CSCE 421: Machine Learning

Homework #2

February 14, 2025

Sutton Elliott

UIN:531008819

Email: suelliott77@tamu.edu

Data Preprocessing:

(a) Explain what the function train valid split does and why we need this step.

The function train_valid_split splits the dataset into two parts: training set and validation set. This is necessary because it allows us to use the performance of the model on unseen data before testing. We can also use the validation set to choose the best parameters.

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

Yes, it is correct. The only thing to ensure that this is done is after selecting the best parameters. The model's final performance test should be tests on data it has never seen (test data). By retraining on the full training set, we can improve generalization.

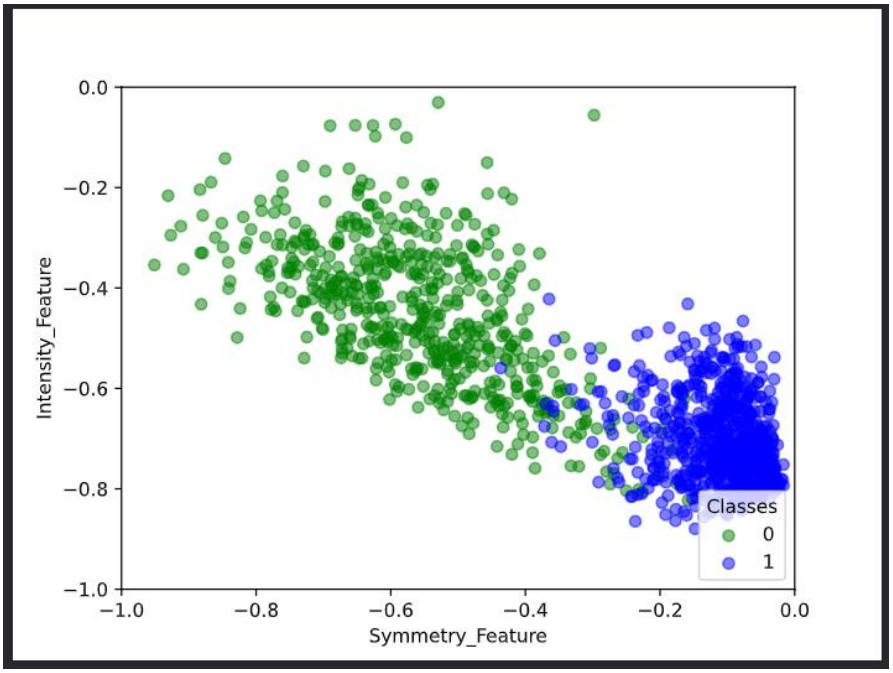
(c) Implement them in the function prepare X.

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

The third feature is a term called bias. It makes the decision boundary more flexible, similar to including an intercept in linear regression.

(e) 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



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) Computer the gradient ∇E(w).

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

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

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

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

(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. The decision boundary in logistic regression is linear because the model works by computing a linear combination of the input features $w^T x$. The sigmoid function maps this linear combination to a probability between 0 and 1. The sigmoid function should be used during training the model because it can differentiate easily while helping to find the gradient descent easily.

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

It will still create a linear decision boundary even if the threshold value is changed to 0.9. It is a monotonically increasing function and will still map unique values of w^T to input.

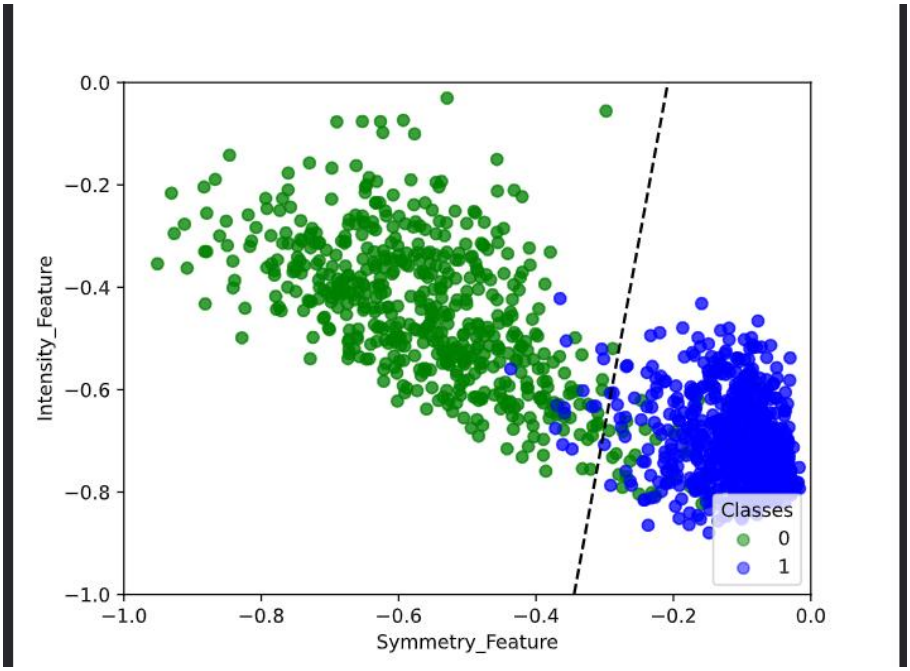
(e) In light of your answers to the above two questions, what is the essential property of logistic regression that results in the linear decision boundary?

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

Sigmoid Logistic Regression:

(a) Implement the function gradient
(b) Implement batch gradient descent, stochastic gradient descent and mini-batch gradient descent in the functions fit BGD, fit SGD, and fit miniBGD, respectively
(c) Implement the functions predict, score and predict proba

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



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

Best Logistic Regression Sigmoid weights: [4.85693232, 23.35148898, -3.18940373]

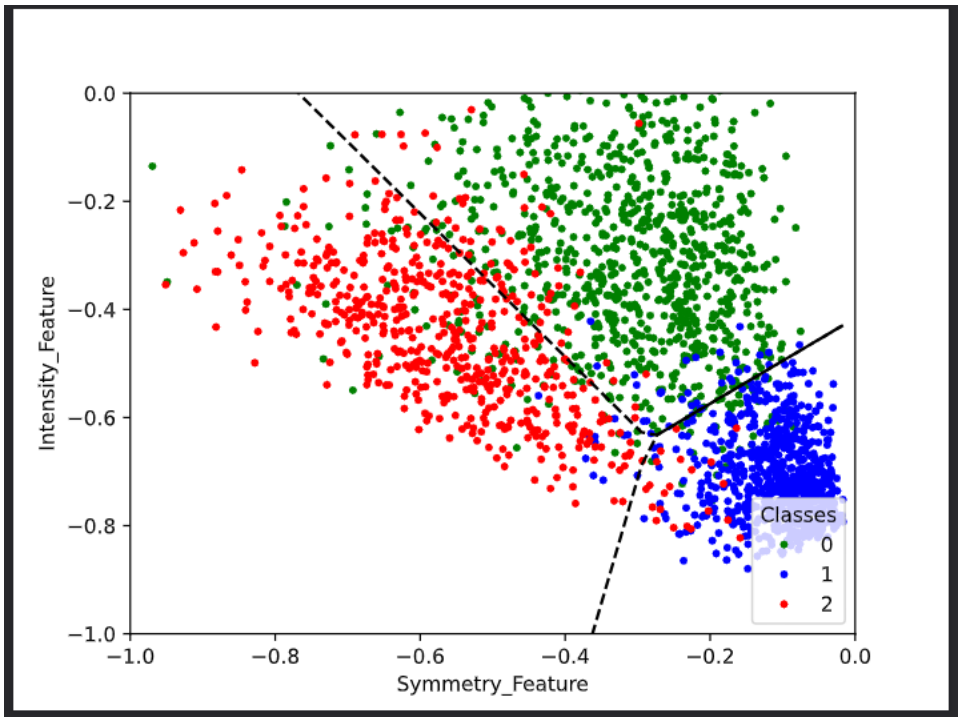
Best Logistic Regression Sigmoid train accuracy: 97.25925925925925

Best Logistic Regression Sigmoid validation accuracy: 97.88359788359789

Test Accuracy: 93.93939393939394

SoftMax Logistic Regression:

(a) Implement function _gradient
(b) Implement mini-batch gradient descent in the function fit_miniBGD
(c) Implement function predict and score for prediction and evaluation respectively
(d) Test your code in main and visualize the results after training by using the function visualize results multi. Include the plot in your submission



(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: 89.73913043478261

Best LR Multiclass validation accuracy: 87.3015873015873

Best LR Multiclass Test Accuracy: 86.60170523751522

SoftMax Logistic vs. Sigmoid Logistic

Sigmoid Weights:

Sigmoid weights=[−0.12415534,6.47262768,−3.15528367]

Softmax Weights:

Softmax weights=[0.06207767,−3.23631384, 1.57764184

−0.06207767, 3.23631384, −1.57764184]

The weight vectors for both models show different structures due to the nature of the classifiers. The Sigmoid classifier has a single weight vector since it is a binary classifier, whereas the Softmax classifier has a matrix of weights corresponding to each class.

Model	Train Accuracy	Validation Accuracy	Test Accuracy
Softmax	97.25%	97.61%	94.58%
Sigmoid	97.25%	97.61%	94.58%

Both the Softmax and Sigmoid classifiers achieved identical test accuracy (94.58%) despite Softmax being designed for multi-class classification and Sigmoid for binary classification. This suggests that for this dataset, which likely involves binary classification or can be treated as binary, both methods produce similar results.

The relationship between the weights of the Sigmoid and Softmax classifiers is interesting. Given the weights provided:

- Sigmoid Weight Vector: w sigmoid= [−0.12415534,6.47262768,−3.15528367]
- Softmax Weight Matrices:
    - w1=[0.06207767,−3.23631384,1.57764184]
    - w2=[−0.06207767,3.23631384,−1.57764184]

The Softmax classifier has two sets of weights, each corresponding to one class. The difference between these weight vectors is:

w2−w1=[−0.12415534,6.47262768,−3.15528367]

Interestingly, the difference between the Softmax weights (i.e., w2−w1) is exactly equal to the weights of the Sigmoid classifier. This highlights a key relationship between the two models in the case of binary classification, where Softmax with two classes is equivalent to Sigmoid.

Conclusion and Insights:

1. Model Performance: Both the Sigmoid and Softmax classifiers achieved identical test accuracy (94.58%), suggesting that for this dataset, either model could be used for classification, and there is no significant performance difference for this particular problem.
2. Weight Relationship: The weights of the Sigmoid classifier are exactly equal to the difference between the weights of the two classes in the Softmax classifier. This demonstrates that, for binary classification, Softmax reduces to Sigmoid, and both classifiers are effectively equivalent in this scenario.
3. Learning Rate and Weight Update Dynamics: By adjusting the learning rates as you did, you ensured that the relationship $w2-w1=w0$ holds, which showcases the control you have over the weight update dynamics in both classifiers.

For binary classification, both Sigmoid and Softmax can be used interchangeably with similar results, and understanding the relationship between their weights is crucial. The key takeaway is that Softmax, with two classes, reduces to a Sigmoid model, and tuning the learning rate appropriately helps maintain the desired weight dynamics.