**Programming Assignment 4: Linear Regression, Linear Classification, and Logistic Regression**

Instructor: Dr. Satish Kumar Thittamaranahalli

Group Member: Yu Hou; Haoteng Tang

In this assignment, we will implement Linear Regression, Linear Classification, and Logistic Regression, these three algorithms. We have two datasets. In the first one, we have 2000 points with 1 or -1 as the class each point belongs, but the 2000 points have two different versions. In the second one, we have 3000 points plotted with some kinds of rules. In figure 1 figure 2, and figure3 shown as these two different datasets.

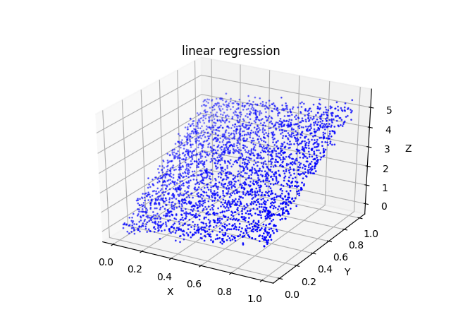
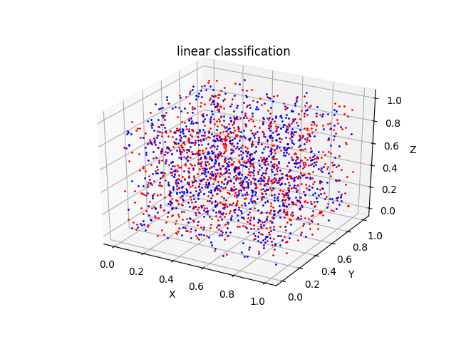
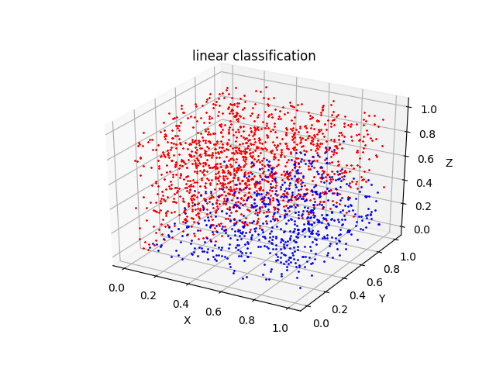


Figure 1 Database 1 version 1 Figure 2 Database 1 version 2 Figure 3 Database2

**1 Procedure of implementing algorithm (Yu Hou)**

(1) procedure

**Perceptron learning algorithm:**

Check every constrain, count the number of unsatisfied constrains

Use unsatisfied data to modify the coefficients

When constrain is zero, stop while.

**Pocket algorithm:**

Check every constrain, count the number of unsatisfied constrains, record the least number of unsatisfied constrains

Use unsatisfied data to modify the coefficients

Iterate the code in 7000 times. Print the best coefficients and the least number.

**Logistic Regression:**

Check every constrain. Use unsatisfied data to modify the coefficients

Iterate the code in 7000 times. Print the best coefficients.

**Linear Regression:**

Input the data, use formula to compute the coefficients.

Draw the formula in 3D view.

(2) Results

**Perceptron learning algorithm:**

The result of Perceptron learning algorithm can be shown as follows:

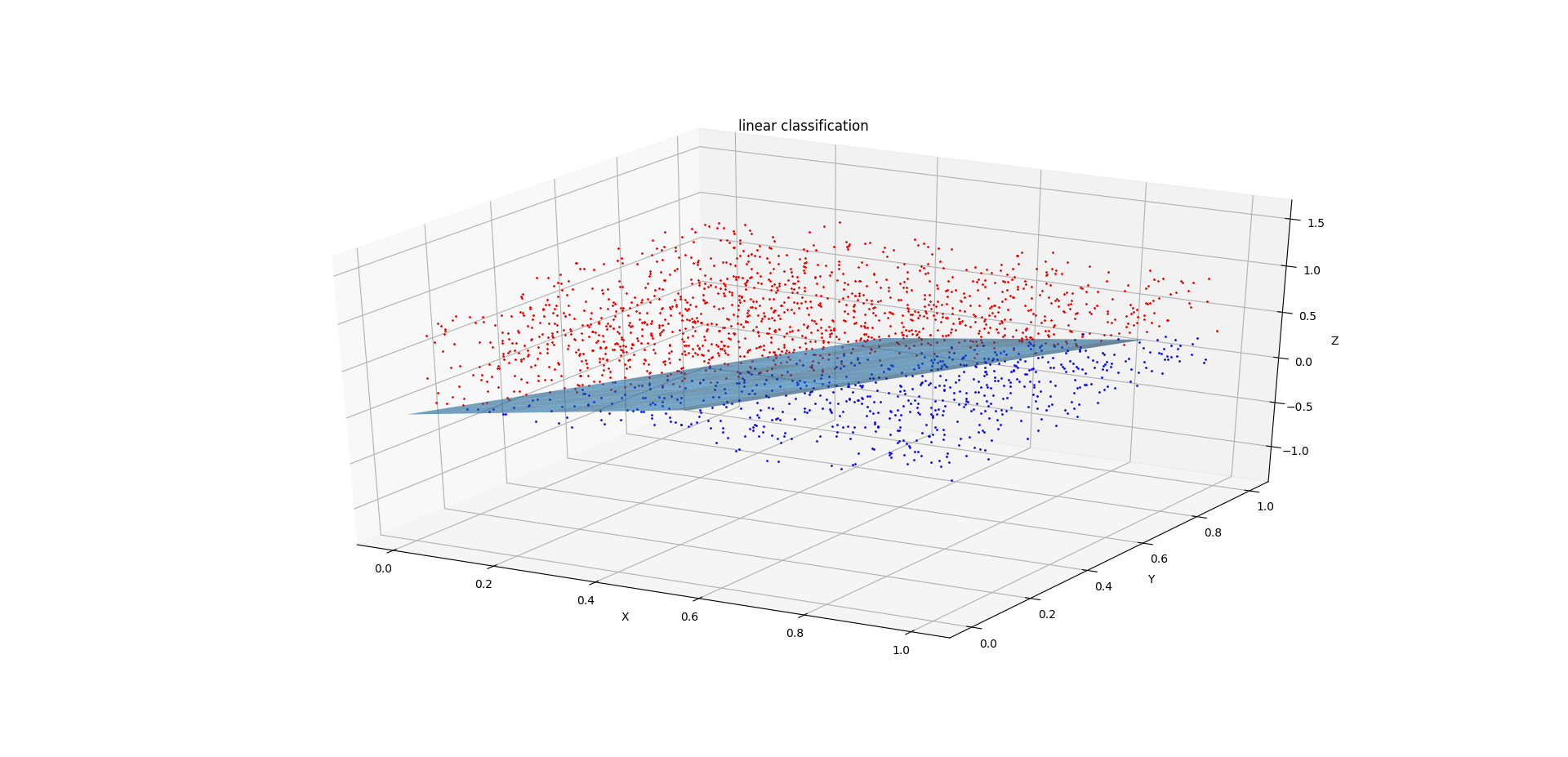


Figure 4 Result of Linear classification

Because these data are linear classification, so we can calculate the coefficients of a Plane to separate them. In the figure 4 the red points and the blue points are separated by the blue plane and the coefficients of the plane are [[-0.00258129 2.05831342 -1.64815778 -1.23164188]]. It can be represented as a formula.

*S =* -0.00258129 *+* 2.05831342 *x +* -1.64815778  *y +* -1.23164188 *z*

In the figure 5 we can know the process of calculating the coefficients. When we first initiate an original w, all the coefficients are set randomly, so 250 points will not satisfy the constrain, shown as the arrow on the left top. After iteration changing the coefficients w, the number is decreasing. However, this number sometimes will increase slightly, but at the end, the number is going to be zero, shown as the arrow on the right bottom.

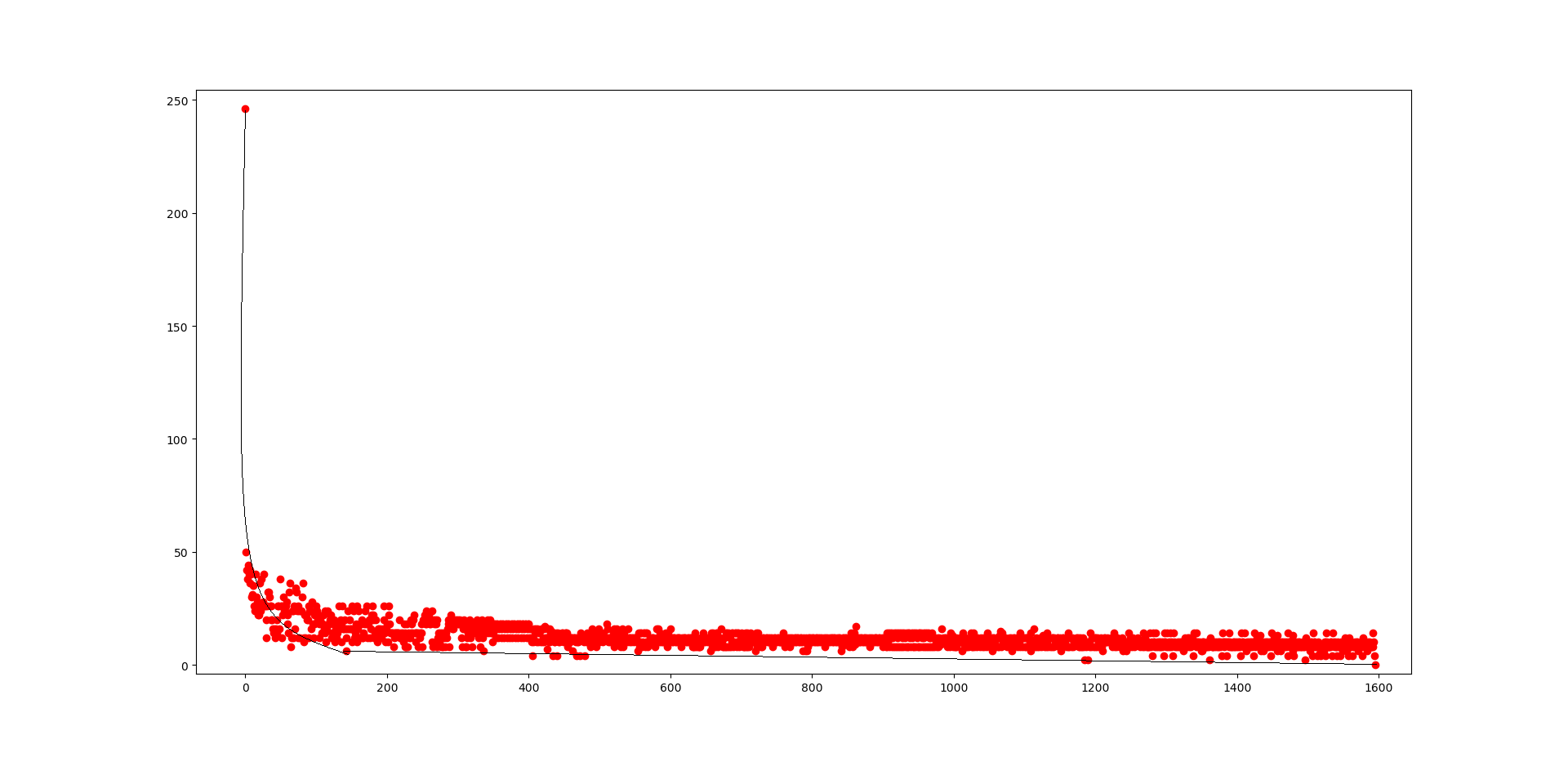
****

Figure 5 The process of calculating the coefficients

**Pocket algorithm:**

The result can be shown as follows. The coefficients are [[ 0.00905698 -0.00748353 0.00940292 -0.01304171]]. It can be represented as this formula:

*S = 0.00905698 + -0.00748353 x + 0.00940292 y + -0.01304171 z*

If S > 0 it belongs to class 1; if S < 0 it belongs to class -1.

To represent the formula directly among the data points, we draw a plane in the 3D view. Shown as figure 6. Because these data are not linear classified, so the plane is not meaningful.

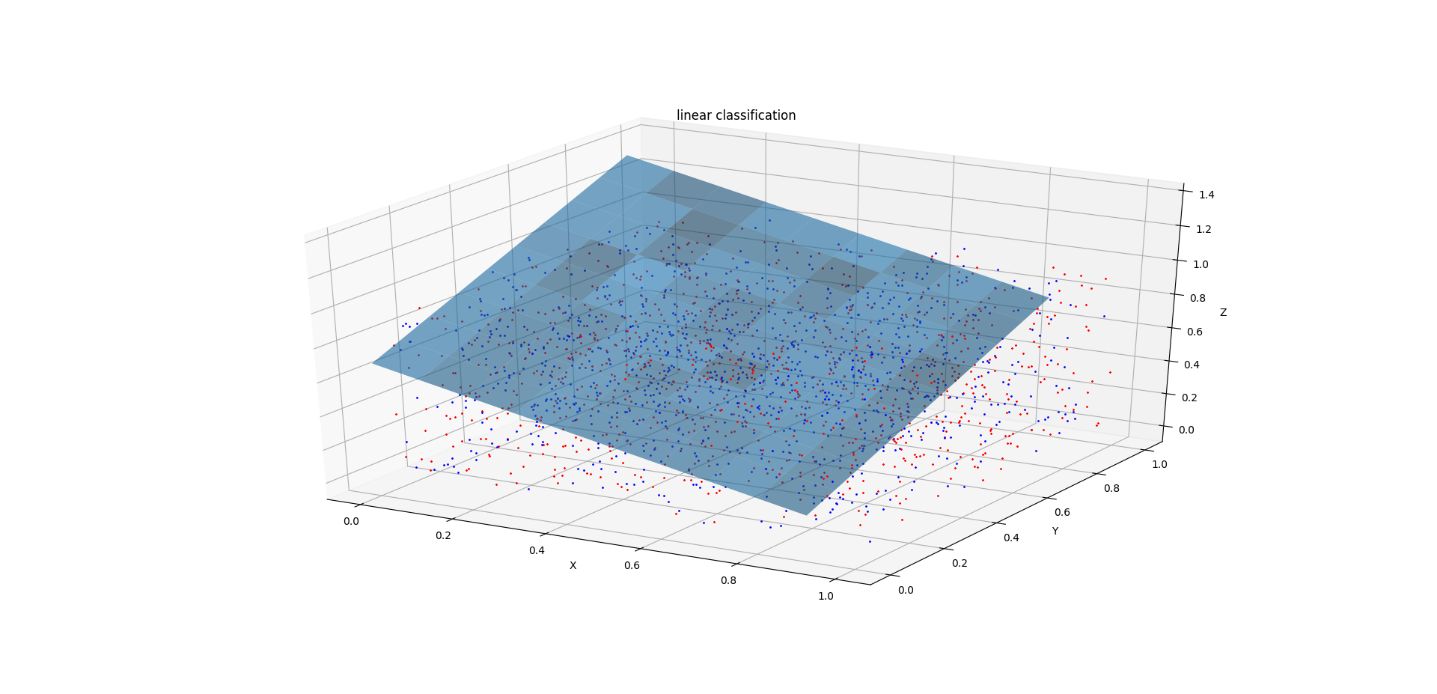
****

Figure 6 The result of linear classification

**Logistic Regression:**

The result can be shown as follows. The coefficients are [[-0.21953169 -0.1452412 0.27220502 0.24614781]]. It means we can represent this formula as

*S = -0.21953169 + -0.1452412 x +0.27220502 y + 0.24614781 z*

So, we can represent the probability as

*P (y(i) = 1 │x (i)) = θ (S) ).*

To represent the formula directly among the data points, we draw a plane in the 3D view. Shown as figure 7. Because these data are not linear classified, so the plane is not meaningful.

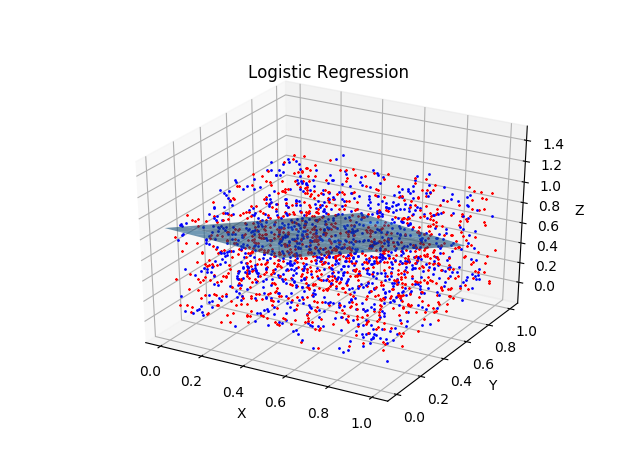
****

Figure 7 The result of Logistic Regression

**Linear Regression:**

The result can be shown as follows. In linear regression, all data are plotted with some rules, so we can find the coefficients to predict them. In figure 8, we can calculate the w (vector storing the coefficients) as [[ 0.01523535], [ 1.08546357],[ 3.99068855]]. Therefore, the linear regression can be presented as

*z = 0.01523535 + 1.08546357 x + 3.99068855 y.*

To represent this formula directly, we draw a plane in the 3D models. However, we also can solve this kind of problem using non-linear regression, for example, represent the formula as

*Z = w0 + w1\*x + w2\*y + w3\* x2 +w4 y2 + w5\* x\*y*

So, after calculation, we will get the formula as follows;

*Z = 0.00767315 + 1.09310895 x + 3.9989562 y + 0.02091292 x2 + 0.02029592 y2 + -0.05697217 x\*y*

Also, we can use much higher powers like x3, x4, or x5

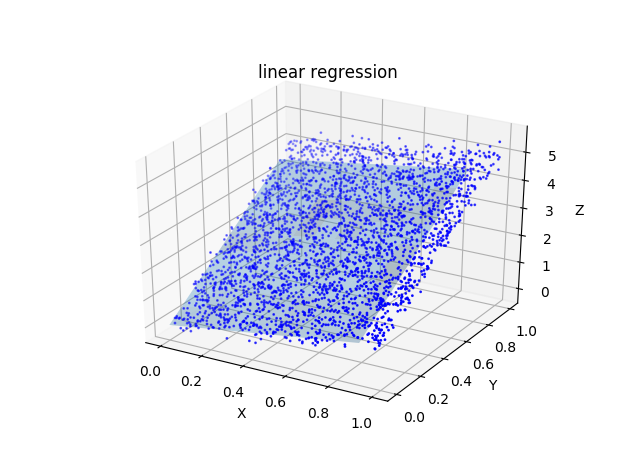


Figure 8 The result of linear regression

(3) data structure

Perceptron learning algorithm, Pocket algorithm, and logistic Regression use the same file, so the data structure can be the same. We create two Array, xDataList and yDataList. xDataList is to be used to store the points. Each row is represented as [ 1, x, y, z ] where 1 means the threshold, x, y and z represents one point. yDataList is to be used to store the classification of each point 1 or -1.

Linear regression is another file, so the data structure is different. We create an array to store each point as [1, x, y] where 1 means the threshold, x, y represents one point and create another array to store the z values as [z]. We also tried to using the same data to do non-linear regression. Therefore, we create the array as [1, x, y, z, x2, y2, xy] and another array as [z]

(4) challenges, code-level optimizations

The biggest challenge is to understand the meaning of Linear classification and Logistic regression, and the difference between these two algorithms. It can say that they used different thoughts to solve the classification problems. Linear classification uses a criterion whether the calculation results are bigger or smaller than 0 to classify them into two class. However, Logistic regression uses probabilities to solve the problems, which means to explore the possibilities each point belongs to which class. Therefore, at the beginning we are confused about the wT *\* x(i*) and θ *(*wT *\* x(i*)*)* which makes us make some mistake in our code. We correct this, and get the right result.

As for the linear regression, we make some coding optimizations. We make some changes on the dataset from [1, x, y] to [1, x, y, z, x2, y2, xy]. Although this is not linear regression any more, this method will make results much precise.

**2 Software Familiarization (Yu Hou,** **Haoteng Tang)**

(1) Python for linear regression

*>>>x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=1)*

*//this method can separate data into testing and training*

*>>>model = Lasso()*

*//define linear regression*

*>>>alpha\_can = np.logspace(-3, 2, 10)*

*>>>lasso\_model = GridSearchCV(model, param\_grid={'alpha': alpha\_can}, cv=5)*

*>>>lasso\_model.fit(x, y)*

*//use linear regression*

*>>>y\_hat = lasso\_model.predict(np.array(x\_test))*

*// get the predict result.*

(2) Python for logistic regression

*>>>from sklearn.linear\_model import LogisticRegression*

*//input package*

*>>>lr = Pipeline([('sc', StandardScaler()), ('clf', LogisticRegression()) ])*

*>>> lr.fit(x, y.ravel())*

*//use logistic Regression*

**3 Application (Haoteng Tang)**