**Gaussian Processes**

**Overview:**

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

**Datasets:**

We have 4 datasets, crime, artsmall, housing, 1D.

1D is a 1 dimensional artificial dataset where we can visualize the predictions.

**Implementing GP:**

**Mean and Covariance functions**:

We will use a zero mean function. The covariance function we use is:

,where  for the linear kernel and

 for the square exponential (a.k.a. RBF) kernel.

In matrix form, this looks like C(X, X) = 1/β \* I + 1/α \* K(X, X) and when examples in X are distinct from examples in Z we get C(X, Z) = 1/α \* K(X, Z).

**Predictive distribution, derivatives, and model selection**:

The equations for the log evidence, predictive distribution and derivatives w.r.t. α, β, s are given below:

Text

Description automatically generated ,

Where A picture containing text

Description automatically generated.

For model selection in this project, we will perform gradient ascent on ln α, ln β, ln s.

**Evaluation:**

We will use MNLL (mean negative log likelihood) on the test set. The negative log likelihood on example i is

 where mi , vi are the mean and variance of the predictive distribution. Then we have

. In addition, to compare to results with Bayesian Linear Regression we calculate the test set MSE:

**.**

**Visualizing performance on the 1D dataset:**

For dataset 1D, we run the algorithm until it stops and record the final learned function. Then we 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 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 is done with both the RBF kernel and the linear kernel, i.e., 2 such plots are made.

**Performance as a function of iterations:**

We 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) we evaluate the prediction every 10 iterations, and after the last iteration (for example, if the algorithm stops at iteration 33 you will have evaluations at 0,10,20,30,33 iterations). Then we plot the MNLL as a function of the number of training iterations. Thus we create 8 plots from 4 datasets ∗ 2 kernels.

**Comparison to Bayesian Linear Regression:**

In addition to the above, we also record the values of α, β and test set MSE after the last iteration (i.e., when the algorithms stops). We then tabulate these results and compare them to the results of Bayesian Linear Regression from the table below.

Table

Description automatically generated

**Info about the code:**

1. The code is divided into three sections: Part 1: GP for Linear Kernel, Part 2: GP for RBF Kernel and Part 3: Visualisation of performance of 1D dataset.
2. Out of these 3 sections of code, Part 1 is left uncommented and the remaining three are commented.

**Steps to run the code:**

1. **Change the paths of the datasets according to the paths on your system.  
     
   Text

   Description automatically generated**
2. **The default dataset chosen is artsmall. In order to test the code on other datasets, just replace the dataset name in all 4 file paths from the list of names given.  
     
   Text

   Description automatically generated**
3. **Part 1: GP with Linear Kernel is uncommented, and can be run right after step 1.  
   Text

   Description automatically generated**
4. **In order to run Part 2: GP with RBF Kernel, just uncomment Part 2 section and comment the Part 1 section and execute the code.  
   Text

   Description automatically generated**
5. **To run Part 3: Visualisation of performance of 1D dataset, just uncomment the section and comment the above two sections and execute the code.  
   Text

   Description automatically generated**
6. **The plots will be automatically shown at the end of the execution.**