ML ASSIGNMENT 4 REPORT

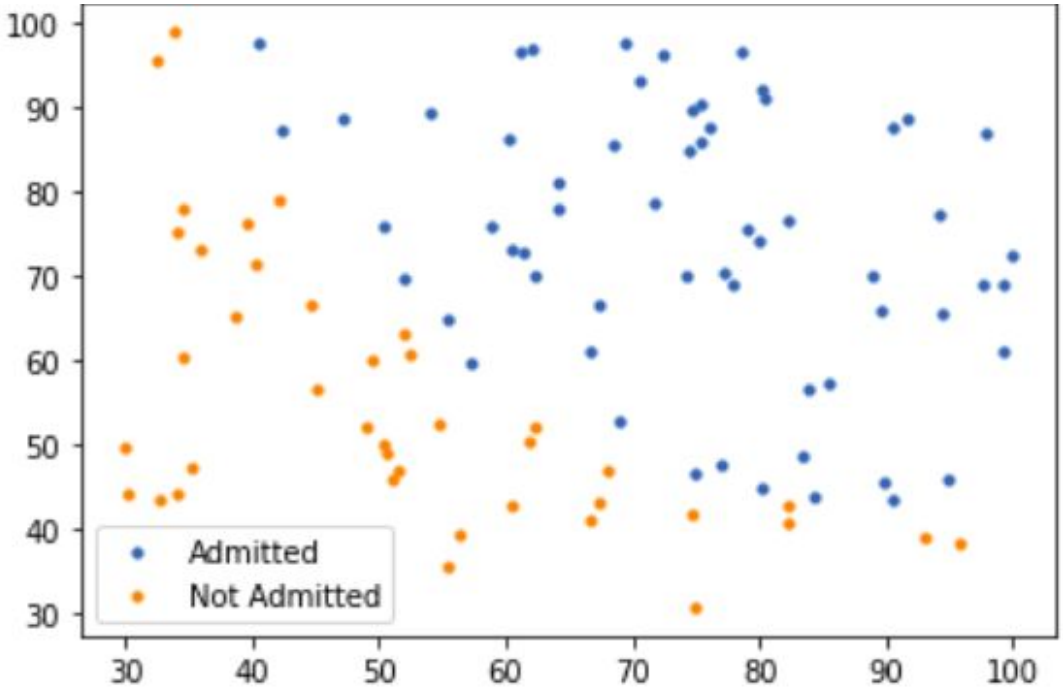
SUBMITTED BY IIT2018174

Question 1.Design a Predictor with two basic features which are given using Batch Gradient Descent Algorithm, Stochastic Gradient Algorithm and mini batch Gradient Descent algorithms (determining minibatch size is your choice- here it could be 10, 20, 30 etc.) with and without **feature scaling** and compare their performances in terms of % error in prediction.(only allowed to use NumPy library of Python, no other functions/libraries are allowed).

INTRODUCTION

Logistic Regression is generally used for classification purposes. Unlike Linear Regression, the dependent variable can take a limited number of values only i.e, the dependent variable is categorical.Here we are just talking about binary logistic regression

We first splitted data to visualize the problem effectively into admitted and non-admitted students.



We then apply the sigmoid function to the output of the linear regression

$$h(x) = \sigma(\theta^T x)$$

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

and then apply the following algorithm to optimize the cost.

BATCH-GRADIENT ALGORITHM:

This is a type of gradient descent which processes all the training examples for each iteration of gradient descent

We took learning rate as 0.1 and no. of iterations to be 1500

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

We applied the above thing in the code:

```
grad=np.dot(x_value.T,(sigmoid(np.dot(x_value,theta)) - y_value))
theta=theta-(learning_rate/m)*grad
```

Now after that we add the regularization term

```
Lambda is r_factor here
r_term=(((learning_rate*r_factor)/m)*np.dot(theta,N))
And futhu update the theta
theta-=r_term
```

In Addition to it we add (6-9) features in the data to make our prediction better:

1,X1,X2, X1^2 ,X2^2 ,X1*X2, X1*(X2^2), X2*(X1^2), X1^3

RESULT:-

```
THE PARAMETER VALUES ARE(without feature scaling in batch gradient descent algorithm):
[-11.80143907   1.02979058  -0.42669957]
THE PARAMETER VALUES ARE(with feature scaling in batch gradient descent algorithm):
[-3.7749129   4.50299371  3.57116159]
The Error in the Gradient Descent Algorithm without feature scaling is:
23.333333333333332
The Error in the Gradient Descent Algorithm with feature scaling is:
13.333333333333334
```

Feature scaling improves the prediction with less error,having the same learning rate and iterations.

```
THE PARAMETER VALUES ARE(with regularization in batch gradient descent algorithm):
[-0.26064925  0.43599428  0.31568161]
The Error in the Gradient Descent Algorithm with regularization is:
6.666666666666667
```

Regularization improved our prediction accuracy and error was less

```
THE PARAMETER VALUES ARE(with adding more features  in batch gradient descent algorithm):
[-1.03348946  0.41194722  0.22011972  0.43536299  0.30464962  0.65376584
  0.60774232  0.70704397  0.41844599]
The Error in the Gradient Descent Algorithm with adding more features is:
6.666666666666667
```

Stochastic Gradient Algorithm:

Same thing what we did in batch gradient but instead of evaluating all the training in one go,we are processing 1 training example per iteration,Hence, the parameters are being updated even after one iteration in which only a single example has been processed.

We have to change the learning rate and number of iterations to get better output

```
for i in range(iterations):
   for j in range(m):
```

```
    //Choosing random index everytime
     index=random.randint(0,m-1)
     x_value=X[index]
     y_value=Y[index]

theta=theta-(learning_rate/m)*np.dot(x_value.T,(sigmoid(np.dot(x_value,the
ta)) - y_value))
       # print(theta
 return theta
```

```
THE PARAMETER VALUES ARE(without feature scaling in Stochastic_Gradient algorithm):
[-1.07809524  0.03221002  0.00728577]
THE PARAMETER VALUES ARE(with feature scaling in Stochastic_Gradient algorithm):
[-0.5082017   0.8963666   0.62481575]
The Error in the Stochastic_Gradient algorithm without feature scaling is:
20.0
The Error in the Stochastic_Gradient algorithm with feature scaling is:
10.0

THE PARAMETER VALUES ARE(with regularization in Stochastic_Gradient algorithm):
[-0.48836557  0.83522034  0.59789634]
The Error in the Stochastic_Gradient algorithm with regularization is:
10.0
```

```
THE PARAMETER VALUES ARE(with addding more features in Stochastic_Gradient algorithm):
[-1.0250747   0.41915718  0.22205326  0.44141456  0.30459108  0.65736254
  0.61113747  0.70841267  0.42372951]
The Error in the Stochastic_Gradient algorithm with adding more features is:
6.666666666666667
```

MINI-BATCH GRADIENT ALGORITHM:

it is processed in batches of b training examples in one go,

```
batch_size=20
 n_batches=int(m/batch_size)
 learning_rate=0.0001
 iterations=2500
```

Deciding batch size ,number of batches ,learning rate ,iterations

```
idx=np.random.permutation(m)
   X=X[idx]
   Y=Y[idx]
   for j in range(0,m,batch_size):
```

```
x_value=X[j:j+batch_size]
y_value=Y[j:j+batch_size]
```

Putting the values in batch of batch_size and evaluating it in one go.
The error was improves with feature scaling.

```
THE PARAMETER VALUES ARE(without feature scaling in minibatch_Gradient algorithm):
[-1.56093914  0.06372165  0.01179531]
THE PARAMETER VALUES ARE(with feature scaling in minibatch_Gradient algorithm):
[-0.71789862  1.13310844  0.78531819]
The Error in the minibatch_Gradient algorithm with feature scaling is:
20.0
The Error in the minibatch_Gradient algorithm with feature scaling is:
3.3333333333333335
```

```
THE PARAMETER VALUES ARE(with regularization in minibatch_Gradient algorithm):
[0.0040277  0.03227463 0.02445943]
The Error in the minibatch_Gradient algorithm with regularization is:
20.0
```

```
THE PARAMETER VALUES ARE(with feature addition in minibatch_Gradient algorithm):
[0.00149315 0.02192424 0.01651016 0.02051782 0.01630323 0.02406145
 0.02160806 0.02386321 0.01854823]
The Error in the minibatch_Gradient algorithm with features addition is:
20.0
```

We can see that feature scaling has less error among the following

QUESTION2: Design a classifier using logistic regression on Cleveland Medical data set for heart disease diagnosis. The processed dataset with some 13 features have been given with a label that a patient has a heart disease (1) or not (0). This design should have a professional touch within your ML knowledge. Below the data set brief description has been provided. Also, if you want to have more knowledge about the very interesting data set, please go through the link provided

INTRODUCTION

For designing the classifier we used a batch gradient algorithm with feature scaling and regularization for better prediction.

The output we got was 18% error :-

The hyperparameter were following:-

```
learning_rate=0.01
 iterations=1500
 r_factor=10


With Feature Scaling
Batch_Gradient
Parameters:
 [-0.6913432  -0.06731282  0.07759242  0.22266571 -0.04701713 -0.06698858
  0.17691127 -0.15166458 -0.5712945   0.50196683  0.20412201  0.26997171
  0.70713152  0.46609437]

Error: 18.27956989247312 %
Accuracy: 81.72043010752688 %
According to Confusion matrix:

Accuracy: 81.72043010752688
Recall: 68.18181818181817
True Negative rate: 93.87755102040816
Precision: 90.9090909090909
```

We also done our prediction with min_batch GDA and stochastic GDA

```
Error with mini_batch was 19%
Error with stochastic gradient was 18.5%
```