ECE 50024

Homework 2

Parth Sagar Hasabnis [phasabni@purdue.edu](mailto:phasabni@purdue.edu)

## Exercise 1:

male\_train = pd.read\_csv('male\_train\_data.csv')

male\_train['male\_bmi'] = male\_train['male\_bmi']/10

male\_train['male\_stature\_mm'] = male\_train['male\_stature\_mm']/1000

male\_train['y'] = 1

male\_train.head(10)

A picture containing funnel chart

Description automatically generatedA picture containing text, black, electronics

Description automatically generated

Female Data

Male data

female\_train = pd.read\_csv('female\_train\_data.csv')

female\_train['female\_bmi'] = female\_train['female\_bmi']/10

female\_train['female\_stature\_mm'] = female\_train['female\_stature\_mm']/1000

female\_train['y'] = -1

female\_train.head(10)

## Exercise 2:

a)

Text, letter

Description automatically generated

Text, letter

Description automatically generated

### b) Analytical Solution in Python

theta = np.matmul(X\_2.T,X\_2)

theta = np.linalg.inv(theta)

temp = np.matmul(X\_2.T, y)

theta = np.matmul(theta, temp)

print(theta)

>> [-10.7017505 -0.12339677 6.67486843]

### c) Using CVXPY

theta = cp.Variable(3)

cost = cp.sum\_squares(X\_2@theta-y)

prob = cp.Problem(cp.Minimize(cost))

prob.solve()

theta.value

print(theta.value)

>> [-10.7017505 -0.12339677 6.67486843]

### d) Gradient Descent Derivation ­­­Text, letter Description automatically generatedA piece of paper with writing Description automatically generated with medium confidence e) Using Gradient Descent

2

2

d = 2

N = X\_2.shape[0]

itr = 50000

theta = np.zeros(d+1)

cost  = np.zeros(itr)

XtX = np.dot( np.transpose(X\_2), X\_2)

# Gradient descent

for itr in range(itr):

  dJ     = np.dot(np.transpose(X\_2), np.dot(X\_2, theta)-y)

  dd     = dJ

  alpha  = np.dot(dJ, dd) / np.dot(np.dot(XtX, dd), dd)

  theta  = theta - alpha\*dd

  cost[itr] = np.linalg.norm(np.dot(X\_2, theta)-y)\*\*2/N

print(theta)

>> [-10.7017505 -0.12339677 6.67486843]

### f) Icon Description automatically generated

### g) GD using momentum

d = 2

N = X\_2.shape[0]

itr = 50000

theta = np.zeros(d+1)

cost  = np.zeros(itr)

dJ\_old = np.zeros(d+1)

XtX = np.dot( np.transpose(X\_2), X\_2)

beta  = 0.9

for itr in range(itr):

  dJ     = np.dot(np.transpose(X\_2), np.dot(X\_2, theta)-y)

  dd     = beta\*dJ\_old + (1-beta)\*dJ

  alpha  = np.dot(dJ, dd) / np.dot(np.dot(XtX, dd), dd)

  theta  = theta - alpha\*dd

  dJ\_old = dJ

  cost[itr] = np.linalg.norm(np.dot(X\_2, theta)-y)\*\*2/N

print(theta)

>> [-10.7017505 -0.12339677 6.67486843]

### h)

A picture containing shape

Description automatically generated

## Exercise 3:

### a)

Chart, scatter chart

Description automatically generated

### b) Evaluation metrics

|  |  |
| --- | --- |
| Type 1 error | 14.17 % |
| Type 2 error | 14.17 % |
| Precision | 0.85 |
| Recall | 0.82 |

## Exercise 4:

Shape

Description automatically generated

Chart, line chart

Description automatically generated

A picture containing shape

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generatedA piece of paper with writing

Description automatically generated with medium confidenceText, letter

Description automatically generatedA picture containing text, document, receipt

Description automatically generatedText, letter

Description automatically generated

## Exercise 5:

Conditional Generative Adverserial Networks (cGANs) addresses a shortcoming of traditional GANS. Traditional GANs generate data by randomly sampling from a latent space and then using a generator network to transform that random input into a new data point. However, the generated data does not follow any given condition, and hence the output can be unpredictable. cGANs, on the other hand, enable the generation of data that is conditioned on a specific input or attribute.

Machine Learning and Deep learning are the premier technologies in today’s world, and both of them run on one single currency - data. Hence, the generation of data has become an important problem, as this data can be used to train autonomous vehicles, smart speakers and virtual assistants, wireless communications channels, and solve may more problems. Using cGANs, the generation of all kinds of data has become possible. We can generate conditioned data that was previously difficult or expensive to collect in the real world. For example – the performance and handling of an autonomous vehicle on a slippery road is an experiment that is difficult and dangerous to conduct in real life, but it is essential for autonomous vehicles to be trained on such conditions. Now we can generate artificial testing data using cGANs, and train vehicles using this artificial data.

There exist multiple implementations of cGANs on the internet, and different people have tried to implement it in their own method. I shall start playing around with a PyTorch implementation trained on the MNIST dataset [1].

While I have created discriminator-like classifier networks, I am not familiar with the architecture of Generator networks. Hence my next step would be to learn about them and develop a traditional GAN, before I move towards creating a cGAN.

[1] <https://github.com/qbxlvnf11/conditional-GAN>

# APPENDIX