Homework 5 Solutions

ESE 402/542

Due November 25, 2020 at 11:59pm

(For problems 1 and 2, no other package except numpy and matplotlib should be used for the programming questions. For problem 3, you can use the packages of your choice.)

Problem 1.

(a) In this problem we will analyze logistic regression learned in class.
Sigmoid function can be written as $S(x) = \frac{1}{1+e^{-x}}$

For a given variable X assume $P(Y = +1|X)$ is modeled as $P(Y = 1|X) = S(\beta_0+\beta_1 X)$. Plot a 3d figure showing the relation between output and variable β_0 and β_1 when X = 1. Take values between [-2,2] for both β_0 and β_1 with a step size of 0.1 to plot the 3d plot.

(b) In class, we have done binary classification with labels Y=$\{0, 1\}$. In this problem, we will be using the labels as Y=$\{-1, 1\}$ as it will be easier to derive the likelihood of the $P(Y|X)$.

- Show that if Y takes values $\{-1, 1\}$, the probability of Y given X can be written as,(Not programming)
$P(Y|X) = \frac{1}{1+e^{-y(\beta_0+\beta_1 \mathbf{x})}}$

- We have learned that the coefficients β_0 and β_1 can be found using MLE estimates. Show that the Log Likelihood function for m data points can be written as(Not Programming)

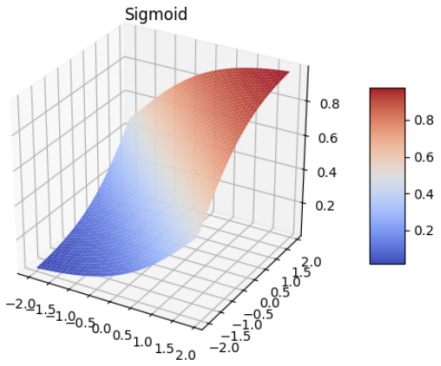
$$\ln L\left(\beta_0, \beta_1\right) = -\sum_{i=1}^{m} \ln\left(1 + e^{-y_i(\beta_0+\beta_1 \mathbf{x_i})}\right)$$

- Plot a 3d figure showing the relation between log likelihood function and variable β_0 , β_1 when X = 1, Y = -1 and X = 1, Y = 1. Take values between [-2, 2] for both β_0 and β_1 with a step size of 0.1 to plot the 3d plot

- Based on the graph, is it possible to maximize this function?

Solution

(a)



(b)

- Let $\eta(x) = \mathbf{P}(Y = +1|X = x)$ and $1 - \eta(x) = \mathbf{P}(Y = -1|X = x)$

$$\begin{aligned}
\eta(\mathbf{x}) &= \frac{1}{1 + e^{-(\beta_0 + \beta_1 \mathbf{x})}} \\
1 - \eta(\mathbf{x}) &= 1 - \frac{1}{1 + e^{-(\beta_0 + \beta_1 \mathbf{x})}} \\
&= \frac{e^{-(\beta_0 + \beta_1 \mathbf{x})}}{1 + e^{-(\beta_0 + \beta_1 \mathbf{x})}} \\
&= \frac{1}{(e^{(\beta_0 + \beta_1 \mathbf{x})}) \cdot (1 + e^{-(\beta_0 + \beta_1 \mathbf{x})})} \\
\mathbf{P}(Y = -1|X) &= \frac{1}{1 + e^{(\beta_0 + \beta_1 \mathbf{x})}}
\end{aligned}$$

Since η and $1-\eta$ differ by ± 1 sign in the power of the exponent in the denominator. We can write the combined probability as

$$p(Y|\mathbf{X}) = \frac{1}{1 + e^{-y(\beta_0 + \beta_1 \mathbf{x})}}$$

Since y $\in -1, 1$

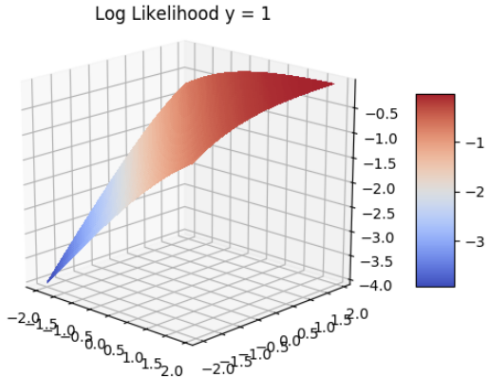
- Likelihood can be written as

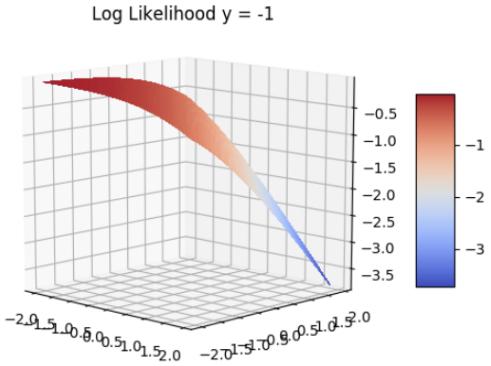
$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= p\left(y_1, \ldots, y_m | \mathbf{x}_1, \ldots, \mathbf{x}_m; \beta_{\mathbf{0}}, \beta_{\mathbf{1}}\right) \\ &= \prod_{i=1}^{m} p\left(y_i | \mathbf{x}_i; \beta_{\mathbf{0}}, \beta_{\mathbf{1}}\right) \\ &= \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(\beta_0 + \beta_1 \mathbf{x})}}\end{aligned}$$

  Log Likelihood can be written as

$$\ln \mathcal{L}(\mathbf{w}) = -\sum_{i=1}^{m} \ln\left(1 + e^{-y_i \mathbf{w}^\top \mathbf{x}_i}\right)$$

•





- Looking at the graphs we can see that the function can be maximized

Problem 2.

1. While we can formalize the Likelihood estimate there is no close form expression for the coefficients β_0, β_1 maximising the above log likelihood. Hence, we will use an iterative algorithm to solve for the coefficients.
   We can see that

$$\max(-\sum_{i=1}^{m} \ln\left(1 + e^{-y_i(\beta_0+\beta_1 x_\mathbf{i})}\right)) = \min(\sum_{i=1}^{m} \ln\left(1 + e^{-y_i(\beta_0+\beta_1 x_\mathbf{i})}\right))$$

   We will describe our function loss as $L = \frac{1}{m}\sum_{i=1}^{m} \ln\left(1 + e^{-y_i(\beta_0+\beta_1 x_\mathbf{i})}\right)$. Our objective is to iterative decrease this loss as we keep computing the optimal coefficients. Here $x_i \in R$

   In this problem we will be working with real image data where the goal is to classify if the image is 0 or 1 using logistic regression.
   The input $X \in R^{m*d}$ where a single data point $x_i \in R^d$, d = 784. The matrix labels $Y \in R^m$, where each label $y_i \in \{0, 1\}$

   - Load the data into the memory and visualize one input as an image for each of label 0 and label 1. (The data should be reshaped back to [28 x 28] to be able to visualize it.)
   - The data is inbetween 0 to 255. Normalise the data to [0,1]
   - Set y_i = 1 for images labeled 0 and y_i = -1 for images labeled 1. Split the data randomly into train and test with a ratio of 80:20.
     Why is random splitting better than sequential splitting in our case?
   - Initialize the coefficients using a univariate "normal" (Gaussian) distribution of mean 0 and variance 1. (Remember that coefficients are a vector of = $[\beta_0, \beta_1 ... \beta_d]$, where d is the dimension of the input)
   - Compute the loss using the above mentioned Loss L.
     (The loss can be written as $L = \frac{1}{m}\sum_{i=1}^{m} \ln\left(1 + e^{-y_i\left(\beta_0+\sum_{j=0}^{d-1}\beta_{(j+1)}\cdot \mathbf{x}_{i,j}\right)}\right)$, where (i, j) represent the data point i for i $\in$ {1 ..m} and jth dimension of the data point x_i for j $\in$ {0 ...d-1})
   - To minimize the loss function a widely known algorithm is going in the direction opposite to the gradients of the loss function.
     (It's helpful to write the coefficients $[\beta_1 ... \beta_d]$ as a vector β and β_0 as a scalar.
     Now β is of size [d] $\in \mathrm{R}^d$ and β_0 is of size [1] $\in$ R)
     We can write the gradients of loss function as

$$\frac{\partial L}{\partial \beta_0} = -\frac{1}{m}\sum_{i=1}^{m} \frac{e^{-y_i\cdot(\beta_0+\beta\cdot x_i^T)}}{1 + e^{-(y_i\cdot(\beta_0+\beta\cdot x_i^T))}} y_i = d\beta_0$$

$$\frac{\partial L}{\partial \beta} = -\frac{1}{m}\sum_{i=1}^{m} \frac{e^{-y_i\cdot(\beta_0+\beta\cdot x_i^T)}}{1 + e^{-(y_i\cdot(\beta_0+\beta\cdot x_i^T))}} y_i x_i = d\beta$$

Write a function to compute the gradients

- Update the parameters as

$$\beta = \beta - 0.05 * d\beta$$
$$\beta_0 = \beta_0 - 0.05 * d\beta_0$$

(Gradient updates should be computed based on the train set)

- Repeat the process for 50 iterations and report the loss after the 50th epoch.
- plot the loss for each iteration for the train and test sets
- Logistic regression is a classification problem. We classify as $+1$ if $P(Y = 1|X) \geq 0.5$. Derive the classification rule for the threshold 0.5.(Not a programming question)
- for the classification rule derived compute the accuracy on the test set for each iteration and plot the accuracy.

The final code should be along this format

```
import numpy as np
from matplotlib import pyplot as plt

def compute_loss(data, labels, B, B_0):

    return logloss

def compute_gradients(data, labels, B, B_0):

    return dB, dB_0

if __name__ == '__main__':
    x = np.load(data)
    y = np.load(label)

    ## Split the data to train and test
    x_train, y_train, x_test, y_test = #split_data

    B = np.random.randn(1, x.shape[1])
    B_0 = np.random.randn(1)

    lr = 0.05

    for _ in range(50):

        ## Compute Loss
        loss = compute_loss(x_train, y_train, B, B_0)
```

```
        ## Compute Gradients
        dB, dB_0 = compute_gradients(x_train, y_train, B, B_0)

        ## Update Parameters
        B = B - lr*dB
        B_0 = B_0 - lr*dB_0

        ##Compute Accuracy and Loss on Test set (x_test, y_test)
        accuracy_test =
        loss_test =

    ##Plot Loss and Accuracy
```

Make sure to vectorize the code. Ideally 50 iterations should run in 15 seconds or less. If possible avoid using for loops, except for the 50 iterations of gradient updates given in the sample code.

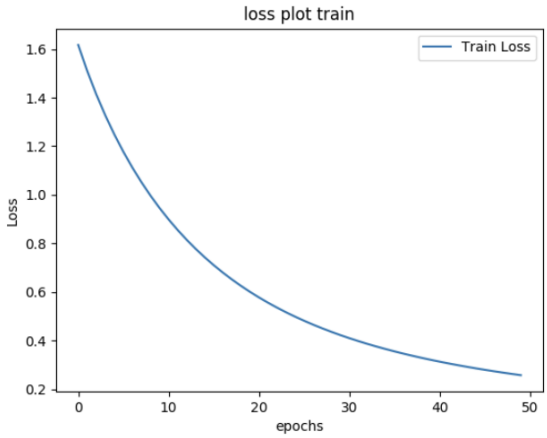
Solution

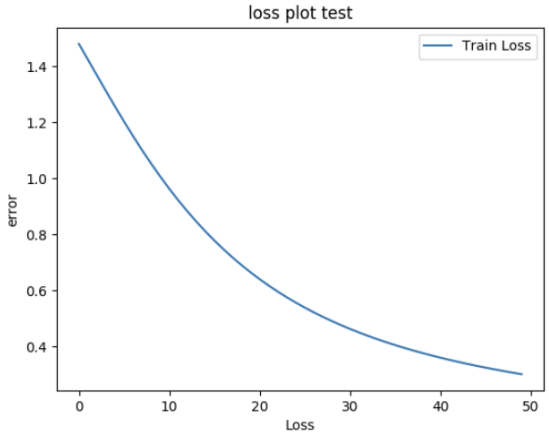
(a) The data is arranged in the order of 1's and 0's. So if we split sequentially, the testset will contain only images and labels belonging to 1's.

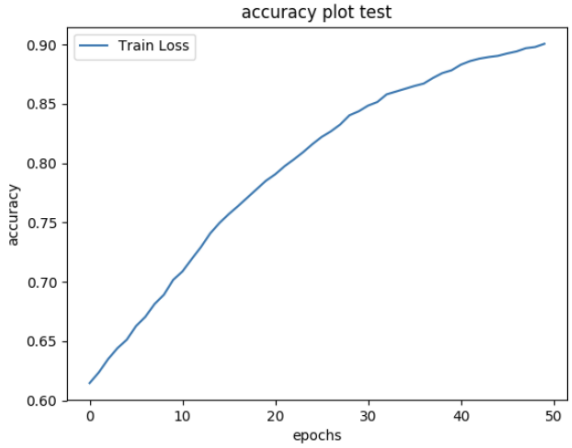
(b) Classification rule can be written as:

$$\begin{aligned} P(Y = +1|X) = \frac{1}{(1 + e^{-(\beta_0 + \beta_1 x_\mathbf{i})})} &>= 0.5 \\ = (\beta_0 + \beta_1 x_\mathbf{i}) &>= 0 \\ So, & \\ P(Y|X) = sign\,(\beta_0 + \beta_1 x_\mathbf{i}) & \end{aligned}$$

(c) The graphs are:


loss plot train
Train Loss
Loss
epochs
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
10
20
30
40
50


loss plot test
Train Loss
error
Loss
1.4
1.2
1.0
0.8
0.6
0.4
0
10
20
30
40
50


accuracy plot test
Train Loss
accuracy
epochs
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
10
20
30
40
50

Problem 3.

Recall that in classification we assume that each data point is an i.i.d. sample from a(n unknown) distribution $P(X = x, Y = y)$. In this question, we are going to design the data distribution P and evaluate the performance of logistic regression on data generated using P. Keep in mind that we would like to make P as simple as we could. In the following, we assume $x \in R$ and $y \in \{0, 1\}$, i.e. the data is one-dimensional and the label is binary. Write $P(X = x, Y = y) = P(X = x)P(Y = y|X = x)$. We will generate $X = x$ according to the uniform distribution on the interval [0,1] (thus $P(X = x)$ is just the pdf of the uniform distribution).

1. Design $P(Y = y|X = x)$ such that (i) $P(y = 0) = P(y = 1) = 0.5$; and (ii) the classification accuracy of any classifier is at most 0.9; and (iii) the accuracy of the Bayes optimal possible classifier is at least 0.8.

2. Using Python, generate $n = 100$ training data points according to the distribution you designed above and train a binary classifier using logistic regression on training data.

3. Generate $n = 100$ test data points according to the distribution you designed in part 1 and compute the prediction accuracy (on the test data) of the classifier that you designed in part 2. Also, compute the accuracy of the Bayes optimal classifier on the test data. Why do you think Bayes optimal classifier is performing better?

4. Redo parts 2,3 with $n = 1000$. Are the results any different than part 3? Why?

Solution

1. Let $h(X)$ be the predicted label and $\eta(X) = P(Y = 1|X)$. We define $\eta(X)$ as

$$\eta(X) = \mathbf{P}(Y = +1|X = \mathbf{x}) = \begin{cases} p & \text{if } x \geq 0.5 \\ q & \text{otherwise} \end{cases}$$

We can design such that $p > 0.5$ and $q < 0.5$. The error for the classification is:

$$\begin{aligned} \text{er}_D[h] &= \mathbf{P}_{(X,Y)\sim D}(h(X) \neq Y) \\ &= \mathbf{E}_{(X,Y)\sim D}[\mathbf{1}(h(X) \neq Y)] \\ &= \mathbf{E}_X\left[\mathbf{E}_{Y|X}[\mathbf{1}(h(X) \neq Y)]\right] \\ &= \mathbf{E}_X[\mathbf{P}(Y = +1|X) \cdot \mathbf{1}(h(X) \neq +1) + \mathbf{P}(Y = 0|X) \cdot \mathbf{1}(h(X) \neq 0)] \\ &= \mathbf{E}_X[\eta(X) \cdot \mathbf{1}(h(X) = 0) + (1 - \eta(X)) \cdot \mathbf{1}(h(X) = +1)] \end{aligned}$$

We can write above as the PDF of the X is given as $f(X) = 1$. The CDF for X will be $P(X \leq x) = x$. For Bayes classifier error, the minimum achievable error, which is given as:

$$\begin{aligned} \text{er}^*_D[h] = \mathbf{E}_X[\min(\eta(X), (1 - \eta(X)))] &\geq 0.1 \\ \Rightarrow 0.5 * q + 0.5(1 - p) &\geq 0.1 \\ p - q &\leq 0.8 \end{aligned}$$

Another equation according the other constraint of the maximum error achievable, is given as

$$\begin{aligned} \mathbf{E}_X[\min(\eta(X), (1 - \eta(X)))] &\leq 0.2 \\ \Rightarrow p - q &\geq 0.6 \end{aligned}$$

We can any value of p from 0.8 to 0.9. Let's take the values as $p = 0.85$ and $q = 0.15$, we can have the following model

$$\mathbf{P}(Y = +1|X = \mathbf{x}) = \begin{cases} 0.85 & \text{if } x \geq 0.5 \\ 0.15 & \text{otherwise} \end{cases}$$

From the above design $p(Y = +1) = 0.5 * 0.85 + 0.5 * 015 = 0.5$.

For part(b) and (c), the accuracy of logistic regression should be less than that of Bayes optimal classifier as the Bayes classifier accuracy is calculated using the true distribution.

For part (d), the accuracy of logistic regression with 1000 data points will be slightly better than 100 data points.

Problem 4.

K-means clustering can be viewed as an optimization problem that attempts to minimize some objective function. For the given objectives, determine the update rule for the centroid, c_k of the k-th cluster C_k . In other word, find the optimal c_k that minimizes the objective function. The data x contains p features.

1. Show that setting the objective to the sum of the squared Euclidean distances of points from the center of their clusters,

$$\sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} (c_{ki} - x_i)^2$$

   results in an update rule where the optimal centroid is the mean of the points in the cluster.

2. Show that setting the objective to the sum of the Manhattan distances of points from the center of their clusters,

$$\sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} |c_{ki} - x_i|$$

   results in an update rule where the optimal centroid is the median of the points in the cluster.

Solution

1. Without the loss of generality, for the k^{th} cluster for the i^{th} point, the gradient for the centroid value c_{ki} are:

$$\begin{aligned}
\frac{\partial}{\partial c_{ki}} \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} (c_{ki} - x_i)^2 &= \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} \frac{\partial}{\partial c_{ki}} (c_{ki} - x_i)^2 \\
&= \sum_{x_i \in C_{ki}} 2 (c_{ki} - x_i) \\
&= 0 \\
\Rightarrow |C_{ki}| \, c_{ki} &= \sum_{x_i \in C_{ki}} x_i \\
\Rightarrow c_{ki} &= \frac{1}{|C_{ki}|} \sum_{x_i \in C_{ki}} x_i
\end{aligned}$$

   Thus, c_k is the mean of the k^{th} cluster.

2. Without loss of generality, the first order condition for the k^{th} centroid at predictor i is

$$\begin{aligned}\frac{\partial}{\partial c_{ki}} \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} |c_{ki} - x_i| &= \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{i=1}^{p} \frac{\partial}{\partial c_{ki}} |c_{ki} - x_i| \\ &= \sum_{x_i \in C_{ki}} \operatorname{sgn}(x_i - c_{ki}) \\ \Rightarrow |\{x_i | x_i \leq c_{ki}\}| &= |\{x_i | x_i \geq c_{ki}\}|\end{aligned}$$

Thus c_k is the median of the k^{th} cluster.