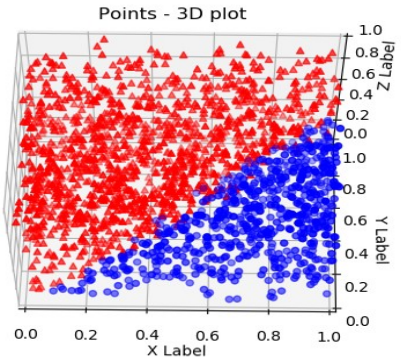
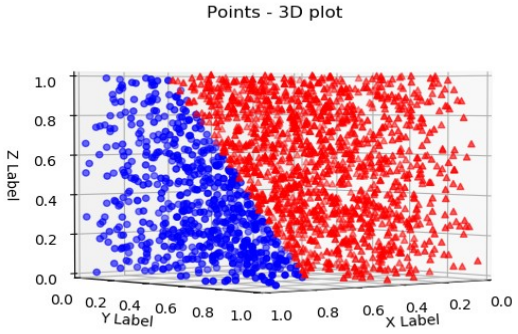
Name: Kushal Sharma
Email: kushals@usc.edu
Course: INF 552 Summer 2020
Assignment: HomeWork4 (Linear Classification, Linear Regression & Logistic Regression)

Linear Classification:

We are considering the data points to be linearly separable in the 3D plane, which is also proven by 3d plot of points and their color coded labels in the figure below. Red for negative labels (-1) and blue for positive labels (+1).





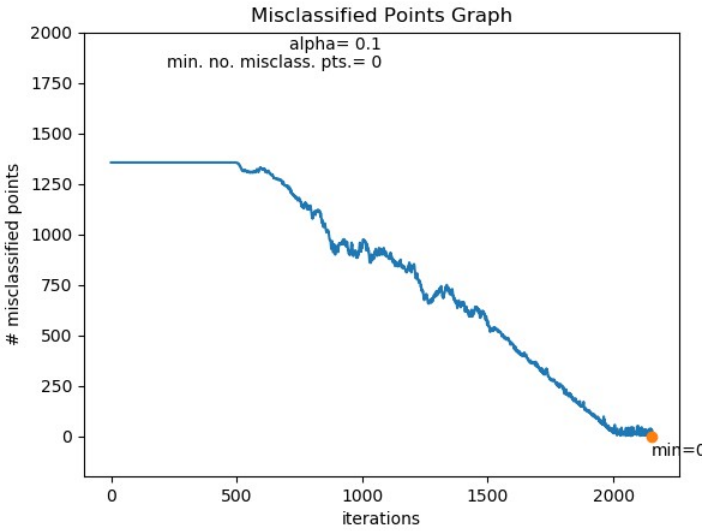
There are 3 hyper-parameters – weights, alpha(learning rate), and number of iterations, which can be tuned to obtain the higher accuracy. Multiple combinations of value of these hyper-parameters were tried in runs of the program to obtain the best results.
Weights are uniformly randomized over wide range of -100 to +100 so that it doesn't stuck at same local minima with every random restart.
Alpha, the learning rate is selected neither at high number so that it doesn't overshoot the optimal value of weights nor at very low number so that it doesn't stuck at local minima.
Number of iteration should be chosen so that program completes within time limit and doesn't run for too long without better results.
Since weights are being updated randomly, the number of violated constraints/miss-classified points are increasing or decreasing erratically, eventually settling at lowest value.

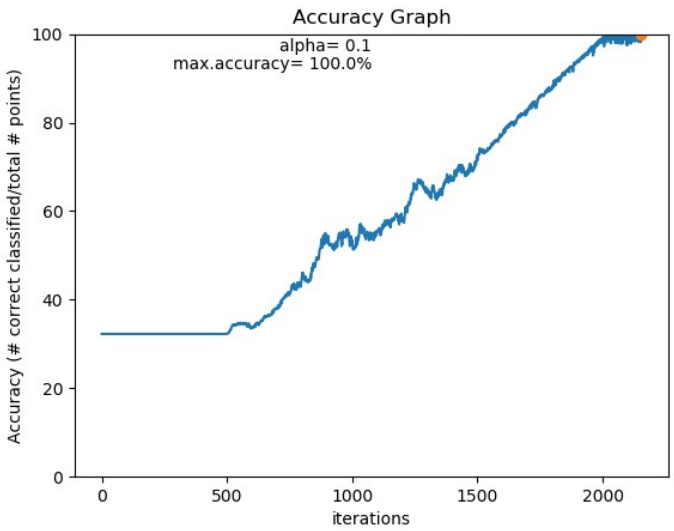
Following steps are taken in the program:

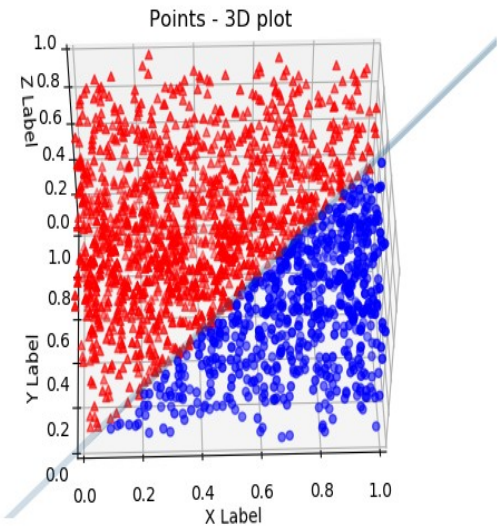
1) Initialize the weights randomly and find the predicted value by taking dot product of weights and every point.
2) Compare the predicted and expected value to find the number of miss-classified points.
3) Pick any one miss-classified point to update the weights
4) Repeat the step 1 – step 3 until no miss-classified points remain.
5) Report the weights.

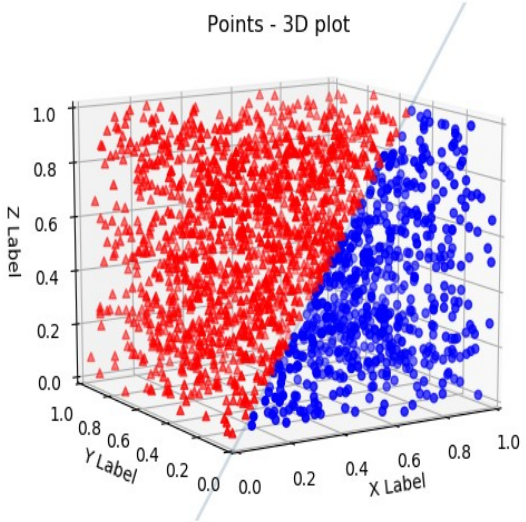
There can be multiple values of weights which can represent a hyperplane/line to correctly classified the points.

alpha(learning rate) = 0.1	Final Iteration	Initial Iteration
Iteration	2155	0
# of Miss-classified points	0	1355
Weight	[0.02383649, 9.53825701, -7.69832774, -5.72126632]	[51.72383649, 27.5871891, 68.81369796, 24.14885514]
Accuracy	100%	32.25%



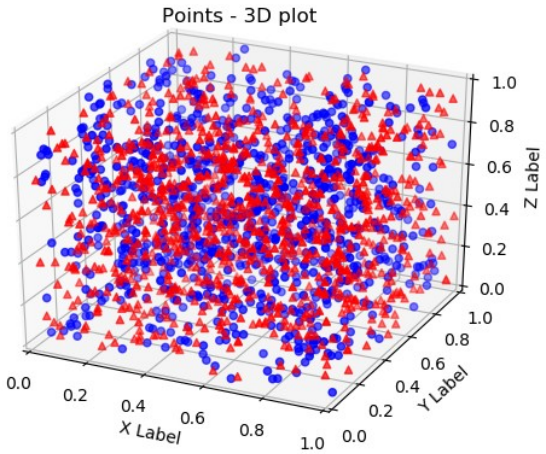


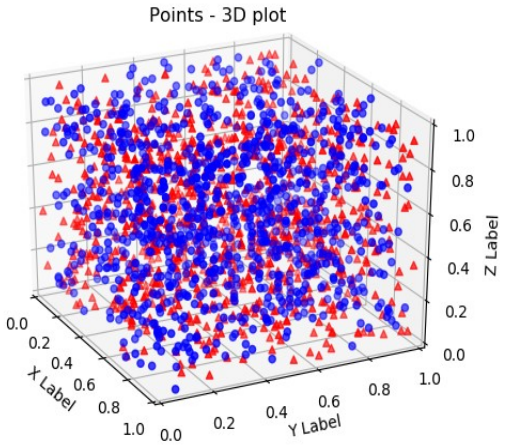




Linear Classification with Pocket algorithm (limited iteration):

For the given set of 3D points, it is evident from the 3D plot below that the points are not linearly separable in 3 dimension.

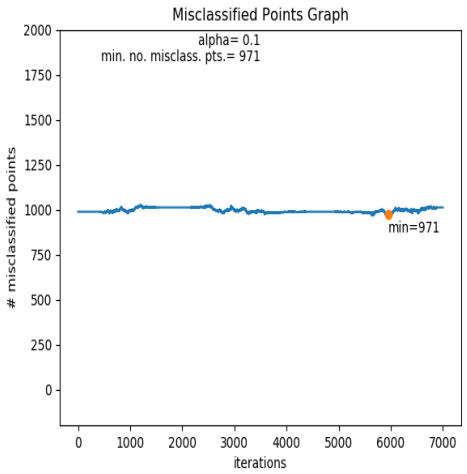


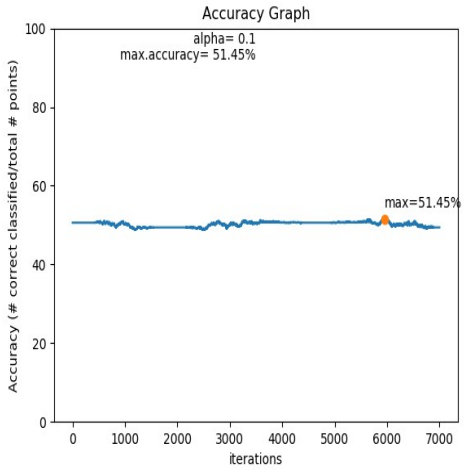


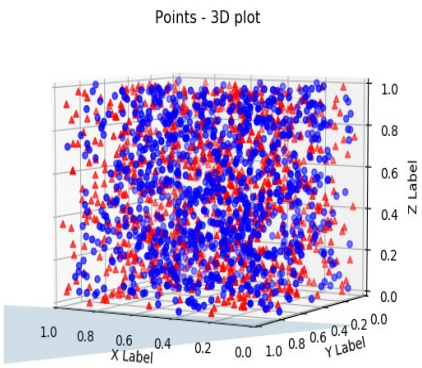
Pocket algorithm is applied to run the program till limited no. of iterations given as 7000 and report the weights at the iteration with minimum number of miss-classified points.

Initial weight = [-92.20438616 -45.26139291 -79.28013827 35.50309874]

alpha (learning rate) = 0.1	Final Iteration	Iteration with min no. of miss-classified points
Iteration	7000	5949
# of Miss-classified points	1012	971
Weight	[7.39561384, 1.5442752 , -4.29045042, 44.77726102]	[-15.70438616, -11.67060349, -20.18414482, 39.05491381]
Accuracy	49.4%	51.45%



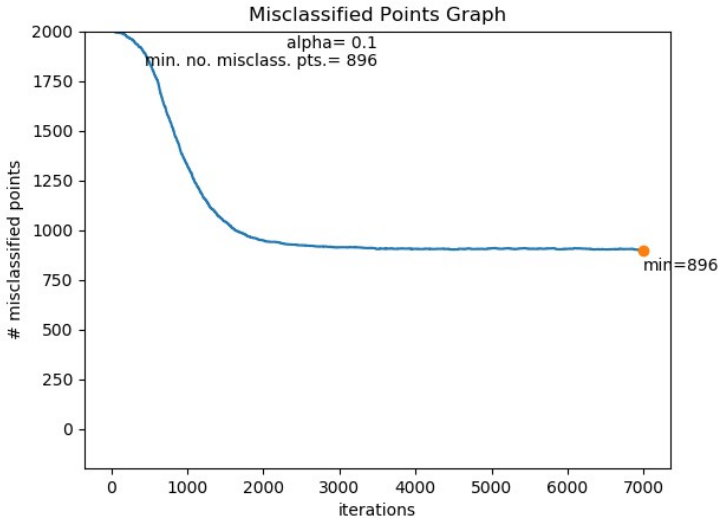


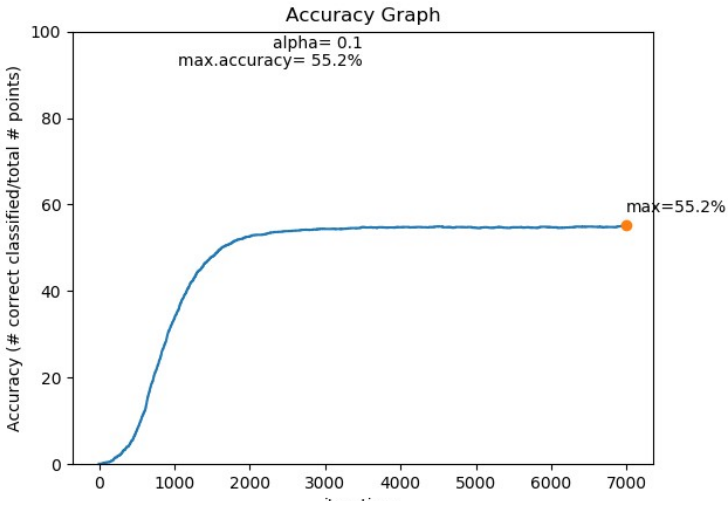


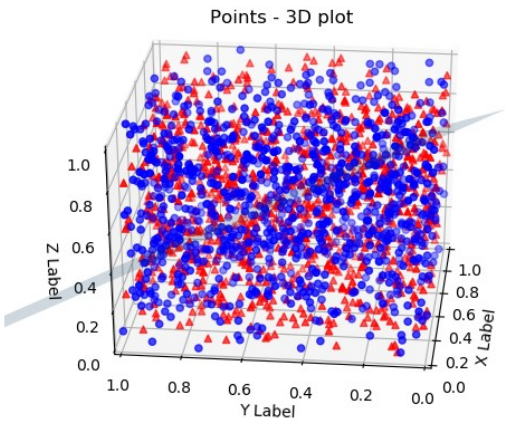
Logistic Regression (limited iterations):

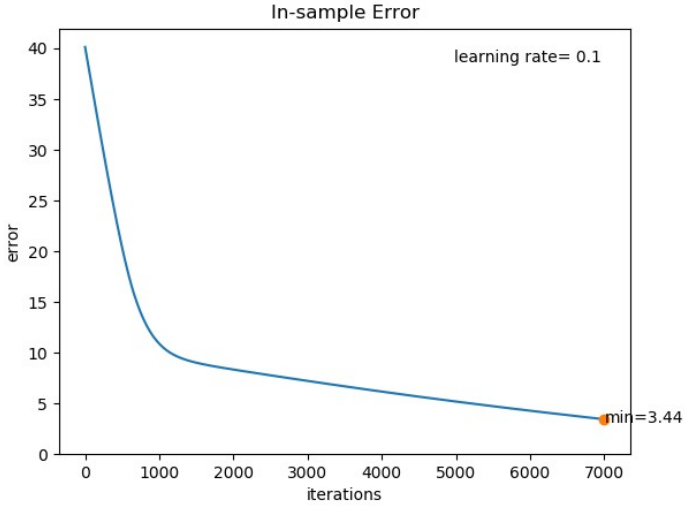
Score can be calculated for each point using sigmoid function which would indicate probability of the point to represent positive label. We can use logistic regression as classifier for positive label if probability value is greater than or equal to 0.5, and negative label if probability is less than 0.5. Logistic regression was applied on same non-linearly separable data-set as in case of pocket algorithm. We can see the rate of number of miss-classified points are changing erratically up to the point of stagnation. Similar pattern with accuracy graph as well, however in-sample error consistently decrease(better) because we intend to minimize the in-sample error with every iteration. That is our main objective function.. Here the learning rate is 0.1. 3D plot of points also show the plane passing through the points. In this case, we have continuous improvement in the results, though at very low rate at the last of iterations. Thus the best results are obtained at final iteration.
Initial weights = [8.82736182 -29.22274782 -53.48353554 -97.61018207]

alpha(learning rate) = 0.1	Final Iteration	Iteration with min no. of miss-classified points
Iteration	7000	7000
# of Miss-classified points	896	896
Weight	[24.53259242, -10.66346888, -15.4912522 , -21.29391733]	[24.53259242, -10.66346888, -15.4912522 , -21.29391733]
Accuracy	55.2%	55.2%









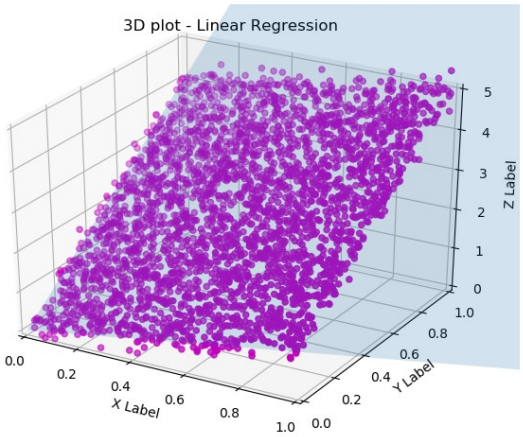
Linear Regression:

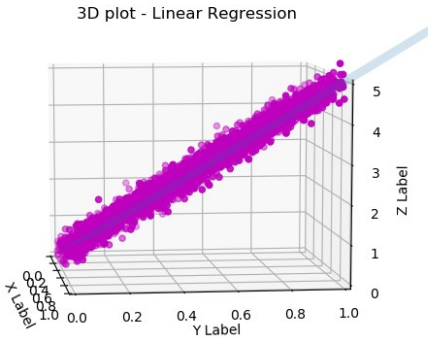
We know the weight value can be determined by simply matrix calculation than running the program in iterations. It is given by following formula:

$$W = (D^T.D)^{-1}.D^T.y$$

where D is the matrix of points of dimension d+1 * n
d is the dimension of points, 2 in our case (x, y), n is the number of points, 2000 in our case

Weights	Root Mean Square Error
[0.01523535 1.08546357 3.99068855]	0.19830914827213394





Data Structure:

(1) Pandas dataframes are used to read from the input files.
(2) Dataframe is converted to Numpy array to perform major linear algebra and matrix calculations.
(3) List is also used to append or concatenate the arrays together.
(4) Dictionary is used to store the results by each iteration which later help plot the graph.

Code-Level optimization:

(1) From the list of miss-classified points, random point is chosen and store in dictionary for that iteration which is later used to update the weights in linear classification.
(2) Score and gradient to updates weights are calculated in same for loop in logistic regression.

Challenges:

(1) Due to a bug in program, Linear classification program was not converging to 0 number of miss-classified points even till 100,000 iterations. After spending couple of days testing it, found a list structure which was not getting initialized properly for every iteration. Thus proper initialization of the variable is very important to learn.
(2) Plotting the 3D points along with the hyperplane which would go through those points to classify positive and negative labels was bit challenging to code.

Part 2 – Software Familiarization :

Linear Classifier using Sklearn:

We can use sklearn.linear_model.SGDClassifier to fit the linear classifier model for the given points and obtain the weights. It is observed that with default parameters of SGDClassifier(), the accuracy of correct classification of points lie between 95% - 99.5% which is not better than our program which can obtain 100% accuracy. However, with tuning of few parameters of SGDClassifier() like alpha, max_iter, n_iter_no_change, learning_rate etc., accuracy can be improved.

Linear Classifier Sklearn:
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
l1_ratio=0.15, learning_rate='optimal', loss='hinge',
max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l2',
power_t=0.5, random_state=None, shuffle=True, tol=0.001,
validation_fraction=0.1, verbose=0, warm_start=False)
Weights: [-0.46529151 17.81376358 -14.05040289 -10.24731023]
Miss-classified points: 18
Accuray: 99.1 %

Logistic Regression using Sklearn:

We can use sklearn.linear_model.LogisticRegression to fit the logistic regression model for the given points and obtain the weights. It is observed that with default parameters of LogisticRegression(), the accuracy of correct classification of points is 52.95%, which is not better than our program which can obtain upto more than 55% correct classification. However, with tuning of few parameters of LogisticRegression() like max_iter etc, accuracy can be imporved.

Logistic Regression Sklearn:
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
warm_start=False)
Score: 0.5295
Weights: [-0.03130434 -0.1735936 0.1117488 0.07496335]
Miss-classified points: 941
Accuray: 52.95 %

Linear Regression using Sklearn:

We can use sklearn.linear_model.LinearRegression to fit the linear regression model for the given points and obtain the weights. The result from sklearn and our program for the weights are very close. However, root mean square error of our program is lesser(better) than sklearn.

Linear Regression Sklearn:
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
Weights: [0. 1.08546357 3.99068855]
Root Mean Square Error: 0.1988935245952035

Part 3 - Applications:

Linear Classification:

1. Email Spam – Based on domain knowledge, set of few most commonly occurring words and punctuation marks in valid spam emails are pre-selected which are then check in every email-message. For new email, text sequence is analyzed and relative frequency of these words are computed.

2. Image Segmentation – For grayscale images, pixels intensity can be represented with number 0 to 255 in a matrix. For color images, which are represented by RGB value, 3 such matrices of pixel intensity ranging from 0 to 255 are there. Images are segmented based on the pixel intensity. Large pictures are also divided into block of pixels to reduce the no. of features.

3. DNA Sequence Classification – Each genome is made up of DNA sequences and each DNA segment has specific biological functions. However there are DNA segments which are non-coding, i.e. they do not have any biological function (or their functionalities are not yet known). One problem in DNA sequencing is to label the sampled segments as coding or non-coding (with a biological function or without). The raw DNA data comprises sequences of letters, e.g., A, C, G, T for each of the DNA sequences.

Logistic Regression:

Basically, linear regression is applicable to all the real world applications where linear classifier is applicable. Only difference is that logistic regression gives us likelihood of a data point belonging to particular class as probability value between 0 and 1.

1. Marketing – Predict if a given user will buy a specific product or not.
2. Geographic Image Processing
3. Healthcare : Analyzing a group of over million people for myocardial infarction within a period of 10 years is an application area of logistic regression.
4. Rain Forecasting

Linear Regression:

1. Financial forecasting
2. Software cost prediction, software effort prediction and software quality assurance.
3. Restructuring of the budget : Organization or Country
4. Crime Data Mining : Predicting the crime rate of a states based on drug usage, number of gangs, human trafficking, and Killings.
5. Credit Scoring
6. Revenue prediction