INF552 Machine Learning for Data Informatics HW4

1. Contribution:

a. Shih-Yu Lai (50%):

Email: shihyula@usc.edu

USC-ID: 7183984563

Implement Linear Regression, Logistic Regression code and write the ReadMe.

b. Shiuan-Chin Huang (50%):

Email: shiuanch@usc.edu

USC-ID: 9815633745

Implement Perceptron Learning, Pocket code and write the ReadMe.

2. Steps of the algorithm and data structure

In perceptron-learning.py, we follow the classifying steps we learned in class and store the data in numpy array:

Step1: start with a random weight

Step2: Pick misclassified point i

Step3: w' = w + y*x

Repeat step2 and step3 until all points are classified correctly.

In Pocket.py, the steps are similar to those of perceptron learning, but it limits the maximum iteration, chooses a random data to see if it's correctly classified so that we can adjust the weight and finally, returns the weight which leads to minimum misclassified points because some data might not be linearly separable.

In linearRegression.py, we follow the below steps to implement:

Step1: read data from txt and store them into numpy array. (XY and Z)

Step2: get the row and column number, then calculate the weight with this formula W = (DD^T)^(-1) * DY

In logisticRegression.py, we follow the below steps to implement:

Step1: read data from txt and store them into numpy array. (X and Y)

Step2: let the iteration be 7000 and the initial learning rate is 0.001.

Step3: random choose the weight can call the train function.

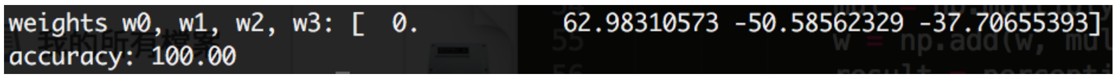
Step4: calculate the sigmoid function and gradient descent and update the values.

Step5: predict data and print the outcome.

3. Our Outcome

Perceptron:

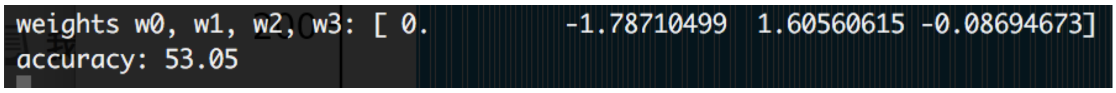
-weights and accuracy after the final iteration



```
weights w0, w1, w2, w3: [  0.         62.98310573 -50.58562329 -37.70655393]
accuracy: 100.00
```

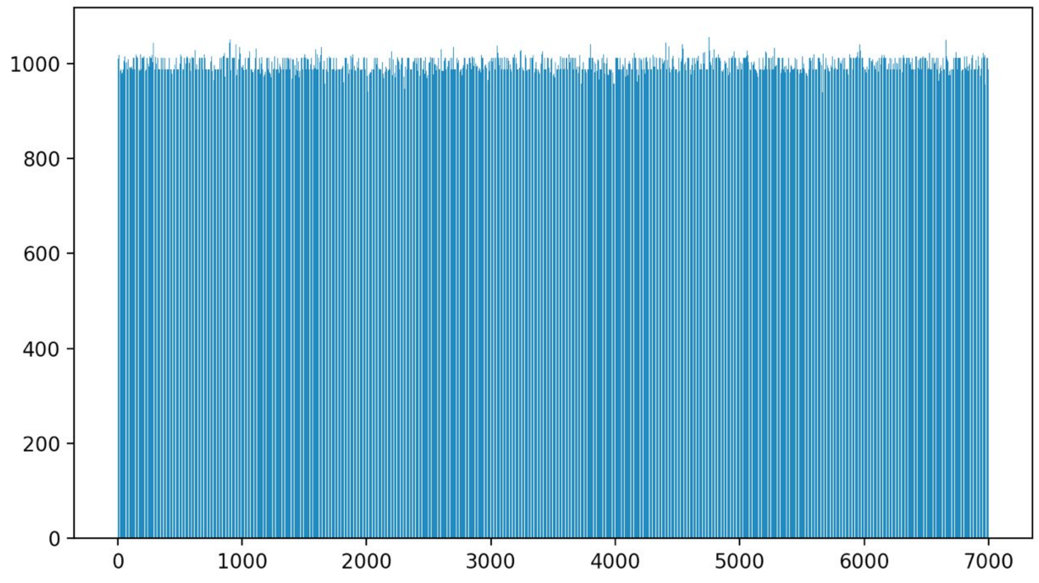
Pocket:

-weights and accuracy after the final iteration



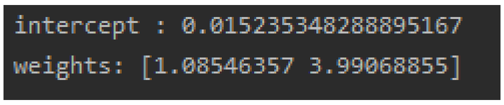
```
weights w0, w1, w2, w3: [ 0.         -1.78710499  1.60560615 -0.08694673]
accuracy: 53.05
```

-number of misclassified points against the number of iterations of the algorithm



Linear Regression:

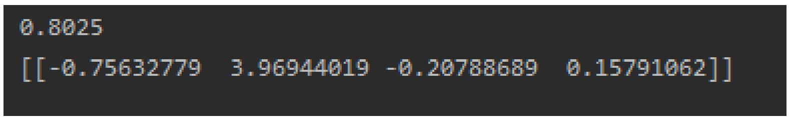
-intercept and weights



```
intercept : 0.015235348288895167
weights: [1.08546357 3.99068855]
```

Logistic Regression:

-accuracy and weights



```
0.8025
[[-0.75632779  3.96944019 -0.20788689  0.15791062]]
```

4. Challenge

The concept of Perceptron Learning and Pocket Algorithms are not difficult, but it takes some time to fully understand the mathematical symbols and equations.

The challenge we met in the linear regression is hard to find the right function in the numpy to help us calculate values. However, in the logistic regression, the challenge is how to choose the random weight at first to let the predict higher.

5. Optimization

Pocket:

Pocket.py is an optimization for perceptron learning because it sets a maximum iteration of 7000 to prevent endless loop (when the data is not linearly separable) and it tracks the best weight which causes minimum misclassified points.

Perceptron:

In sklearn-perceptronLearning.py, we set the maximum iteration to be 1300, which results in a 100% accuracy of classification without too many extra iterations.

Linear Regression:

In the previous homework, we calculate matrix by ourselves, but this time, we use functions in numpy which is faster than before.

Logistic Regression:

I use three different ways to find the random weight. The first method is the just use random function, and the predict accuracy after I run the code 100 times is 0.46 to 0.9625. The second method is based on random function and multiply 10, and the predict accuracy after I run the code 100 times is 0.653 to 1. The last method is He Initialization, and the predict accuracy after I run the code 100 times is 0.3655 to 0.9025. So I finally choose the second method which has the highest accuracy.

6. Software Familiarization

Sklearn.linear_model.Perceptron:

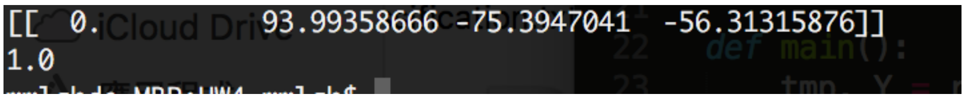
There are some parameters we can adjust in this library, so that makes the classification more efficient and in case the data is not linearly separable.

fit_intercept: the default is True, but here we set it to be False because we assume the data is already centered.

max_iter: in case the data is not linearly separable, default=1000, here we set it to be 1300 and all data can be correctly classified in these iterations.

shuffle: default=True but here we set it to be False because we don't have to shuffle the training data.

-result of sklearn.linear_model.Perceptron



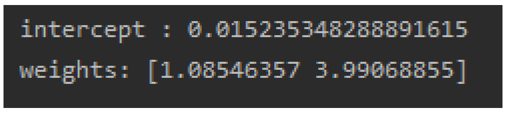
```
[[  0.          93.99358666 -75.3947041   -56.31315876]]
1.0
```

Sklearn.linear_model.LinearRegression:

coef_: array of shape. Estimated coefficients for the linear regression problem.

intercept_ : float or array of shape of (n_targets,). Independent term in the linear model. Set to 0.0 if fit_intercept = False.

-result of sklearn.linear_model.LinearRegression



```
intercept : 0.015235348288891615
weights: [1.08546357 3.99068855]
```

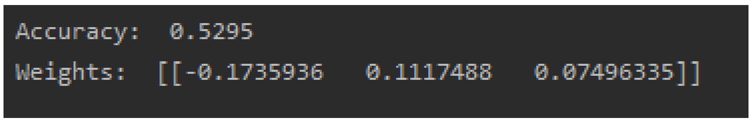
We got the same intercept (intercept_) and weights (coef_) with our code.

Sklearn.linear_model.LogisticRegression:

fit(self, X, y[, sample_weight]): fit the model according to the given training data.

predict(self, X): predict class labels for samples in X.

-result of sklearn.linear_model.LogisticRegression



```
Accuracy:  0.5295
Weights:  [[-0.1735936   0.1117488   0.07496335]]
```

7. Applications

Linear classification(Perceptron):

1. Machine translation software:

   Using each sentence in the corpus as dataset to train the machine, so that it can translate a source text in certain language into another.

2. Speech recognition:

   Using the speech signals as data set for multilayer perceptron to do classifications.

3. Image recognition:

   Every image can be represented as a two-dimensional array of its pixels for multilayer perceptron to do image recognition.

Linear Regression:

1. House price predict
2. Stock predict
3. Salary predict

Every problem is linear can use linear regression.

Logistic Regression:

1. NLP
2. Image recognition

Every problem need to classify to two groups can use logistic regression.