_1}$ **= 14.233847598602836**

Problem 2.

1. The Value of X [128,128] is **173**.

2. The Scaled Value of X [128,128] is **1.2794041765058435**

3. If we perform PCA on the given matrix X_{scaled}, the value of the first element of the first principal component is **0.05842033740661623** (own code).

   If we perform PCA (using SVD) on the given matrix X_{scaled}, the value of the first element of the first principal component is **-0.05842033740661623** (Using own SVD function, and extracting the first term U [0,0], from Professors' slides).

4. If we only use the first 50 principal components of X_{scaled} to reconstruct this matrix, the reconstruction error is **0.008731721488750455** (own code).

   If we only use the first 50 principal components of X_{scaled} (using SVD) to reconstruct this matrix, the reconstruction error is **0.046444313764157766** (Professors slides).

Problem 3.

Suppose that we obtain a bootstrap sample from a set of observations.

1. What is the probability that the first bootstrap observation is not the j^{th} observation from the original sample? Justify your answer.
   - The number of observations to sample from is n.
   - There are 'n' observations in the original sample. Since bootstrap sampling draws items with replacement, we are sampling from the same pool with the same probability every time. There are (n-1) items in the (n) that are not in the j^{th} observation. So, there is a fraction [**(n-1) / (n)]** chance that the first item is not j.
   - Since the probability is equal for any of the n observations to be selected, the probability that the j^{th} observation is selected as the first bootstrap observation is **(1/n)**
   - Thus, the probability that the first bootstrap observation is not the j^{th} observation from the original sample is $\mathbf{(1 - 1/n)}$

2. What is the probability that the j^{th} observation is not in the bootstrap sample?
   - With the bootstrap sample we do 'n' draws. This means there are 'n' chances to draw something other than 'j' that all must succeed for 'j' not to be in the bootstrap. This is a simple product of 'n' of these probabilities, which can be written as $\mathbf{(1 - 1/n)^{n}}$
   - We are sampling from a set of 'n' observations. We sample n times, randomly and with replacement, and want to know the probability that the observation indexed by j is not in the final bootstrap sample, which we denote by $S= \{S_1, S_2,\dots,S_n\}$.
   - We know from the previous case that $P(S_1 \neq j) = P(S_2 \neq j) = (1-1/n)$. Using the same logic, we can say: $\mathbf{P\ (S_i \neq j) = (1-1/n)}$ $(\forall\ I \in \{1, 2,\dots,n\})$.

- Because the samples are independent, we can multiply these probabilities together (using the multiplication rule for independent events), and say that the probability that the j^{th} observation is in none of these independent bootstrap samples S_1, S_2, Sn is:

$$= \mathbf{P\ (S_1 \neq j) \cap P\ (S_2 \neq j) \cap \cdots \cap P\ (S_n \neq j)}$$
$$= \prod\nolimits^{n}{}_{i=1}[\mathbf{P\ (S_i \neq j)}]$$
$$= \prod\nolimits^{n}{}_{i=1}\ \mathbf{(1-1/n)}$$
$$= \mathbf{(1-1/n)^{n}}$$

3. When n=10000, what is the value of the probability mentioned in section 2?
   - The probability that the j^{th} observation is not in the bootstrap sample when n = 10000.
   - $\prod^{10000}{}_{i=1}[\mathbf{P\ (S_i \neq j)}] = \mathbf{(1-1/n)^{n}}$
   - $\prod^{10000}{}_{i=1}[\mathbf{P\ (S_i \neq j)}] = \mathbf{(1-1/10000)^{10000}} = \underline{\mathbf{0.36786}}$

Appendix:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Nov  8 18:50:04 2021

@author: SACHIN F DSOUZA
"""


import pandas as pd
from urllib.request import urlopen
from sklearn.decomposition import PCA
from tqdm.notebook import tqdm
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

data = np.genfromtxt("aw3.txt",delimiter=',')

x_ar = data[:,0]
```

```
y_ar = data[:,1]

#Values being Initialized
B0 = 1;
B1 = 1;
n = len(data);

#1.3 : for B0 = 1 and B1 = 1
y_pr = B0 + B1*x_ar
errors = y_ar-y_pr
sqrd_errors = errors**2
sum_errors = sum(sqrd_errors)

loss_value = sum_errors/n

print("1.3  Loss Value for B0 = 1 and B1 = 1 is : ",loss_value)

#1.4
trials = 10
rate_of_learning = 0.001

for iteration in tqdm(range(trials)):
    y_pr = B0 + B1*x_ar
    change_B0 = -2*sum(y_ar-y_pr)/n
    change_B1 = -2*sum(x_ar*(y_ar-y_pr))/n
    B0 = B0 - rate_of_learning*change_B0
    B1 = B1 - rate_of_learning*change_B1

print("1.4  After {} iterations, B0 = {} AND B1 = {}".format(trials,B0,B1))

image_data =
Image.open(urlopen("https://raw.githubusercontent.com/changyaochen/MECE4520/master/lectures/le
cture_1/leena.png"))
X = np.array(image_data)

plt.figure()
plt.imshow(image_data, cmap="gray")
plt.show()

#2.1
```

```
print("2.1  Element at index (128,128) of matrix X is",str(X[128][128]))

#2.2
X_scal = []
for i in tqdm(range(X.shape[1])):
    column = X[:,i]
    new_column = (column-column.mean())/column.std()
    X_scal.append(new_column)

X_scal = np.asarray(X_scal).transpose()

print("2.2  The Scaled Value of X [128,128]",str(X_scal[128][128]))

#2.3

# performing PCA
pca = PCA()
pca_X = pca.fit_transform(X_scal)
print("2.3.a      The value of the first element of the first principal component",pca.components_[0][0])
# perform SVD
n = len(X_scal)
U, S, Vh = np.linalg.svd(X_scal.T @ X_scal / n)
print("2.3.b The value of the first element of the first principal component", U[0,0])

#2.4

#own code
cov_matrix = X - np.mean(X, axis = 1)
eig_val, eig_vec = np.linalg.eigh(np.cov(cov_matrix))
p = np.size(eig_vec, axis = 1)
idx = np.argsort(eig_val)
idx = idx[::-1]
eig_vec = eig_vec[:,idx]
eig_val = eig_val[idx]
pri_comp = 50
if pri_comp <p or pri_comp >0:
    eig_vec = eig_vec[:, range(pri_comp)]
    score = np.dot(eig_vec.T, cov_matrix)
    recon = np.dot(eig_vec,score) + np.mean(X, axis = 1).T
```

```
    recon_matrix = np.uint8(np.absolute(recon))


k_sum = 0

n_sum = 0

for i in range (len(recon_matrix)):

    k_sum += recon_matrix[i][i]

    n_sum += X[i][i]


print("2.4.a  If we only use the first 50 principal components of Xscaled to reconstruct this matrix, the reconstruction error is",str(1-k_sum/n_sum))


#from professors slides, calculate reconstruction error

U_reduced = U[:,:50]

Z = X_scal @ U_reduced

X_approx = Z @ U_reduced.T

reconstruction_error = (

        np.sum(np.square(np.linalg.norm((X_scal - X_approx), ord=2, axis=1)))

        / np.sum(np.square(np.linalg.norm(X_scal, ord=2, axis=1)))

    )

print("2.4.b  If we only use the first 50 principal components of Xscaled to reconstruct this matrix, the reconstruction error is",reconstruction_error)
```

```
In [39]: runcell(0, 'D:/Columbia University/Data Science for Mechanical Systems/aw3.py')
1.3  Loss Value for B0 = 1 and B1 = 1 is :  30068.440558989987
  0%|          | 0/10 [00:00<?, ?it/s]
1.4  After 10 iterations, B0 = 0.6855626678299438 AND B1 = 14.233847598602836
2.1  Element at index (128,128) of matrix X is 173
  0%|          | 0/512 [00:00<?, ?it/s]
2.2  The Scaled Value of X [128,128] 1.2794041765058435
2.3.a  The value of the first element of the first principal component 0.05842033740661623
2.3.b The value of the first element of the first principal component -0.058420337406615785
2.4.a  If we only use the first 50 principal components of Xscaled to reconstruct this matrix,
the reconstruction error is 0.008731721488750455
2.4.b  If we only use the first 50 principal components of Xscaled to reconstruct this matrix,
the reconstruction error is 0.046444313764157766
```