INF552 MACHINE LEANING FOR DATA INFOMATICS

ASSIGNMENT 4: PLA/ LOGIT REGRESSION/ OLS

PART 1: IMPLEMENTATION

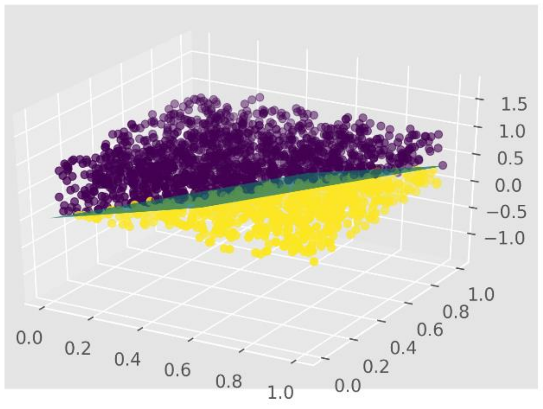
All codes are written in Python 3.

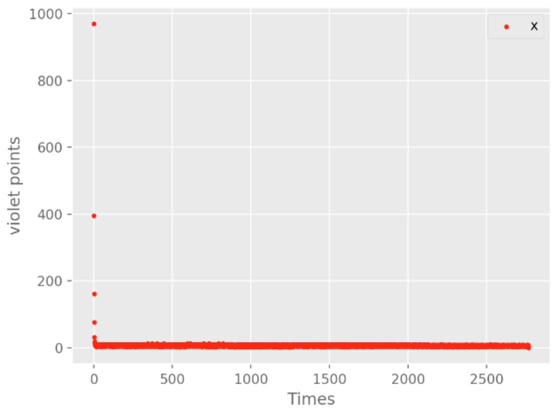
In this assignment, we produced several linear models for both classification and regression. Firstly, in Perceptron Learning Algorithm (PLA), we applied an idea of decreasing learning rate to reduce running time. And the running time significantly dropped. On the other hand, we can't avoid Pocketing Learning from iterating up to 7000 times since the data isn't linearly separable. However, with the probability and overall score concept introduced in Logistic Regression, we can approach the best solution using gradient descent. Secondly, we also implemented a Linear Regression Model using Ordinary Least Squares method.

PERCEPTRON LEARNING (PERCEPTRON.PY AND PERCEPTRONORDINARY.PY)

RESULT

The weight factor of perceptron [w0, w1, w2, w3] = [-0.000261, 0.159829, -0.127846, -0.095625]





THE DATA STRUCTURES USED AND DESCRIPTION

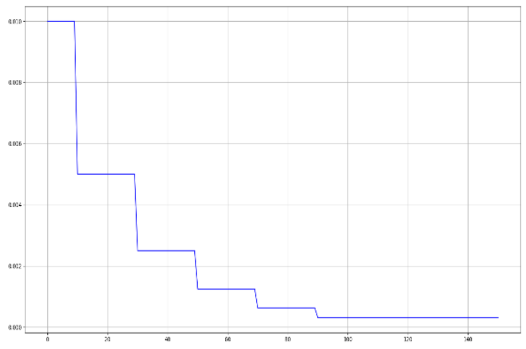
I used pandas.DataFrame as the data structure.

The algorithm iterates through each data point, and updates w (the weight vector) once violation appears. This process will stop only if there's no more violated constraints. However, the number of total violation doesn't always decrease. It's because the weight vector doesn't change slightly. Instead, by adding the element consisting of learning rate and data point, the vector often goes back and forth around the fitting solution. And this sometimes put the vector more far away from the solution.

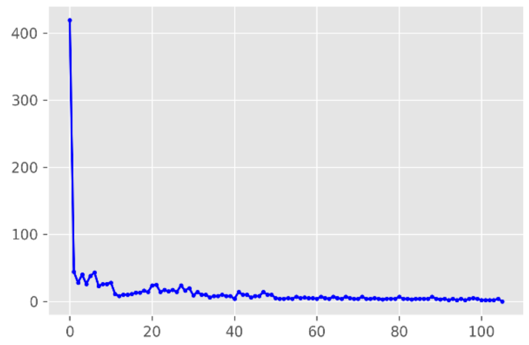
OPTIMIZATION

As mentioned above, the vector is updated with elements consisting of learning rate and data points. Thus, one of the reasons why the weight vector oscillates is that, the learning rate is too big. A big learning rate results in a big step, results in inaccuracy. Hence, I developed a method to decrease learning rate like a downstairs.

As a result, the iteration time for PLA reduced from around 3000 to 100.



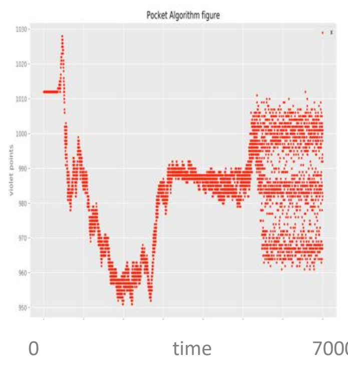
Alpha

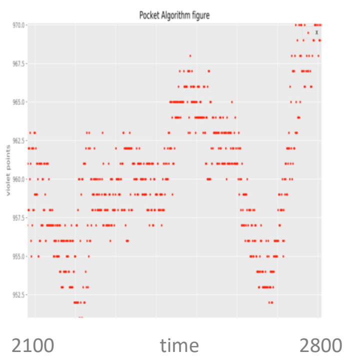


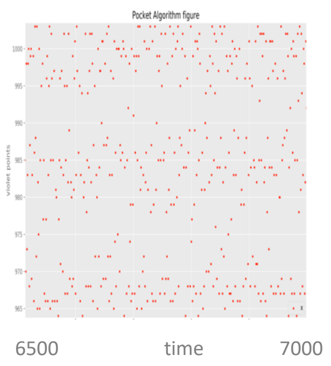
Optimized PLA

POCKET LEARNING (POCKET.PY)

RESULT





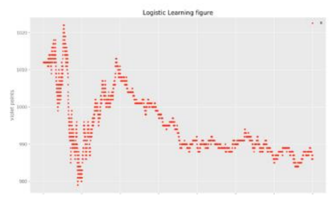


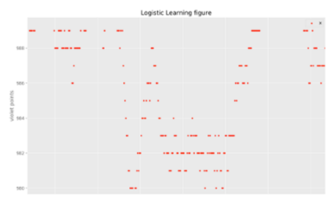
DESCRIPTION

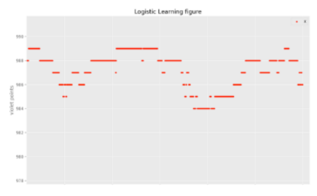
Since there is no line could correctly separate the points into two clusters (label 1 and label -1), the algorithm would be iterated 7000 times to stop. From the figure, we can notice that the violation numbers don't decrease to zero. Besides, we couldn't even end at the local minimum point. However, this happens in the pocket algorithm.

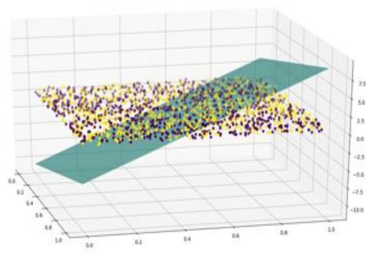
LOGISTICE LEARNING (LOGISTICREGRESSION.PY)

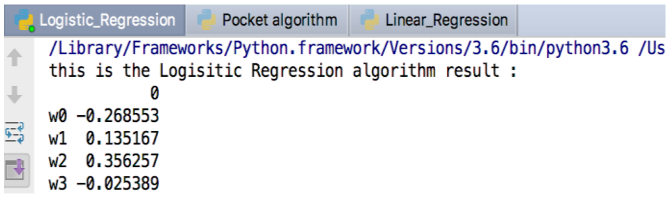
RESULT











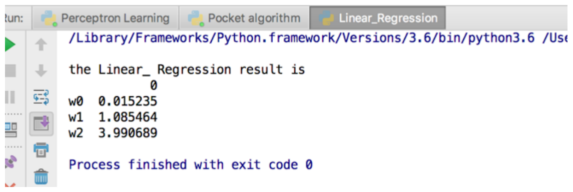
```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Us
this is the Logisitic Regression algorithm result :
             0
w0 -0.268553
w1  0.135167
w2  0.356257
w3 -0.025389
```

DESCRIPTION

The w is used to calculate the possibility of whether one point belongs to 1 or -1. Under my algorithm, if the possibility is greater than 0.5, this point is classified as 1. On the contrary, if the possibility is smaller than 0.5, this point is classified as -1. With the 7000 iterations, I record the violation numbers in each iteration and find out the violation figure.

LINEAR LEARNING (LINEARREGRESSION.PY)

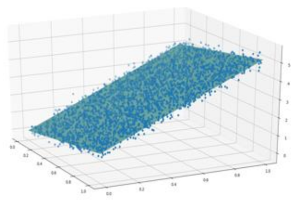
RESULT



```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Use

the Linear_ Regression result is
            0
w0  0.015235
w1  1.085464
w2  3.990689

Process finished with exit code 0
```



DESCRIPTION

Linear regression is calculated with the formula: $W = (DD^T)^{-1}Dy$

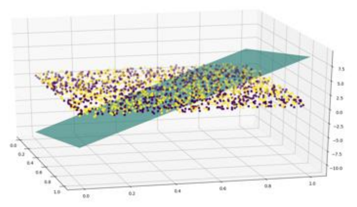
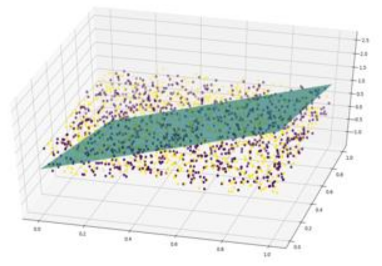
Where D is consisting of independent X and Y variables, and y stands for the dependent Z variable.

PART 2: COMPARISON

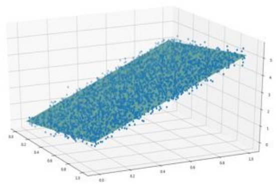
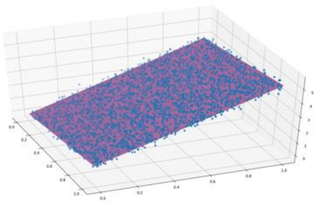
PERCEPTRON LEARNING ALGORITHM

	Perceptron.py	PerceptronOrdinary.py	sklearn.linear_model.Perceptron	
Epoch	126 epochs	Around 3000 epochs	5 epochs	120 epochs
Weight Vector	[0.244, -0.195, -0.146, -0.000721]	[0.159829, -0.127846, -0.095625, -0.000261]	[0.17713184 -0.12792947 -0.10781926, 0.]	[0.504128, -0.395141, -0.296163, 0.]
Scaled Weight Vector	[1, -0.7991803, -0.5983607, -0.0029549]	[1, -0.7998924, -0.5982957, -0.001633]	[1, -0.7222274, -0.608695, 0.]	[1, -0.7838109, -0.5874758, 0.]
Score	100%	100%	97.55%	98.65%
Conclusion	The Epoch outcomes that we designed are greater than scikit-learn package (when the EPOCH number of package is 5) Our scaled weight vectors are very similar, but differ from the package. However, we noticed that if we let the package execute around 120 times, the score that of we designed is better than the score of package (100%> 98.65%). Besides, we believed one of the reasons why the package could end its execution faster is that, it may sacrifice the accuracy to measure the best solution. Therefore, it could stop and obtain an answer much rapidly.			

LOGISTIC REGRESSION

	LogisticRegression.py	sklearn.linear_model.LogisticRegression
Weight Vector	[0.135167, 0.356257, -0.025389, -0.268553]	[-0.17769439, 0.1144508, 0.0767, -0.0157501]
Output Result		
Conclusion	To be honest, we are not sure about the reason why our weight vector is different from the package. Firstly, we noticed that neither our designation nor the package can successfully classify data because of its high entropy. Furthermore, we believed that the package sacrificed accuracy in order to get results faster. However, our designation takes every constraint into consideration. Therefore, based on the entropy of data and trading-off on time and accuracy, we have distinct outcomes.	

LINEAR REGRESSION

	LinearRegression.py	sklearn.linear_model.LinearRegression
Operation Time	0.016 sec	0.006 sec
Weight Vector	[1.085464, 3.990689, 0.015235]	[1.08546357, 3.99068855, 0.]
Output Result		
Conclusion	Basically, our weight vector are the same as the package. However, the operation time of package is much faster than ours.	

PART 3: APPLICATION

PERCEPTRON LEARNING ALGORITHM (PLA)/ POCKET LEARNING:

PLA could be used to decide whether an applicant could apply for a credit card. Banks could take each applicant as a vector x, which is a combination of applicant's various dimension features such as applicant's age, annual salary, length of service and so on. Each dimension would impose different positive or negative impacts on the decision whether banks would approve the credit card request. In this case, we can integrate these dimensions and give applicants a score. If the score exceeds a certain threshold, we can know an applicant could get the credit card. Otherwise, an applicant could not get the card.

LOGISTIC REGRESSION:

The output of Logistic regression can take only two values, "0", "1", which represent two outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients. Besides, Logistic regression may be used to predict whether a patient has a given disease, based on observed characteristics of the patients such as their age, sex, results of various blood tests and so on. Another example might be to predict whether an American voter will vote Democratic or Republican, based on age, income, sex, race, state of residence, votes in previous elections.

LINEAR REGRESSION:

Linear regression refers to a model that can show relationship among variables and how one can impact others. In essence, it involves showing how the variation in the “dependent variable” can be captured by change in the “independent variables”.

Take assessing risk in financial services or insurance domain as an example to illustrate Linear Regression, a car insurance company might conduct a linear regression to come up with a suggested premium table using predicted claims to Insured Declared Value ratio. The risk can be assessed based on the attributes of the car, driver information or demographics. The results of such an analysis might guide important business decisions.

GROUP MEMBER

Jung-Kang, Su	2389753352	Research for application; package comparison; report
Yuhsi Chou	6048573191	Create Perceptron.py, Visualization.py; report
Zhang Mujie	2621330761	Implement Pocket, Linear Regression, Logistic Regression

REFERENCE

http://analyticstraining.com/2014/popular-applications-of-linear-regression-for-businesses/

https://en.wikipedia.org/wiki/Logistic_regression