CSCE 636: Deep Learning (Fall 2022)

Assignment #1

TEXAS A&M UNIVERSITY

10 Sep 2022

Submission By: Ashutosh Chauhan

UIN: 232009024

Email: ashutosh@tamu.edu

I. Objective

Prepare a Linear Models for Handwritten Digits Classification.

A) Programming Assignment
B) Non-Programming Assignment

II. Non-Programming Assignment

1) Data Preprocessing:

(a) Explain what the function train_valid_split does and why we need this step.

Train _valid_split function divide our input data into two different subsets - training subset and validation subset. Training subset is used to train the model parameters. Validation subset is used to check the model accuracy prior to testing it on final testing data. Difference in accuracy of model on training set and validation set help us to analyze our model. If training accuracy is high and validation accuracy is less it means that our model has overfitted on the training data. If both training and validation accuracy are low then we could make out that model is underfitted.

(b) Before testing, is it correct to re-train the model on the whole training set? Explain your answer.

No, it is not correct to re-train the model on complete (training + validation) set before finally testing it because it would again invalidate our final optimized parameters and could overfit the model. And there will be no further data available to again validate our new model parameters.

(c) Programming: Implement prepare_X

(d) In the function prepare X, there is a third feature which is always 1. Explain why do we need it?

The third feature that we use in input X is called Bias term. We need bias term because it adds a constant term w0 to our model which does not depends on input. It provides more flexibility to movement of our hypothesis in solution space.

For example: If our model is y = x1 * w1 , than it has restriction to pass from origin only and depends only on input. Whereas, in case our model is y = x1*w1 + w0 , than the model is more flexible and can represent any linear term. Bias term in input facilitate us in getting that w0 term.

(e) Programming: Run prepare_y

(f) Test your code in "code/main.py" and visualize the training data from class 1 and 2 by implementing the function visualize features. The visualization should not include the third feature. Therefore it is a 2-D scatter plot. Include the figure in your submission.

Scatter plot was plotted showing spread of intensity and symmetry feature. We can observe that class 0 has high intensity and low symmetry whereas class 1 has high symmetry and low intensity. Refer Figure 1 for scatter plot of training data.

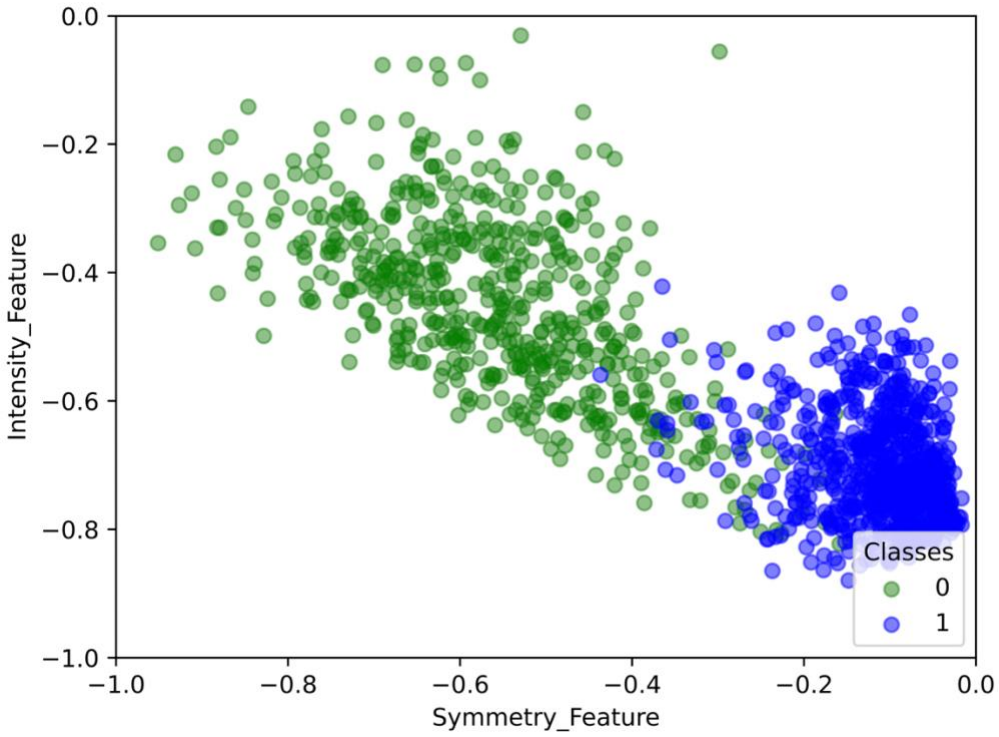


Figure 1: Scatter Plot of Training Data

2) Cross Entropy Loss:

(a) Write the loss function E(w) for one training data sample (x,y).

$$E(w) = ln(1 + e^{(-yw^{(T)}x)})$$

(b) Compute the gradient ∇E(w).

$$\nabla E(w) = \nabla ln(1 + e^{-yw^T x})$$

$$= \frac{1}{1 + e^{(-yw^T x)}} \nabla\left(1 + e^{(-yw^T x)}\right)$$

$$= \frac{1}{1 + e^{(-yw^T x)}} (-\text{yx}e^{(-yw^T x)})$$

$$= -y.\frac{e^{(-yw^T x)}}{1 + e^{(-yw^T x)}} .\text{x}$$

$$= -y.\theta(yw^T x).x$$

(c) This is not the most efficient way since the decision boundary is linear. Why? Explain it. When will we need to use the sigmoid function in prediction?

It is not the most efficient way to predict output once the w is already calculated because it will involve extra time complexity of solving sigmoid function value.

Instead, we could directly use following prediction method:

$$\text{y^ = x(predicted value)} = \begin{cases} 1, & if\ w^T x >= 0 \\ -1, & if\ w^T x < 0 \end{cases}$$

Sigmoid function should be used while training the model because it can be differentiated easily which could help in finding the gradient descent easily and hence minimizing the loss efficiently.

(d) Is the decision boundary still linear if the prediction rule is changed to the following? Justify briefly.

True, it will still create a linear decision boundary even if the threshold value is changed from 0.5 to 0.9 because it is a monotonically increasing function and will still map unique value of $w^T x$ to input.

(e) In light of your answers to the above two questions, what is the essential property of logistic regression that results in the linear decision boundary?

Logistic regression decision boundary is created using the prediction method y = $w^T x$ which is a linear function wrt input x. $w^T x$ is a monotonically increasing function and maps a unique value to each input x and hence results in linear decision boundary.

3) Sigmoid logistic regression:

(a) Programming: implement the function gradient.

(b) Programming: Implement batch gradient descent, stochastic gradient descent and mini-batch gradient descent in the functions fit BGD, fit SGD and fit miniBGD, respectively.

(c) Programming: Implement the functions predict, score and predict proba.

(d) Test your code in "code/main.py" and visualize the results after training by using the function visualize results. Include the figure in your submission.

The decision boundary, symmetry feature and intensity feature are plotted on the 2D scatter plot. It can be observed from the plot that the decision boundary is separating both classes of input data accurately.

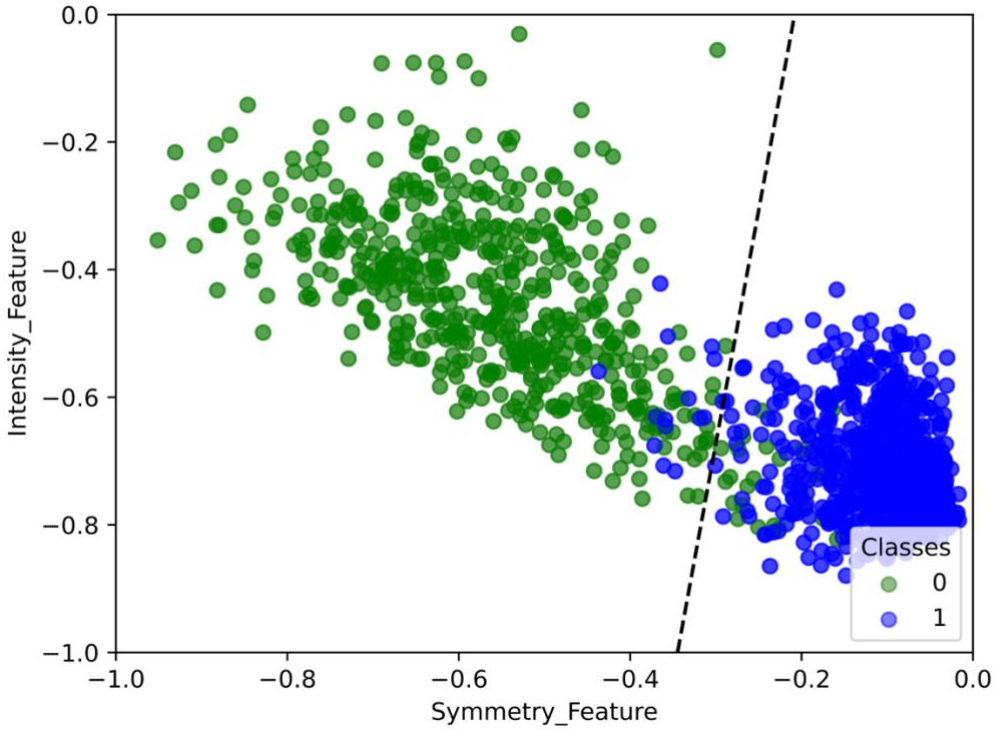


Figure 2: Scatter Plot with decision boundary in Sigmoid LR

(e) Implement the testing process and report the test accuracy of your best logistic regression model.

- Best Logistic Regression Sigmoid weights:

  [4.85693232 23.35148898 -3.18940373]

- Best Logistic Regression Sigmoid train accuracy: 97.25
- Best Logistic Regression Sigmoid validation accuracy: 97.88
- Test Accuracy: 93.93

4) Softmax Logistic Regression:

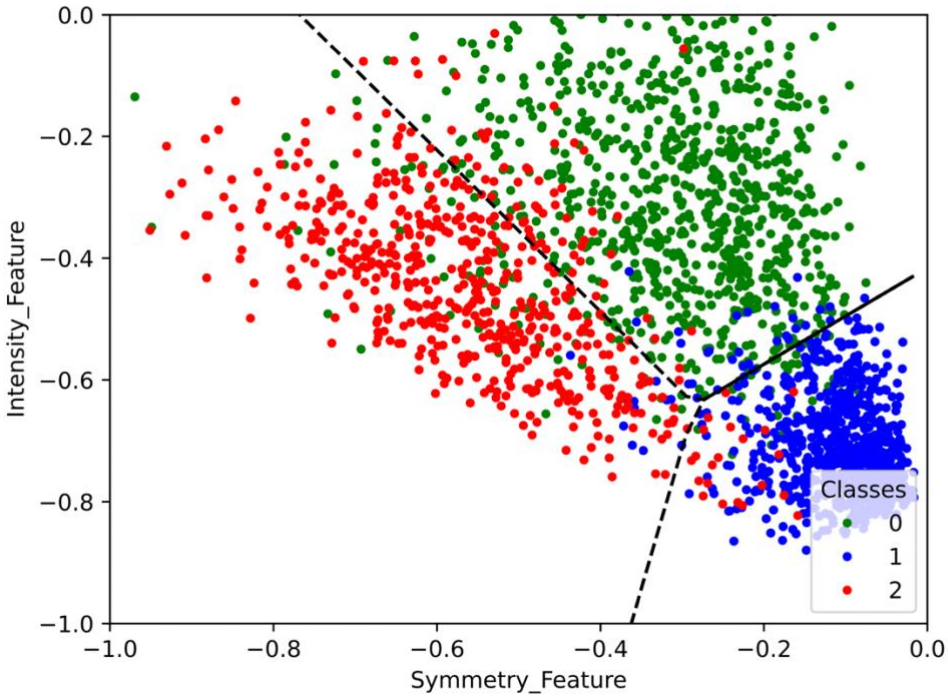
(a) Programming: Implement the function gradient.

(b) Programming: Implement mini-batch gradient descent.

(c) Programming: Implement the functions predict and score.

(d) Test your code in "code/main.py" and visualize the results after training by using the function visualize results multi. Include the figure in your submission.

The decision boundary, symmetry feature and intensity feature are plotted on the 2D scatter plot. It can be observed from the plot that the decision boundaries are separating all 3 classes of input data accurately.



(e) Implement the testing process and report the test accuracy of your best logistic regression model.

- Best LR Multiclass weights:

  [[6.72824027 0.38752899 10.32514648]

  [-1.11471254 15.29841055 -8.5262314]

  [-5.61352773 -15.68593954 -1.79891509]]

- Best LR Multiclass train accuracy: 89.73
- Best LR Multiclass validation accuracy: 87.30
- Best LR Multiclass Test Accuracy: 86.60

5) Softmax logistic vs Sigmoid logistic:

(a) Train the softmax logistic classifier and the sigmoid logistic classifier using the same data until convergence. Compare these two classifiers and report your observations and insights.

Both softmax and sigmoid classifier are trained using same data for 1000 iterations and the accuracy, and score are noted. It is observed that both softmax and sigmoid give same accuracy of 94.588. Its shows that both can be used interchangeably.

Softmax Classifier	Train Accuracy	97.25
	Validation Accuracy	97.61
	Test Accuracy	94.58
Sigmoid Classifier	Train Accuracy	97.25
	Validation Accuracy	97.61
	Test Accuracy	94.58

Table 1: Softmax and Sigmoid classifier accuracy

(b) Explore the training of these two classifiers and monitor the graidents/weights. How can we set the learning rates so that $w_1 - w_2 = w$ holds for all training steps?

Weights are calculated for both the classifier by training on same data and using same number of iterations. Learning rate of sigmoid classifier is kept double of softmax classifier as was taught in lecture and it was verified that difference of weights of softmax (w_2-w_1) was coming out to be equal to weight of sigmoid (w0).

Sigmoid	w_0	[-0.12415534 6.47262768 -3.15528367]
Softmax	w_1	[0.06207767 -3.23631384 1.57764184]
	w_2	[-0.06207767 3.23631384 -1.57764184]
	$w_2 - w_1$	[-0.12415534 6.47262768 -3.15528367]

Table 2 : Weights of Sigmoid and Softmax classifier

III. Analysis of Programming Assignment

1) Sigmoid Logistic Regression

Sigmoid logistic regression is trained on different types of gradient descent functions batch gradient descent, mini- batch gradient descent and stochastic gradient descent and accuracy is calculated for all on different learning rate and iteration value. The best test accuracy of 93.93% is achieved when mini-batch gradient is used with 0.1 learning rate, 1200 iterations and 25 batch size.

Mini Batch Gradient Descent	**Batch size = 1**	**Batch size =25**
Learning Rate	0.1	0.1
Iterations	1200	1200
Train Accuracy	96.97	97.25
Validation Accuracy	96.82	97.88
Test Accuracy	92.85	93.93

Table 3: Accuracy analysis of sigmoid logistic regression

2) Softmax Multiclass Logistic Regression

Softmax logistic regression model is trained using mini-batch gradient descent and multiple iterations were done using various combination of batch size, learning rate and number of iterations. The best accuracy was achieved while using batch size of 25, learning rate of 0.1 and number of iterations of 1000. Best accuracy was coming out to be 86.60%. Validation accuracy is less as compared to training accuracy, which show that model is getting overfitted. Using regularization term can improve the overall accuracy.

Mini Batch Gradient Descent	**Batch size = 1**	**Batch size =25**
Learning Rate	0.1	0.1
Iterations	1000	1000
Train Accuracy	89.13	89.7
Validation Accuracy	87.93	87.30
Test Accuracy	85.87	86.60

Table 4: Accuracy analysis of Softmax logistic regression

```
ashutoshchauhan@client-10-228-10-169 code % /usr/local/bin/python3 /Users/ashutoshchauhan/Downloads/HW1/code/main.py
[ 0.18761413  3.55944904 -2.06327285]
94.44444444444444
[ 0.18761413  3.55944904 -2.06327285]
94.44444444444444
[ 0.44848307  3.17200854 -1.87898148]
90.5925925925926
[10.38612223 32.55240052  2.81104834]
97.11111111111111
[ 4.89709838 23.59676087 -2.96524442]
97.18518518518519


 Best Logistic Regression Sigmoid
Best Logistic Regression Sigmoid weights: [ 4.85693232 23.35148898 -3.18940373]
Best Logistic Regression Sigmoid train accuracy:  97.25925925925925
Best Logistic Regression Sigmoid validation accuracy:  97.88359788359789
Test Accuracy: 93.93939393939394
[[  6.73317989   0.43440348  10.54833281]
 [ -1.14978962  15.73698023  -8.53979319]
 [ -5.58339027 -16.17138371  -2.00853961]]
89.04347826086956


 Best LR Multiclass
Best LR Multiclass weights:
 [[  6.72824027   0.38752899  10.32514648]
 [ -1.11471254  15.29841055  -8.5262314 ]
 [ -5.61352773 -15.68593954  -1.79891509]]
Best LR Multiclass train accuracy:  89.73913043478261
Best LR Multiclass validation accuracy:  87.3015873015873
Best LR Multiclass Test Accuracy:  86.60170523751522


2-Class Softmax LR
Softmax LR weights:
 [[-1.25362205 -9.59062015  2.56390865]
 [ 1.25362205  9.59062015 -2.56390865]]
Softmax LR train accuracy:  97.25925925925925
Softmax LR validation accuracy:  97.61904761904762
Softmax LR test accuracy:  94.58874458874459


Binary Sigmoid LR
Binary Sigmoid LR weights:
 [ 2.5072441  19.18124029 -5.12781729]
Binary Sigmoid LR train accuracy:  97.25925925925925
Binary Sigmoid LR validation accuracy:  97.61904761904762
Binary Sigmoid LR test accuracy:  94.58874458874459
sigmoid weights:
 [-0.12415534  6.47262768 -3.15528367]
softmax weights:
 [[ 0.06207767 -3.23631384  1.57764184]
 [-0.06207767  3.23631384 -1.57764184]]
ashutoshchauhan@client-10-228-10-169 code %
```