PRML04

March 16, 2022

1 Report

1.1 Question-1

50 uniformly distributed random numbers between [0,1] are generated.

1.2 Question-2

Training dataset is generated using the 50 X values generated in question-1.

1.3 Question-3

Similarly testing dataset is generated using the same functions generate_X() and generate_Y() for 500 points.

1.4 Question-4

Regularized least square kernel regression method is used over training data to generate estimations. These estimations weights are used to calculated estimated Y values for both train and test. We further calculated the RMSE for train and test datasets. (To check plottings, go to section Question-4. Observations are discussed in question-6)

1.5 Question-5

Same mathematical calculations are performed after generating 5 random outliers in training dataset. We calculated RMSE for both training and testing datasets. Training estimation function is plotted against the actual mean estimates. Check section Question-5 for graph.

1.6 Question-6

Observations---------> 1.) Training RMSE without outliers = 0.042218699335702735

2.) Testing RMSE without outliers = 0.0690993470657655

3.) Training RMSE with outliers = 0.47231799934807783

4.) Testing RMSE with outliers = 0.18510363304553143

After adding the outliers, the curve moves towards the outliers little bit but not completely. These outliers increase the training RMSE and because of the moved and readjusted estimation function, testing RMSE also increases.

This basically means that normally the estimation process is more sensitive towards existence of outliers.

1.7 Question-7

L1-norm loss kernal regression method is used to calculate the estimated weights after performing the gradient descent techniques on both without outliers and with outliers training dataset. Further, Training and testing RMSE is calculated using L1-norm loss kernel regression. To check the plots, go to section Question-7.

Observations --------> 1.) Training RMSE using L1-norm loss kernel regression without outliers = 0.19059812434904114

2.) Testing RMSE L1-norm loss kernel regression without outliers = 0.156841121339253

3.) Training RMSE L1-norm loss kernel regression with outliers = 0.8777220051710518

4.) Testing RMSE L1-norm loss kernel regression with outliers = 0.1517411766204316

Here we can see that even after adding the outliers in the training data, testing RMSE is not much affected. Although training RMSE is high becuase since estimation process didn't get much affected by outliers, therefore these outlier points is dominating for training RMSE.

Therefore, we can conclude that L1-norm loss kernal regression method is less sensitive towards existence of outliers.

2 Creating Datasets

3 Question-1

Generate 50 real numbers for the variable X from the uniform distribution U [0,1]

[ ]:
```
import numpy as np
import math
```

[ ]:
```
#### generating N uniformly distributed random variable.

def generate_X(N):
  X = np.random.uniform(size = N)
  return X
```

[ ]:
```
X = np.array(sorted(generate_X(50)))
print('50 Uniformly distributed random variables are :')
print(X)
```

```
50 Uniformly distributed random variables are :
[0.00140182 0.00764259 0.01888481 0.02885813 0.04034469 0.05051534
 0.06409562 0.08271648 0.11475257 0.14991701 0.17905035 0.22883379
 0.25524781 0.28933171 0.29069825 0.29159221 0.29343318 0.31769698
 0.33665388 0.34104922 0.36332431 0.37403382 0.4146254  0.43962655
 0.44898783 0.46099921 0.46713775 0.50108536 0.53560617 0.54699746
 0.60868223 0.65076134 0.69740676 0.71382742 0.71962655 0.7283669
 0.78271708 0.80050888 0.80612169 0.80905062 0.81787923 0.81896692
 0.83149017 0.8731152  0.88712687 0.88851544 0.91915003 0.96992502
 0.98332207 0.98974758]
```

4 Question-2

Construct the training set T = { (x1,y1),(x2,y2),.....,(x50,y50)} using the relation Yi = sin(2 (xi)) + i where i ∼ N(0,0.25) and xi is from R.

[ ]:
```
##### Generating Y values according to the given X values

def generate_Y(X, N):
  PI = math.pi
  Exp = 2*PI*X
  Epsilon = np.random.normal(0, 0.25, size = N)
  # print(Epsilon)
  Y = np.sin(Exp) + Epsilon
  return Y
```

[ ]:
```
Y = generate_Y(X, 50)
print('50 output for X datapoints are :')
print(Y)
```

```
50 output for X datapoints are :
[-0.12939004  0.69207138 -0.2187501   0.15149516  0.03790919  0.27831142
  0.31540969  0.2914958   0.7014624   0.89152253  0.89974374  0.83017699
  0.96437178  0.54233672  1.17077542  1.36990923  0.71751888  1.10909592
  0.65266102  0.75508381  0.70342181  0.91828724  0.2844156   0.27476816
  0.44221709  0.32747211  0.70342242  0.11049073 -0.18644321 -0.3659722
 -0.75430831 -1.14557802 -0.67694015 -0.89193314 -1.02179756 -1.03237084
 -1.16241297 -0.78087212 -0.99884548 -1.23455921 -0.68113206 -1.318849
 -0.91508754 -0.93362068 -0.67122065 -0.32687158 -0.44838001  0.18348578
  0.21517101  0.2855739 ]
```

[ ]:
```
##### Making a zipped dataset based on the X and Y values

def create_XY_set(X, Y):
  result = np.array(list(zip(X,Y)))
  # print(result)
  return result
```

[ ]:
```
training_set = create_XY_set(X, Y)
training_set
```

[ ]:
```
array([[ 0.00140182, -0.12939004],
       [ 0.00764259,  0.69207138],
       [ 0.01888481, -0.2187501 ],
       [ 0.02885813,  0.15149516],
       [ 0.04034469,  0.03790919],
       [ 0.05051534,  0.27831142],
       [ 0.06409562,  0.31540969],
       [ 0.08271648,  0.2914958 ],
       [ 0.11475257,  0.7014624 ],
       [ 0.14991701,  0.89152253],
       [ 0.17905035,  0.89974374],
       [ 0.22883379,  0.83017699],
       [ 0.25524781,  0.96437178],
       [ 0.28933171,  0.54233672],
       [ 0.29069825,  1.17077542],
       [ 0.29159221,  1.36990923],
       [ 0.29343318,  0.71751888],
       [ 0.31769698,  1.10909592],
       [ 0.33665388,  0.65266102],
       [ 0.34104922,  0.75508381],
       [ 0.36332431,  0.70342181],
       [ 0.37403382,  0.91828724],
       [ 0.4146254 ,  0.2844156 ],
       [ 0.43962655,  0.27476816],
       [ 0.44898783,  0.44221709],
       [ 0.46099921,  0.32747211],
       [ 0.46713775,  0.70342242],
       [ 0.50108536,  0.11049073],
       [ 0.53560617, -0.18644321],
       [ 0.54699746, -0.3659722 ],
       [ 0.60868223, -0.75430831],
       [ 0.65076134, -1.14557802],
       [ 0.69740676, -0.67694015],
       [ 0.71382742, -0.89193314],
       [ 0.71962655, -1.02179756],
       [ 0.7283669 , -1.03237084],
       [ 0.78271708, -1.16241297],
```

```
       [ 0.80050888, -0.78087212],
       [ 0.80612169, -0.99884548],
       [ 0.80905062, -1.23455921],
       [ 0.81787923, -0.68113206],
       [ 0.81896692, -1.318849  ],
       [ 0.83149017, -0.91508754],
       [ 0.8731152 , -0.93362068],
       [ 0.88712687, -0.67122065],
       [ 0.88851544, -0.32687158],
       [ 0.91915003, -0.44838001],
       [ 0.96992502,  0.18348578],
       [ 0.98332207,  0.21517101],
       [ 0.98974758,  0.2855739 ]])
```

[ ]:
```
print(training_set.shape)
```

```
(50, 2)
```

5 Question-3

In the similar way construct a testing set of size 500 i,e. Test = {(x'1,y'1),(x'2,y'2),.....,(x'500,y'500)}.

[ ]:
```
##### Appling same procedure as Question 1 and 2

test_X = np.array(sorted(generate_X(500)))
```

[ ]:
```
test_y = generate_Y(test_X, 500)
```

[ ]:
```
testing_set = create_XY_set(test_X, test_y)
```

[ ]:
```
print(testing_set.shape)
print(testing_set)
```

```
(500, 2)
[[ 2.09916520e-03 -3.17983055e-01]
 [ 2.14933710e-03 -1.20765683e-01]
 [ 2.65433488e-03  1.13099677e-01]
 [ 3.89023896e-03 -9.32264432e-02]
 [ 5.18137208e-03 -2.60240721e-01]
 [ 6.73712904e-03  1.87883329e-01]
 [ 8.32177660e-03 -1.47867066e-01]
 [ 9.33221873e-03  1.32868659e-01]
 [ 9.69345139e-03  3.28212993e-01]
 [ 1.39437317e-02 -2.36117980e-01]
 [ 1.45779998e-02  1.01065007e-01]
```

```
[ 1.51062718e-02  2.64722959e-01]
[ 1.52031583e-02  1.24633684e-01]
[ 1.73156934e-02  2.66262895e-01]
[ 1.98025381e-02  2.61655614e-01]
[ 2.00552122e-02 -1.47819127e-01]
[ 2.16664001e-02  1.52359984e-01]
[ 2.46377537e-02  3.58520423e-01]
[ 2.47092635e-02  1.89652649e-01]
[ 3.13261252e-02 -7.24296549e-02]
[ 3.13364401e-02  2.90674592e-02]
[ 3.14795392e-02  3.84731977e-01]
[ 3.18219593e-02  4.36252403e-01]
[ 3.24093461e-02  2.79684604e-01]
[ 3.63527419e-02 -1.95774892e-01]
[ 3.99173069e-02  5.32449563e-01]
[ 4.05726075e-02  7.05877491e-01]
[ 4.79054122e-02  4.00727774e-01]
[ 4.91742706e-02  1.57827709e-01]
[ 5.09987649e-02  1.89604260e-01]
[ 5.10005472e-02  3.11003543e-01]
[ 5.49632614e-02  3.95287874e-01]
[ 5.55315671e-02  6.08593752e-01]
[ 5.56032466e-02  2.81138288e-01]
[ 5.78504488e-02  3.00684522e-01]
[ 6.35913692e-02  1.43858287e-01]
[ 6.58076218e-02  1.40547447e-01]
[ 6.70878301e-02  4.78894993e-01]
[ 6.88462180e-02  7.47594508e-01]
[ 7.10167710e-02 -1.48017567e-01]
[ 7.34658757e-02  9.09580034e-01]
[ 7.53052147e-02  6.51907899e-01]
[ 7.70785750e-02  3.70638753e-01]
[ 7.72203653e-02  7.17555117e-01]
[ 8.39196625e-02  3.34550530e-01]
[ 8.53719296e-02  2.23340135e-01]
[ 8.56753731e-02  6.44519634e-01]
[ 8.61181713e-02  4.31662728e-01]
[ 8.87255025e-02  4.67995390e-01]
[ 9.55668310e-02  5.43798692e-01]
[ 9.66612558e-02  6.20033105e-01]
[ 9.74831030e-02  6.15003777e-01]
[ 9.91711084e-02  5.32746106e-01]
[ 1.00192910e-01  6.75717395e-01]
[ 1.04730342e-01  8.90095165e-01]
[ 1.06979043e-01  3.75615415e-01]
[ 1.07408568e-01  5.71629597e-01]
[ 1.08860320e-01  2.27469466e-01]
[ 1.16319254e-01  4.37293244e-01]
```

```
[ 1.17169672e-01  5.80069382e-01]
[ 1.18170898e-01  6.68790374e-01]
[ 1.20693771e-01  6.51316942e-01]
[ 1.21433940e-01  6.80110900e-01]
[ 1.23418197e-01  6.02640844e-01]
[ 1.26369399e-01  7.13557167e-01]
[ 1.26563818e-01  6.76644732e-01]
[ 1.31864920e-01  8.16215077e-01]
[ 1.34479489e-01  1.14338361e+00]
[ 1.34645595e-01  1.14111782e+00]
[ 1.36060611e-01  7.64703039e-01]
[ 1.38727054e-01  7.70358298e-01]
[ 1.41444243e-01  6.09966753e-01]
[ 1.46395778e-01  7.26690268e-01]
[ 1.47474516e-01  6.21526472e-01]
[ 1.47544499e-01  1.02282578e+00]
[ 1.47586907e-01  1.01351711e+00]
[ 1.49444510e-01  1.06879079e+00]
[ 1.50473519e-01  8.55849975e-01]
[ 1.51236273e-01  3.37790822e-01]
[ 1.51875203e-01  1.19314098e+00]
[ 1.52132257e-01  9.28314975e-01]
[ 1.56862506e-01  6.69453182e-01]
[ 1.60108310e-01  5.87397228e-01]
[ 1.62029921e-01  8.72361993e-01]
[ 1.62885733e-01  9.04690362e-01]
[ 1.67922077e-01  8.32864489e-01]
[ 1.69849001e-01  5.76253013e-01]
[ 1.70320348e-01  1.65068727e-01]
[ 1.70397893e-01  9.77437675e-01]
[ 1.70573010e-01  1.01817260e+00]
[ 1.70581077e-01  1.02469513e+00]
[ 1.73798601e-01  8.55923954e-01]
[ 1.76135409e-01  9.82727335e-01]
[ 1.78425464e-01  1.17292307e+00]
[ 1.81699118e-01  7.50120203e-01]
[ 1.87150492e-01  7.01520088e-01]
[ 1.89979318e-01  9.32231628e-01]
[ 1.95057130e-01  3.96267825e-01]
[ 1.95486545e-01  1.19313561e+00]
[ 1.96765430e-01  1.26641917e+00]
[ 1.99380182e-01  1.31479886e+00]
[ 2.00480582e-01  9.11774993e-01]
[ 2.03214892e-01  9.93308664e-01]
[ 2.03472830e-01  9.88544456e-01]
[ 2.05131614e-01  7.33696777e-01]
[ 2.07244013e-01  1.18326034e+00]
[ 2.08737543e-01  7.96158880e-01]
```

```
[ 2.09661257e-01  1.06388365e+00]
[ 2.09870158e-01  1.21438848e+00]
[ 2.11862656e-01  1.03437328e+00]
[ 2.15121544e-01  9.80642818e-01]
[ 2.16853872e-01  9.77892140e-01]
[ 2.20634380e-01  1.32497526e+00]
[ 2.20963360e-01  1.30808695e+00]
[ 2.31671716e-01  1.17893700e+00]
[ 2.34970596e-01  7.31855964e-01]
[ 2.35354495e-01  1.13279108e+00]
[ 2.43048168e-01  1.40279456e+00]
[ 2.49834512e-01  1.06942516e+00]
[ 2.53483071e-01  8.74211676e-01]
[ 2.53856565e-01  8.69646808e-01]
[ 2.55710937e-01  8.58037742e-01]
[ 2.57319947e-01  6.55882523e-01]
[ 2.57551520e-01  1.49442521e+00]
[ 2.58837765e-01  1.17779389e+00]
[ 2.66599101e-01  1.20580389e+00]
[ 2.69566013e-01  1.17726969e+00]
[ 2.72188821e-01  9.36316305e-01]
[ 2.75550173e-01  9.81676898e-01]
[ 2.77651752e-01  1.25233843e+00]
[ 2.77795099e-01  7.85330547e-01]
[ 2.80455061e-01  1.02461910e+00]
[ 2.83049674e-01  9.31395892e-01]
[ 2.83794713e-01  1.38802539e+00]
[ 2.84302121e-01  6.47311134e-01]
[ 2.85911497e-01  1.26156631e+00]
[ 2.87529685e-01  9.30863201e-01]
[ 2.90905153e-01  1.08694552e+00]
[ 2.92913447e-01  1.17945421e+00]
[ 2.93605853e-01  1.26998794e+00]
[ 2.93678188e-01  1.15744404e+00]
[ 2.95257903e-01  1.18878825e+00]
[ 2.95781302e-01  1.32107530e+00]
[ 2.98027570e-01  1.07480244e+00]
[ 3.02381963e-01  5.85574101e-01]
[ 3.04939625e-01  1.04466075e+00]
[ 3.05506832e-01  9.41055827e-01]
[ 3.07040818e-01  1.12807962e+00]
[ 3.09064887e-01  9.00388642e-01]
[ 3.10984988e-01  8.51710302e-01]
[ 3.12681907e-01  8.36340962e-01]
[ 3.18033733e-01  3.72010173e-01]
[ 3.19421054e-01  4.47434107e-01]
[ 3.22608354e-01  8.83249278e-01]
[ 3.22999563e-01  6.51924019e-01]
```

```
[ 3.24206297e-01  7.07864190e-01]
[ 3.25746852e-01  1.05861527e+00]
[ 3.27153798e-01  1.03965930e+00]
[ 3.29087261e-01  9.13743081e-01]
[ 3.29643694e-01  9.14865652e-01]
[ 3.30104566e-01  7.78693625e-01]
[ 3.31991638e-01  1.47245525e+00]
[ 3.33193553e-01  5.95274932e-01]
[ 3.40080558e-01  1.00351111e+00]
[ 3.40510870e-01  5.33049733e-01]
[ 3.41540236e-01  9.16687954e-01]
[ 3.41660270e-01  1.47085744e+00]
[ 3.43666166e-01  1.20182119e+00]
[ 3.45549251e-01  6.96288923e-01]
[ 3.54181244e-01  5.86257056e-01]
[ 3.56382282e-01  7.57944598e-01]
[ 3.64002491e-01  8.86622343e-01]
[ 3.64149175e-01  8.11135895e-01]
[ 3.64588816e-01  6.88006786e-01]
[ 3.67570869e-01  4.79797741e-01]
[ 3.72332914e-01  1.03276354e+00]
[ 3.73034449e-01  6.88659584e-01]
[ 3.73927426e-01  6.23592185e-01]
[ 3.76413964e-01  8.51376674e-01]
[ 3.78619285e-01  5.63796197e-01]
[ 3.81528803e-01  4.62119165e-01]
[ 3.82167064e-01  8.25950878e-01]
[ 3.83254718e-01  6.66867766e-01]
[ 3.85348072e-01  4.35117958e-01]
[ 3.88969709e-01  4.84485398e-01]
[ 3.90982315e-01  6.01173668e-01]
[ 3.92684441e-01  5.29767065e-01]
[ 3.92958783e-01  4.13935074e-01]
[ 3.94641827e-01  3.25793237e-01]
[ 3.94845458e-01  8.66304176e-01]
[ 3.95429896e-01  5.13963470e-01]
[ 3.96121924e-01  2.94123653e-01]
[ 3.96139477e-01  6.62885432e-01]
[ 3.99092829e-01  7.05039400e-01]
[ 4.02300643e-01  3.93787329e-01]
[ 4.03398313e-01  1.38528886e-01]
[ 4.08139532e-01  7.80172664e-01]
[ 4.09506284e-01 -6.07172339e-03]
[ 4.10673353e-01  7.71305020e-01]
[ 4.12901395e-01  6.42096046e-01]
[ 4.18477517e-01  3.88553948e-03]
[ 4.22620512e-01  7.17005362e-01]
[ 4.26612083e-01  7.57439366e-01]
```

```
[ 4.27073601e-01  9.27008600e-01]
[ 4.29003008e-01  2.49429223e-01]
[ 4.29074716e-01  5.55213148e-01]
[ 4.29852191e-01  1.31439288e-01]
[ 4.33489935e-01  2.49787048e-01]
[ 4.34307608e-01  5.48820269e-01]
[ 4.34685762e-01  2.03429910e-01]
[ 4.39319214e-01  6.66007059e-01]
[ 4.43733123e-01  1.86920455e-01]
[ 4.44097439e-01  3.11444112e-01]
[ 4.44825467e-01  3.82305134e-01]
[ 4.48587286e-01  2.31769771e-01]
[ 4.48843563e-01  5.56601871e-01]
[ 4.49665871e-01  2.28583133e-01]
[ 4.50482639e-01  2.28099821e-01]
[ 4.51338681e-01  2.04268279e-01]
[ 4.51385260e-01  9.62152408e-01]
[ 4.51593058e-01  6.00202088e-01]
[ 4.53450710e-01  5.55592125e-01]
[ 4.55479899e-01  2.19206747e-01]
[ 4.58795659e-01  5.96943031e-01]
[ 4.59090626e-01  2.82071651e-01]
[ 4.59413665e-01  2.24828979e-01]
[ 4.64269782e-01  1.46681531e-01]
[ 4.64533512e-01  4.25461056e-03]
[ 4.64722506e-01  4.24417770e-01]
[ 4.71829182e-01 -1.70511002e-02]
[ 4.75568297e-01 -5.64761189e-01]
[ 4.76060626e-01 -5.78155162e-02]
[ 4.78532046e-01  2.52289573e-01]
[ 4.79897442e-01  3.61077746e-01]
[ 4.85307981e-01 -3.63495502e-02]
[ 4.92582652e-01  1.52084810e-01]
[ 4.92654210e-01  2.67598862e-02]
[ 4.92677333e-01  1.52239831e-01]
[ 4.94369348e-01  6.19334156e-04]
[ 4.95861028e-01 -1.10763862e-01]
[ 4.98037031e-01 -1.52530558e-01]
[ 4.99516427e-01 -2.35150640e-01]
[ 5.03953996e-01  3.25823753e-01]
[ 5.07018584e-01  4.86475599e-02]
[ 5.07242130e-01 -1.70647630e-01]
[ 5.13522832e-01 -6.46448959e-01]
[ 5.13828877e-01 -4.89734074e-01]
[ 5.18631426e-01  2.75586493e-01]
[ 5.19794694e-01 -1.74279009e-01]
[ 5.21296866e-01 -1.12029301e-01]
[ 5.23640154e-01  8.69923687e-02]
```

```
[ 5.23813474e-01 -5.81217469e-01]
[ 5.25039040e-01 -3.18445869e-01]
[ 5.30333286e-01 -3.18690212e-01]
[ 5.30781906e-01 -9.97007769e-02]
[ 5.31999191e-01 -4.97693946e-01]
[ 5.34942539e-01 -2.91334512e-01]
[ 5.36116693e-01  1.92310537e-01]
[ 5.37868958e-01  4.17204098e-02]
[ 5.38723666e-01 -2.83866019e-01]
[ 5.42766572e-01 -6.32137232e-01]
[ 5.44061041e-01 -1.13039970e-01]
[ 5.45363884e-01 -6.52306958e-01]
[ 5.46850172e-01 -4.72733349e-01]
[ 5.50661977e-01 -3.64677925e-01]
[ 5.53548338e-01 -8.78533222e-02]
[ 5.53621140e-01  4.49795025e-02]
[ 5.55505985e-01 -6.95763210e-01]
[ 5.56480166e-01 -3.42239219e-01]
[ 5.56711234e-01  1.08940276e-01]
[ 5.59220390e-01 -5.24526754e-01]
[ 5.61741836e-01 -1.92657639e-01]
[ 5.64500405e-01 -5.90075466e-01]
[ 5.64560190e-01 -6.96818041e-01]
[ 5.68352034e-01 -6.23181745e-01]
[ 5.73495078e-01 -6.54322473e-01]
[ 5.77740201e-01 -4.09875026e-01]
[ 5.78032902e-01 -4.78527883e-01]
[ 5.78097213e-01 -5.45369690e-01]
[ 5.82301741e-01 -6.97099914e-01]
[ 5.90110191e-01 -2.14518899e-01]
[ 5.95560678e-01 -6.61518183e-01]
[ 5.96605881e-01 -1.61823625e-01]
[ 5.97665873e-01 -6.40355304e-01]
[ 5.99363059e-01 -1.14949749e+00]
[ 6.02809222e-01 -5.66386776e-01]
[ 6.08721284e-01 -6.30869609e-01]
[ 6.08908662e-01 -7.05402095e-01]
[ 6.09977958e-01 -3.03205915e-01]
[ 6.10792531e-01 -6.48014835e-01]
[ 6.10837810e-01 -4.36402891e-01]
[ 6.13432612e-01 -5.64356919e-01]
[ 6.13625126e-01 -7.82403879e-01]
[ 6.14185563e-01 -5.95367662e-01]
[ 6.17573375e-01 -7.31529298e-01]
[ 6.23846834e-01 -9.22510748e-01]
[ 6.24546746e-01 -8.44281603e-01]
[ 6.28958501e-01 -8.01698339e-01]
[ 6.30415209e-01 -6.94891693e-01]
```

```
[ 6.31966771e-01 -6.40870459e-01]
[ 6.34575210e-01 -8.31693702e-01]
[ 6.34581847e-01 -6.72086760e-01]
[ 6.34741151e-01 -6.47133657e-01]
[ 6.40014450e-01 -8.50794374e-01]
[ 6.41304565e-01 -1.34963013e+00]
[ 6.43426548e-01 -5.94584993e-01]
[ 6.43787569e-01 -4.39570345e-01]
[ 6.46500028e-01 -4.47688844e-01]
[ 6.48679490e-01 -8.65661854e-01]
[ 6.51065070e-01 -9.01937168e-01]
[ 6.51701566e-01 -6.47391966e-01]
[ 6.55347611e-01 -1.23184706e+00]
[ 6.58279191e-01 -9.81262674e-01]
[ 6.59816779e-01 -7.04474890e-01]
[ 6.61607816e-01 -7.95998348e-01]
[ 6.61894290e-01 -8.49475738e-01]
[ 6.62838030e-01 -9.75320481e-01]
[ 6.64284351e-01 -9.46033351e-01]
[ 6.70792053e-01 -5.83552980e-01]
[ 6.71165214e-01 -8.64185721e-01]
[ 6.72903065e-01 -9.37772279e-01]
[ 6.72969660e-01 -7.16769101e-01]
[ 6.74574286e-01 -7.34134287e-01]
[ 6.76584869e-01 -9.57944137e-01]
[ 6.80682617e-01 -9.44675710e-01]
[ 6.84090678e-01 -1.01998669e+00]
[ 6.84530834e-01 -1.25873794e+00]
[ 6.85779671e-01 -8.49317327e-01]
[ 6.86267997e-01 -3.27297309e-01]
[ 6.89698287e-01 -8.72684239e-01]
[ 6.96657327e-01 -8.62016859e-01]
[ 6.99090295e-01 -1.06294889e+00]
[ 6.99544320e-01 -9.62335078e-01]
[ 7.01416016e-01 -5.66857216e-01]
[ 7.01483480e-01 -9.67291617e-01]
[ 7.02313423e-01 -1.18422467e+00]
[ 7.03312248e-01 -9.48371436e-01]
[ 7.04261953e-01 -1.05718485e+00]
[ 7.07595718e-01 -1.24078137e+00]
[ 7.10918530e-01 -9.19770180e-01]
[ 7.14725540e-01 -1.30475443e+00]
[ 7.15865303e-01 -7.95270784e-01]
[ 7.16037140e-01 -9.74123020e-01]
[ 7.16662981e-01 -5.83587780e-01]
[ 7.21312328e-01 -1.02805928e+00]
[ 7.22081112e-01 -1.00207132e+00]
[ 7.23677038e-01 -7.50879363e-01]
```

```
[ 7.25964727e-01 -8.36848458e-01]
[ 7.26902299e-01 -1.09374708e+00]
[ 7.27407362e-01 -8.97762036e-01]
[ 7.27544364e-01 -1.02026720e+00]
[ 7.27574906e-01 -5.73116731e-01]
[ 7.27663999e-01 -5.76394637e-01]
[ 7.31406414e-01 -7.30673705e-01]
[ 7.32102088e-01 -1.02442343e+00]
[ 7.33786605e-01 -1.26339289e+00]
[ 7.36252786e-01 -7.88328378e-01]
[ 7.38858762e-01 -8.61456455e-01]
[ 7.40239733e-01 -1.04270616e+00]
[ 7.43490852e-01 -7.45362050e-01]
[ 7.50848344e-01 -1.15741868e+00]
[ 7.50917169e-01 -1.22264007e+00]
[ 7.51837980e-01 -6.98235656e-01]
[ 7.52428422e-01 -1.54222708e+00]
[ 7.53864767e-01 -1.23395489e+00]
[ 7.55549760e-01 -1.06804449e+00]
[ 7.57217364e-01 -5.48602979e-01]
[ 7.61593125e-01 -1.04969137e+00]
[ 7.69080703e-01 -8.19347203e-01]
[ 7.69704526e-01 -7.26148349e-01]
[ 7.70943415e-01 -7.80197768e-01]
[ 7.71537871e-01 -1.20406262e+00]
[ 7.72205958e-01 -8.05601816e-01]
[ 7.73055072e-01 -1.00067952e+00]
[ 7.73634033e-01 -9.06390620e-01]
[ 7.74066556e-01 -1.04515336e+00]
[ 7.74172401e-01 -7.20028012e-01]
[ 7.74266715e-01 -4.67274289e-01]
[ 7.75777719e-01 -5.64545207e-01]
[ 7.79110574e-01 -1.38765145e+00]
[ 7.79567482e-01 -1.14312650e+00]
[ 7.85723294e-01 -8.63758965e-01]
[ 7.87096398e-01 -9.49037225e-01]
[ 7.90869828e-01 -1.10786000e+00]
[ 7.91184756e-01 -9.11412440e-01]
[ 7.93477324e-01 -9.98499717e-01]
[ 7.97585358e-01 -8.20161326e-01]
[ 7.98834564e-01 -1.38989645e+00]
[ 8.02089090e-01 -1.24348535e+00]
[ 8.05453979e-01 -8.45401783e-01]
[ 8.06355760e-01 -1.30111555e+00]
[ 8.08528923e-01 -7.33718491e-01]
[ 8.08856970e-01 -1.06866737e+00]
[ 8.13711421e-01 -8.74084908e-01]
[ 8.17363154e-01 -9.05530703e-01]
```

```
[ 8.18472243e-01 -9.80896450e-01]
[ 8.19887157e-01 -1.11262640e+00]
[ 8.20072578e-01 -7.02065419e-01]
[ 8.21167750e-01 -8.60682242e-01]
[ 8.22660051e-01 -7.16752476e-01]
[ 8.23832953e-01 -3.81278465e-01]
[ 8.23955366e-01 -6.30577521e-01]
[ 8.24641970e-01 -1.25673158e+00]
[ 8.25772902e-01 -1.28920932e+00]
[ 8.27286204e-01 -7.31654555e-01]
[ 8.29579808e-01 -7.75326786e-01]
[ 8.29864698e-01 -6.79299913e-01]
[ 8.34203611e-01 -8.37162545e-01]
[ 8.35403633e-01 -9.86760354e-01]
[ 8.36087270e-01 -7.13287412e-01]
[ 8.36671440e-01 -7.86874501e-01]
[ 8.37938315e-01 -1.10215076e+00]
[ 8.45299677e-01 -6.17547038e-01]
[ 8.48033098e-01 -1.03895828e+00]
[ 8.49025655e-01 -6.54886973e-01]
[ 8.51769766e-01 -3.24526303e-01]
[ 8.52239012e-01 -7.71829469e-01]
[ 8.59783226e-01 -6.20537424e-01]
[ 8.59792510e-01 -8.58650722e-01]
[ 8.63251926e-01 -6.78762422e-01]
[ 8.66116926e-01 -5.56634675e-01]
[ 8.67231289e-01 -2.92455368e-01]
[ 8.68089161e-01 -1.11407994e+00]
[ 8.70288911e-01 -1.22963078e+00]
[ 8.80354169e-01 -6.15308923e-01]
[ 8.81584062e-01 -6.34379204e-01]
[ 8.83824167e-01 -1.01978947e+00]
[ 8.85683370e-01 -4.97895403e-01]
[ 8.91073004e-01 -2.35507088e-01]
[ 8.92518718e-01 -6.82415203e-01]
[ 8.97506029e-01 -4.45581232e-01]
[ 9.01963522e-01 -5.96706754e-02]
[ 9.03120611e-01 -4.29946440e-01]
[ 9.03892573e-01 -4.87126277e-01]
[ 9.04026527e-01 -4.60530993e-01]
[ 9.04396124e-01 -4.83955966e-01]
[ 9.06777791e-01 -1.91786597e-01]
[ 9.07355691e-01 -2.89957106e-01]
[ 9.08203710e-01 -6.31870991e-01]
[ 9.08668449e-01 -6.01221161e-01]
[ 9.10001199e-01 -3.17947149e-01]
[ 9.10001730e-01 -8.60436363e-01]
[ 9.10169235e-01 -7.08913856e-01]
```

```
[ 9.11009881e-01 -1.19209756e-01]
[ 9.11619995e-01 -8.96748807e-01]
[ 9.12835789e-01 -4.30555253e-01]
[ 9.15334571e-01 -5.50907177e-01]
[ 9.15624842e-01 -5.69333928e-01]
[ 9.18024932e-01 -3.41720408e-01]
[ 9.22931986e-01 -8.13964678e-01]
[ 9.24630695e-01 -7.11148457e-01]
[ 9.26939825e-01 -6.39276592e-01]
[ 9.29177907e-01 -3.62709901e-01]
[ 9.31009982e-01 -5.05738837e-01]
[ 9.31719522e-01 -6.33914952e-01]
[ 9.31998137e-01 -7.23647084e-01]
[ 9.36909448e-01 -1.92156741e-01]
[ 9.37893742e-01 -1.95353255e-01]
[ 9.37920732e-01  2.26574004e-03]
[ 9.40282885e-01 -6.91553318e-01]
[ 9.40956653e-01 -6.67141249e-01]
[ 9.41073059e-01  1.71205615e-01]
[ 9.41618653e-01 -5.62457870e-01]
[ 9.42834517e-01 -3.75634134e-01]
[ 9.45698588e-01 -4.12219715e-01]
[ 9.46648499e-01 -4.99491737e-01]
[ 9.48713855e-01 -6.42550368e-02]
[ 9.49840410e-01 -7.28670844e-01]
[ 9.52091501e-01 -5.48726760e-01]
[ 9.53619920e-01 -2.14503659e-01]
[ 9.54564215e-01 -2.98580351e-01]
[ 9.54730671e-01 -5.80470872e-01]
[ 9.55463862e-01 -1.64258747e-01]
[ 9.56751680e-01 -8.53586521e-02]
[ 9.57109103e-01 -1.90487454e-01]
[ 9.58559805e-01 -3.53341293e-01]
[ 9.61930293e-01 -2.17686324e-01]
[ 9.65666159e-01 -3.62408112e-01]
[ 9.66731343e-01 -6.46274038e-02]
[ 9.66956859e-01 -4.35595841e-01]
[ 9.68124592e-01 -1.85748542e-01]
[ 9.70274726e-01 -2.73234798e-01]
[ 9.76473816e-01 -3.55176043e-01]
[ 9.77829672e-01 -3.73662343e-01]
[ 9.81642653e-01  7.93435840e-02]
[ 9.82577302e-01 -2.81102140e-01]
[ 9.84171132e-01 -7.05234429e-01]
[ 9.85703375e-01 -2.50626819e-01]
[ 9.86007481e-01 -3.34694272e-02]
[ 9.87434307e-01 -9.88744509e-03]
[ 9.87853663e-01 -5.17969566e-01]
```

```
[ 9.88865362e-01 -1.16350344e-01]
[ 9.89102198e-01  8.01049684e-02]
[ 9.90223491e-01  1.05053954e-01]
[ 9.90834085e-01  3.81149245e-01]
[ 9.93600485e-01 -2.61289201e-01]
[ 9.98182713e-01  5.60377054e-02]
[ 9.98297026e-01 -3.45496481e-01]
[ 9.98464792e-01 -5.15522541e-02]
[ 9.98531508e-01 -3.90615696e-01]]
```

6 Question-4

Estimate the Regularized Least Square kernel regression model by using the training set T. Plot the training estimate along with the original target function. Also, compute the Test RMSE.

[ ]:
```
def calculate_kernel_block(X1, X2, sigma):
  value = math.exp((-1* abs(X1-X2)) / sigma)
  return value
```

[ ]:
```
def construct_kernel(row_X, col_X, sigma):
  kernel = np.zeros((len(row_X), len(col_X)+1))
  kernel[:,-1] = 1
  for i in range(len(row_X)):
    for j in range(len(col_X)):
      kernel[i][j] = calculate_kernel_block(row_X[i], col_X[j], sigma)
  return kernel
```

[ ]:
```
def calculate_U(kernel, Y, lamb):
  result = np.linalg.inv( (kernel.T @ kernel) + lamb * np.
 ↪identity(int(len(kernel[0])))) @ kernel.T @ Y
  return result
```

[ ]:
```
###### Function to calculate the RMSE based on actual and estimated Y values

def calculate_RMSE(Y, Y_real):
  error = np.subtract(Y, Y_real)
  squared_error = np.square(error)
  total_squared_error = squared_error.sum()     #### to sum all the values in␣
 ↪each row
  total_mean_squared_error = total_squared_error/len(Y)
  total_root_mean_squared_error = np.sqrt(total_mean_squared_error)
  return total_mean_squared_error
```

[ ]:
```
train_X = training_set[:, 0]
train_Y = training_set[:, 1]
```

```
test_X = testing_set[:, 0]
test_Y = testing_set[:, 1]
```

[ ]:
```
sigma = 5

train_kernel = construct_kernel(train_X, train_X, sigma)
```

[ ]:
```
train_kernel.shape
```

[ ]:
```
(50, 51)
```

[ ]:
```
lamb = 0.001

U = calculate_U(train_kernel, train_Y, lamb)
```

[ ]:
```
U.shape
```

[ ]:
```
(51,)
```

[ ]:
```
train_Y_est = train_kernel @ U
train_RMSE = calculate_RMSE(train_Y_est, train_Y)
train_RMSE
```

[ ]:
```
0.042218699335702735
```

[ ]:
```
test_kernel = construct_kernel(test_X, train_X, sigma)
test_kernel.shape
```

[ ]:
```
(500, 51)
```

[ ]:
```
test_Y_est = test_kernel @ U
test_Y_est.shape
```

[ ]:
```
(500,)
```

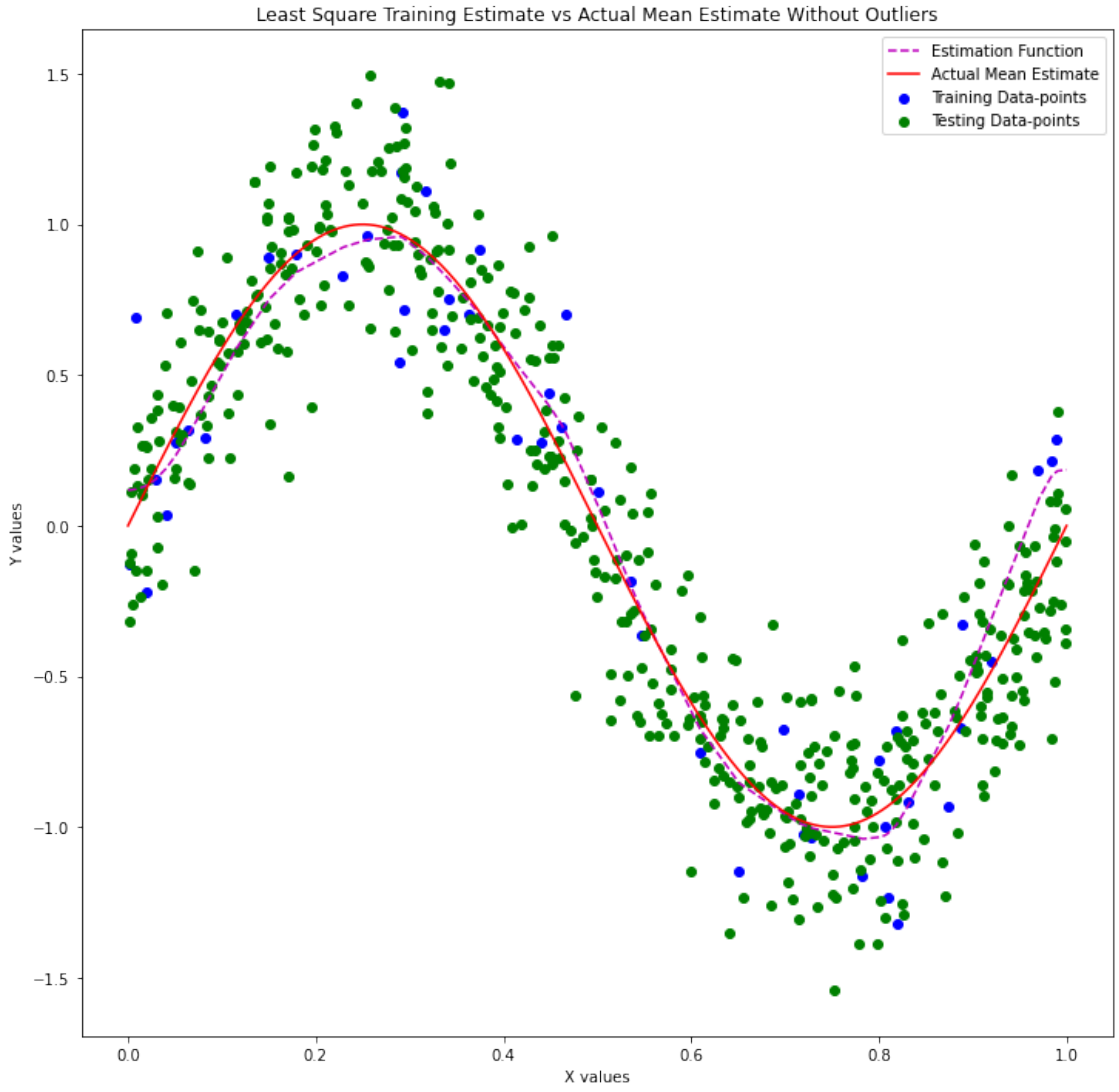
[ ]:
```
test_RMSE = calculate_RMSE(test_Y_est, test_Y)
test_RMSE_RKR_Q1 = test_RMSE
test_RMSE_RKR_Q1
```

[ ]:
```
0.0690993470657655
```

6.0.1 Plotting estimate function

[ ]:
```
####### for plotting the estimated values

from matplotlib import pyplot as plt

def plot_values(X, real_Y, X1, real_Y1, weights):
  PI = math.pi
  myline = np.linspace(0, 1, 5000)
  actual_mean_estimate = np.sin(2*PI*myline)

  plt.figure(figsize = (12,12))
  plt.scatter(X, real_Y, color = 'b', label = 'Training Data-points')
  plt.scatter(X1, real_Y1, color = 'g', label = 'Testing Data-points')

  plotting_kernel = construct_kernel(myline, train_X, sigma)
  myline_Y_est = plotting_kernel @ weights

  plt.plot(myline, myline_Y_est, color = 'm', linestyle='--', label =␣
 ↪'Estimation Function')
  plt.plot(myline, actual_mean_estimate, '-r', label = 'Actual Mean Estimate')
  plt.legend()
  return plt
```

[ ]:
```
plt = plot_values(train_X, train_Y, test_X, test_Y, U)
plt.title('Least Square Training Estimate vs Actual Mean Estimate Without␣
 ↪Outliers')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```



7 Question-5

Randomly select 5 training points to say x'1 , x'2 , x'3 , x'4 , x'5 and scale(multiply) their corresponding y'1 , y'2 , y'3 , y'4 , y'5 by 4. So these 5 points are to be considered as outliers. Estimate the Regularized Least Square kernel regression model by using the modified training set T. Plot the training estimate along with the original target function. Also, compute the RMSE.

[ ]:
```
import random

indexes = []
for i in range(5):
    indexes.append(random.randint(0,49))
```

```
print(indexes)
```

```
[1, 12, 34, 11, 24]
```

[ ]:
```
train_Y_outlier = np.copy(train_Y)
for index in indexes:
    train_Y_outlier[index] = train_Y_outlier[index] * 4
```

[ ]:
```
U_outlier = calculate_U(train_kernel, train_Y_outlier, lamb)
U_outlier.shape
```

[ ]:
```
(51,)
```

[ ]:
```
train_Y_est_outlier = train_kernel @ U_outlier
train_RMSE_outlier = calculate_RMSE(train_Y_est_outlier, train_Y_outlier)
train_RMSE_outlier
```

[ ]:
```
0.47231799934807783
```

[ ]:
```
test_Y_est_outlier = test_kernel @ U_outlier
test_Y_est_outlier.shape
```

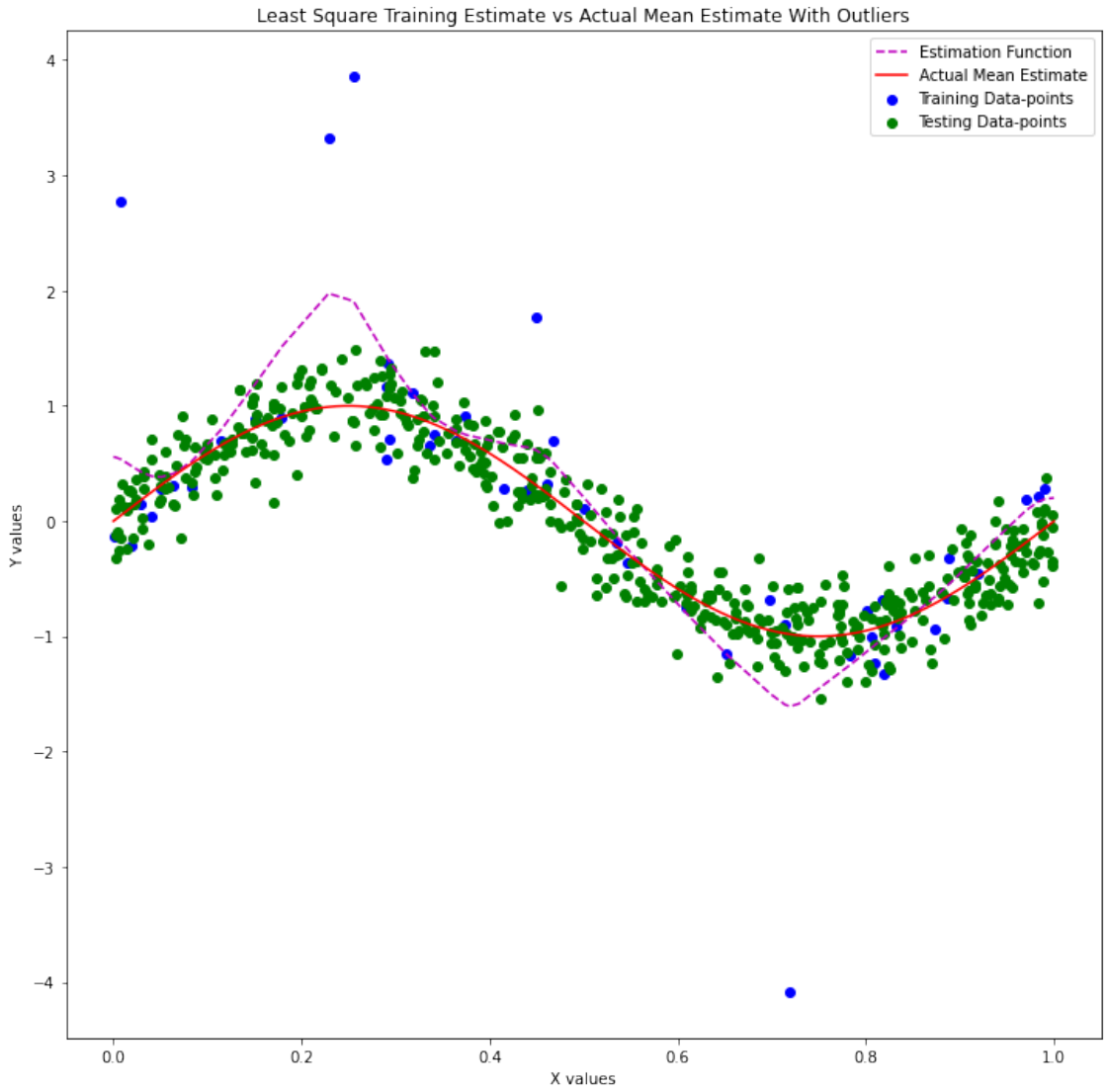
[ ]:
```
(500,)
```

[ ]:
```
test_RMSE_outlier = calculate_RMSE(test_Y_est_outlier, test_Y)
test_RMSE_RKR_Q1_outlier = test_RMSE_outlier
test_RMSE_RKR_Q1_outlier
```

[ ]:
```
0.18510363304553143
```

7.0.1 Plotting estimate function

[ ]:
```
plt = plot_values(train_X, train_Y_outlier, test_X, test_Y, U_outlier)
plt.title('Least Square Training Estimate vs Actual Mean Estimate With␣
 ↪Outliers')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```



8 Question-6

Write your observation regarding the performance of the Least Square Regression model in presence of outliers.

8.0.1 Observations

1.) Training RMSE without outliers = 0.042218699335702735

2.) Testing RMSE without outliers = 0.0690993470657655

3.) Training RMSE with outliers = 0.47231799934807783

4.) Testing RMSE with outliers = 0.18510363304553143

After adding the outliers, the curve moves towards the outliers little bit but not completely. These outliers increase the training RMSE and because of the moved and readjusted estimation function, testing RMSE also increases.

This basically means that normally the estimation process is more sensitive towards existence of outliers.

9 Question-7

Repeat the experiment (1-5) for the L1-norm loss kernel regression model. For obtaining the solution of the L1-norm loss kernel regression model, you need to use the gradient descent or stochastic gradient descent algorithm. Write your observation regarding the performance of L1-norm kernel Regression model in presence of outliers.

9.1 Part A.) L1-norm loss kernel Regression Without outliers

[ ]:
```
def transform_gradient(v):
    for i in range(len(v)):
        if(v[i] > 0):
            v[i] = -1
        else:
            v[i] = 1
    return v
```

[ ]:
```
def compute_gradient_vector(lamb, U, Y, kernel):
    loss_vector = Y - (kernel @ U )
    loss_vector_transformed = transform_gradient(loss_vector)
    gradient_vec = (lamb*U) + (kernel.T @ loss_vector_transformed)
    return gradient_vec
```

[ ]:
```
weights = np.array( [1] * (len(train_X)+1) )

eta = 0.0002
lamb = 0.001
tolerance = 1.1

gradient_vector = compute_gradient_vector(lamb, weights, train_Y, train_kernel)
gradient_norm = []

counter = 1
while np.linalg.norm(gradient_vector) > tolerance:
    weights = weights - eta * gradient_vector
    gradient_norm.append(np.linalg.norm(gradient_vector))
```

```
    gradient_vector = compute_gradient_vector(lamb, weights, train_Y,␣
↪train_kernel)

    print('Iteration no : '+str(counter))
    counter = counter+1
    print('Current gradient\'s norm value is : '+str(np.linalg.
↪norm(gradient_vector)))
    print('================================================================')
```

[ ]:
```
train_Y_est_l1_norm = train_kernel @ weights
train_RMSE_l1_norm = calculate_RMSE(train_Y_est_l1_norm, train_Y)
train_RMSE_l1_norm
```

[ ]:
```
0.19059812434904114
```

[ ]:
```
test_Y_est_l1_norm = test_kernel @ weights
test_Y_est_l1_norm.shape
```

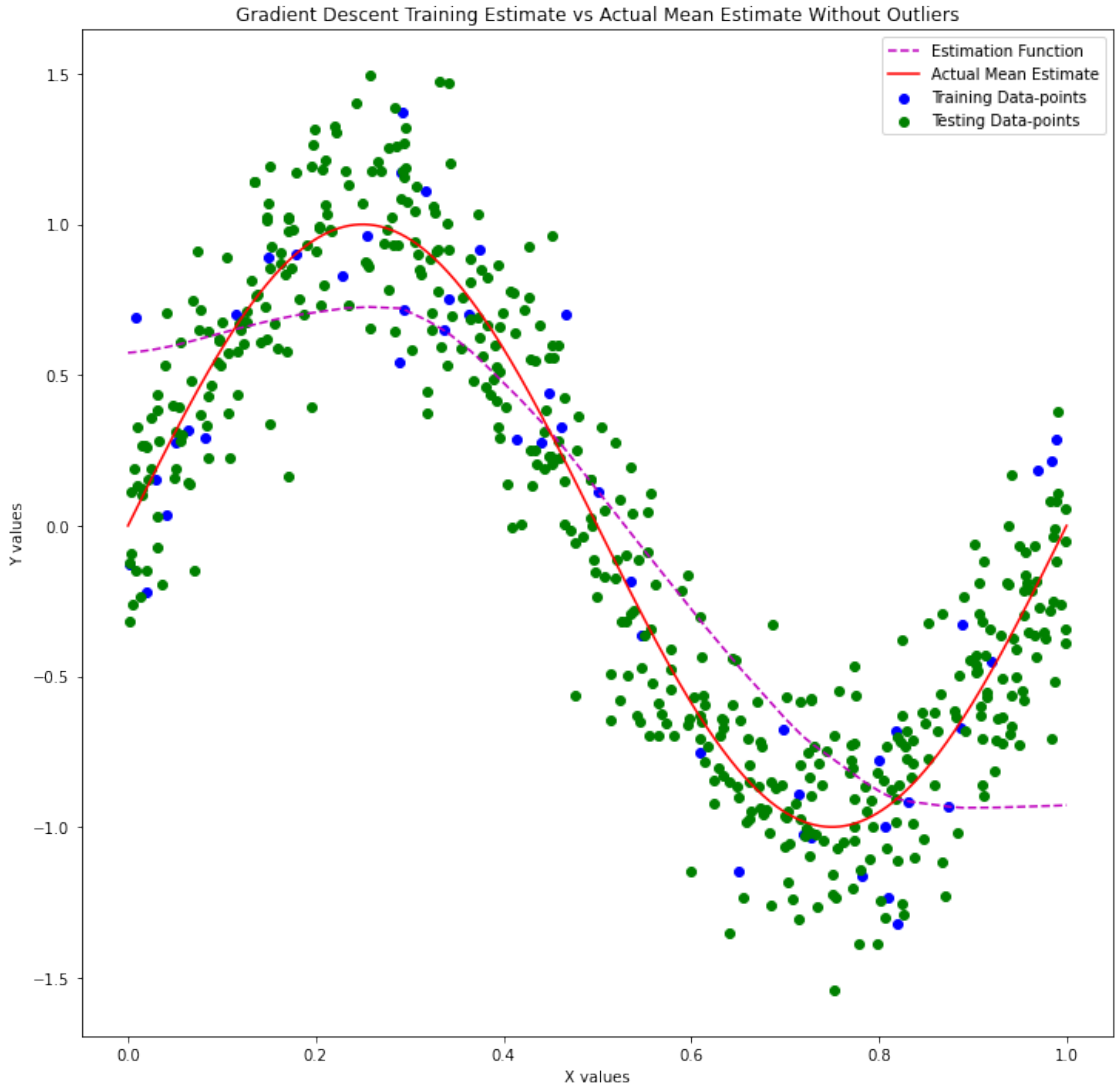
[ ]:
```
(500,)
```

[ ]:
```
test_RMSE_l1_norm = calculate_RMSE(test_Y_est_l1_norm, test_Y)
test_RMSE_RKR_Q1_l1_norm = test_RMSE_l1_norm
test_RMSE_RKR_Q1_l1_norm
```

[ ]:
```
0.156841121339253
```

9.1.1 Plotting estimated function

[ ]:
```
plt = plot_values(train_X, train_Y, test_X, test_Y, weights)
plt.title('Gradient Descent Training Estimate vs Actual Mean Estimate Without␣
↪Outliers')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```



9.2 Part B.) L1-norm loss kernel Regression With outliers

[ ]:
```
weights_outlier = np.array( [1] * (len(train_X)+1) )

eta = 0.0002
lamb = 0.001
tolerance = 1.1

gradient_vector_outlier = compute_gradient_vector(lamb, weights_outlier,␣
↪train_Y_outlier, train_kernel)
gradient_norm_outlier = []
```

```
counter_outlier = 1
while np.linalg.norm(gradient_vector_outlier) > tolerance:
    weights_outlier = weights_outlier - eta * gradient_vector_outlier
    gradient_norm_outlier.append(np.linalg.norm(gradient_vector_outlier))
    gradient_vector_outlier = compute_gradient_vector(lamb, weights_outlier,␣
 ↪train_Y_outlier, train_kernel)

    print('Iteration no : '+str(counter))
    counter_outlier = counter_outlier+1
    print('Current gradient\'s norm value is : '+str(np.linalg.
 ↪norm(gradient_vector_outlier)))
    print('==============================================================')
```

[ ]:
```
train_Y_est_l1_norm_outlier = train_kernel @ weights_outlier
train_RMSE_l1_norm_outlier = calculate_RMSE(train_Y_est_l1_norm_outlier,␣
 ↪train_Y_outlier)
train_RMSE_l1_norm_outlier
```

[ ]:
```
0.8777220051710518
```

[ ]:
```
test_Y_est_l1_norm_outlier = test_kernel @ weights_outlier
test_Y_est_l1_norm_outlier.shape
```

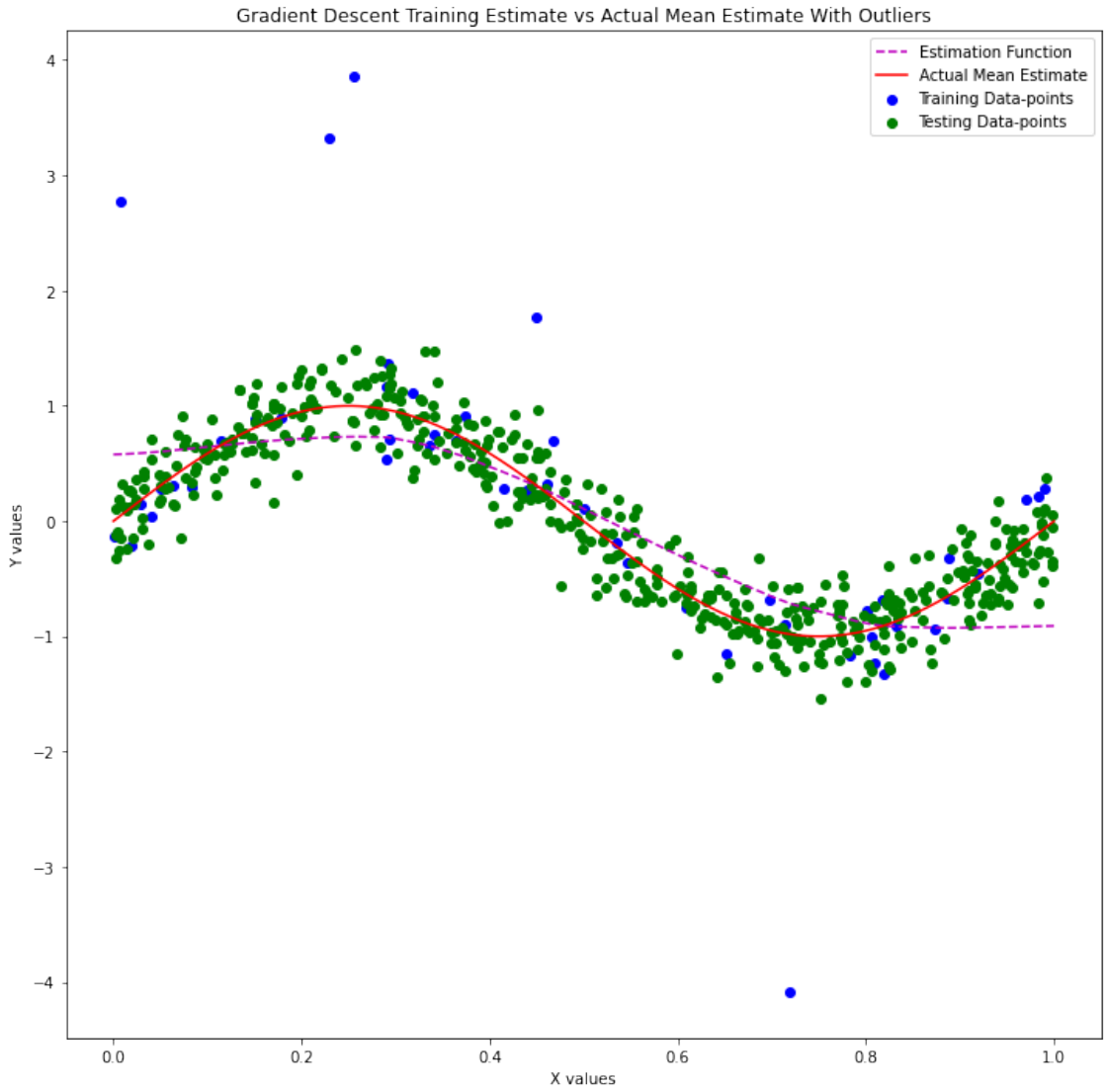
[ ]:
```
(500,)
```

[ ]:
```
test_RMSE_l1_norm_outlier = calculate_RMSE(test_Y_est_l1_norm_outlier, test_Y)
test_RMSE_RKR_Q1_l1_norm_outlier = test_RMSE_l1_norm_outlier
test_RMSE_RKR_Q1_l1_norm_outlier
```

[ ]:
```
0.1517411766204316
```

9.2.1 Plotting estimated function

[ ]:
```
plt = plot_values(train_X, train_Y_outlier, test_X, test_Y, weights_outlier)
plt.title('Gradient Descent Training Estimate vs Actual Mean Estimate With␣
 ↪Outliers')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```



9.2.2 Observations

1.) Training RMSE using L1-norm loss kernel regression without outliers = 0.19059812434904114

2.) Testing RMSE L1-norm loss kernel regression without outliers = 0.156841121339253

3.) Training RMSE L1-norm loss kernel regression with outliers = 0.8777220051710518

4.) Testing RMSE L1-norm loss kernel regression with outliers = 0.1517411766204316

Here we can see that even after adding the outliers in the training data, testing RMSE is not much affected. Although training RMSE is high becuase since estimation process didn't get much affected by outliers, therefore these outlier points is dominating for training RMSE.

Therefore, we can conclude that L1-norm loss kernal regression method is less sensitive towards existence of outliers.

10 Thank You!

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('PRML04.ipynb')
```

```
--2022-03-16 13:06:48--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.110.133, 185.199.111.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-03-16 13:06:48 (20.7 MB/s) - ‘colab_pdf.py’ saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab
Notebooks/PRML04.ipynb to pdf
[NbConvertApp] Support files will be in PRML04_files/
[NbConvertApp] Making directory ./PRML04_files
[NbConvertApp] Making directory ./PRML04_files
[NbConvertApp] Making directory ./PRML04_files
[NbConvertApp] Making directory ./PRML04_files
[NbConvertApp] Writing 88112 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
```

```
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 297592 bytes to /content/drive/My Drive/PRML04.pdf
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

[2]:
```
'File ready to be Downloaded and Saved to Drive'
```

[ ]:
```

```