INTRODUCTION TO MACHINE LEARNING PROJECT

Ajay Anand Kumar: 013711933

Exercise 1a

Instructions: Copy the files “Exercise1.1.r” in the folder “mnist/”

Aim: To implement the perceptron algorithm for N= 4 sample of two class dataset in 2 dimension. Visually check if points are linearly separable and hence find the corresponding hyperplane or decision boundary.

Methodology:

Commands:>source(“Exercise1.1.r”)
>ex1.1()

The digits selected were zeros and one and 4 sample digits randomly selected. Since individual digits has dimension of 1 × 784. As such it was required to reduce the dimension to 1 × 2. The dimensions of digits were reduced by using Principal component analysis method. But before that the digits were processed such that mean of every digits is subtracted from each digit and it is rescaled to unit variance. Then PCA is done on processed digits. The procedure for PCA is as follows:

1. Input the digit matrix (4 samples) that has dimension 4 × 784.
2. Calculate the covariance matrix which results to 4×4 dimension covariance matrix.
3. Calculate the corresponding Eigen values (2 values) and Eigen vectors (2 × 2 dimension).
4. Project the input digits matrix to these Eigen vectors.
5. Resultant matrix is reduced dimension of original digits.

Since sample digits of 0 and 1 are randomly chosen as such example plot of digits in two dimension is shown in **Figure 1.1**. The red dots indicate zeros and blue dots indicate ones. For digits having index 1513 and 2161 for zeros (red) and 3042 and 471 for ones (blue) a linearly non separable plot was obtained in **Figure 1.2.**

For Linearly separable: For linearly separable points the algorithm implementation ran for two iterations and converged. The resultant linear decision boundary was found and it is shown in Figure 1.3. From the figure it can be seen that a linear decision boundary was found and it perfectly separates the blue and red points.

For Non-linearly separable: For these points the algorithm was run for 10000 iterations and it did not converge. The error rate oscillates from 0.25 and 0.5. The resultant plot is shown in

Figure 1.4. It can be clearly seen from the figure that the decision boundary does not separate the red and blue points.

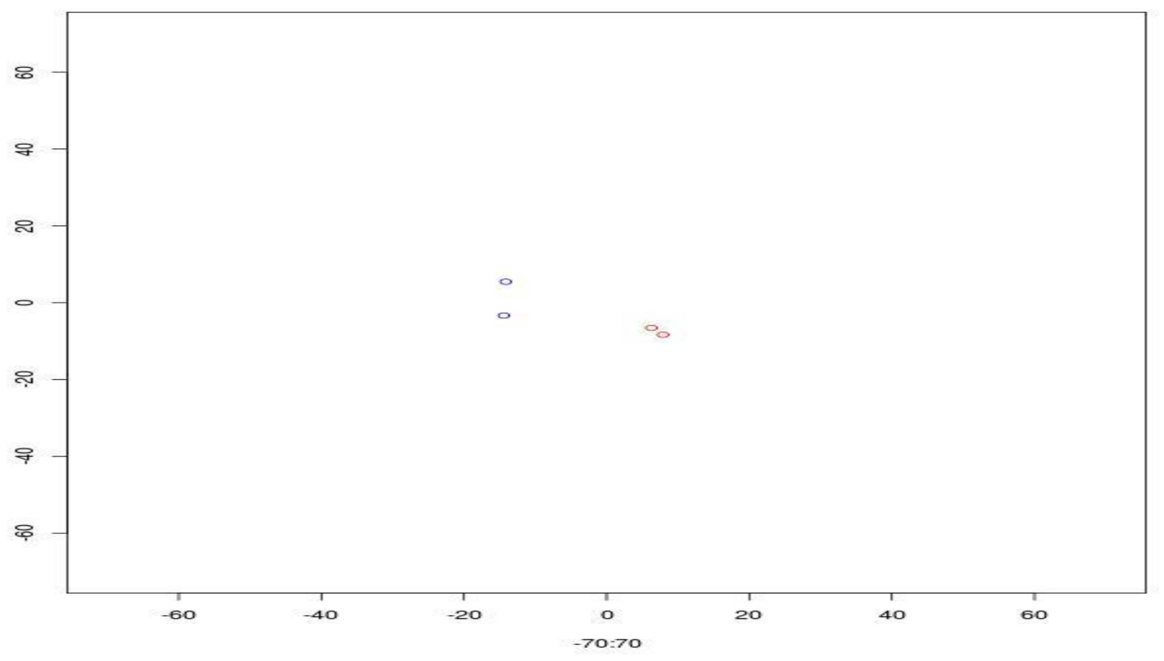


Figure 1.1: Linearly separable points. Red dots indicate zeros and blue dots indicate ones.

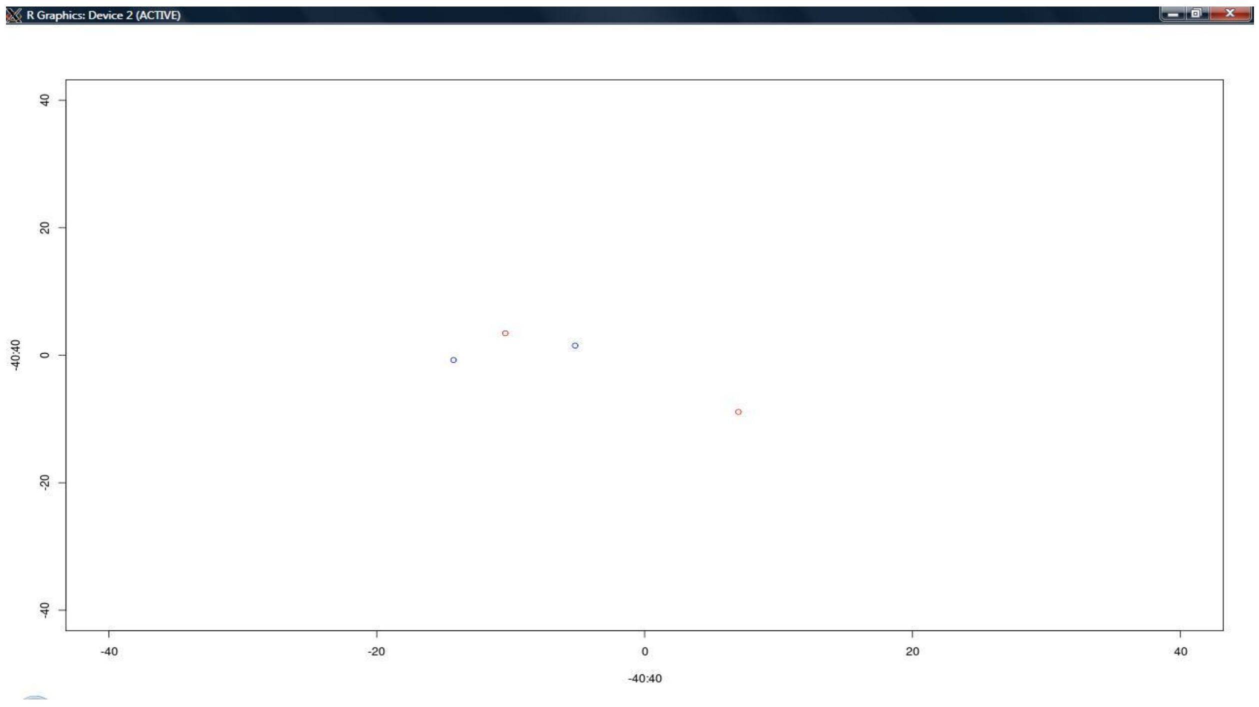


Figure 1.2: Visualization of non separable points. Red dots indicate zeros and blue dots indicate ones.

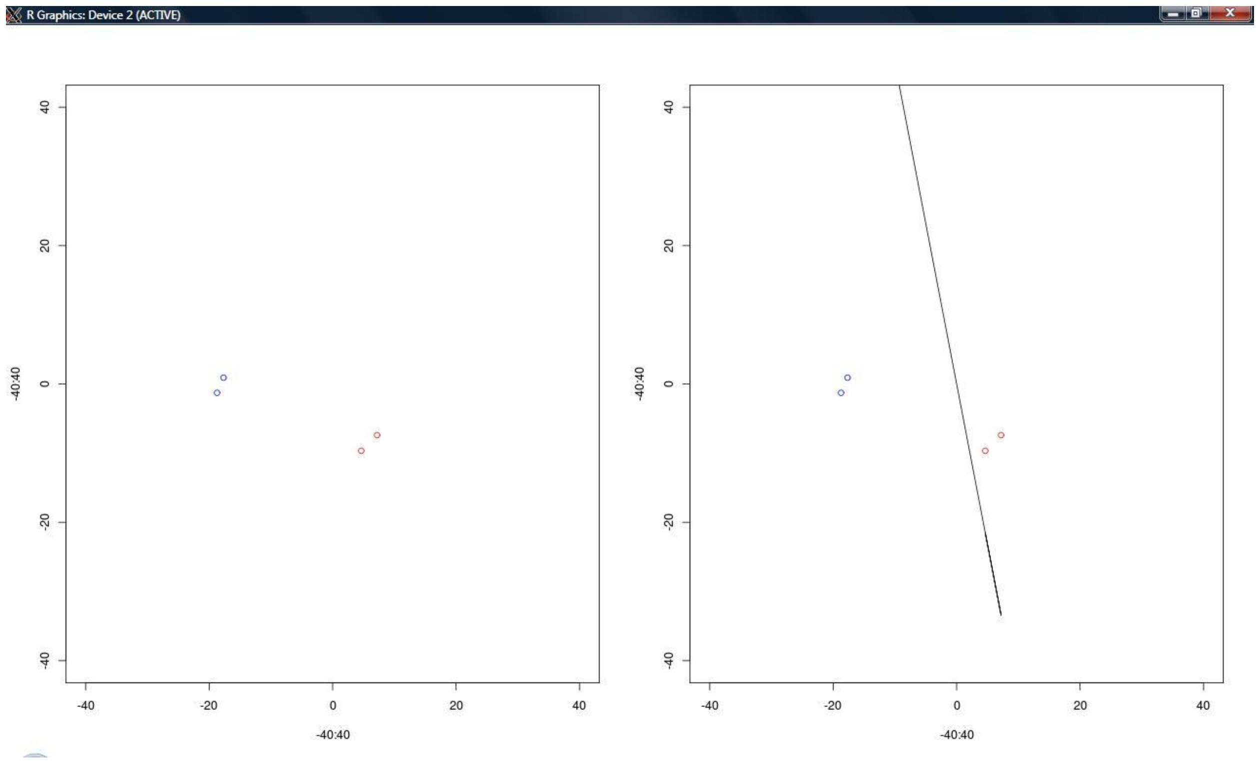


Figure 1.3: Linear separable points with decision boundary (black line). Red dots indicate zeros and blue dots indicate ones.

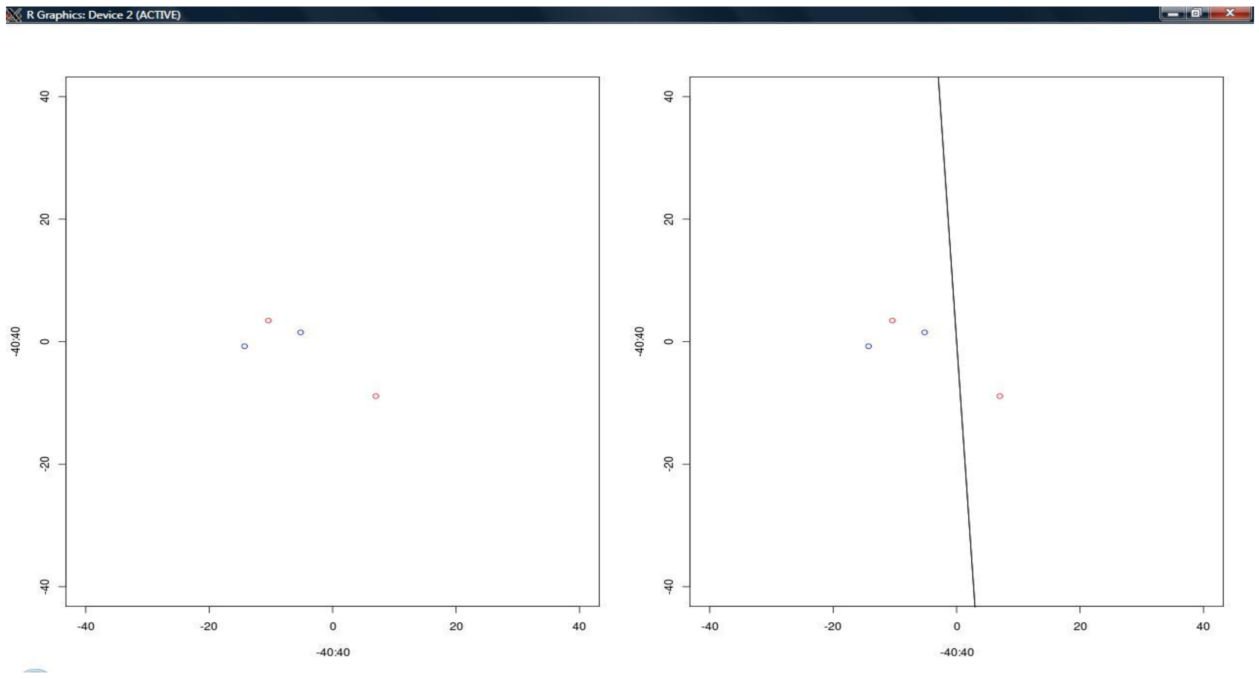


Figure1.4: Linearly non separable points and no perfect decision boundary found.

Exercise1.b

Aim: To apply implementation of perceptron algorithm in binary classification of hand written digits (only zeros and ones).

Methodology:

Commands:>source("Exercise1.1.r")
>ex1.2()

As compared to previous exercise, there is no dimensionality reduction process applied to it. Training set of zeros and ones were created for first N = 2500 digits and test set was created for remaining set.

Initial weight vector of dimension (1 × 784) was initialized to zeros. The bias term **c** was chosen = 3000. The target class was converted to (-1 ,+1) where -1 indicates class label for zeros and +1 indicate class label for ones. The algorithm ran for 4 iterations and with each iteration with training set the error rates are:

0.003913894, 0.003913894, 0 and 0 for respective iterations.

The final misclassification error rate for test data is: 0.001883239.

The resultant weight vector looks like zero. This is because after learning the training set and since the algorithm converged which indicates it found an hyperplane that is more closer towards points belonging to zero class. This can be shown in **Figure 1.5.**

Indeed the perceptron algorithm did some misclassification and the error rate is: 0.001883239. The number of iteration for convergence varies with different values of constant term.

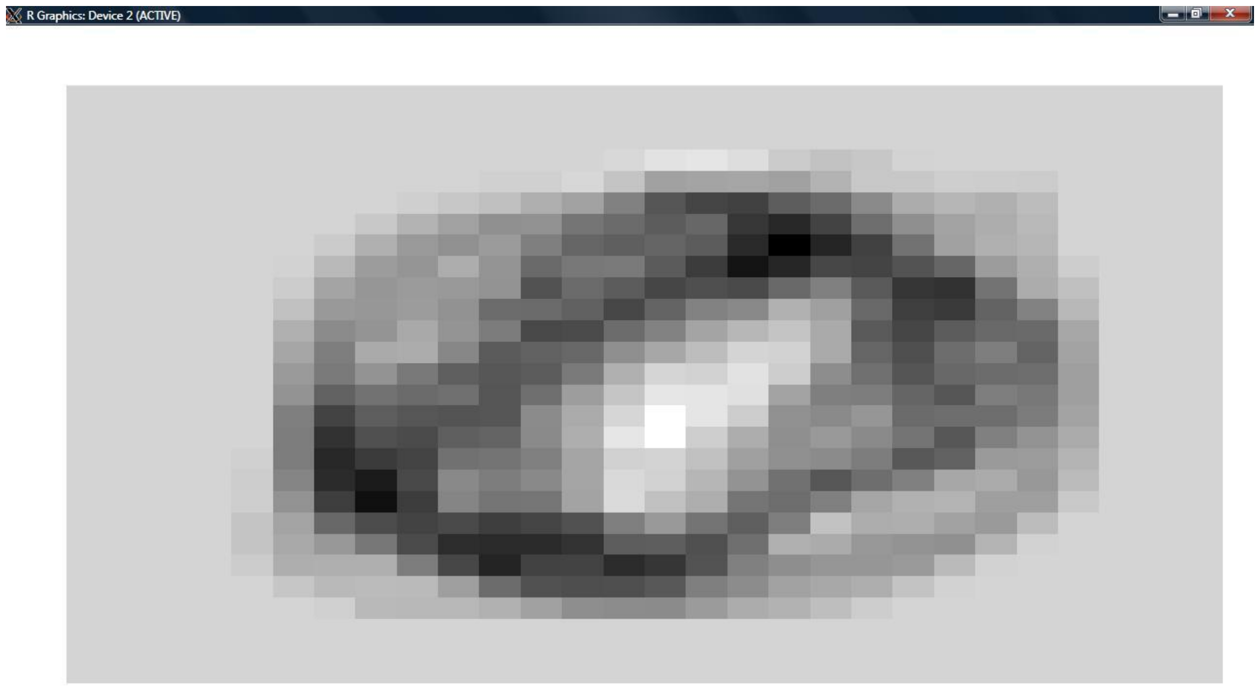


Figure 1.5: Visualization of learned weight vector.

Exercise 2

Aim: To test linear regression on sample dataset generated in the interval [-3, 3]. The target function is

$$Y = F(x) = 2 + x - 0.5 * x^2.$$

a) Fit polynomial of order k = 0 to 10 the sample data set using linear regression, minimizing the squares error.

b) test the regression model based on 10-fold cross validation.

Solutions:

Commands: >source("Exercise2.r")

Packages to be installed in R: MASS. The reason to install this package is that it has function to compute the pseudo inverse of the matrix. This computation is required in the code for calculating the coefficient matrix for the *k* order polynomial function.

a)

1. Generate 300 data points in interval [-3,3]

Commands: >data

Output:

```
           x         y
1   2.35917613  1.1112361
2  -2.39346303 -3.1369911
3  -1.54646940 -0.3628256
……
```

Gives a list of 30 pair of data points.

2. Fitting a polynomial of order k = 0,1,2,3,….,10 can be done and the corresponding plots can be obtained by this command:

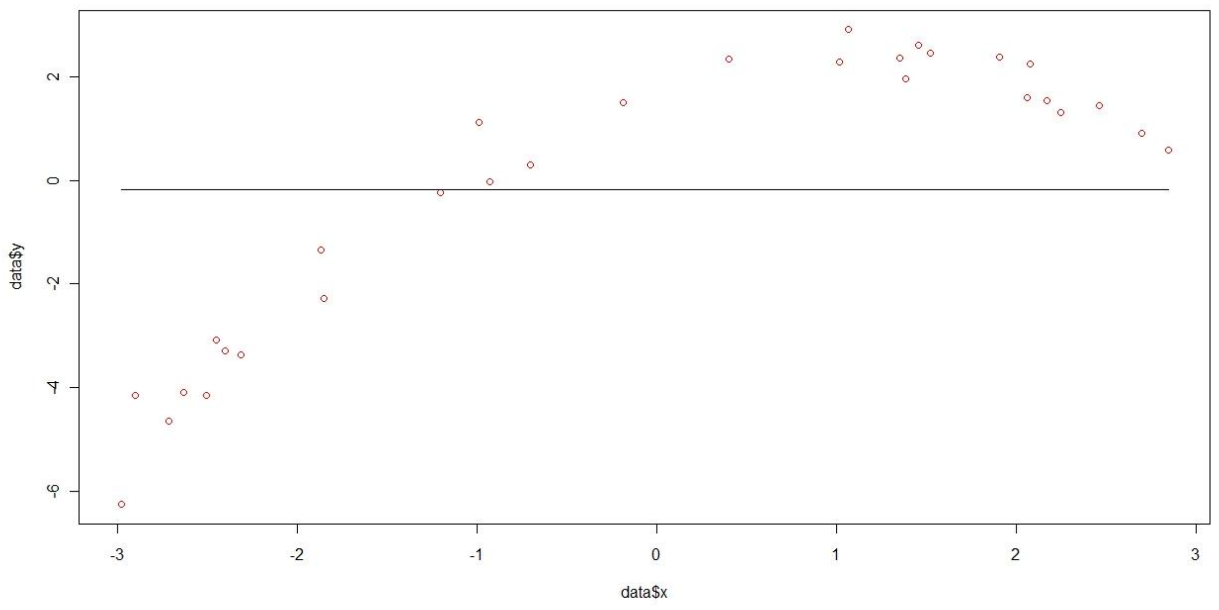
```
> poly_plot(0,data)
```

For individual values of ***k*** the corresponding plots can be obtained by simply using the function *poly_plot(k,data)* where the input parameter k has value 0 to 10 and data is the set of 30 pair of data points which is obtained initially by running the source code.

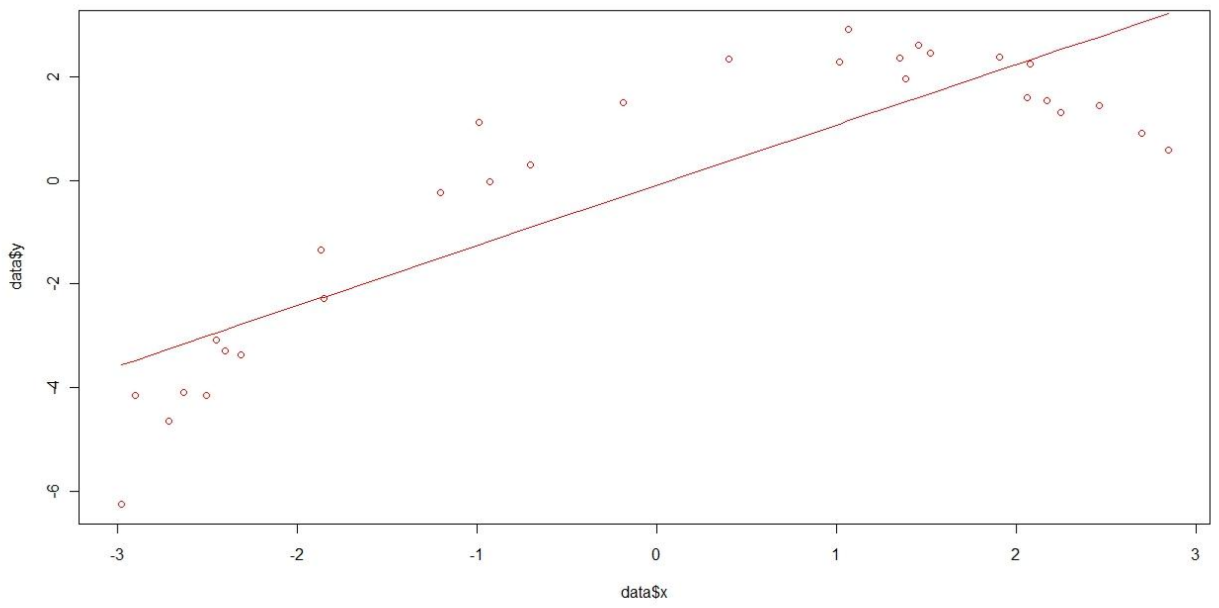
All the plots for k = 0 to 10 are shown below.

We can see from the below plots that as the value of K increases the curve fits closely to the data points.

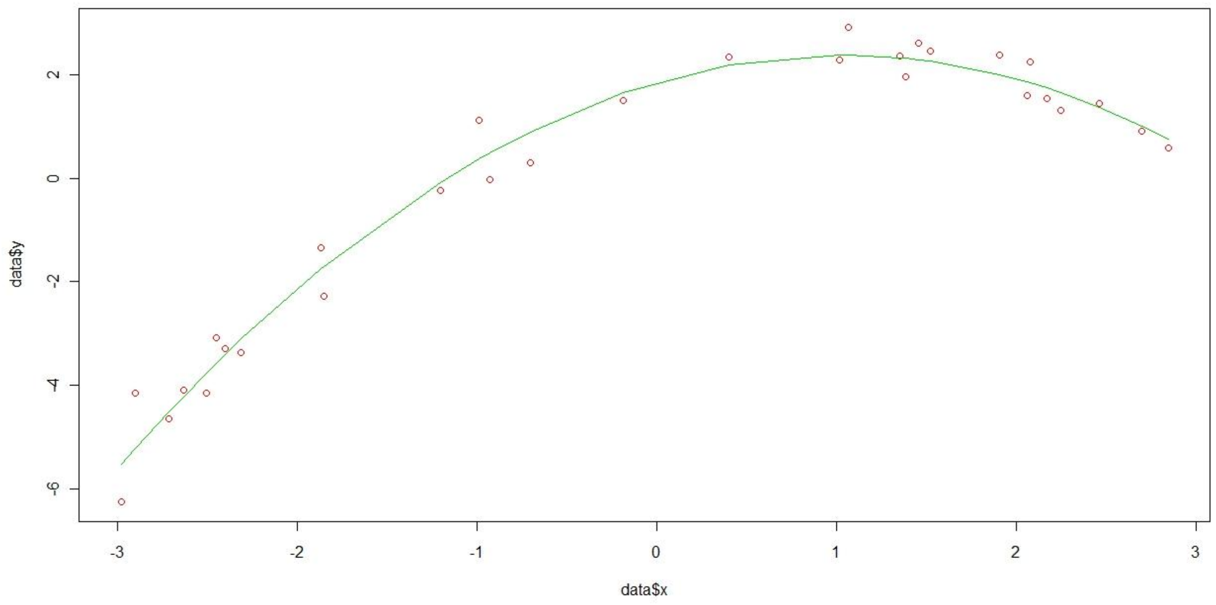
K = 0



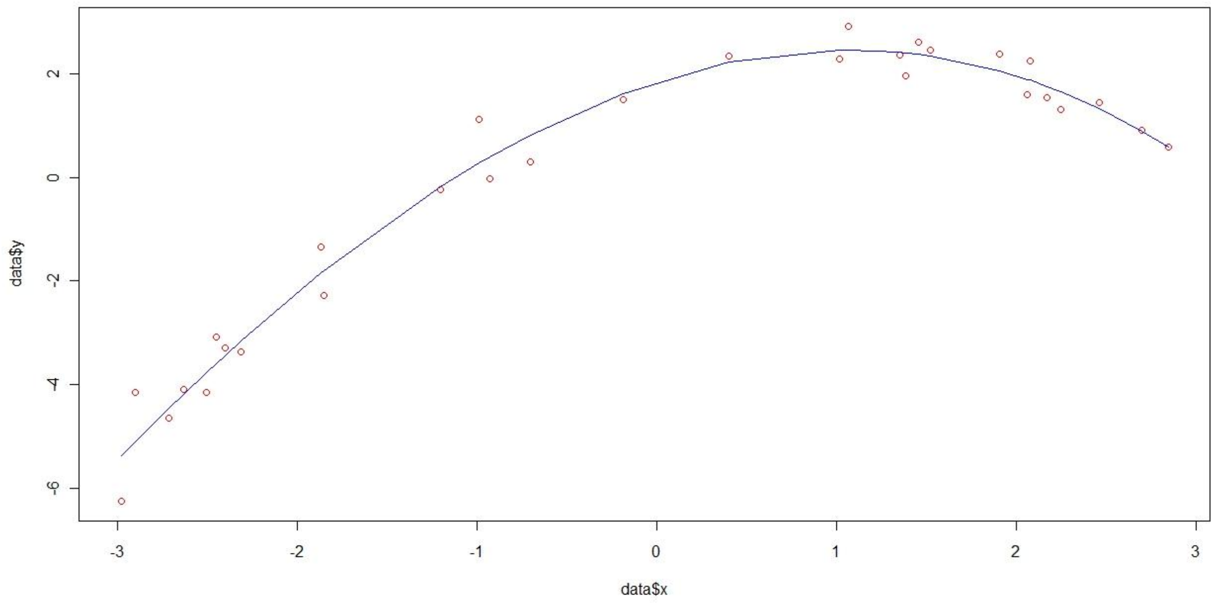
K = 1



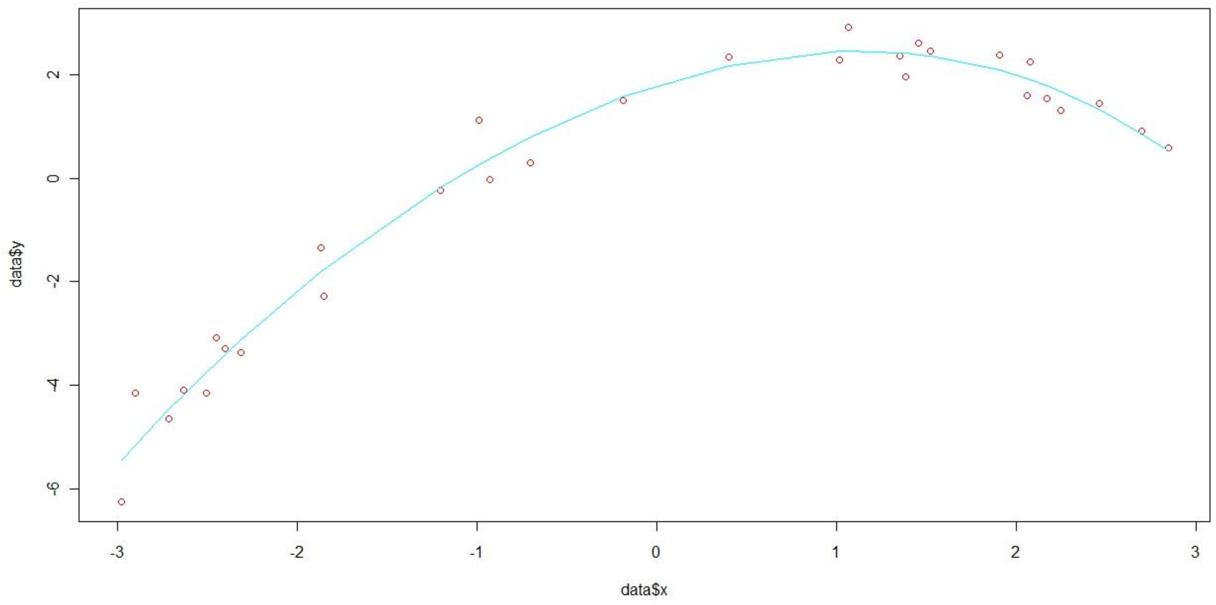
K = 2



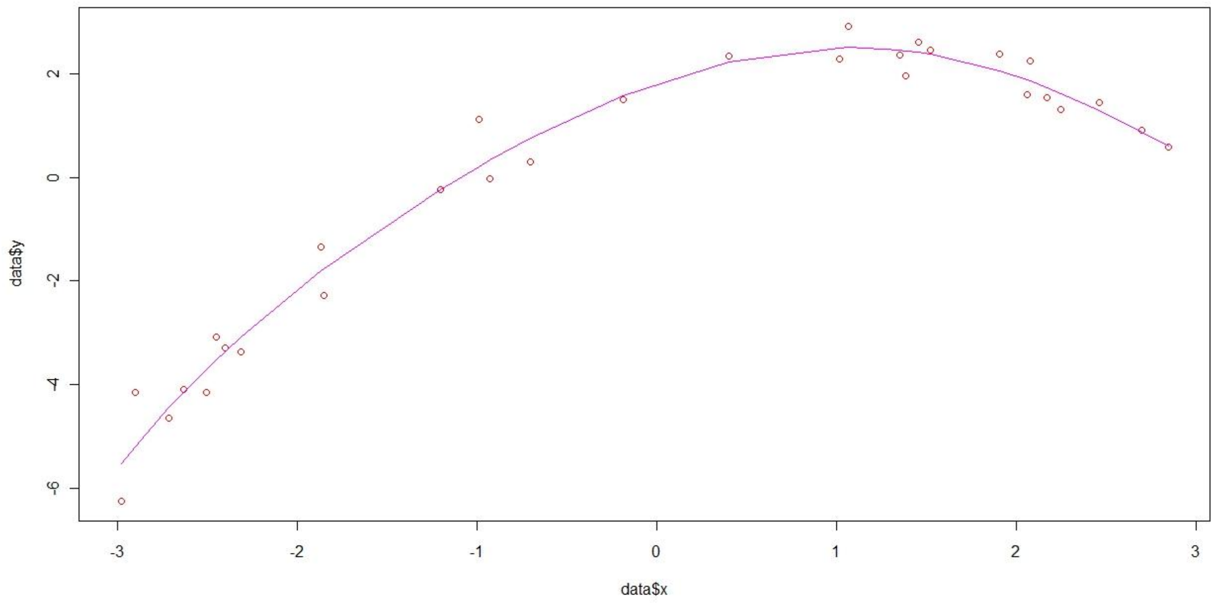
K = 3



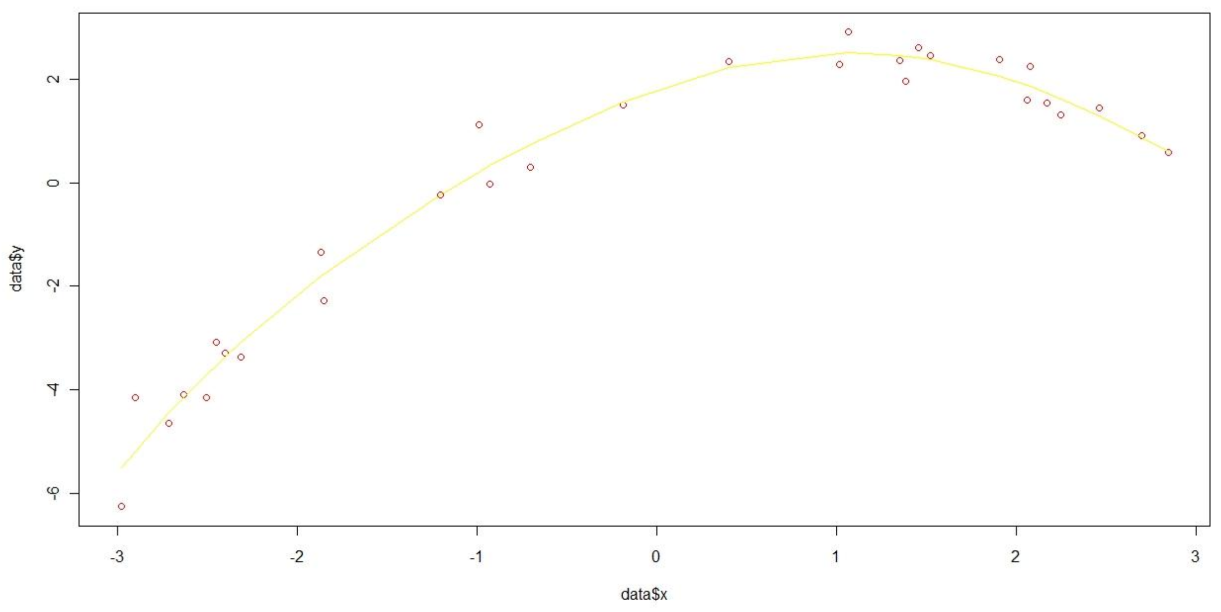
K = 4



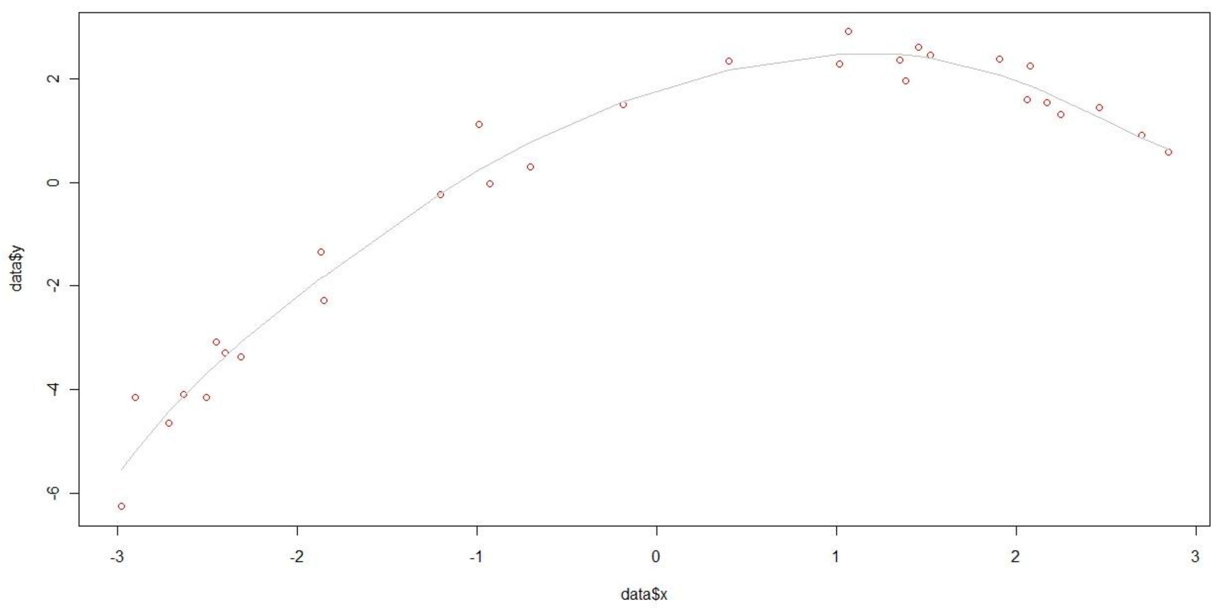
K = 5



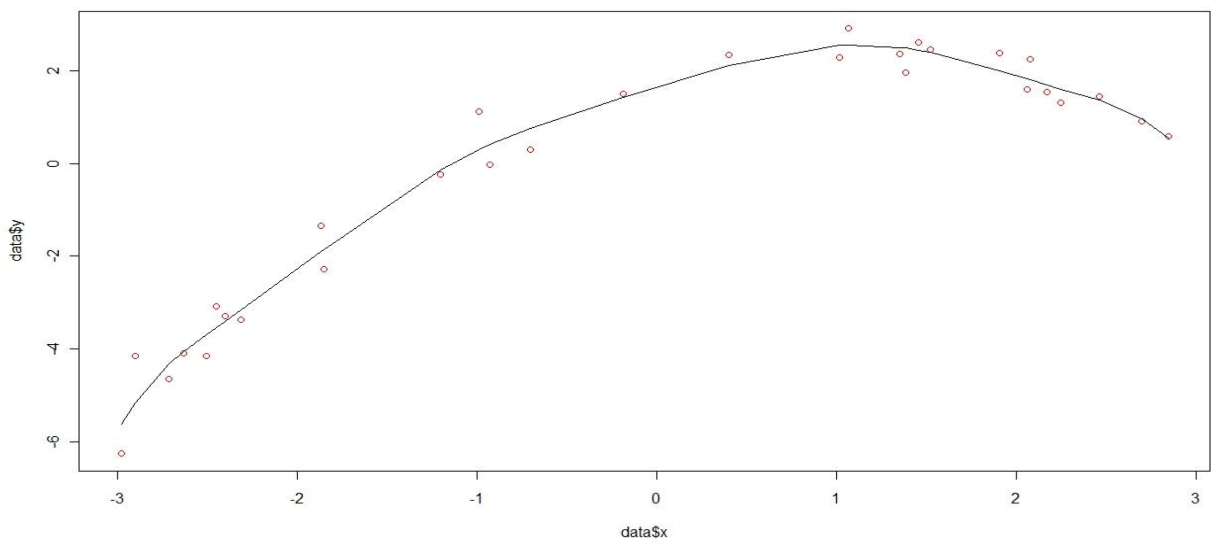
K = 6



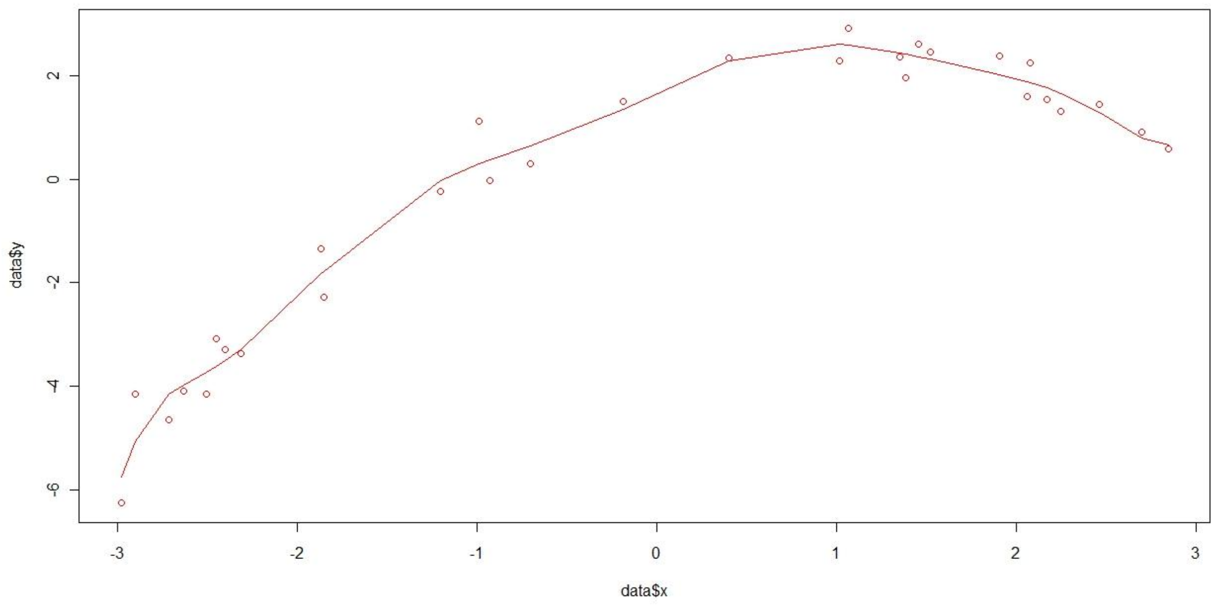
K = 7



K = 8



K = 9



K = 10

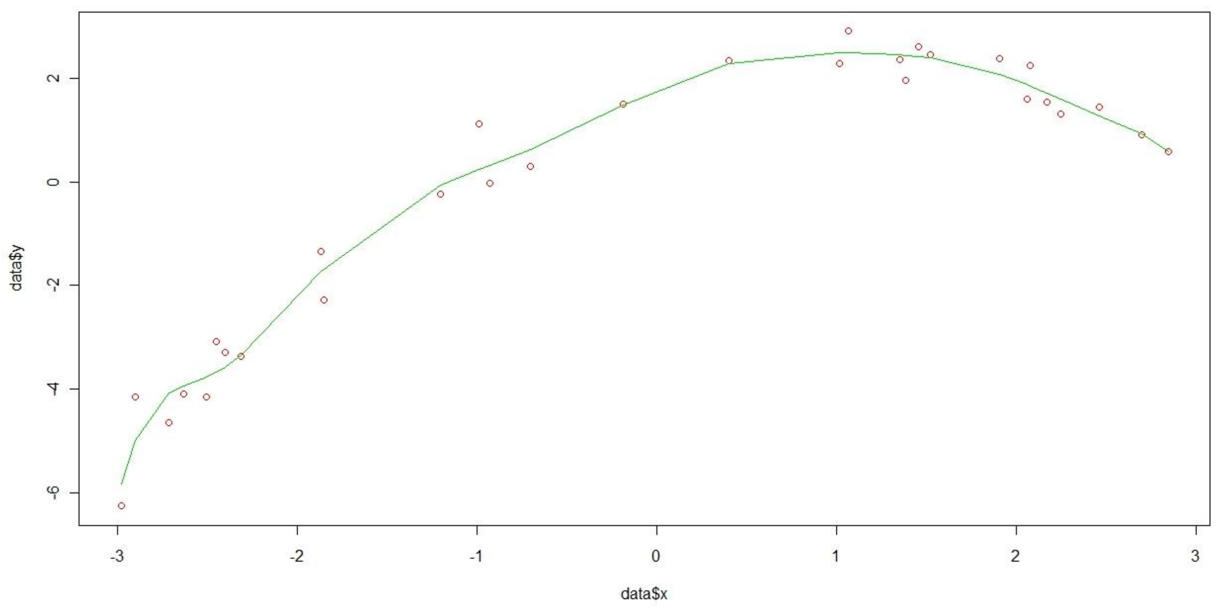


Figure 2.1: 11 plots (k = 0 to 10) that show curve fitting on sample data points.

b) Computing coefficient of determination $\boldsymbol{R^2}$:
Commands: >poly_coeff(data)

Ouptut:

```
[1] 0.0000000 0.7518009 0.9762191 0.9771467 0.9772756 0.9775266 0.9775272
[8] 0.9776297 0.9782560 0.9795144 0.9801740
```

Plotting the coefficient data w.r.t K :

```
>coeff = poly_coeff(data)
>k = c(0:10)
>plot(k ,coeff,col=”red”)
```

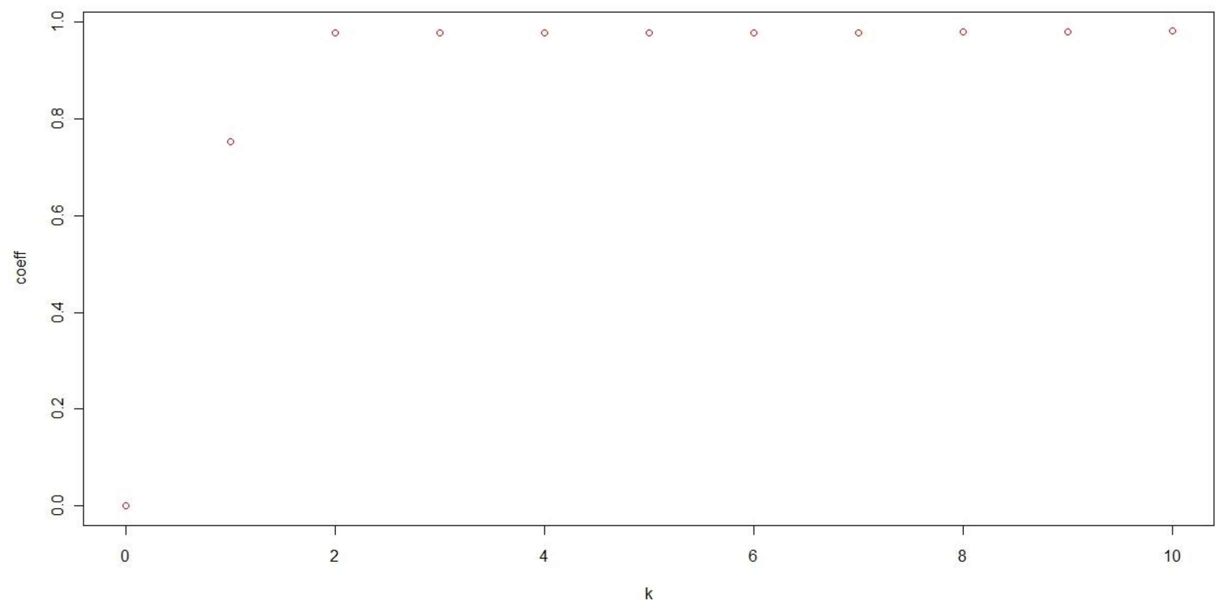


Figure 2.2: Plot of Coeff of R^2 and K.

This suggests that as the value of K increases the coefficient value increases but it saturates and almost becomes constant at higher value of k. This implies that the curve fitting is correct and higher degree of polynomial for higher value of k fits the data more appropriately.

Exercise 2b

1. 10-Fold cross validation set. This is obtained by running the source code in the initial stage. The variable data_new holds the partitioned data in to 10 subsets.

```
>data_new
b        x          y
1   1  1.4541011  2.62029347
2   1 -2.5063658 -4.15910660
3   1  1.9056847  2.38195754
4   2 -0.9271486 -0.02466637
5   2 -1.8666740 -1.34763835
```

2. The plot for sum of errors for each K is:

```
>poly_func_new(data_new)

[1] 230.372888  66.520509   6.420849   6.738077   7.622034   8.391782
[7]  10.428114  12.979676  14.115809  15.570560  14.003656
```

The corresponding plot is:

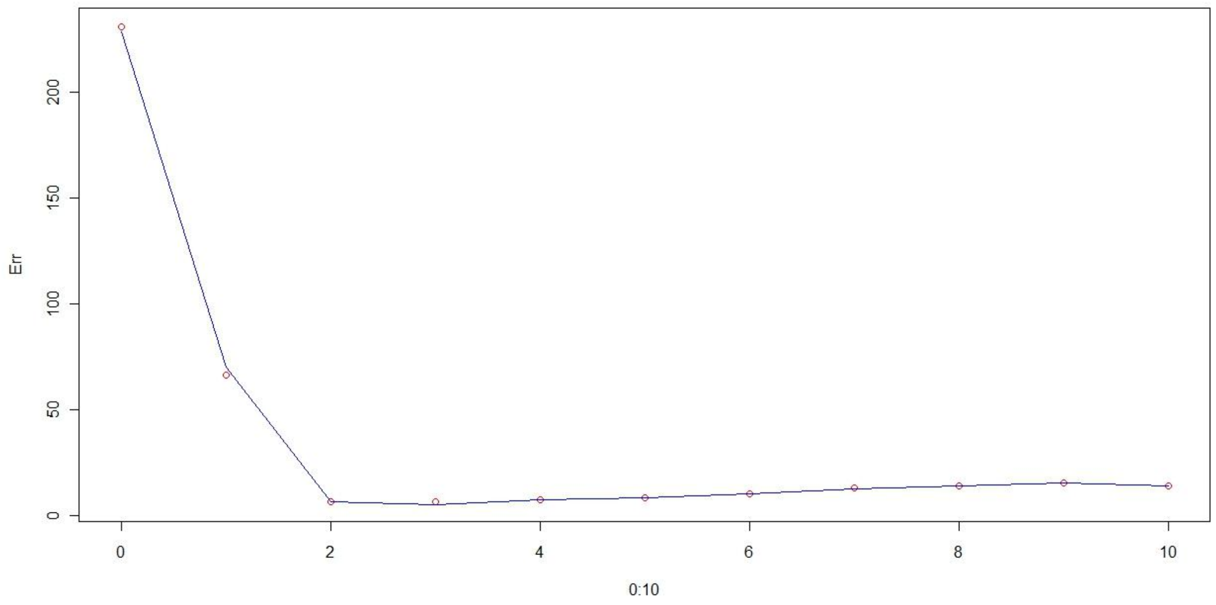


Figure 2.4: Plot of Error vs K = 0 to 10.

The curve initially has very high sum of square Errors for K = 0, then it starts decreasing reaches a minimum at K = 2 which has value of 5.00 and then it increases until k = 9 and finally it decreases to lower value at K = 10.
Hence we can conclude from this result is that for K = 2 the polynomial has minimum sum of squared error.
The cross-validated error does not improve with increasing values of K.
The corresponding coefficients can be obtained for K= 2 by:

```
> R = polynomial_func(2,data)
> R[[2]]
         [,1]       [,2]        [,3]
[1,] 1.858595   1.012805    -0.4910542
```

If we compare this result with given function y = 2+ 1*x + (-0.5)*x^2
We can observe that the respective coefficient values are approximately similar.
Hence the polynomial of order k = 2 fits the data approximately.
N.B For different iterations of running the source code i.e > source('Exercise5.r') the value of k for which the sum of error is minimized also changes. But still the coefficient values obtained as above, the first 3 values are approximately similar to that of the given polynomial. But in most of the test runs the minimum was found for K= 2.

Exercise 3

Instructions: Copy the files “Exercise3a.r”, “Exercise3b.r” and “Exercise3b_new.r” “word_doc” into 20news/ folder.

1. Create training set of 90% of documents from each newsgroup and test set wit 10% of documents.

Ans: Commands: >source(“Exercise3a.r”)

This results in output of train_set and test_set. Samples can be viewed by:

```
>train_set[[1]]
 [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
[19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
[37]  37  38  39 … 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
```

This gives list of documents that belong to newsgroup 1.

```
>test_set[[1]]
[1] 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451
[20] 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470
[39] 471 472 473 474 475 476 477 478 479 480
```

This gives list of documents that belong to news group 2.

2. Estimate a) Marginal probability of each newsgroup.
b) Probability that a document from that newsgroup that contains that word.

Ans:

Commands:>source(“Exercise3b.r”)

From Bayes theorem we have:

$P(g|d_i) = P(g) \times \Pi_{w \varepsilon d} P(w|g)$

where, $P(g|d_i)$ = the posterior probability of a newsgroup given it contains a list of documents.

$P(g)$ = prior probability of the newsgroup calculated as = N_g/N

N_g = number of documents that particular newsgroup contains.

N = Total number of documents available.

$P(w|g) = (N_{wg} + 1)/(N_g + 2)$

Thus, logarithm of $P(g|d)$ was estimated and stored as file “dg_mat_log”.

Top 200 words for each newsgroup based on their decreasing probability was printed out as:

Most probable word list of newsgroup : 1

the of in to and that is writes it you not be edu for this are have but on if article as or what with so an one by there do can all your they was no about from at don we would people com some who my any which more think me than has god say just then other will he only how like could out know were their does apr why because well when being even see been up said such them time many his should way here something must these keith those believe very now get make also into most things point re did thing right much good am us religion take seems same had read mean may ve doesn first really fact atheism since sure might our him true anyone wrote own nothing someone again little cannot example want too anything where its world isn question wrong made another life religious claim before course rather after atheists agree try without over seem actually however case never still long come system evidence two go makes jon reason moral better cs therefore atheist saying yes part give bible different others christian ll above find every understand going says

This output clearly indicates that stop words such as the, of, in, to etc occurs more frequently and should have more probability in general. As such estimation is correct.

Estimation of Marginal probability of each newsgroup: From the equation above the marginal probability was estimated. This can be viewed in the command prompt as:
>unlist(mrg.prob)
The output gives a list of probabilities:

```
[1] 0.04259473   0.05155737   0.05075872   0.05208980   0.05102494   0.05253350
 [7] 0.05164611  0.05253350   0.05288846   0.05271098   0.05306593   0.05271098
[13] 0.05244476 0.05271098   0.05262224   0.05315467   0.04836277    0.05004881
[19] 0.04117490  0.03336587
```

Exercise 3b

Commands: >source("Exercise3b_new.r")
>conf.train
>conf.test

Output:
Confusion matrix for training set and test set are shown in Figure 3.1 and 3.2 :

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	411	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	5	4	5
2	0	363	5	11	1	16	2	1	0	0	1	52	0	0	4	3	2	11	43	8
3	1	2	403	13	0	14	0	0	0	0	0	35	0	0	1	6	3	5	32	0
4	0	2	1	425	1	3	2	0	0	0	0	44	0	1	1	5	7	6	28	2
5	0	2	1	8	367	2	0	0	1	0	0	48	1	1	1	3	7	7	66	3
6	1	1	3	1	0	478	0	0	0	0	0	21	0	0	0	4	2	1	21	0
7	0	1	2	33	6	0	274	16	1	1	2	36	6	0	3	4	22	17	93	7
8	1	0	0	0	0	2	1	433	0	1	0	5	0	0	0	1	14	16	58	1
9	1	1	0	0	0	0	3	1	450	0	0	2	0	0	0	2	19	3	54	0
10	0	1	0	0	0	1	0	0	0	461	6	1	0	0	1	2	6	9	45	2
11	0	0	0	0	0	0	0	0	0	0	516	2	1	1	0	1	0	1	14	2
12	0	0	0	0	0	0	0	0	0	0	0	526	0	0	0	0	2	1	6	0
13	0	1	0	14	0	0	0	1	0	0	0	48	383	0	3	6	16	9	45	6
14	0	0	0	0	0	0	0	0	0	0	0	2	1	503	0	6	2	3	17	1
15	0	1	0	0	0	1	0	0	0	0	0	1	0	0	495	3	1	4	25	3
16	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	529	2	3	2	2
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	482	2	5	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	506	2	0
19	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0	3	411	0
20	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	4	3	5	315

Table 3.1: Confusion matrix table for training set. Rows indicate the actual class and columns indicate the predicted class.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0
2	0	40	0	1	0	2	0	0	0	0	0	6	0	0	0	0	1	1	6	1
3	0	0	40	0	0	2	0	0	0	0	0	11	0	0	0	0	0	0	4	0
4	0	0	0	50	0	1	1	0	0	0	0	2	0	0	0	0	2	0	3	0
5	0	0	1	1	48	0	0	0	0	0	0	4	0	1	0	0	0	0	2	0
6	0	1	0	0	0	53	0	0	0	0	0	1	0	0	0	0	0	2	2	0
7	0	0	0	1	0	1	33	3	0	0	0	5	1	0	1	1	2	1	7	2
8	0	0	0	0	0	0	0	44	0	0	0	1	0	0	0	2	2	2	8	0
9	0	0	0	0	0	0	0	0	50	0	0	1	0	0	0	1	1	0	7	0
10	0	0	0	0	0	0	0	0	0	49	0	0	0	0	0	0	2	1	7	0
11	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	1	3	0
12	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	3	0
13	0	0	0	0	0	0	0	0	0	0	0	7	34	1	1	1	3	2	10	0
14	0	0	0	0	0	0	0	0	0	0	0	1	0	58	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	54	0	0	1	3	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57	1	2	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	1	1	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	55	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	35

Table 3.2: Confusion matrix table for test set. The rows indicate actual class and columns indicate the predicted class.

Form table 3.1 we can find that newsgroup 7 got mixed up with most of other newsgroup items.
From table 3.2 we can find that newsgroup 7 got mixed up with most of other newsgroup items.

Exercise 4
Instructions: Put the source codes “jaccard_fun.r”, “Exercise4.1.r”, “Exercise4.2.r” in the directory “movielens/”

Output:

1. Create a function that gives two different movie ids and outputs the Jaccard coefficient.

    Jaccard Coefficient is given by J = |A intersect B|/ |A union B|
    where, A = number of users who rated a movie 1
    B = number of users who rated a movie 2

    Commands:
    ```
    >source(“jaccard_fun.r”)
    >jaccard_fun(1, 4)
    ```

    Output: 0.2910156

For all the below questions the command is: >source(“Exercise4.1.r”)

2. What is the Jaccard coefficient between 'Three Colors: Red' and 'Three Colors: Blue'?

Ans:
```
    [1] 0.5978261
```

3. What are the 5 movies with highest Jaccard coefficient to 'Monty Python and the Holy Grail'?

Ans:
```
Movies with highest Jaccard Coefficient to Monty python and Holy Grail
[1] Monty Python and the Holy Grail (1974)
[2] Raiders of the Lost Ark (1981)
[3] Blues Brothers, The (1980)
[4] Back to the Future (1985)
[5] Indiana Jones and the Last Crusade (1989)
```

4. What are the 5 movies with highest Jaccard coefficient to “Basic Instinct (1992)”?

Ans:
```
Movies with highest Jaccard Coefficient to Basic Instinct (1992)
[1] Firm, The (1993)
[2] Platoon (1986)
[3]Basic Instinct (1992)
[4] Abyss, The (1989)
[5]Die Hard 2 (1990)
```

Exercise 4b:

Commands: >source("Exercise4.2.r")

I Selected 20 movies based on the the genre only. The genre are: war,scifi,action,documentary, musical,unknown. The movie list ids are:

Genre	**Movie id**
Action	2,4,17,21
Documentary	32,48,75,115
Musical	21,71,91,95
Sci-Fi	7,38,39,50
War	10,22,31,50
Unknown genre	267, 1373

Here movie ID 21 belongs to Musical as well as Action and Movie ID 50 belongs to Sci-Fi as well as War genre.
Hence the priori statement states that in hierarchical clustering these 20 movies should cluster according to their distance value of binary genre attributes. Those with unknown genres might or might not get clustered separately since genre is unknown and those with similar genre attributes should get clustered together.

1. Jaccard coefficient was calculated in perspective of genre binary attributes of these movies as matrix.
2. Coefficient matrix was converted to 'dist' format.
3. 'hclust' function was used with methods 'simple' and 'complete' to evaluated the cluster.
4. Finally the dendogram was plotted.

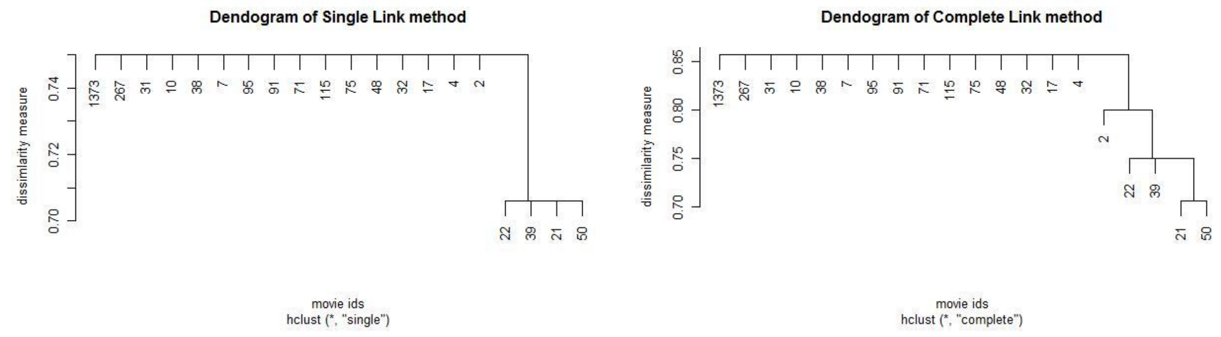


Figure 4.1: Dendogram plot for Single link and Complete link hierarchical clustering.

Output :

The plotted Dendogram tree for "Single" and "Complete".

Conclusion: For 'single-link 'the movie IDs 22, 39, 21, 50(War, Sci-Fi, and Action). This is quite reasonable as all Sci-Fi and War movie are Action based movies. For 'complete-link' the movie IDs 21, 50 form cluster (Action and War) and they are dissimilar from 22 and 39 (War and Sci-Fi) .