Cmpe 462 Machine Learning Project 1 Report

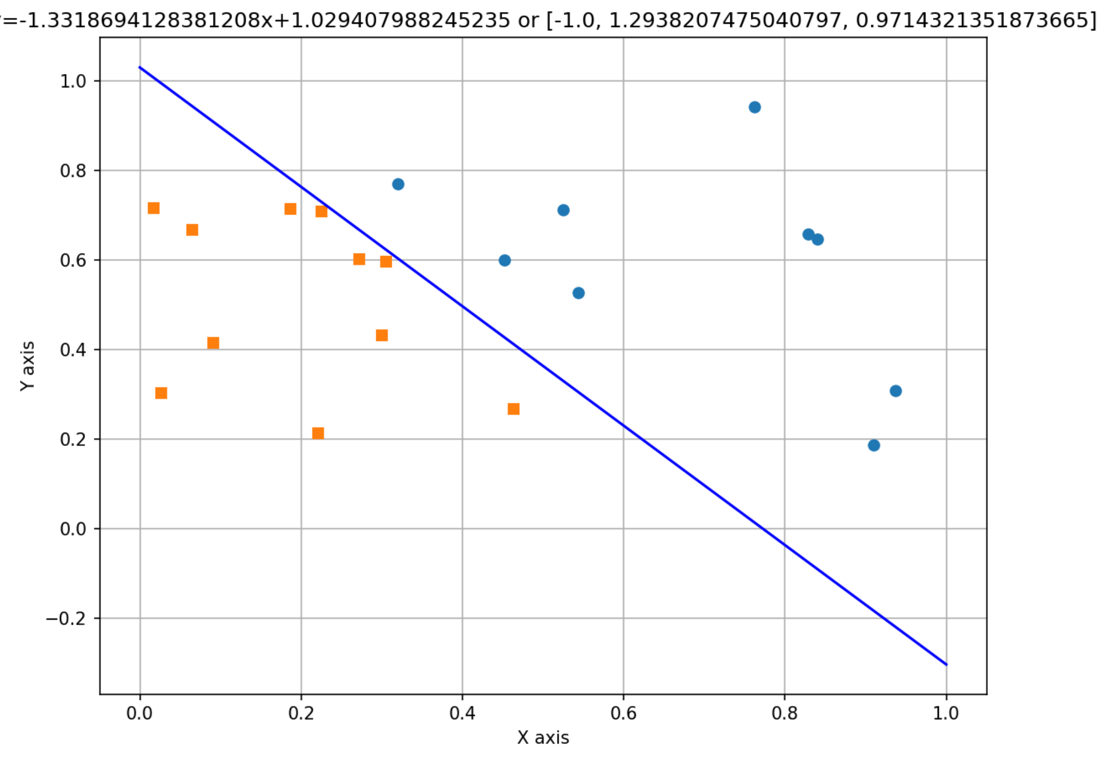
2020400147 – Muhammet Ali Topcu
2019400309 – Halil İbrahim Gürbüz
2021400171 – Mücahit Erdoğan Ünlü

Link to our Code

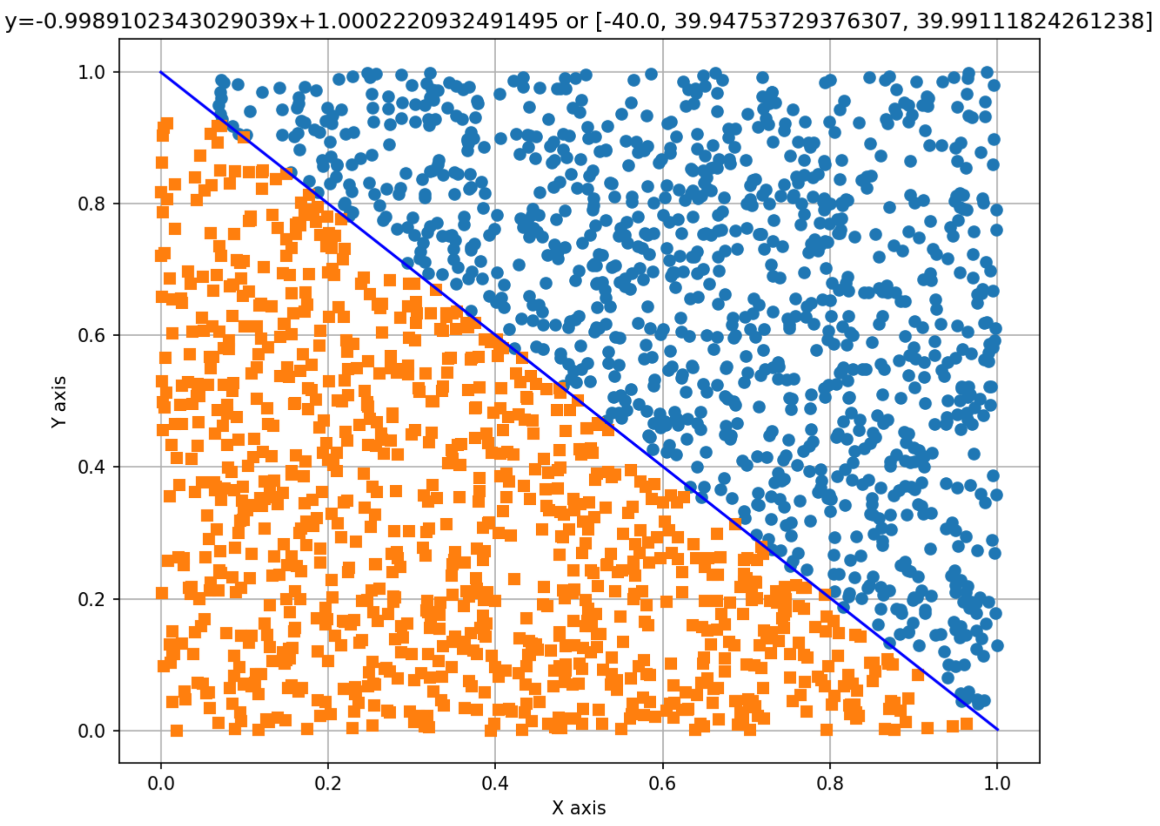
https://github.com/alitpc25/cmpe462projects/tree/main/HW1

Part 1: Perceptron Learning Algorithm

1) For both small and large data sets, weights are initialized as zeros. For the small dataset, it takes the algorithm 21 iterations to converge for a decision boundary. This number is 4294 for the large dataset. The large dataset requires more iterations to converge, as expected.
2) **Small dataset and its decision boundary**



3) Large dataset and its decision boundary



Using the small dataset, the initial weights and their learned decision boundaries are as follows:

Initial weights	Decision boundary
[0, 0, 0]	[-1.0, 1.2938207475040797, 0.9714321351873665]
[10, -30, 1]	[-1.0, 1.7050412477236225, 0.7774515928630655]
[3, 2, 1]	[-2.0, 2.9542224356805655, 1.7147730636919647]
[-1.0, 1.2938207475040797, 0.9714321351873665]	[-1.0, 1.2938207475040797, 0.9714321351873665]
[-1, 1, 1]	[-1, 1, 1]

With different initial weights, we may observe different decision boundaries. The reason for that is that each dataset has a set of decision boundaries. The PLA algorithm tries to find one of them. This search is governed by the following rule: The PLA updates its current weights by adding them to a misclassified data vector. Therefore, the next weights are determined solely by the previous weights and the last misclassified sample. The ultimate decision boundary, defined by the final weights, is influenced by this iterative process. Hence, the selection of initial weights impacts the eventual configuration of weights.

Depending on the dataset, the solution set may consist of infinitely many elements as is the case with our dataset. Nevertheless, despite this variability, it's noteworthy that the decision boundaries can be the same as seen in rows 1 and 4.

Part 2: Logistic Regression

1)

The last column of the data consists of strings to name the two classes. The logistic regression model needs two numerical labels. Therefore, we converted “Cammeo” to 1, and “Osmancik” to -1.
The data consists of 7 attributes with values that have different ranges. This can mislead our machine learning model to give more weight to some of the attributes than others. To eliminate this, we applied Z-score normalization. It converts all input values to a common measure with a standard deviation of one and an average of zero. (Guide)
We also added a constant 1 as the first attribute of the data. This is to include the bias term in the decision boundary while doing computations.

2)
The data are split randomly into 5 parts. The model is trained according to the lambda values for [0.3, 0.2, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0]. Since the data is shuffled in each execution, the best lambda can get different but close values each time.

(Accuracy means average test accuracy.)
Lambda: 0.3
Accuracy: 92.17847769028872
Lambda: 0.2
Accuracy: 92.23097112860893
Lambda: 0.1
Accuracy: 92.49343832020998
Lambda: 0.05
Accuracy: 92.65091863517061
Lambda: 0.01
Accuracy: 92.86089238845143
Lambda: 0.005
Accuracy: 92.83464566929134
Lambda: 0.001
Accuracy: 92.75590551181102
Lambda: 0.0005
Accuracy: 92.78215223097112
Lambda: 0
Accuracy: 92.78215223097112
Best regularization parameter (lambda) 0.01

3)

Model Type	Test Accuracy	Training Accuracy
logistic regression trained with GD	92.78215223097112	92.78215223097114
logistic regression trained with SGD	92.44094488188976	92.58530183727034
regularized logistic regression trained with GD	92.86089238845143	92.75590551181102
regularized logistic regression trained with SGD	92.54593175853019	92.42125984251969

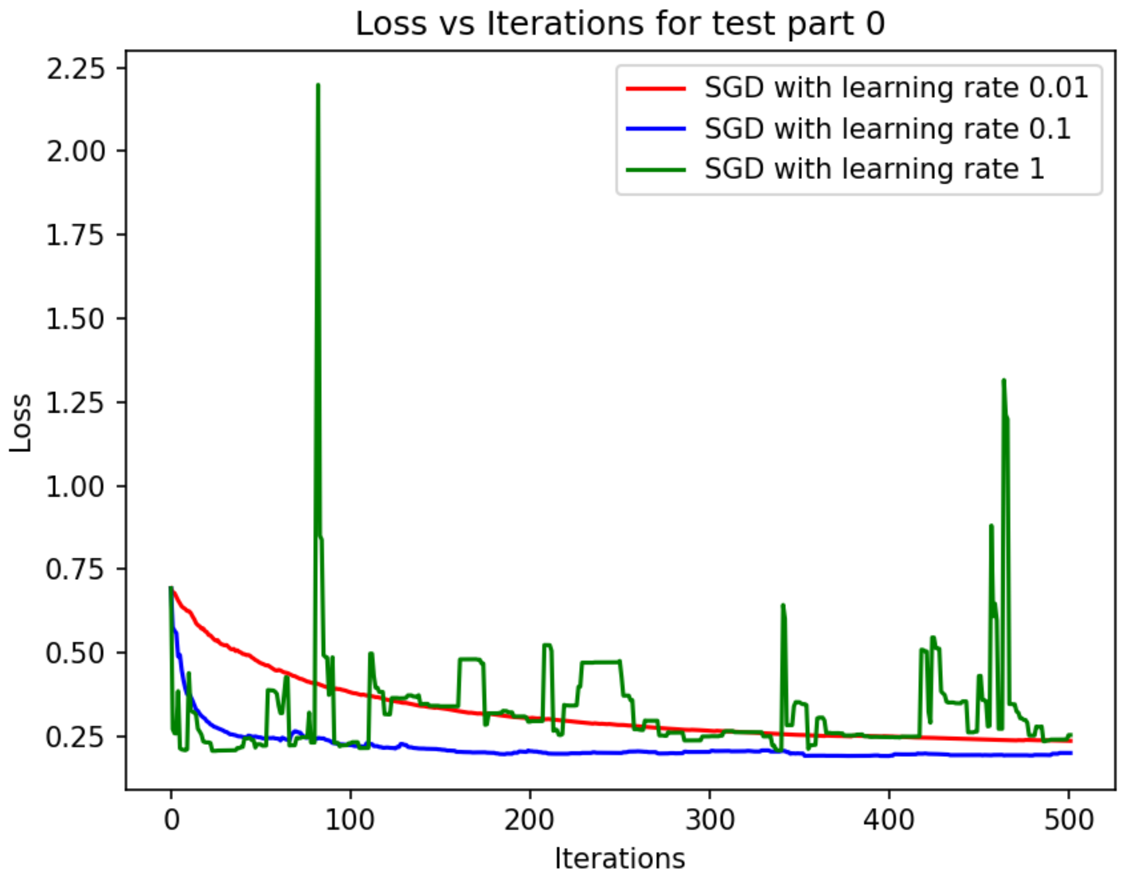
Comments on the results:

- LG/GD has the same test and training accuracies which can happen sometimes.
- LG/SGD has lower test accuracy than its training accuracy. This is because the model overfits according to the training data.
- RLG/GD has higher test accuracy than its training accuracy. This is because the model alleviates overfitting by using the regularization parameter.
- RLG/SGD has higher test accuracy than its training accuracy. This is because the model alleviates overfitting by using the regularization parameter.

4)
Average test accuracy for GD: 0.9278215223097112
Average test accuracy for SGD: 0.9244094488188976

average execution time for GD 0.21239275932312013
average execution time for SGD 0.10313901901245118
SGD is twice as efficient in terms of execution time than GD.

5)



The best learning rate is 0.1. For a higher learning rate, 1, the fluctuations in the loss function are so high that it is hard to determine a convergence value. For a lower learning rate, 0.01, the loss function does not converge to the best loss value even after 500 iterations. On the other hand, the model converges to the best loss value at around 100 iterations with the best learning rate.

A fourth learning rate, 0.1, is used to show that the model cannot find a better loss value. We used this part to determine the best learning rate as well.

Part 3: Naive Bayes

1)
We have used 5-fold cross-validation to get robust accuracy values.
naive bayes test accuracy: 93.14392175128084
naive bayes training accuracy: 93.9365721997301
As expected, naive Bayes has higher accuracy on the training dataset than the test dataset.

2)
Without conditional independence assumption:

$$\#variance\ parameters = \frac{30 * 30 - 30}{2} + 30 = 465$$
$$\#mean\ parameters = 30$$
$$\#classes = 2$$
$$\begin{aligned}\#parameters\ to\ be\ estimated &\\ &= \#classes * (\#variance\ parameters + \#mean\ parameters)\\ &= 2 * (465 + 30) = 990\end{aligned}$$

With conditional independence assumption:

$$\#variance\ parameters = 30$$
$$\#mean\ parameters = 30$$
$$\#classes = 2$$
$$\begin{aligned}\#parameters\ to\ be\ estimated &\\ &= \#classes * (\#variance\ parameters + \#mean\ parameters)\\ &= 2 * (30 + 30) = 120\end{aligned}$$

As can be seen, the Naive Bayes assumption makes the computation a lot easier.

3)

naive bayes test accuracy: 93.14392175128084
logistic regression test accuracy: 97.18677224033534

Logistic regression performs better than Naive Bayes on this dataset.

Naive Bayes is a generative model. This means that the model estimates the parameters of the likelihoods of features given label and the prior. In doing so, it assumes conditional independence and makes an assumption on its distribution. Since it does not overfit, NB requires a small amount of training data to estimate the parameters. NB has a high bias and low variance due to its strong assumption of feature independence.

Logistic regression does not make either of these assumptions. It is a discriminative model. This means that it predicts the occurrence of a specific categorical event estimating the parameters of a logistic model where the variables are in the linear combination. It does not need the conditional independence assumption due to this linear combination. LR requires a large amount of training data to avoid overfitting. Logistic regression typically has a lower bias if the features and the target have a linear relationship. Also, it has high variance especially when the number of features is large and if model complexity is not properly controlled such as through regularization.

Naive Bayes and Logistic Regression produce asymptotically the same model if the Naive Bayes assumption holds. ("Lecture 5: Bayes Classifier and Naive Bayes") Since this is not the case most of the time and also with our dataset, Logistic Regression performs better than Naive Bayes. Another reason is that Logistic regression does not suffer from overfitting because the number of data is enough.

Work Cited

“Guide, Step. “Effects of Normalization Techniques on Logistic Regression in Data Science.” *Turing*, https://www.turing.com/kb/effects-of-normalization-techniques-on-logistic-regression-in-data-science.

“Lecture 5: Bayes Classifier and Naive Bayes.” *Cornell CS*, https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote05.html.

“Choice of a Regularization Parameter “ Statlect, https://www.statlect.com/machine-learning/choice-of-a-regularization-parameter#:~:text=How%20do%20we%20choose%20the%20regularization%20parameter%3F&text=as%20follows%3A,on%20the%20validation%20set)%3B

“The Perceptron Algorithm: Some Limitations” https://data-intelligence.hashnode.dev/the-perceptron-algorithm-some-limitations

“Useful Numpy Functions in Data Science” https://data-intelligence.hashnode.dev/the-perceptron-algorithm-some-limitations

“Pandas Functions in Python” https://www.geeksforgeeks.org/pandas-functions-in-python/

“NumPy Reference” https://numpy.org/doc/stable/reference/index.html