Birla Institute of Technology and Science
Foundations of Data Science (CS F320)
I Sem 2021-22
Lab Sheet #3: Linear & Polynomial Regression

a) Linear Regression

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

The general mathematical equation for a linear regression is −

$y = ax + b + \in$

Following is the description of the parameters used −

y is the response variable.

x is the predictor variable.

a (the intercept) and b (the slope) are constants which are called the coefficients.

$\in$ is called the error term.

Steps to Establish a Regression

The steps to create the relationship is −

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the **lm()** functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called residuals.
- To predict the weight of new persons, use the **predict()** function in R.

A Simple Example

A simple example of regression is predicting weight of a person when his height is known. A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

Below is the sample data representing the observations −

# Values of height

151, 174, 138, 186, 128, 136, 179, 163, 152, 131

# Values of weight.

63, 81, 56, 91, 47, 57, 76, 72, 62, 48

<u>lm() Function</u>

This function creates the relationship model between the predictor and the response variable.

<u>Syntax</u>

The basic syntax for lm() function in linear regression is −

lm(formula,data)

Following is the description of the parameters used −

- formula is a symbol presenting the relation between x and y.
- data is the vector on which the formula will be applied.

<u>Create Relationship Model & get the Coefficients</u>

```
# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
# The response vector.
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)
print(relation)
```

Execution of above code shows you the regression coefficients.

Now, print summary of relation and observe the values.

<u>predict() Function</u>

<u>Syntax</u>

The basic syntax for predict() in linear regression is −

predict(object, newdata)

Following is the description of the parameters used −

- object is the formula which is already created using the lm() function.
- newdata is the vector containing the new value for predictor variable.

Predict the weight of new persons

```
# Find weight of a person with height 170.
a <- data.frame(x = 170)
result <-  predict(relation,a)
```

```
print(result)
```

Visualize the Regression

Type the following to see the graphical line fit:

```
plot(y,x,col = "blue",abline(lm(x~y)),xlab = "Weight in Kg",ylab = "Height in cm")
```

Exercise 1

For this analysis, we will use the cars dataset that comes with R by default. Observe the dataset.

You will do simple regression model that you can use to predict Distance (dist).

1. Build a linear regression model with dist as a predictor variable. Identify the regression coefficients.

2. Identify the linear regression equation of the form: dist = a + b.speed

3. Plot the graph.

Exercise 2

Once you have done Exercise-1, its time to see more details about linear regression. See the article using following URL for the same.

https://www.machinelearningplus.com/machine-learning/complete-introduction-linear-regression-r/

b) Polynomial Regression

Now that you know how to fit a line/plane/hyperplane through data, let us now learn how to fit a polynomial of degree M into a given dataset. Note that with Linear regression is a special case of polynomial regression with M=1.

Input Data:

Generate some data by using the function sin(2πx) in [0,1]

```
x = seq(0,1,length=11)

y = sin(2*pi*x)

Some noise is generated and added to the real signal (y):

noise = y + rnorm(11, sd=0.3)

#we will make y the response variable and x the predictor

#the response variable is usually on the y-axis

plot(x,noise,pch=19)
```

The basic syntax for lm() function in linear regression is -
lm(formula,data)

```
#fit first degree polynomial equation:

lm1  <- lm(y~x)

#second degree

lm2 <- lm(y~poly(x,2)
```

Now try the model it with degree varying from 3 to 9 and plot the fits.

Write down your observations as degree increases from 1 to 9.

Now increase the number of training points, N as mentioned below for M=9

N=10

N=15

N=30

N=50

N=100

Write down your observations as N increases.

Overfitting Problem:

1) **Plot a graph of training error and testing error**

```
#getting the test and training errors

lms = list(lm1,lm2,lm3,lm4,lm5,lm6,lm7,lm8,lm9)

# set up two null vectors for the errors

train = test = rep(0,10) # train = vector(length=10) also works

# this is SSE/10

for(i in 1:10){train[i]=var(lms[i][[1]]$residuals)}

# now compute similarly scaled prediction error for test data
```

```
newy = sin(2*pi*x)+rnorm(11,sd=.3) # note we used the same x's

for(j in 1:10){test[j] = (1/10)*sum((newy-predict(lms[j]


[[1]],newdata=data.frame(x)))^2)}

lines (1:10,train, ylim=c(0,.5), col='black',lwd=3)

lines(1:10,test,pch="X", col='red',lwd=3)

cbind(train,test)
```

2) **Show all coefficients of the polynomial in a tabular form. And write down your observations.**
   Coefficients are the intercept and the slope of the regression line, but more informative results about the model can find by:

```
#summary

summary(lm1)

#coefficient

C1 <- coef(lm1)
```

Use data frame for table generation

```
max_length <- max(c(length(C1), length(C2))) # Find out maximum
length for unequal table

#Now, we can use this information to create a data frame with NA
values at the bottom of each column that is shorter as the
maximum length.

data <- data.frame(col1 = c(C1,rep(NA, max_length - length(C1))),

                   col2 = c(C2,rep(NA, max_length - length(C2))))
```

Regularization:

Linear regression algorithm works by selecting coefficients for each independent variable that minimizes a loss function. However, if the coefficients are large, they can lead to over-fitting on the training dataset, and such a model will not generalize well on the unseen test data. To overcome this shortcoming, we'll do regularization, which penalizes large coefficients.

We will be using the glmnet() package to build the regularized regression models. The glmnet function does not work with dataframes, so we need to create a numeric matrix for the training features and a vector of target values.

```
lambdas <- 10^seq(3, -2, by = -.1)

fit <- glmnet(x, y, alpha = 0, lambda = lambdas)
```

Exercise

- Load “mtcar” dataset and split it into training and testing part. Plot a graph with various polynomial degree on training dataset. Then choose suitable degree for testing data.
- Apply ridge regularization by using glmnet function and plot the graph between lambda & error.