Logistic Regression

February 2022

1 Machine Learning Lab#7

1.1 Aim: Implement Logistic Regression Algorithm on the given dataset

2 Introduction

Logistic Regression is a supervised learning classification algorithm. It is used for predicting the categorical dependent variable using a given set of independent variables. In this lab, we will perform the implementation of the Logistic Regression algorithm in Python using Pytorch and Tensorflow library. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

2.1 Binary Logistic Regression

The main goal of a Binary Logistic Regression algorithm is to train a classifier and make binary decision for the new observation that is passed into the model. Given some observation $x_1, x_2, ..., x_n$ and parameters $\theta_0, \theta_1, ..., \theta_n$, the logistic regression model outputs a probability. It uses sigmoid fuction to perform classification. After learning the values of parameters $\theta_0, \theta_1, ..., \theta_n$, the classifer first performs the weighted sum of input features $x_1, x_2, ..., x_n$ and parameters $\theta_0, \theta_1, ..., \theta_n$.

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n$$

Later , this value is passed into a sigmoid function which squashes the output value between 0 and 1 and provides us a probability $P(1|x)$ for the given input features x. The sigmoid has a number of advantages; it takes a real-valued number and maps it into the range [0,1], which is just what we want for a probability. The sigmoid has the following equation,

$$\sigma(z) = \frac{1}{1 + e^{-}z}$$

For a given x, we say yes if the probability $P(y = 1|x)$ is more than 0.5, and no otherwise. We call 0.5 the decision boundary.

2.1.1 Loss Function

We need a loss function that expresses, for an observation x, how close the classifier output $\hat{y} = \sigma(\theta^T X)$ is to the correct output (y, which is 0 or 1). The loss function used in Logistic regression is Binary cross entropy loss.

The loss function for Logistic Regression is defined as follows:

$$Loss(y, \hat{y}) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y}]$$

The loss will be smaller if the model's estimate is close to correct, and bigger if the model is perfroming misclassification.

2.1.2 Gradient Descent

The main target of Logistic Regression is to find a best fit line that separates the data. The model tries to minimize the loss.

Gradient descent is an iterative optimization algorithm to find the minimum of a function.

Repeat until convergence

{

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

}

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y^i} - y^i) x_j^i$$

where $J(\theta)$ is the cost function and $\frac{\partial J(\theta)}{\partial \theta}$ is the update or rate of change.
The value of $\frac{\partial J(\theta)}{\partial \theta}$ for logistic regression is similar to that of linear regression. But here the sigmoid of z is our hypothesis function.

2.2 Multimomial Logistic Regression

Sometimes we need more than two classes. Perhaps we might want to do 3-way sentiment classification (positive, negative, or neutral). In such cases we use multinomial logistic regression, also called softmax regression. In multinomial logistic regression we want to label each observation with a class k from a set of K classes, under the stipulation that only one of these classes is the correct one.

The multinomial logistic classifier uses a generalization of the sigmoid, called the softmax function, to compute $P(y_k = 1|x)$. The softmax function takes a vector $z = [z_1, z_2, ..., z_K]$ of K arbitrary values and maps them to a probability distribution, with each value in the range (0,1), and all the values summing to 1.

The formula for softmax regression is given as follows:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

for i=1,2,...,K where K represents the number of classes.

The loss function for multinomial logistic regression generalizes the loss function for binary logistic regression from 2 to K classes.

$$Loss(y, \hat{y}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i$$

In this lab we will create a logistic regression model that predicts whether a person buys computer or not.

[ ]:
```
#Importing libraries
import numpy as np
import pandas as pd
import io
import matplotlib.pyplot as plt
```

[ ]:
```
# reading the csv file, del 2 columns from the file, checking first few rows of the␣
 ↪file
from google.colab import files
uploaded = files.upload()

data = pd.read_csv(io.BytesIO(uploaded['BuyComputer.csv']))

data.drop(columns=['User ID',],axis=1,inplace=True)
data.head()
```

```
<IPython.core.display.HTML object>


Saving BuyComputer.csv to BuyComputer.csv
```

[ ]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

[ ]:
```
#Declare label as last column in the source file
```

[ ]:
```
#Declaring X as all columns excluding last
```

[ ]:
```
# Splitting data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = 0,0,0,0
```

[ ]:
```
# Sacaling data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[ ]:
```
#Variabes to calculate sigmoid function
y_pred = []
len_x = len(X_train[0])
w = []
b = 0.2
print(len_x)
```

[ ]:
```
entries = len(X_train[:,0])
entries
```

[ ]:
```
for weights in range(len_x):
    w.append(0)
w
```

[ ]:
```
# Sigmoid function
def sigmoid(z):
  pass
```

[ ]:
```
def predict(inputs):
    pass
```

[ ]:
```
#Loss function
def loss_func(y,a):
    J = 0
    return J
```

[ ]:
```
dw = []
db = 0
J = 0
alpha = 0.1
for x in range(len_x):
    dw.append(0)
```

[ ]:
```
#Repeating the process 3000 times
```

[ ]:
```
#Print weight
```

[ ]:
```
#print bias
```

[ ]:
```
#predicting the label
```

[ ]:
```
#print actual and predicted values in a table
```

[ ]:
```
# Calculating accuracy of prediction
```

#Using sklearn LogisticRegression model

[ ]:
```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(random_state = 0)

#Fit

#predicting the test label with LR. Predict always takes X as input
```

Exercise:

Try logistic regression on BuyComputer dataset and set Random state=Your_RollNumber (last 3 digit of ID, incase if you don't have ID)