CSE 353
HomeWork 4

JeongYoon Lee (114133199)

a) Introduction. Brief summary of what you think the assignment is about

This assignment is about classifying with various classification method (Linear Regression, Logistic Regression, Stochastic Gradient Descent).

Training dataset contains 40 training samples of dimension 3 in X.txt and Y.txt is ground truth binary labels.

First, I will find weight of linear regression algorithm, and will compare PLA(perceptron learning algorithm) model with W_linearRegression and W_0 (which is 0 vector).

Second, I will build logistic regression model and compare with W_logisticRegression and W_LinearRegression in logistic regression algorithm. In this model, there is also the other hyper parameter called "learning rate", so I can find the optimal learning rate and iteration number through tracking error rate for each model.

Lastly, I will extract K random data from original training dataset and running in logistic regression algorithm, and it is called Stochastic Gradient Descent(SGD) Logistic Regression. This also help to know some relationships between changing the number of training samples.

b) Method. Brief outline of your (algorithmic) approach

- Linear Regression

$$W\_LinearRegression = (X^T X)^{-1} X^T Y$$

$$h(x) = sign(W\_LinearRegression^T X)$$

If $sign(W\_LinearRegression^T X)$!= Label_groundTruth, this is error, so we can find error rate.

And then, try PLA algorithm with this W_LinearRegression, and find error rates and iteration times to build optimal model.

- Logistic Regression

$$W\_LogisticRegression(t+1) = W\_LogisticRegression(t) + \alpha \frac{1}{N} \sum_{n=1}^{N} sigmoid(-y_n * W\_LogisticRegression(t)\text{^}T * x_n)(y_n x_n)$$

$$h(x) = sign(W\_LogisticRegression^T X)$$

Try to update new W_LogisticRegression(t+1) until enough iteration or until error is small enough.

- SGD Logistic Regression

$$W\_LogisticRegressionSGD(t+1) = W\_LogisticRegressionSGD(t) + \alpha \frac{1}{K} \sum_{k=1}^{K} sigmoid(-y_k * W\_LogisticRegression(t)^{\wedge}T * x_k)(y_k x_k)$$

Try to do same thing with logistic regression in the previous problem, but extract random k data from given dataset, and used this as train dataset in this algorithm.

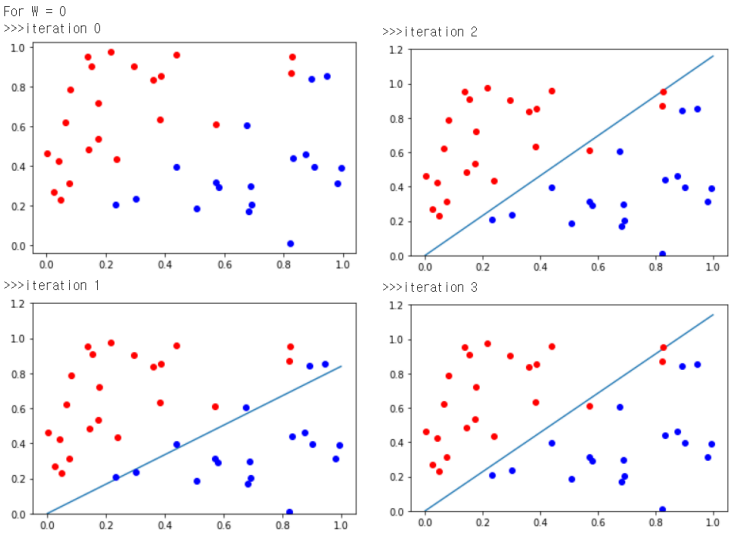
And then find relationship between different k's.

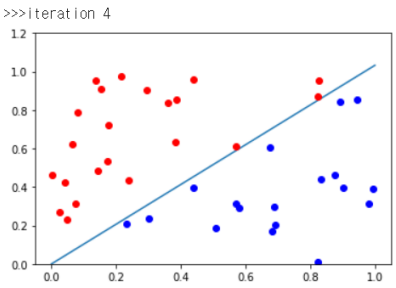
c) Experiments. Tables and/or pictures of intermediate and final results that convince us that the program does what you think it does.

1. Linear Regression

```
1. Linear Regression
W_linearRegression:  [-0.07882655  2.04885018 -1.79061198]
error rate :  0.025
```

>> (a) Find W_linearRegression and error rate



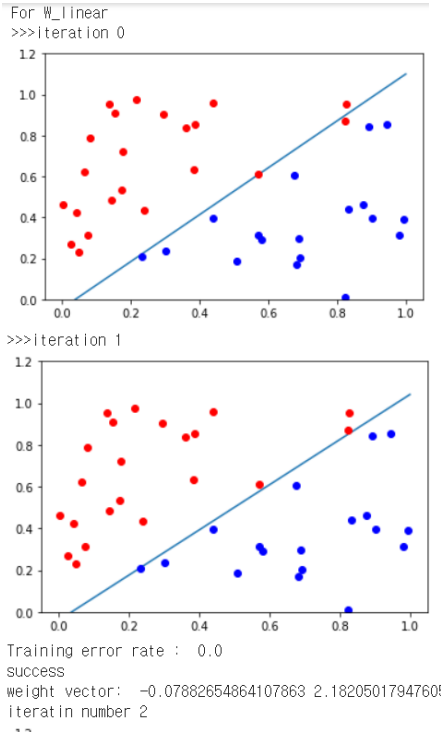



```
Training error rate :  0.0
success
weight vector:  0.0 3.5302999999999995 -3.4182
iteratin number 5
```

- Total 5 iterations

- Error rate : 0.0

>> [Linear Regression with W=0]



(b)

- Total 2 iterations

- Error rate : 0.0

>> [Linear Regression with W=W_linearRegression]

(c)
Iteration times with W_LinearRegression is much smaller than using with W = 0, since we W_inearRegression is much closer to optimal weight than W=0, so it reduce the iteration time to find optimal weight for this algorithm, but both can also reach to similar result for enough iteration, but to save cost, it is better to set proper initial weight before training.

2. Logistic Regression

- (a) , (b) Finding W_logisticRegression & finding error rate with W_logisticRegression

```
---Using W_LogisticRegression as the initialization---
learning rate : 0.3, Iteration : 100
error rate :  0.1
>>W_logisticRegression :  [-0.27532635  2.24845968 -1.80811357]
```

>>> Weight obtained with logistic regression algorithm is [-0.27532635 2.24845968 -1.80811357].
>>> Error rate for logistic regression to an initialization with the zero vector is 0.1

- Error rate when using W_linearRegression as the initialization in logistic regression

```
---Using W_LinearRgression as the initialization---
learning rate : 0.3, Iteration : 100
error rate :  0.05
```

>>> Weight obtained with logistic regression algorithm is [-0.27532635 2.24845968 -1.80811357].
>>>(c) Error rate for logistic regression to an initialization with the W_linearRegression is 0.05, and it is smaller than the model with initialization with the zero vector in logistic regression.

As we compare both models for same iteration number and same learning rate, we can find that model is better for using W_linearRegression than zero vector. We can know that the initial weight can affect to the result of logistic regression result.

```
---Using different learning rates---
>>> Learning rate : 0.001, >>>Iteratioin : 10, >>>Weight : W_logisticRegression
error rate :  0.1

---Using different learning rates---
>>> Learning rate : 0.001, >>>Iteratioin : 100, >>>Weight : W_logisticRegression
error rate :  0.1

---Using different learning rates---
>>> Learning rate : 0.001, >>>Iteratioin : 10000, >>>Weight : W_logisticRegression
error rate :  0.05
```

```
---Using different learning rates---
>>> Learning rate : 0.7, >>>Iteratioin : 10, >>>Weight : W_logisticRegression
error rate :  0.075

---Using different learning rates---
>>> Learning rate : 0.7, >>>Iteratioin : 100, >>>Weight : W_logisticRegression
error rate :  0.05

---Using different learning rates---
>>> Learning rate : 0.7, >>>Iteratioin : 1000, >>>Weight : W_logisticRegression
error rate :  0.0
```

[Result of Small learning rate (0.001)] [Result of Large learning rate (0.7)]

(d) We can find that when the learning rate is small, the error rate does not decrease any more even if the iteration is large enough, and I think this is because of underfitting since the learning rate is too small.

So I tried with large learning rate to compare with smaller one, and with large learning rate, the error rate converge to 0 when we do enough iteration.

Usually, if the learning rate is too big, overfitting can happen, but in this dataset, the error rate tends to converge to zero with large learning rate. But if I should train with more large various dataset, I should check the error rate (loss) is converge or not when scaling learning rate and iteration to prevent overfitting.

Also for this experiment, I used initial weight as weight from logistic Regression that I got before, but we can still use with other weight like zero vector or weight from linear regression.

3. SGD Logistic Regression

(a), (b), (c)

```
---Using W_logisticRegression as the initialization---
Learning rate : 0.7  , Iteration : 100  , size of sample :  30
randomlist (random index list):  [19 17 30  4 26 38 13 31 11 22 23 37 36 18 21 39  5 10  1  0 16  2  3 20
 29 32 14  7 12 33]
Random sample size :  30
error rate :  0.06666666666666667
>>> W_logisticRegressionSGD :  [-0.46242557  3.47475517 -2.46709936]
```

>> Random size K : 30

```
---Using W_logisticRegression as the initialization---
Learning rate : 0.7  , Iteration : 100  , size of sample :  10
randomlist (random index list):  [31  7 33 10 37 13 19 27 22 38]
Random sample size :  10
error rate :  0.1
>>> W_logisticRegressionSGD :  [-0.84650309  1.56049848 -1.26938278]
```

>> Random size K : 10

```
---Using W_logisticRegression as the initialization---
Learning rate : 0.7  , Iteration : 100  , size of sample :  5
randomlist (random index list):  [22 29  2 16 37]
Random sample size :  5
error rate :  0.4
>>> W_logisticRegressionSGD :  [-0.11709129  0.22118673 -0.65053112]
```

>> Random size K : 5

The error rate tends to decrease when we have more number of sample data, but it doesn't have clear causation between K and error rate. It is because our data doesn't have big error rate with any number of sample data, and also our given data is too small, but I can see that this still have small tendency that the large number of sample have lower error rate.

I think computational cost will decrease if K gets smaller value, but it should be used when the size of sample data is large enough, and when extracting k samples don't affect to find the optimal model. Also, it should not be biased when extracting data. So in this case, I think extracting random K dataset does not have meaning since dataset is too small and with this way, the extracted K dataset can be biased. So I want to try the other way to extract data, not randomly if I want to extract part of sample training data.

d) **Discussions and Conclusions. Any design decisions you had to make and your experimental observations. What do you observe about the behavior of your program when you run it? Does it seem to work the way you think it should? Play around a little with different setting to see what happens. Note, your open-ended exploration is highly valued.**

As a result of substituting W obtained by linear regression into the PLA algorithm, it was possible to find the optimal classifier for the given data. The most interesting thing was keep updating the weight value in logistic regression, and it was also interesting to continue to find the optimal w value using the gradient descent method.

Also, there are many hyperparameters that can be adjusted, such as learning rate and iteration times, so I think I learned a better understanding of how to train data. I think that logistic regression can quickly find the best classifier by scaling even for non-linear separable data.

Also, in the SGD method, we could not find a big difference because there were not many samples given this time, but it seems to be useful for training using batches or validation process later if we can extract not biased dataset. Also later, instead of binary data, I want to classify datasets that have multi-classes too.