Binary classification using logistic regression

Practice#1-2

Report by: 완(2019007901)

Running Environment

- OS: Windows 10 Home
- CPU: AMD Ryzen 7 5800H
- Language: Python 3.10

Introduction

For this assignment, we will train a model for binary classification using logistic regression similar to the previous assignment. The only changes are the datasets are built differently and the input vector will have different dimension. At the same amount of dataset as previous assignment, random degree value (0-360) will be inputted and the program will correctly identify if the sine value of the input degree is positive or negative. If positive, the output label y will be labelled 1 or otherwise 0.

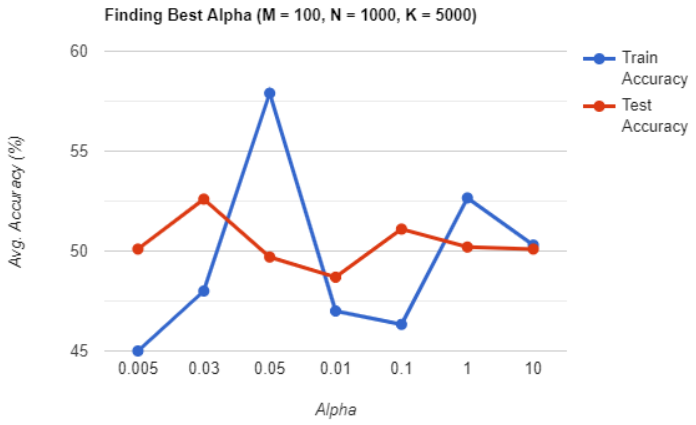
Estimated unknown function W & b

For our experiments, we will set the number of samples according to the assignment specifications (M=10000, N = 1000) and the number of iterations K will be set to 5000. After experimenting a few times with the model, we reached a conclusion where for every iteration, the values of W and b are increasing. The figure below shows the estimated function parameters W and b for this training.

```
[w1, b] = [-105.24273737911399, -0.015135976982106225]
Cost: 0.000000
[w1, b] = [-11.915964800425854, 157.51730619707882]
Cost: -1.000000
[w1, b] = [-1.628193029271926, 237.56532225650315]
Cost: 0.000000
[w1, b] = [-1.6281930291412527, 237.56532225650315]
Cost: 0.000000
[w1, b] = [-1.6281930290105795, 237.56532225650315]
Cost: 0.000000
[w1, b] = [-1.6281930288799062, 237.56532225650315]
Cost: 0.000000
[w1, b] = [-1.628193028749233, 237.56532225650315]
Cost: 0.000000
[w1, b] = [-1.6281930286185597, 237.56532225650315]
Cost: 0.000000
[w1, b] = [-1.6281930284878865, 237.56532225650315]
Cost: 0.000000
[w1, b] = [-1.6281930283572132, 237.56532225650315]
Cost: 0.000000
Cost with n test samples =  -4.8599999999776164
Accuracy for 'm' train samples: 60.0%
Accuracy for 'n' test samples: 51.4%
```

Empirically determined (best) hyper parameter, α

To find the alpha value, we will use the same test-and-run method in previous assignment and observe which one yields the highest average accuracy on the training and test samples. The graph below shows the result.



From the graph, we can conclude that when α=0.05, it gives the best accuracy for train accuracy. The problem compared to the previous assignment was, the margin between the train accuracy and test accuracy are more distorted and unstable. Observation also shows that the accuracy is not consistent either with margin about ~10% between tests compared to previous assignment which only has about ~3% margin difference.

Accuracy

	m=10, n=1000, K=5000	m=100, n=1000, K=5000	m=10000, n=1000, K=5000
Accuracy ('m' train samples)	60.0%	52.0%	49.55%
Accuracy ('n' test samples)	51.4%	49.6%	48.0%

	m=10000, n=1000, K=10	m=10000, n=1000, K=100	m=1000, n=1000, K=5000
Accuracy ('m' train set)	49.89%	50.29%	49.8%
Accuracy ('n' test samples)	49.7%	49.2%	49.8%

Discussion

From the accuracy table above, we can see that the accuracy for this dataset is way lesser than our previous assignment. Although the behavior of the result stays the same; as m increase, the accuracy gap between test and train samples decreases and while as K increase, the accuracy increases too. Now we will have to discuss the reason of why the accuracy for this dataset is too low. In the previous assignment, the input dataset was not altered much; it was only a matter of whether the sum of both number in input vector is positive or negative. Therefore, using backward propagation in the training algorithm can easily track down the original value of the input. But in this assignment, the input value has been converted to radians, and then the sine value was determined by using math.sin(). Therefore, it will be harder for the training algorithm to track back the input value by only using backward propagation. Thus, it results in much lower accuracy for this program.