Coursework	Fall 2021
Machine Learning (B555 - 17011)	Prof. Roni Khardon
Department of Computer Science, IU Bloomington	

Topic : Programming project 4 - Implementation of Gaussian Process with Linear and RBF Kernel Report
Submitted by : Pratap Roy Choudhury, MS in Data Science, Fall202

1. Overview: Gaussian Processes

In this programming project we will be working with Gaussian processes using the RBF kernel and the polynomial kernel with degree 1 (i.e., linear) kernel.

2. Data

Data for this assignment is provided in a zip file pp4data.zip. We have 4 datasets viz. crime, artsmall, housing, 1D whose files per train/test and features/labels are organized as in project 2.

The first 2 datasets are drawn from project 2 but we normalized the features and labels and reduced the size of the train and test sets to reduce the run time for the project. 1D is a 1-dimensional artificial dataset where we can visualize the predictions.

3. Implementing GP

3.1. Mean and Covariance functions:

We will use a zero-mean function. The covariance functions we use are exactly as in lecture slides.

$C(x_1, x_2) = \delta(x_1, x_2)\frac{1}{\beta} + \frac{1}{\alpha}k(x_1, x_2)$, where $k(x_1, x_2) = (x_1^\top x_2) + 1$ for the linear kernel and $k(x_1, x_2) = e^{-0.5\|x_1 - x_2\|^2/s^2}$ for the square exponential (a.k.a. RBF) kernel.

In matrix form, this looks like $C(X, X) = \frac{1}{\beta}I + \frac{1}{\alpha}K(X, X)$ and when examples in X are distinct from examples in Z we get $C(X, Z) = \frac{1}{\alpha}K(X, Z)$.

3.2. Predictive distribution, derivatives, and model selection:

The equations for the log evidence, predictive distribution, and derivatives w.r.t. $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{s}$ are given in the slides. As suggested in the slides, for model selection in this project we need to perform gradient ascent on $ln\alpha$, $ln\beta$, lns.

Note: implementing the GP equations normally requires some care to avoid numerical instability in calculations (for example using the Cholesky decomposition to avoid computing inverses). However, for this project a direct implementation of the equations works fine.

3.3. Optimization:

Initialize $\boldsymbol{\alpha} = 1$, $\boldsymbol{\beta} = 1$ and $\boldsymbol{s} = 5$, except for dataset 1D where $\boldsymbol{s}$ should be initialized to 0.1 (of course $\boldsymbol{s}$ is only relevant when using the RBF kernel). For model selection use gradient ascent on the log values of hyperparameters with learning rate of $\eta = 0.01$ for up to 100 iterations. The value of the log evidence during optimization should be tracked and need to stop the optimization early if the log evidence does not change “too much”. Specifically let **a** be the log evidence before the update, **b** be the value after the update and **c = (b - a)/|a|** be the relative change. Then we can stop the optimization if $\mathbf{c \leq 10^{-5}}$

Notes: It is a good idea for debugging to make sure that the log evidence is increasing, i.e., b > a, with each gradient step. Setting the learning rate is tricky in general but using the value specified works well for our datasets. Similarly, the results are somewhat sensitive to the initial value of $\boldsymbol{s}$ but the values above yield good results for our datasets.

4. Implementing GP

We will use MNLL (mean negative log likelihood) on the test set. The negative log likelihood on example $\boldsymbol{i}$ is NLL = $-lnN(t_i|m_i, v_i)$ where m_i, v_i are the mean and variance of the predictive distribution. Then we have MNLL = $\frac{1}{N}\Sigma_i - lnN(t_i|m_i, v_i)$. In addition, to compare to results with Bayesian Linear Regression we calculate the test set MSE: MSE = $\frac{1}{N}\Sigma_i(m_i - t_i)^2$.
(Whereas in project 2 we had $m_i = m_N{}^T\phi(x_{i)})$.

4.1. Visualizing performance on the 1D dataset:

For dataset 1D run the algorithm until it stops and record the final learned function. Then visualize the results as follows. The true function for this dataset is:

- if *x* > 1:5 then: *f(x)* = -1;
- if *x* < -1:5 then: *f(x)* = 1;
- otherwise, *f(x) = sin(6*x).*

We need to plot the true function, and the mean of the predictive distribution with +/- 2 standard deviations around the mean, where x is in the range [-3,3]. This should be done with both the RBF kernel and the linear kernel, i.e., need to provide 2 such plots.

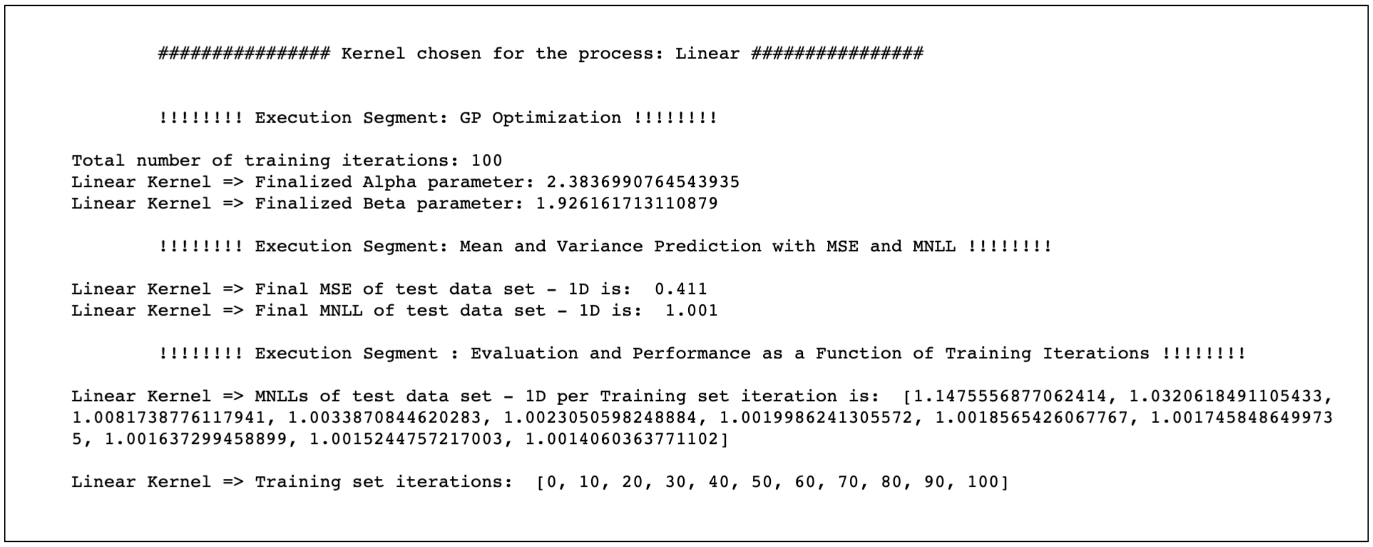
4.2. Performance as a function of iterations:

We will run the algorithm, on each of the 4 datasets, with both the RBF and Linear kernels, and record the test set MNLL performance as a function of the number of training iterations. To save in compute time (since evaluation of GPs is time consuming) evaluate the prediction every 10 iterations, and after the last iteration (for example, if the algorithm stops at iteration 33, we will have evaluations at 0,10,20,30,33 iterations). Then we need to plot the MNLL as a function of the number of training iterations. Thus, we expect 8 plots from 4 datasets * 2 kernels.

Let’s have a look on the outputs from all the 4 datasets as the algorithm designed above.

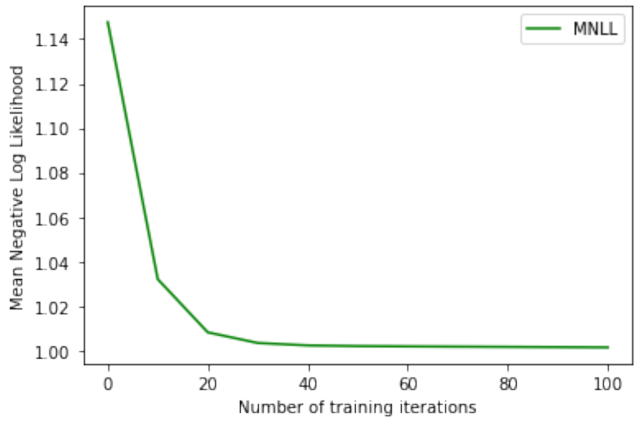
<u>Dataset: 1D</u>

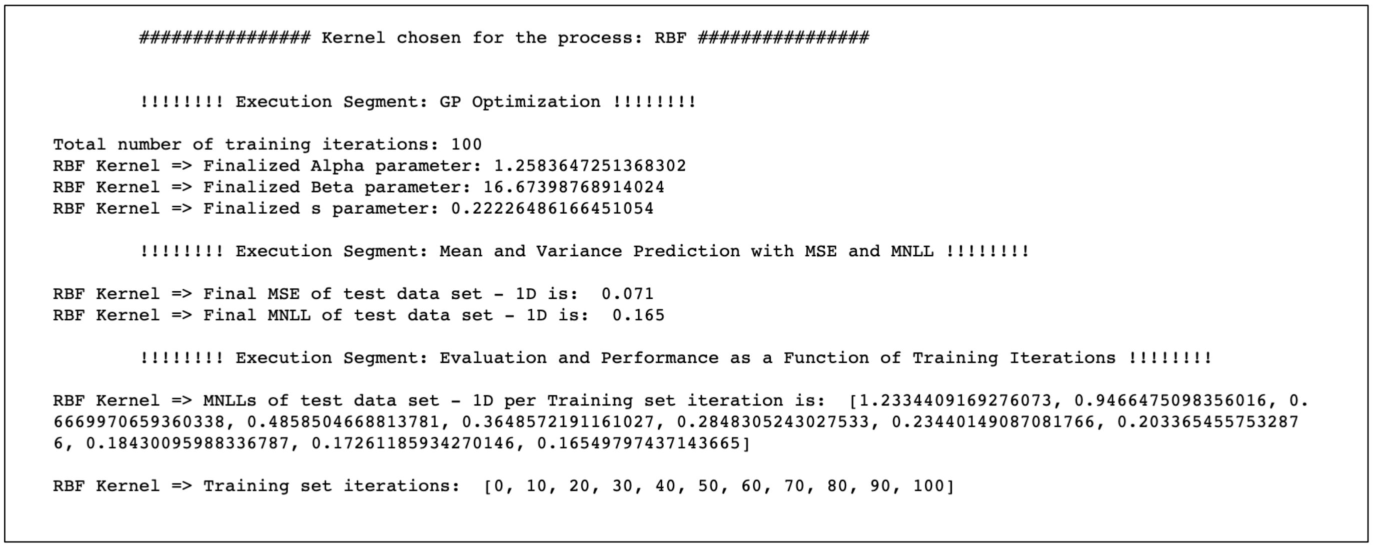
```
        ============ Processing the data file: 1D ============

TRAINING SET:  (30, 1) (30, 1)
TESTING SET:  (60, 1) (60, 1)

Training data size: 30
Training feature size: 1

Testing data size: 60
Testing feature size: 1
```



```
        ################ Kernel chosen for the process: Linear ################


        !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

Total number of training iterations: 100
Linear Kernel => Finalized Alpha parameter: 2.3836990764543935
Linear Kernel => Finalized Beta parameter: 1.926161713110879

        !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

Linear Kernel => Final MSE of test data set - 1D is:  0.411
Linear Kernel => Final MNLL of test data set - 1D is:  1.001

        !!!!!!!! Execution Segment : Evaluation and Performance as a Function of Training Iterations !!!!!!!!

Linear Kernel => MNLLs of test data set - 1D per Training set iteration is:  [1.1475556877062414, 1.0320618491105433,
1.0081738776117941, 1.0033870844620283, 1.0023050598248884, 1.0019986241305572, 1.0018565426067767, 1.001745848649973
5, 1.001637299458899, 1.0015244757217003, 1.0014060363771102]

Linear Kernel => Training set iterations:  [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

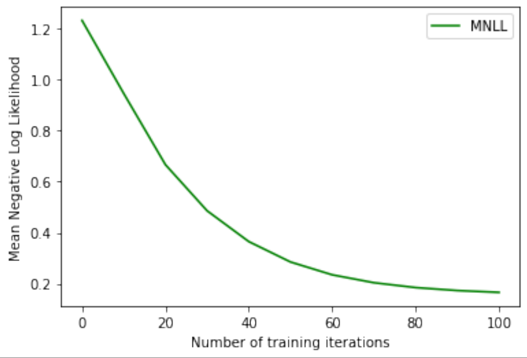
********* MNLL performance vs Number of Training Iterations for Kernel: Linear & dataset: 1D *********





```
        ################ Kernel chosen for the process: RBF ################


        !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

Total number of training iterations: 100
RBF Kernel => Finalized Alpha parameter: 1.2583647251368302
RBF Kernel => Finalized Beta parameter: 16.67398768914024
RBF Kernel => Finalized s parameter: 0.22226486166451054

        !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

RBF Kernel => Final MSE of test data set - 1D is:  0.071
RBF Kernel => Final MNLL of test data set - 1D is:  0.165

        !!!!!!!! Execution Segment: Evaluation and Performance as a Function of Training Iterations !!!!!!!!

RBF Kernel => MNLLs of test data set - 1D per Training set iteration is:  [1.2334409169276073, 0.9466475098356016, 0.
6669970659360338, 0.4858504668813781, 0.3648572191161027, 0.2848305243027533, 0.23440149087081766, 0.203365455753287
6, 0.18430095988336787, 0.17261185934270146, 0.16549797437143665]

RBF Kernel => Training set iterations:  [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

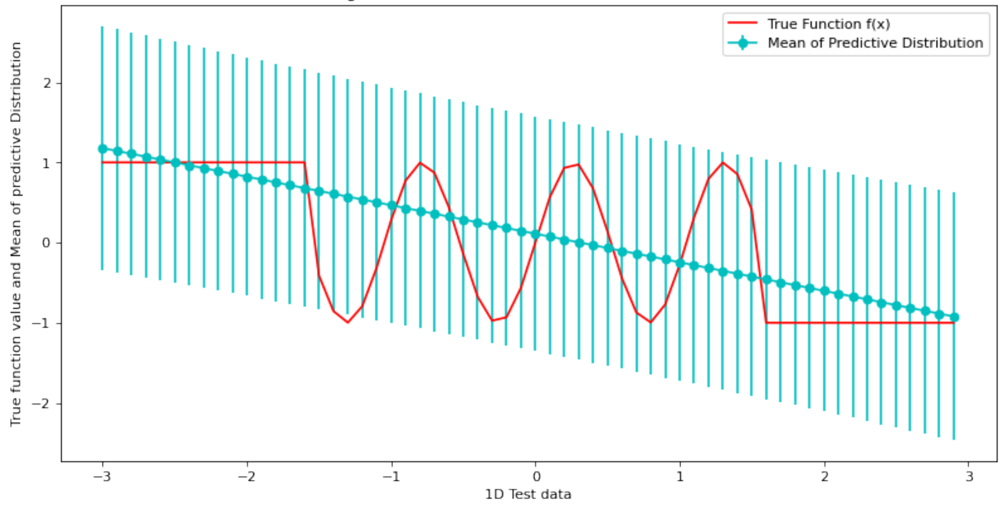
```
********* MNLL performance vs Number of Training Iterations for Kernel: RBF & dataset: 1D *********
```



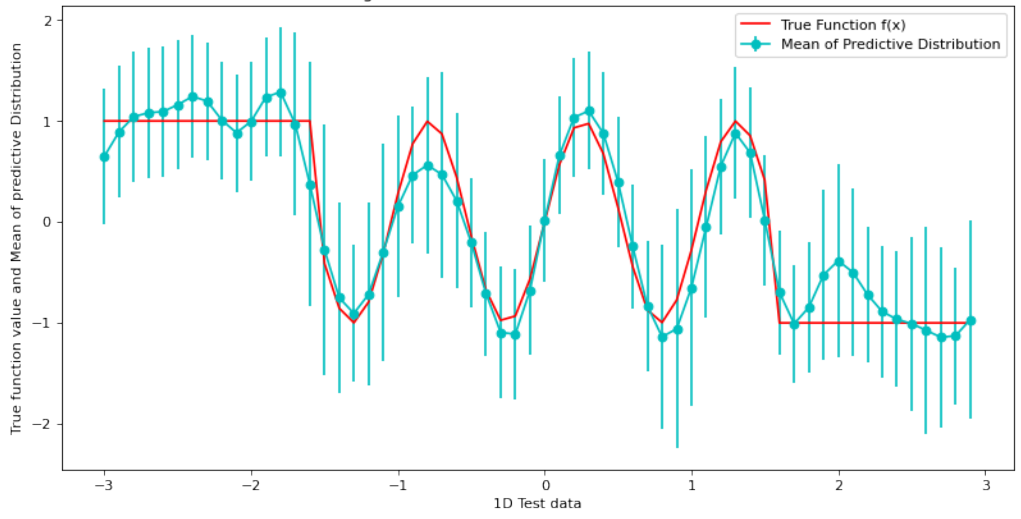
Visualizing performance on the 1D dataset :

```
!!!!!!!! Execution Segment: Visualizing Performance on the 1D Dataset !!!!!!!!
```

```
********* Visualizing Performance on the 1D Dataset with kernel: Linear *********
```



```
********* Visualizing Performance on the 1D Dataset with kernel: RBF *********
```



```
!!!!! Total Execution time for completing the RBF process: 1.6283502578735352 seconds!!!!!

======= Total Execution time for completing the entire process: 2.526801824569702 seconds =======
```

The MSE and Final MNLL of the test set with optimized hyper-parameters are significantly different in both Linear and RBF kernels. Linear kernel having very high MSE and MNLL compared to that of RBF kernel. For both the Linear and RBF kernels, it takes 100 iterations before the algorithm stops which is also the limiting condition. The log evidence was not converged for sure till then. The hyper-parameters are also very differently ranged in both the algorithms.

We can observe that the MNLL curve drops significantly from Iteration 0 to Iteration 10 and again a slow decrement at iteration 20, then gradually decreases till iteration 100 in case of Linear kernel. Whereas in case of RBF kernel, the MNLL decreases almost exponentially from iteration 0 to iteration 100.

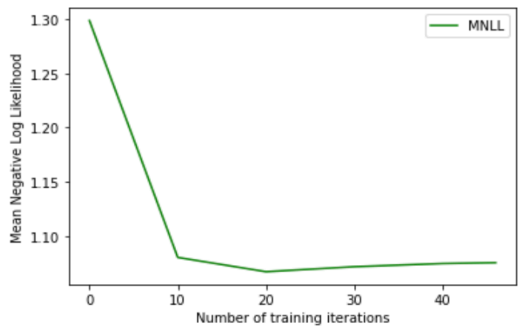
The true function value that has been computed as given formulation, stays 1 from -3 to -1.5 and 1 between 1.5 to 3 but fluctuates and exhibits sinusoidal nature between -1.5 to 1.5.

In case of Linear curve, the mean of predictive distribution linearly decreases almost from 1 to -1 within the range of -3 to 3. Whereas for RBF kernel, it follows similar sinusoidal nature with almost similar mean value as the true function value between -1.5 and 1.5. But between -3 and 1.5 and below 1.5 till 3 the mean value fluctuates sinusoidally but at a lesser rate.

<u>Dataset: crime</u>

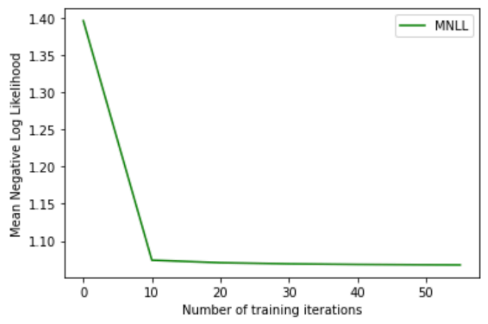
```
TRAINING SET:  (298, 100) (298, 1)
TESTING SET:  (400, 100) (400, 1)

Training data size: 298
Training feature size: 100

Testing data size: 400
Testing feature size: 100


        ################ Kernel chosen for the process: Linear ################


        !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

Total number of training iterations: 46
Linear Kernel => Finalized Alpha parameter: 353.3792778969981
Linear Kernel => Finalized Beta parameter: 2.604925838601952

        !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

Linear Kernel => Final MSE of test data set - crime is:  0.503
Linear Kernel => Final MNLL of test data set - crime is:  1.075

        !!!!!!!! Execution Segment: Evaluation and Performance as a Function of Training Iterations !!!!!!!!

Linear Kernel => MNLLs of test data set - crime per Training set iteration is:  [1.2986804486556085, 1.08025037863531
61, 1.067070659916206, 1.0716850074062576, 1.074624451811614, 1.0754070879266502]

Linear Kernel => Training set iterations:  [0, 10, 20, 30, 40, 46]
```

********* MNLL performance vs Number of Training Iterations for Kernel: Linear & dataset: crime *********



```
!!!!! Total Execution time for completing the Linear process: 1.3158400058746338 seconds!!!!!
```

```
        ################ Kernel chosen for the process: RBF ################


        !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

 Total number of training iterations: 55
 RBF Kernel => Finalized Alpha parameter: 0.7
 RBF Kernel => Finalized Beta parameter: 2.8
 RBF Kernel => Finalized s parameter: 21.3

        !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

 RBF Kernel => Final MSE of test data set - crime is:  0.499
 RBF Kernel => Final MNLL of test data set - crime is:  1.068

        !!!!!!!! Execution Segment: Evaluation and Performance as a Function of Training Iterations !!!!!!!!

 RBF Kernel => MNLLs of test data set - crime per Training set iteration is:  [1.3965973032132848, 1.0741574062051553,
 1.0707503722223668, 1.0692352745874032, 1.0684239673377218, 1.0679392565726828, 1.0677675692705468]

 RBF Kernel => Training set iterations:  [0, 10, 20, 30, 40, 50, 55]

  ********* MNLL performance vs Number of Training Iterations for Kernel: RBF & dataset: crime *********
```



```
 !!!!! Total Execution time for completing the RBF process: 67.22704887390137 seconds!!!!!


 =========== Total Execution time for completing the entire process: 69.44945812225342 seconds ===========
```

The MSE and Final MNLL of the test set with optimized hyper-parameters are almost similar in both Linear and RBF kernels. For Linear kernel, it takes 46 iterations and whereas for RBF kernel, it takes 55 iterations before the algorithm stops and finds its optimal hyper-parameters. The Alpha parameter for Linear kernel is very large compared to that of RBF kernel whereas the Beta parameter is almost similar.

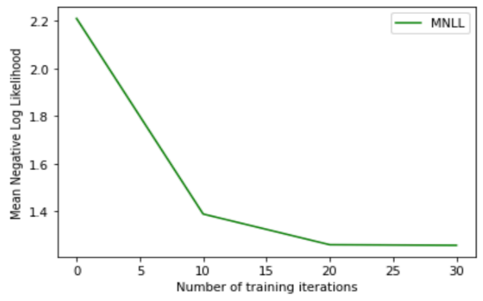
We can observe that the MNLL curve drops significantly from Iteration 0 to Iteration 10 and again a slow decrement at iteration 20, then increases very slowly till iteration 46 in case of Linear kernel. Whereas in case of RBF kernel, the MNLL drops significantly from iteration 0 to iteration 10 and then stays almost same till iteration 55.

<u>Dataset: artsmall</u>

```
         ============ Processing the data file: artsmall ============

 TRAINING SET:  (100, 100) (100, 1)
 TESTING SET:  (400, 100) (400, 1)

 Training data size: 100
 Training feature size: 100

 Testing data size: 400
 Testing feature size: 100
```

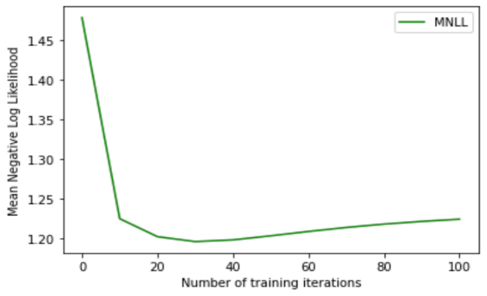
```
        ################ Kernel chosen for the process: Linear ################


        !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

Total number of training iterations: 30
Linear Kernel => Finalized Alpha parameter: 143.8
Linear Kernel => Finalized Beta parameter: 4.1

        !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

Linear Kernel => Final MSE of test data set - artsmall is:  0.71
Linear Kernel => Final MNLL of test data set - artsmall is:  1.258

        !!!!!!!! Execution Segment: Evaluation and Performance as a Function of Training Iterations !!!!!!!!

Linear Kernel => MNLLs of test data set - artsmall per Training set iteration is:  [2.2105129586023198, 1.38894232517
8849, 1.2599161403116108, 1.25758656547072]

Linear Kernel => Training set iterations:  [0, 10, 20, 30]
```

********* MNLL performance vs Number of Training Iterations for Kernel: Linear & dataset: artsmall *********



```
!!!!! Total Execution time for completing the Linear process: 0.364621639251709 seconds!!!!!

         ################ Kernel chosen for the process: RBF ################


         !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

 Total number of training iterations: 100
 RBF Kernel => Finalized Alpha parameter: 0.4
 RBF Kernel => Finalized Beta parameter: 6.5
 RBF Kernel => Finalized s parameter: 16.2

         !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

 RBF Kernel => Final MSE of test data set - artsmall is:  0.678
 RBF Kernel => Final MNLL of test data set - artsmall is:  1.224

         !!!!!!!! Execution Segment: Evaluation and Performance as a Function of Training Iterations !!!!!!!!

 RBF Kernel => MNLLs of test data set - artsmall per Training set iteration is:  [1.4778762619113235, 1.22425777268862
 61, 1.201593384254731, 1.1953938040130034, 1.1975713598665745, 1.2026654761755282, 1.2081891679447672, 1.213201498729
 9506, 1.2174294784828836, 1.2208674325722775, 1.223605745128963]

 RBF Kernel => Training set iterations:  [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

********* MNLL performance vs Number of Training Iterations for Kernel: RBF & dataset: artsmall *********



```
 !!!!! Total Execution time for completing the RBF process: 23.95816922187805 seconds!!!!!

 =========== Total Execution time for completing the entire process: 26.482202291488647 seconds ===========
```

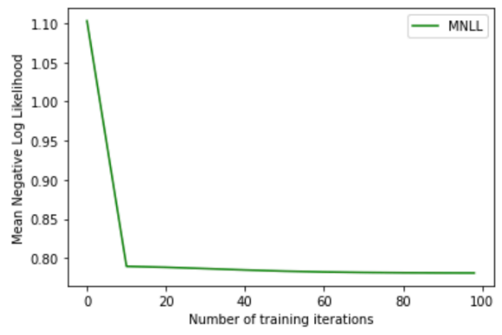
The MSE and Final MNLL of the test set with optimized hyper-parameters are nearby in both Linear and RBF kernels. For Linear kernel, it takes 30 iterations and whereas for RBF kernel, it runs for 100 iterations before the algorithm stops and returns the hyper-parameters. The Alpha parameter for Linear kernel is very large compared to that of RBF kernel whereas the Beta parameters are 4.1 and 6.5 respectively.

We can observe that the MNLL curve drops significantly from Iteration 0 to Iteration 10 and again a slow decrement at iteration 20 and stay similar till iteration 30 in case of Linear kernel. Whereas in case of RBF kernel, the MNLL drops significantly from iteration 0 to iteration 10 and then gradually decreases till iteration 30 but again increases till iteration 100 slowly and reaches similar MNLL as iteration 10.

<u>Dataset: housing</u>

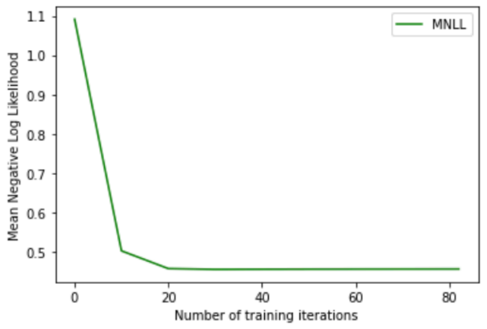
```
        ============ Processing the data file: housing ============

 TRAINING SET:  (102, 13) (102, 1)
 TESTING SET:  (404, 13) (404, 1)

 Training data size: 102
 Training feature size: 13

 Testing data size: 404
 Testing feature size: 13


        ################ Kernel chosen for the process: Linear ################


        !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

 Total number of training iterations: 98
 Linear Kernel => Finalized Alpha parameter: 21.2
 Linear Kernel => Finalized Beta parameter: 4.0

        !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

 Linear Kernel => Final MSE of test data set - housing is:  0.288
 Linear Kernel => Final MNLL of test data set - housing is:  0.781

        !!!!!!!! Execution Segment: Evaluation and Performance as a Function of Training Iterations !!!!!!!!

 Linear Kernel => MNLLs of test data set - housing per Training set iteration is:  [1.1028226326076007, 0.789551564889
 4359, 0.7883658069035611, 0.7867022433826781, 0.7849244232369855, 0.783417297879097, 0.7823727881829248, 0.7817439910
 692316, 0.7813928707180676, 0.7812015164760565, 0.7811130284370567]

 Linear Kernel => Training set iterations:  [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 98]
```

********* MNLL performance vs Number of Training Iterations for Kernel: Linear & dataset: housing *********



```
 !!!!! Total Execution time for completing the Linear process: 0.4816439151763916 seconds!!!!!
```

```
        ################ Kernel chosen for the process: RBF ################


        !!!!!!!! Execution Segment: GP Optimization !!!!!!!!

Total number of training iterations: 82
RBF Kernel => Finalized Alpha parameter: 0.3
RBF Kernel => Finalized Beta parameter: 12.8
RBF Kernel => Finalized s parameter: 4.8

        !!!!!!!! Execution Segment: Mean and Variance Prediction with MSE and MNLL !!!!!!!!

RBF Kernel => Final MSE of test data set - housing is:  0.178
RBF Kernel => Final MNLL of test data set - housing is:  0.456

        !!!!!!!! Execution Segment: Evaluation and Performance as a Function of Training Iterations !!!!!!!!

RBF Kernel => MNLLs of test data set - housing per Training set iteration is:  [1.0912721488480353, 0.502528907085271
3, 0.45739819919078306, 0.4554305089325091, 0.45582837793753456, 0.45607877610201614, 0.4562252486870092, 0.456336334
47600835, 0.45643120148830485, 0.4564486624062925]

RBF Kernel => Training set iterations:  [0, 10, 20, 30, 40, 50, 60, 70, 80, 82]

 ********* MNLL performance vs Number of Training Iterations for Kernel: RBF & dataset: housing *********
```



```
!!!!! Total Execution time for completing the RBF process: 21.392690896987915 seconds!!!!!


=========== Total Execution time for completing the entire process: 24.163004875183105 seconds ===========
```

The MSE and Final MNLL of the test set with optimized hyper-parameters are slightly higher for Linear kernel than RBF kernels. For Linear kernel, it takes 98 iterations and whereas for RBF kernel, it runs for 82 iterations before the algorithm stops and returns the optimized hyper-parameters. The Alpha parameter for Linear kernel is very large compared to that of RBF kernel whereas the Beta parameter is higher and almost thrice for RBF kernel than that of Linear kernel.

We can observe that the MNLL curve has 'L' shape and drops significantly from Iteration 0 to Iteration 10 and stays almost same till 98 iterations in case of Linear kernel. Whereas in case of RBF kernel, the MNLL drops significantly from iteration 0 to iteration 10 and then drops at iteration 20, but thereafter stays same till iteration 82.

4.3. Performance as a function of Iteration:

In addition to the above, we need to record the values of α, β and test set MSE after the last iteration (i.e., when the algorithms stops). Tabulate these results and compare them to the results of Bayesian Linear Regression, either by running code from project 2, or taken from the table given in the problem statement.

Given				Kernel = Linear				Kernel = RBF			
Dataset	MSE	α	β	Iterations	MSE	α	β	Iterations	MSE	α	β
1D	0.39	7.5	1.9	100	0.411	2.4	1.9	100	0.071	1.3	16.7
crime	0.5	357.5	2.6	46	0.503	353.4	2.6	55	0.499	0.7	2.8
artsmall	0.716	141.4	4.23	30	0.71	143.8	4.1	100	0.678	0.4	6.5
housing	0.288	20.4	4	98	0.288	21.2	4	82	0.178	0.3	12.8

4.4. Discuss the results:

With all these results recorded, what can you observe w.r.t. the performance of the algorithms? Are the BLR and GP with linear kernel behaving similarly w.r.t. α, β, MSE as expected? How does the performance of GP compare when changing RBF vs. linear kernel? What are potential advantages or disadvantages of each method?

Answer: The effects of optimized hyper-parameter on the MNLL per training iterations and corresponding plots are shown in the section 4.2. We can observe that the algorithm comparatively performs better for RBF kernel than Linear kernel in terms of the MSE of the predictive distributions. Though it takes a greater number of training iterations to converge the log evidence and produce the optimized hyper-parameter set.

The BLR and GP with Linear kernel behaves similarly w.r.t α, β, MSE as expected. We can refer to the table above and observe that the α, β, MSE are all same for BLR (given) and GP with Linear kernel excepts the 1D dataset where the α is lesser in GP and the MSE is little higher than that of BLR.

When we change the GP to RBF kernel, it performs better compared to Linear kernel. Referring to the above table, we can observe that the MSEs are far better in GP with RBF kernel. The α parameter values in RBF kernel GP are way lesser than the Linear kernel but the β parameter values are greater in GP with RBF kernel than Linear kernel. It comparatively takes a greater number of training iterations before the algorithm stops in RBF kernel than the number of iterations in GP with Linear kernel.