Report: PRML Assignment 2

Keval Dodiya (CS19M023)

Karan Jivani (CS19M030)

Problem 1

Part 1: Plot the Dataset.

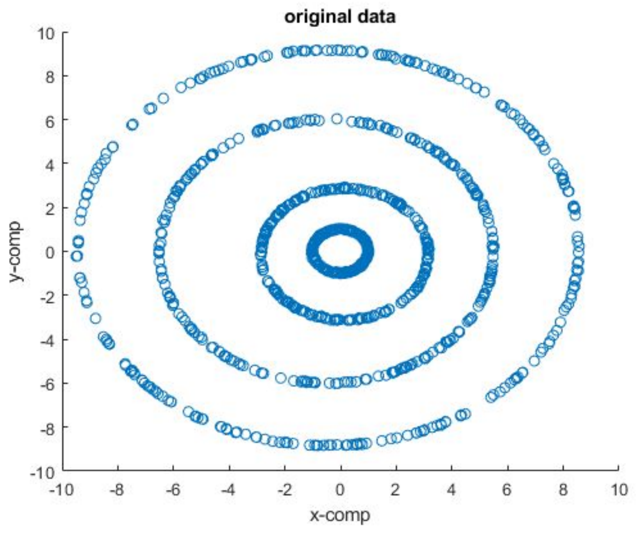
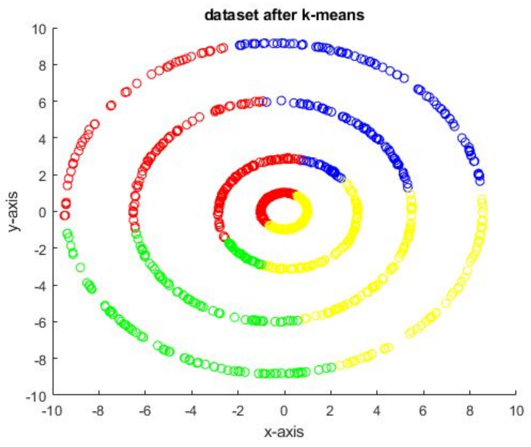
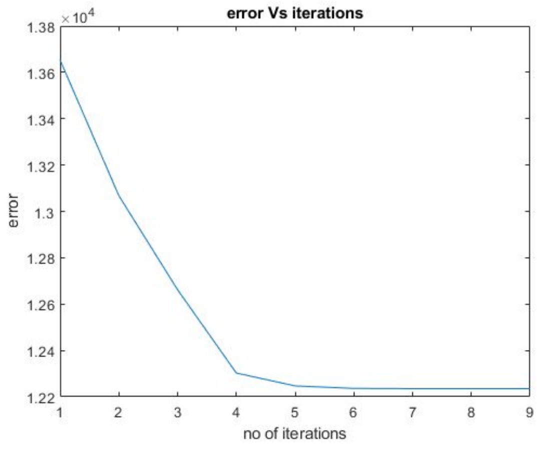


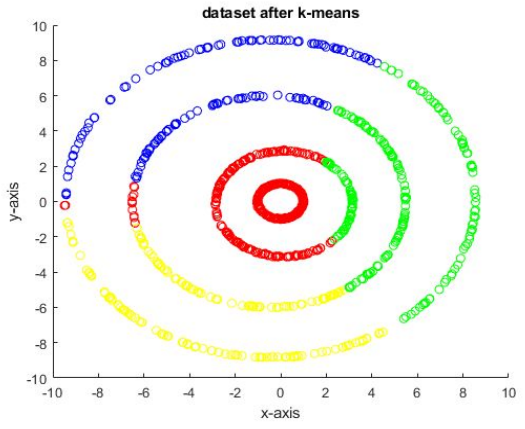
Figure 1.1 : Original Data

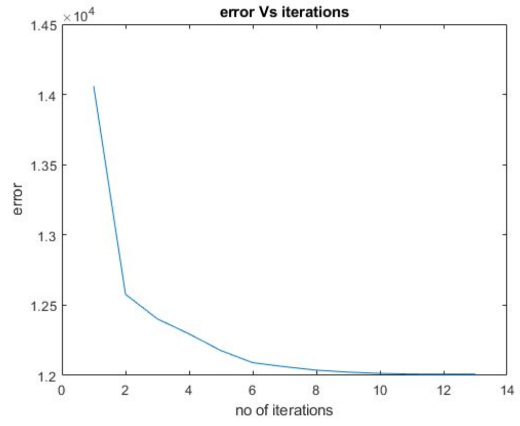
1st random initialization:



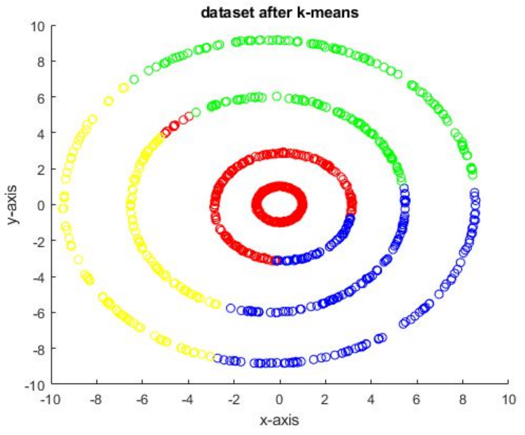


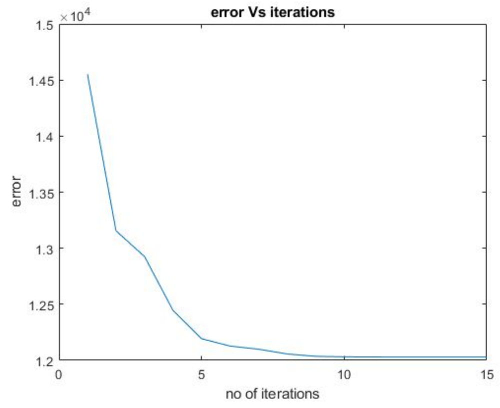
2)


dataset after k-means
y-axis
x-axis

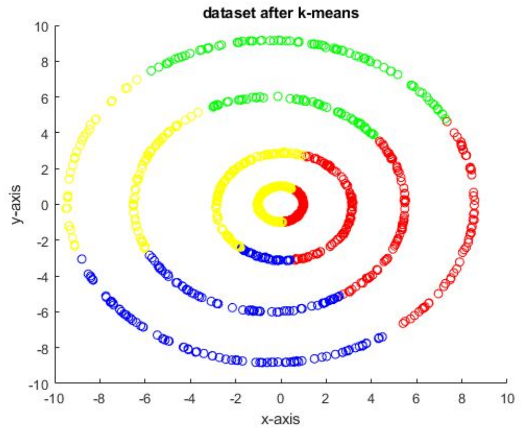

error Vs iterations
error
no of iterations

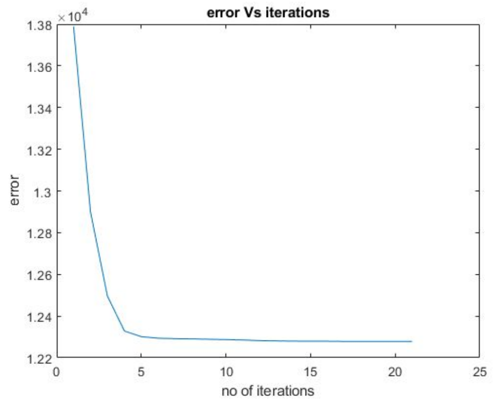
3)


dataset after k-means
y-axis
x-axis


error Vs iterations
error
no of iterations

4)


dataset after k-means
y-axis
x-axis


error Vs iterations
error
no of iterations

5)

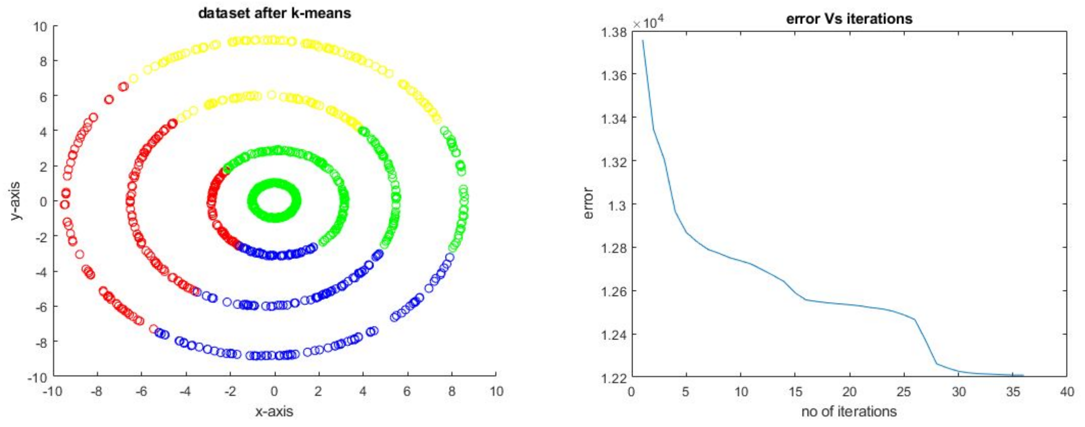
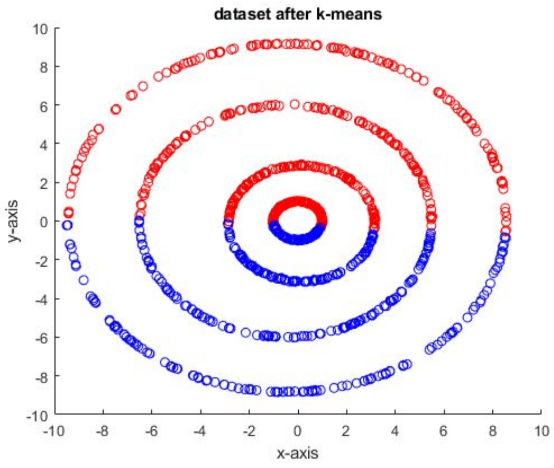
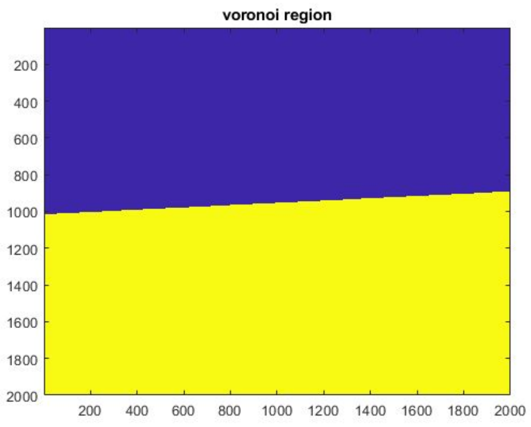


Figure 1.2 : clustered data with its error Vs iterations graph

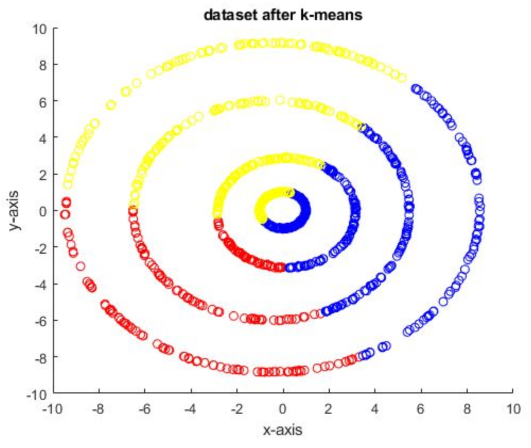
Part 2 :

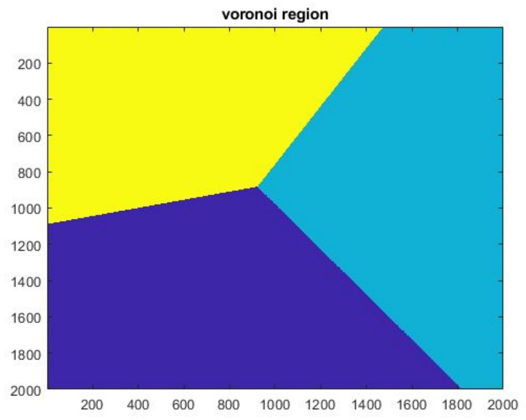
k=2 :



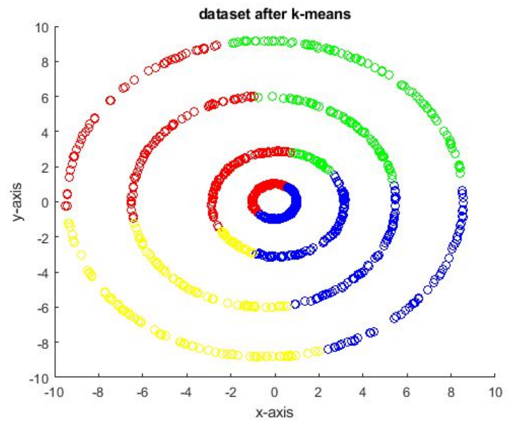


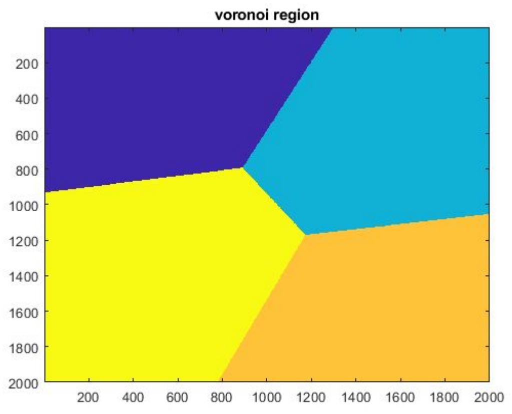
k=3 :



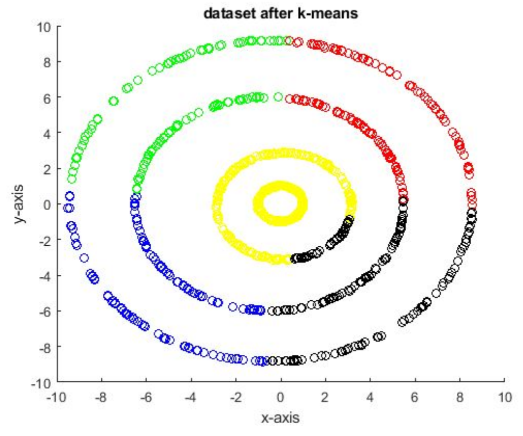


k=4 :





k=5:



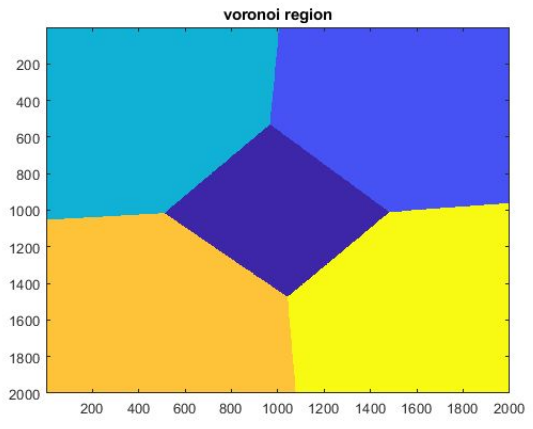
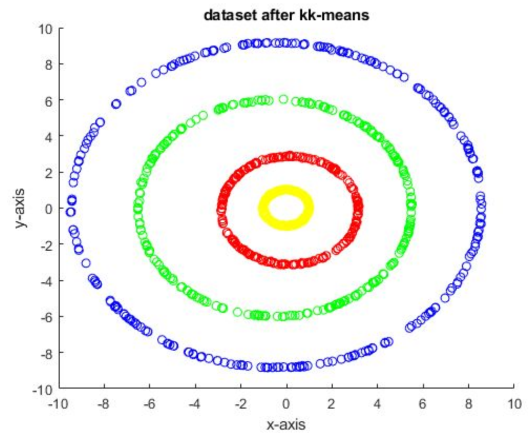


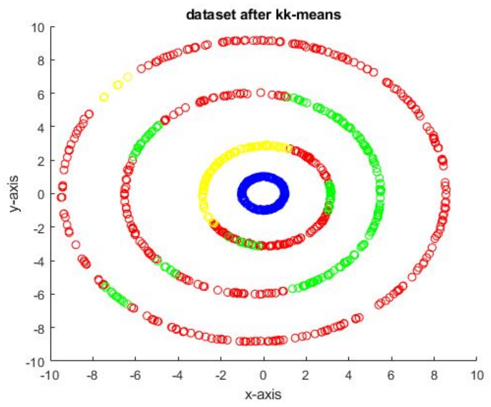
Figure 1.3 : Voronoi region(right image) with its cluster image.

Part 3 :

We took gaussian kernel with different sigmas(1,2,3,4,5). And observes clusters in each case. And we got best cluster when sigma=4 that images given below.



(a)



(b)

Figure 1.4 : left image indicates clustering with sigma = 4 ,right image indicates clustering with sigma = 1.

Here we can observe that (a) gives best clusters ,
So in conclusion gaussian kernel with sigma = 4 is best kernel.

Problem 2

Part 1: Find the least squares solution w_{ML} to the regression problem using the closed form expression.

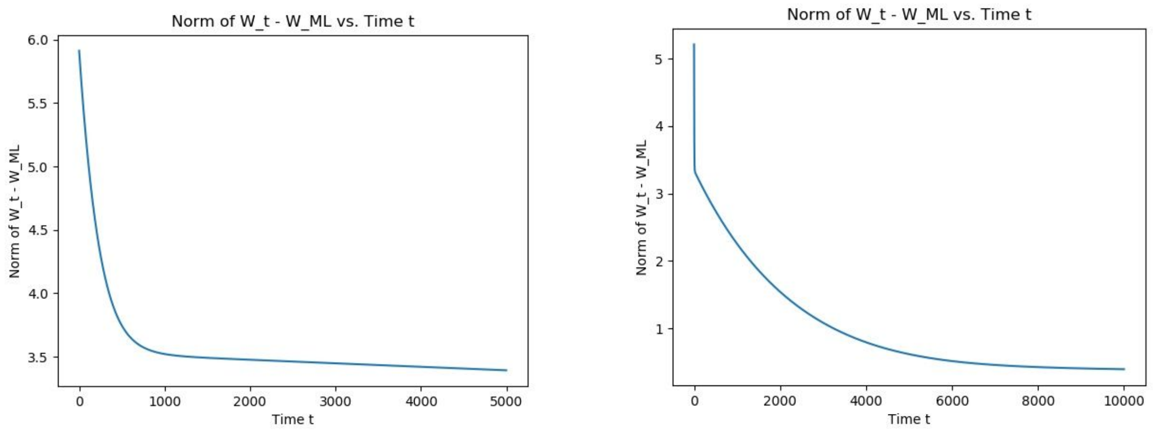
To calculate w_{ML}, closed form solution is given as,

$$w_{ML} = (X^T X)^{-1} X^T Y$$

w_{ML} :

```
[ 1.92296782e-02   -8.22857432e-03   -1.40735081e-02   -4.02942767e-03
2.31220046e-03     1.88157124e-01    2.23820069e-03    9.11822774e-03
1.79432430e-01     3.32884968e-03    4.99250666e-01    7.98628410e-03
3.94689676e-03     1.38609783e-02    3.59213519e-03    8.95875251e-03
-1.39801208e-03    3.00560684e-03    8.39456862e-04   -9.80758689e-03
-2.86553616e-03   -1.21436606e-02   -1.05889398e-02    7.54500426e-03
-1.04598177e-02    5.75759844e-03   -5.05271340e-03   -3.41199334e-03
7.73777160e-03     1.16039534e-02   -7.23226013e-03   -8.98924102e-03
6.16057636e-04     4.57696298e-03    5.58297065e-03   -1.38113067e-02
7.16306770e-04     1.28915820e-02   -1.18363478e-02   -2.05111364e-02
5.84691974e-01     1.37222430e-05   -1.18194742e-03   -3.11318620e-03
-9.89442065e-03   -5.82715877e-03    9.37398615e-03   -7.15157795e-03
-8.01112765e-04    6.61921837e-03    3.12067233e-02    4.51354488e-01
-9.08772338e-03    2.17671679e-03    3.84588863e-03    2.89396702e-01
6.70717489e-03    -2.34622352e-03    1.37897608e-02   -1.10592729e-02
7.72355598e-01    -6.10805912e-03   -1.02578098e-03    6.13349340e-03
-5.17696632e-03    4.24392631e-03    4.12683901e-03   -3.42663565e-03
7.99425703e-03    -2.84380183e-03    4.96739446e-04    7.35029087e-04
-2.21806185e-03   -1.08161666e-02   -9.69382179e-03   -1.61355795e-03
5.97350857e-01    -5.16556909e-03   -1.17370075e-02    2.50269313e-03
8.48727704e-01    -1.13316243e-02    1.88729575e-03   -1.69365387e-03
-6.74076221e-03    5.45921745e-03    5.37247426e-03    4.95715764e-03
-8.59069143e-03    1.26217457e-02   -2.73665605e-03   -4.92873681e-04
8.21922290e-03    -4.74208032e-03    5.30529728e-03    7.15170746e-03
-9.81804403e-03    1.78895132e-02   -1.53275896e-03   -3.01499638e-03
-9.00650232e-03]
```

Part 2: Solve using gradient descent and Plot || w^t - w_{ML} || as a function of time *t*.



(a) (b)

Figure 2.1: Plots of || w^t - w_{ML} || vs. time t with, Number of Iterations = 10000 and (a) Learning Rate = 0.0001 and (b) Learning Rate = 0.005. Training done with gradient descent.

- w_{ML} is same as Part 1, w^t after running gradient descent, with learning rate = 0.005 and number of iterations = 10000 (Fig. (2.1 b)), is :

```
W using gradient descent :
[-3.60311007e-01  -7.51378008e-03   -4.86994323e-03   8.36910560e-03
1.50310773e-02    1.94862331e-01    5.41556250e-03    1.09366426e-02
1.77163525e-01    1.81634634e-02    5.06600647e-01    1.09550706e-02
1.94066434e-02    2.88064968e-02    9.16210025e-03    1.82545249e-02
1.15483784e-02    1.68027187e-02    6.42324742e-03    2.15519483e-03
1.03974528e-02    1.54523718e-04    -1.03549033e-02   2.14329407e-02
6.21685767e-05    1.20476000e-02    1.24149503e-02    4.23807634e-03
2.05474602e-02    2.59140219e-02    8.45322452e-04    -1.00660662e-02
6.49525195e-03    2.44078991e-02    1.81015953e-02    4.09628535e-04
1.42050533e-02    1.70281511e-02    7.44829090e-03    -1.60603980e-02
5.76271323e-01    8.07754254e-03    8.10922241e-03    1.32427356e-03
5.62471581e-04    5.03722654e-03    1.17497614e-02    -1.08583697e-02
1.06886884e-03    6.20081787e-03    4.63883475e-02    4.53213514e-01
-5.46194485e-03   1.61313289e-02    1.10592993e-03    3.05650493e-01
9.40874950e-03    7.37212304e-03    2.35038385e-02    -1.26691439e-03
7.54923394e-01    -2.73282744e-03   3.48182279e-03    1.96383587e-02
-1.37648421e-03   1.30729753e-02    1.55837784e-02    8.01850208e-03
2.48898484e-02    -2.63630889e-04   -2.79362546e-03   5.93138306e-03
2.58376435e-03    -1.16539924e-02   5.00652930e-03    3.71181778e-03
5.98997357e-01    9.52999769e-03    -2.92489734e-03   9.11618490e-03
8.39108772e-01    -6.27232929e-03   5.51071424e-03    7.29794405e-03
-4.62310880e-03   1.07331408e-02    6.15556053e-03    1.99069023e-02
-1.84071500e-03   2.60334340e-02    -3.11878372e-03   1.28354346e-02
2.40313150e-02    5.11527489e-03    1.04878807e-02    2.25994833e-02
-3.72239557e-03   3.10255345e-02    1.70675214e-02    -2.50381278e-03
-1.07304252e-03]
```

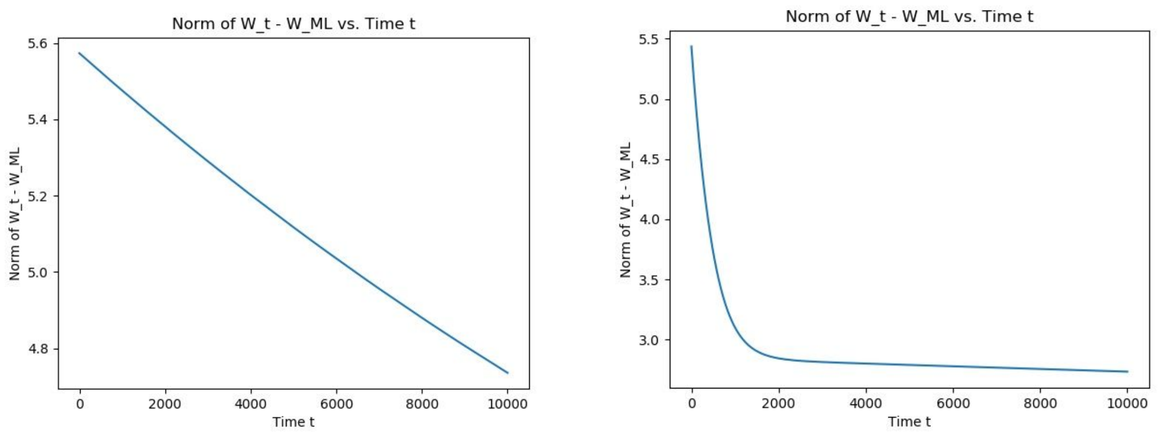
- It can be observed from the Fig. (2.1 a) and Fig. (2.1 b), that error in w^t with respect to w_{ML} is quite less, 3.342384 if learning rate is 0.0001 and 0.392643 if learning rate is 0.005. Thus, w^t converges to w_{ML} after running through gradient descent. Rate of this convergence depends on learning rate of gradient descent. Higher learning rate means faster convergence (Fig. (2.1 b)).

- Cost function value on training set with:
    - w_{ML} is 0.01984260543439583
    - W^t (with Learning Rate = 0.0001) is 0.46947137981616466
    - W^t (with Learning Rate = 0.005) is 0.020246648405339806

Where, cost function is given as,

$$\frac{1}{2*n}\sum_{i=1}^{n}(x_i^T w^t - y_i)^2$$

Part 3: Solve using stochastic gradient descent with batch size = 100 and Plot || w^t - w_{ML} || as a function of time *t*.



(a) (b)

Figure 2.2: Plots of || w^t - w_{ML} || vs. time t with, Number of Iterations = 10000 and (a) Learning Rate = 0.0001 and (b) Learning Rate = 0.005. Training done with stochastic gradient descent and batch size = 100.

- w_{ML} is same as Part 1, w^t after running stochastic gradient descent, with batch size = 100, learning rate = 0.005 and number of iterations = 10000 (Fig. (2.2 b)), is :

W using stochastic gradient descent :

```
[-0.3376374   0.41872901  0.13643909  0.09173577  0.31496115  0.21796965
 0.2998409  -0.26951456  0.12158804 -0.23729124  0.10945596 -0.2441048
 0.20627842  0.11075743 -0.40402632 -0.06687975 -0.42358567  0.41957762
-0.08664896  0.24319359 -0.16449209 -0.27902217 -0.19801669 -0.40384242
-0.32708433  0.30188863  0.4687764   0.3143786  -0.07285282  0.24180079
 0.09953274 -0.00539733  0.11100276 -0.05620358 -0.28964756 -0.13137568
-0.04021705  0.27342427  0.419832    0.13865569  0.23857831 -0.01814736
 0.28068715 -0.19470078 -0.36554479 -0.29612938  0.03425724 -0.38652586
-0.32168769  0.20274032  0.19657802 -0.09265113  0.42046704 -0.07564955
 0.09889004  0.12799842  0.13122208  0.42748557 -0.11549499  0.13065538
 0.49956404 -0.20524126 -0.42867504  0.20598038  0.09380004 -0.41197516
 0.20113629  0.26107476 -0.35423468  0.47018307 -0.11824101  0.36657025
-0.05783251  0.1676098   0.49531748  0.11646753  0.4807265   0.4096124
 0.1962029  -0.01525237  0.31981287  0.4466133   0.06560276  0.02963396
 0.09169389 -0.14233376 -0.24411641  0.00529877  0.36557084 -0.20870676
 0.37138202  0.12708071 -0.13957802 -0.01175306  0.52540612  0.05631798
-0.32810696 -0.11228083  0.03982309 -0.35447296  0.02657466]
```

- It can be observed from Fig. (2.2 a) and Fig. (2.2 b), that error in w^t with respect to w_{ML} is around 2.7 even if the learning rate is 0.005 and if the learning rate is 0.0001, error is around 4.7 which is quite high compared to gradient descent.

- But, here in stochastic gradient we use batches of data of certain fixed size to train the value of w^t and data points for these batches are selected randomly. Since, the data points are selected randomly and for each iteration of training, only batch size number of data points are used, which is quite less compared to the total number of data points (*n*), training rate is quite slow, and therefore, even with learning rate = 0.005, convergence is quite slow, that is in 10000, error in w^t relative to w_{ML} is still higher compared to relative error in w^t trained with gradient descent. This is also reflected in cost function value on training set.

- Cost function value on training set with:
  - w_{ML} is 0.01984260543439583
  - W^t (with Learning Rate = 0.0001) is 164.50629036882694
  - W^t (with Learning Rate = 0.005) is 0.32133036849302893

  Where, cost function is given as,

$$\frac{1}{2*n}\sum_{i=1}^{n}(x_i^T w^t - y_i)^2$$

Problem 3

Part 2 : Use the gradient descent for ridge regression to cross validate for various values of λ on the validation set extracted from the original training set. Here the ratio of validation set to new training set is 2:8. Plot of validation error vs. λ is given as,

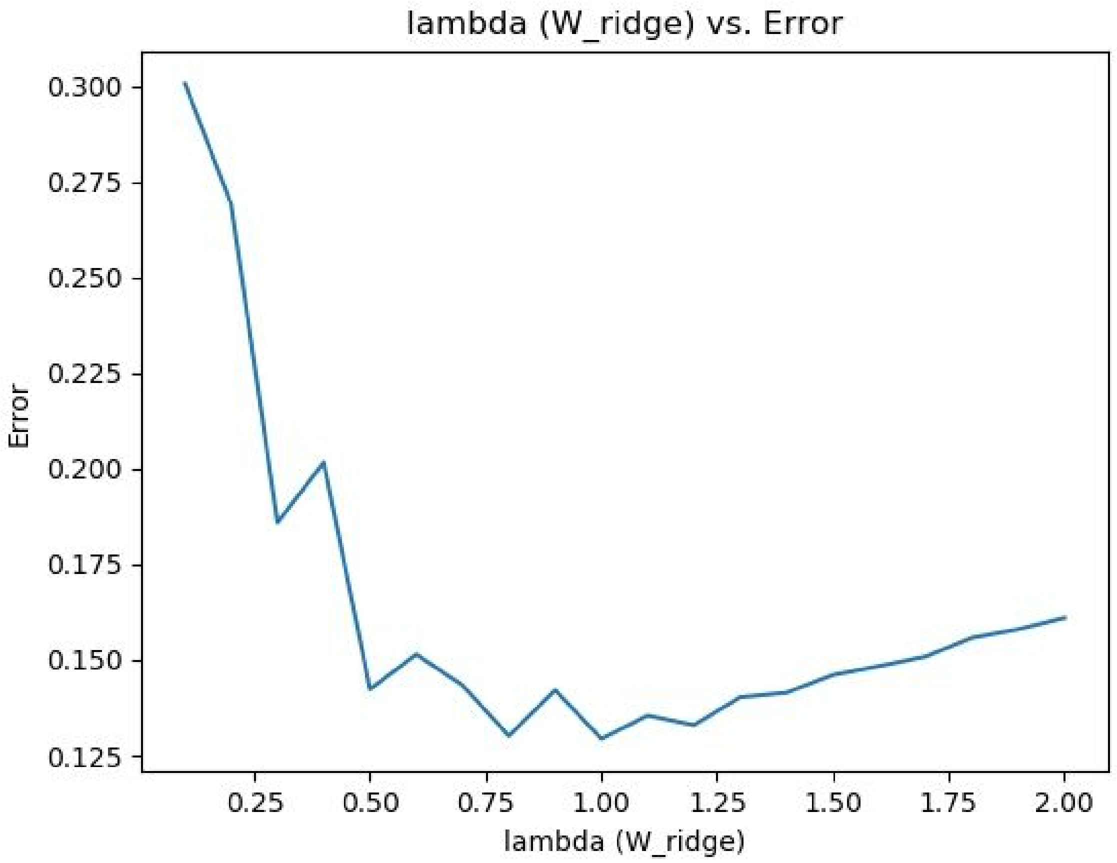


Figure 3.1: Plots of Validation Error vs. λ with gradient descent for ridge regression. Values of λ is taken from the set {0.1, 0.2, 0.3, …, 2.0}.

- From the Fig. (3.1), we chose the λ for which the error is least and w^r is the corresponding solution for ridge regression with the best λ value. Also, learning rate (0.0001) and number of iterations (10000) were chosen high enough that value of w^t for each value of λ can converge.

- Error (value of cost function) in test set using:
  - w_{ML} is 0.18607622512696873
  - W^r is 0.12072888075800238

- Thus, we can observe that, w^r gives less error in test set compared to w_{ML}. Thus, even though w_{ML} fits better in training set, w^r predicts better for testing set.

Part 3 : Use the coordinate descent for LASSO regression to cross validate for various values of λ on the validation set extracted from the original training set. Here the ratio of validation set to new training set is 2:8. Plot of validation error vs. λ is given as,

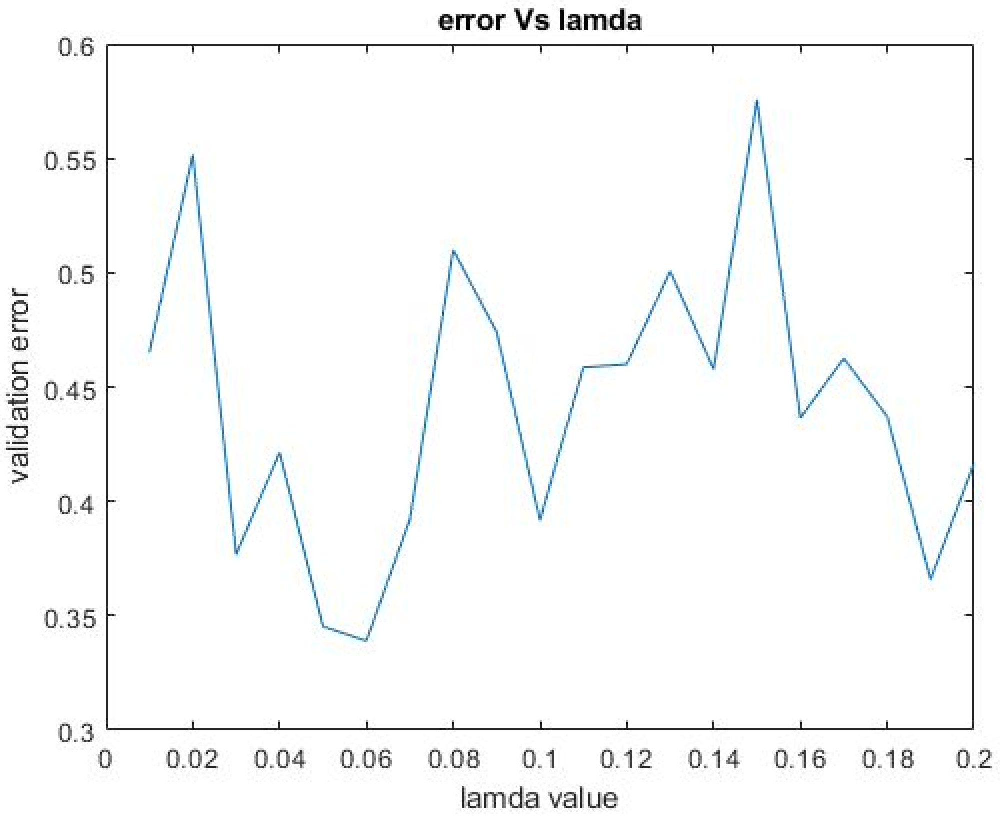


Figure 3.2: Plots of Validation Error vs. λ with coordinate descent for LASSO regression. Values of λ is taken from the set {0.01, 0.02, 0.03, …, 0.2}.

- From the Fig. (3.2), we chose the λ for which the error is least and w^{lasso} is the corresponding solution for LASSO regression with the best λ value by coordinate descent.

- Error (value of cost function) in test set using:
    - w_{lasso} is 0.003776240919419 with 10000 iterations
    - W^r is 0.12072888075800238 with 10000 iterations.

- Thus, we can observe that, w^{lasso} gives less error in test set compared to w_{ML} because its add bias and it sets parameter to zero which is near to zero(kind of remove them). Thus, w^{lasso} is better when more parameters are near to zero.

Note: Tools Used: Matlab, Python [Numpy (Array, Zeros, Ones, Sum), Matplotlib]