Hao Fang

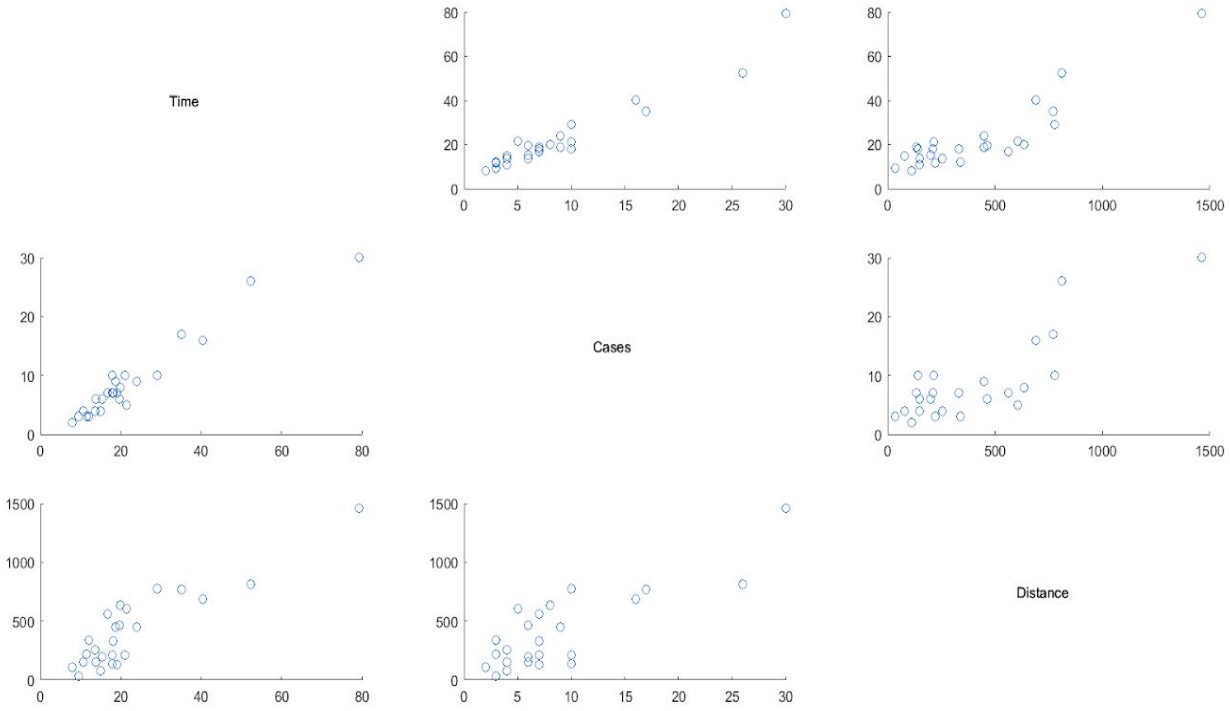
Haoyang Su

# CSCI-484 Project 1 Report

## Part 1: Linear Regression Model

For the part 1 of the project 1, we are going to predict the delivery time (y in min) based on the the number of cases of product stocked (x1) and the distance walked by the driver (x2 in feet) from data set “DeliveryTimes.txt” by using linear regression model.

Before dealing with the data, we should visualize the data set and see what the data looks like, and here is the graph of the data:




By looking at data, we can find some kind relationship between different variables which can help to find out correct way to implement the linear regression.

There are 2 features in this data set, which means we need three β's in the linear regression model (includes the β0). In order to estimate the multiple regression equation, we to use our data to train the linear regression model and find the best β's to fit the model.

Therefore, in this project we are going to use the least square equation can help to find out the best β's, the equation below:

$$\beta\text{'s} = \text{pinv}(X' * X) * X' * y$$

After train the data, the result of β's is :

$$\beta 0 = 2.341231$$

$$\beta 1 = 1.615907$$

$$\beta 2 = 0.014385$$

the mean square error (MSE) is :

$$3.057657$$

The mean square error is between the predicted time values using your model and the actual time values given in the dataset, and MSE here is a small number, which means we did a good prediction of this data set.

In the next a few pages, there shows the codes in matlab:

Part 2 : Support Vector Machine

This part of the project will use the Support Vector Machine to classify the different classes of the data set. As our case, we are going to do a binary classification which means we have only two classes in our samples. The data set we choose is “dota2Test.csv” , which is a game data set, the classes of the samples are “win”(1) or “lose”(-1).

Since the dataset has more than 100 features, and they will be nonlinearly separable data, using linear SVM is impossible to separate those data. Therefore, we are going to use different kernel functions to find out the best classifier. In this part, we choose two kernel functions: Polynomials, and RBF kernel functions.

By using the code from class, the main thing to change the C value in order to get a good parameters for predicting. Normally, the larger C value we use, the better parameters we get. But we should get a reasonable C value when we apply the different kernel functions to train the data set.

Analysis of Polynomials kernel function:

The Polynomial kernel function looks like:

$$K=(x*y + 1)^2$$

In order to find the best C value, we choose 4 different C values, and then calculate the b value which is the weight based on the first 600 samples from the dataset. Here is some code and analysis:

```
%=========================================================================%
    %   (Kernel function: Polynomials)
    %  The program will use 900 samples from dota2Test.csv, and 600 samples will be
    % used to train the model,and the other 300 samples will be used to test the model.
%=========================================================================%
```

%==========================================================%

% Fold 1 (First 600 samples for training, last 300 samples for testing)

%==========================================================%

% When Cval = 0; b = NaN; Test result:  Accuracy rate: 46%

% When Cval = 10; b = 3.039336087336221; Test result: Accuracy rate: 59.3%

% When Cval = 50; b = 3.039339503491064; Test result: Accuracy rate: 59.3%

% When Cval = 100; b = 3.039339687450168; Test result: Accuracy rate: 59.3%

%==========================================================%

% Fold 2 (Last 600 samples for training, first 300 samples for testing)

%==========================================================%

% When Cval = 0; b = NaN; Test result:  Accuracy rate: 52.3%

% When Cval = 10; b = 1.292208904051222; Test result: Accuracy rate: 50.3%

% When Cval = 50; b = 1.292205382745403; Test result: Accuracy rate: 50.3%

% When Cval = 100; b = 1.292205457598126; Test result: Accuracy rate: 50.3%

%==========================================================%

% Fold 3(600 samples for training, middle 300 samples for testing)

%==========================================================%

% When Cval = 0; b = NaN; Test result:  Accuracy rate: 44.6%

% When Cval = 10; b = -0.985859142422366; Test result: Accuracy rate: 67%

% When Cval = 50; b = -0.985864941227871; Test result: Accuracy rate: 67%

% When Cval = 100; b = -0.985865296742413; Test result: Accuracy rate: 67%

%==========================================================%

%==========================================================================%

% Polynomials:

% C = 0 ;The average accuracy rate: 47.63%

% C = 10 ;The average accuracy rate: 58.87%

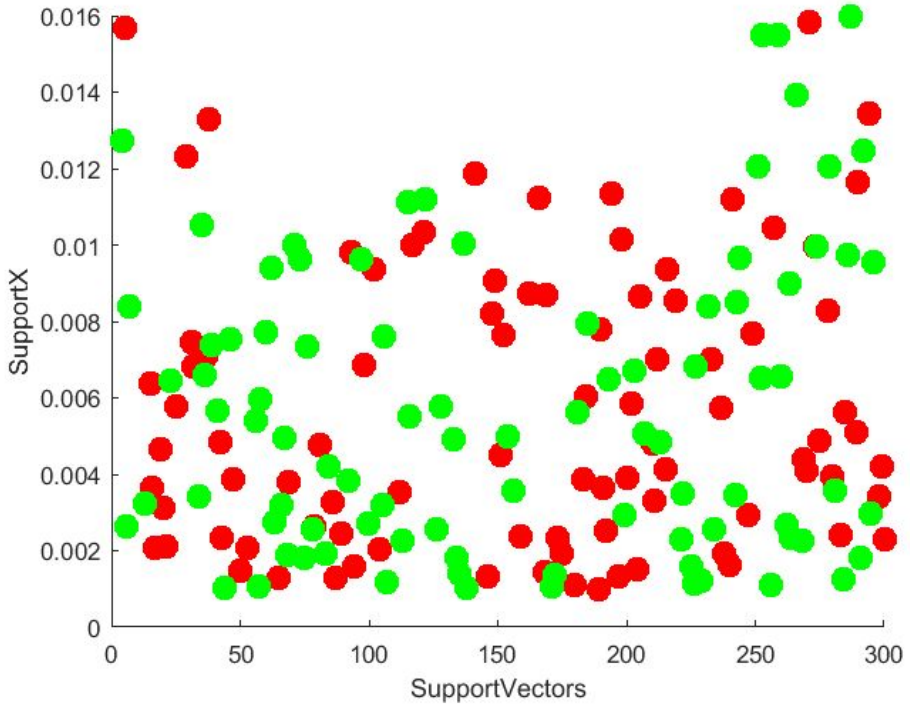
% C = 50 ;The average accuracy rate: 58.87%

% C = 100 ;The average accuracy rate: 58.87%

%==========================================================================%

From the code above, we can find out the best C value we can choose from 10 - 100, since the accuracy rate are almost the same. However, this is not a good model to predict the data set, because the highest average accuracy rate: 58.87%, which is not good for prediction.

By choosing the best C value for polynomials kernel function, we plot the data based on the testing data in matlab:



From the picture of the data, we can see that the data is not separated very well by using Polynomial kernel function, the Polynomial kernel function may not be able to predict this data set very well since the highest average accuracy is about 58.87%.

Analysis of RBF kernel function:

The Polynomial kernel function looks like:

```
var_mean = 0;
for i=1 : num_features
    var1 = var(Data(i,:));
    var_mean = var1 + var_mean;
end
var_mean = var_mean / 116;
disp(var_mean);
                    kval = x1 - x2 ;
            K = exp( -(kval * kval')/ var_mean );
```

In order to find the best C value, we choose 4 different C values, and then calculate the b value which is the weight based on the first 600 samples from the dataset. Here is some code and analysis:

```
%=============================================================%
    %   (Kernel function: RBF)
    %   The program will use 900 samples from dota2Test.csv,
    % and the first 600 samples will be used to train the model,and the
    % other 300 samples will be used to test the model.
%=============================================================%
```

%==========================================================%

% Fold 1 (First 600 samples for training, last 300 samples for testing)

%==========================================================%

% When Cval = 0; b = NaN; Test result:  Accuracy rate: 46%

% When Cval = 10; b = 0.127861867198121; Test result: Accuracy rate: 83%

% When Cval = 50; b = 0.046846318936093; Test result: Accuracy rate:  96%

% When Cval = 100; b = 0.170427079452457; Test result: Accuracy rate: 99%

%==========================================================%

% Fold 2 (Last 600 samples for training, first 300 samples for testing)

%==========================================================%

% When Cval = 0; b = NaN; Test result:  Accuracy rate: 52.3%

% When Cval = 10; b = -0.200159289808129; Test result: Accuracy rate: 98.6%

% When Cval = 50; b = 0.261868032357953; Test result: Accuracy rate: 99%

% When Cval = 100; b = 0.300522967602285; Test result: Accuracy rate: 99%

%==========================================================%

% Fold 3 (600 samples for training, middle 300 samples for testing)

%==========================================================%

% When Cval = 0; b = NaN; Test result:  Accuracy rate: 44.667%

% When Cval = 10; b = 0.023980197327936; Test result: Accuracy rate: 96.3%

% When Cval = 50; b = 0.326834159848469; Test result: Accuracy rate: 96%

% When Cval = 100; b = 0.426165209137314; Test result: Accuracy rate: 95%

%==========================================================%

% RBF kernel:

% C = 0 ;The average accuracy rate: 47.66%

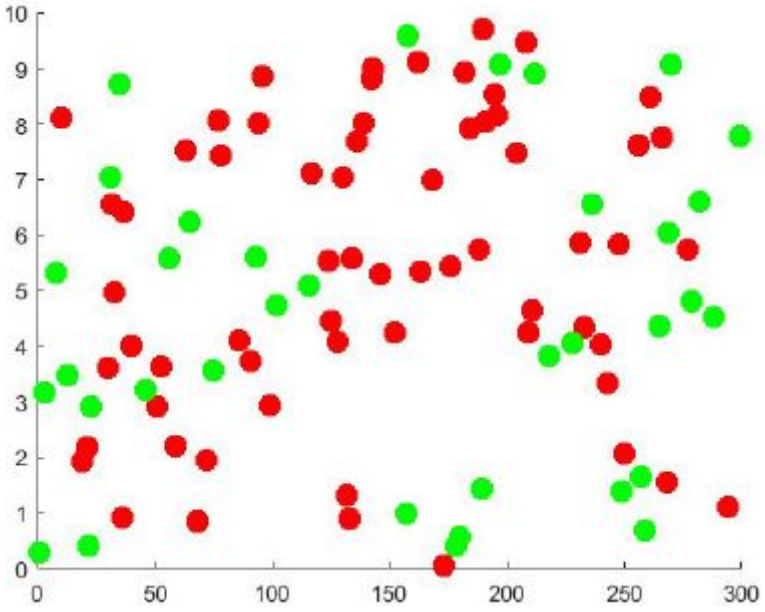
% C = 10 ;The average accuracy rate: 92.63%

% C = 50 ;The average accuracy rate: 97%

% C = 100 ;The average accuracy rate: 97.67%

%==========%

The data above shows that, when C value increases from 0 to 100, the average accuracy rate increases from 47.66% to 97.67%, which means the accuracy of prediction is better when we choose the bigger C value.




From the picture of the data, we can see that the data is still not separated very well by using RBF kernel function, but we can see that the class 1 and class 2 are more separated than the picture above, the RBF kernel function may be a better option for this data set since the data set may to complex and the there are so many different blocks of one class. Since we have already got about 97.67% correct of prediction, the RBF kernel function work really well for this data set.