CS 4210- HW 4

In HW3, we are going to write codes for (1) gradient descent algorithm for logistic regression and (2) K-fold cross-validation.

Task 1. Classifying MNIST data

Download the MNIST data MNIST_CV.csv contains 8,200 hand-written digit image data for label of 6 and 8. So, we will solve a binary classification problem.

For training and test data, split the data into K=10 sets for 10-fold cross-validation. Then you will have ten sets of training and test data. Note that training and data sets must be “disjoint”.

For each fold, you need to train a logistic regression by the gradient descent algorithm. And compute True Positive Rates and False Positive Rates for ROC curve. Then, make the ROC Curve of 10-fold cross-validation (use averaged TPR and FPR). Consider the thresholds: [0, 0.1, 0.2, 0.3, …, 0.9, 1]

You can use built-in function to split the dataset for cross-validation. But don’t use any library for logistic regression. You have to implement gradient descent algorithm to find the optimal parameter of the logistic regression

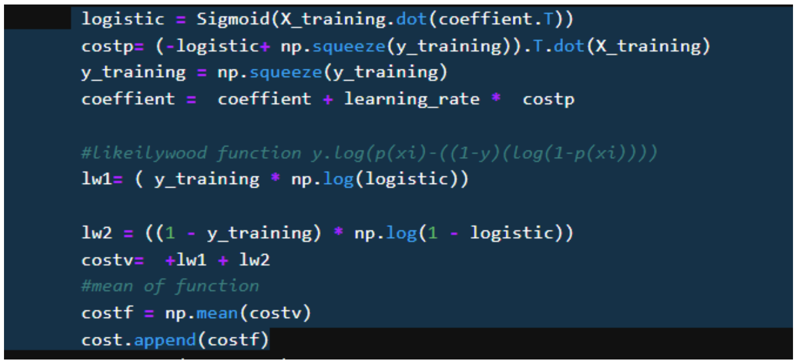
Example code is given below, you need to fill Sigmoid function before using that code

1. MS word file
   - Describe what you have done for the homework assignment.
   - **MUST include tables of averaged TPR and FPR on the various thresholds.**
   - **MUST include a ROC Curve**
   - **MUST include AUC of the ROC Curve. Use a built-in function to compute AUC from TPR and FPR.**
2. Python source code file(s)
   - Must be well organized (comments, indentation, …)
   - You need to upload the “original python file (*.py)” and also its “PDF” version.
     - For the PDF file, you can just convert the source file to PDF. One way is to print the source file and save to “PDF”.

You have to submit the files SEPERATELY. DO NOT compress into a ZIP file.

Deadline:

The deadline is **11:59pm Monday, November 4, 2024**.



```
logistic = Sigmoid(X_training.dot(coeffient.T))
costp= (-logistic+ np.squeeze(y_training)).T.dot(X_training)
y_training = np.squeeze(y_training)
coeffient =  coeffient + learning_rate *  costp

#likeilywood function y.log(p(xi)-((1-y)(log(1-p(xi))))
lw1= ( y_training * np.log(logistic))

lw2 = ((1 - y_training) * np.log(1 - logistic))
costv=  +lw1 + lw2
#mean of function
costf = np.mean(costv)
cost.append(costf)
```