Homework 2 – Gaussian Process Regression

Submitted in partial fulfillment of the requirements for the course of

CMSC818B – Decision Making for Robotics

By

Adheesh Chatterjee

Course Faculty
Prof. Pratap Tokekar

A. JAMES CLARK
SCHOOL OF ENGINEERING

GAUSSIAN PROCESS REGRESSION

Gaussian Process Regression calculates the probability distribution over all admissible functions that fit the data rather than calculating the probability distribution of parameters of a specific function. A Gaussian process is like an infinite-dimensional multivariate Gaussian distribution, where any collection of the labels of the dataset are joint Gaussian distributed.

IMPLEMENTATION

The general process of implementation is as follows -

1) We import all libraries reqd to process and plot our data points
2) The files for train and test data are imported and the variables for x_train, y_train and x_test are stored in variables. Our goal is to predict y for the x_test values.
    2.1) In 4a, Since we know our input is y=sin(3x). We know ground truth values for sin(3x) for the given x_test
    2.2) In 4b, we have the solution provided, so those serve as ground truth values
3) We import the gaussian process regressor and the different kernels that we will test our GP regression on.

KERNEL	**FORMULA**	**REASON OF CHOICE**
RBF Kernel (RBF)	k(x_i, x_j) = exp(-1 / 2 d(x_i / length_scale, x_j / length_scale)^2)	The RBF kernel is generic kernel used to model the error as it is gaussian
Constant Kernel (C)	k(x_1, x_2) = constant_value for all x_1, x_2	Constant kernel is used to account for noise in the data (We later make noise gaussian)
Rational Quadratic Kernel (RQ)	k(x_i, x_j) = (1 + d(x_i, x_j)^2 / (2*alpha * length_scale^2))^-alpha	This kernel accounts for non linear noise
Exponential Sine Squared Kernel (ESS)	exp(-2 (sin(pi / periodicity * d(x_i, x_j)) / length_scale) ^ 2)	In 4a) since it is periodic, this function should work well

We essentially combine kernels and use them to get better results
As seen from the code

Kernel 1 is C1*RBF1+C2
Kernel 2 is C1*RBF1+RBF2
Kernel 3 is C1*RQ1 + RBF2

- Kernel 4 is C1*ESS1 + RBF2, we use this kernel only for problem 4a as it should fit perfectly for a periodic function

4) Now we use the in built gaussian process regressor function from sklearn for all 4 kernels defined above individually to create the model.
5) We then fit our x_train and y_train dataset to the model created.

6) Now our model is ready for prediction. Using the x_test values we input them into the model and predict the output y_pred.
7) We then proceed to check the values of y_pred with the ground truth values using the Mean Squared Error.
8) Finally, we plot our results for all 3/4 kernels to visualize the results

COVARIANCE FUNCTION USED

Our first covariance function is **Kernel 1 = C1*RBF1+C2**
Here we use a 2 constant kernels and one rbf kernel. The first constant kernel is combined with the rbf kernel to account for gaussian noise

Our second covariance function is **Kernel 2 = C1*RBF1+RBF2**
Here we use 1 constant kernel and 2 rbf kernels. The first constant kernel is combined with rbf kernel to account for gaussian noise, the second rbf kernel is used to model the data

Our third covariance function is **Kernel 3 = C1*RQ1 + RBF2**
Here we use 1 constant kernel, 1 Radial Quadratic kernel and 1 rbf kernel. The constant kernel combined with RQ kernel accounts for non linear noise in our data.

Our fourth covariance function is **Kernel 4 = C1*ESS1 + RBF2 (This is used only for 4a)**
Here we use 1 constant kernel, 1 Exponential Sine Squared kernel and 1 rbf kernel. The constant kernel combined with ESS kernel is to account for periodicity of the function

HYPERPARAMETERS SET/LEARNED

We set the hyperparameters randomly to initialize them.

They are learned as follows -

For Problem 4a)

Our first covariance function is **Kernel 1 = C1*RBF1+C2**
Kernel 1 = 0.7332 * RBF(length_scale=0.481) + 0.0316**2**

Our second covariance function is **Kernel 2 = C1*RBF1+RBF2**
Kernel 2 = 0.03162 * RBF(length_scale=0.561) + RBF(length_scale=0.561)**

Our third covariance function is **Kernel 3 = C1*RQ1 + RBF2**
Kernel 3 = 0.03162 * RationalQuadratic(alpha=1e+05, length_scale=0.561) + RBF(length_scale=0.561)**

Our fourth covariance function is **Kernel 4 = C1*ESS1 + RBF2**
Kernel 4 = 3.112 * ExpSineSquared(length_scale=6.09, periodicity=2.09) + RBF(length_scale=100)**

For Problem 4b)

Our first covariance function is **Kernel 1 = C1*RBF1+C2**
Kernel 1 = 17.12 * RBF(length_scale=0.0122) + 31.6**2**

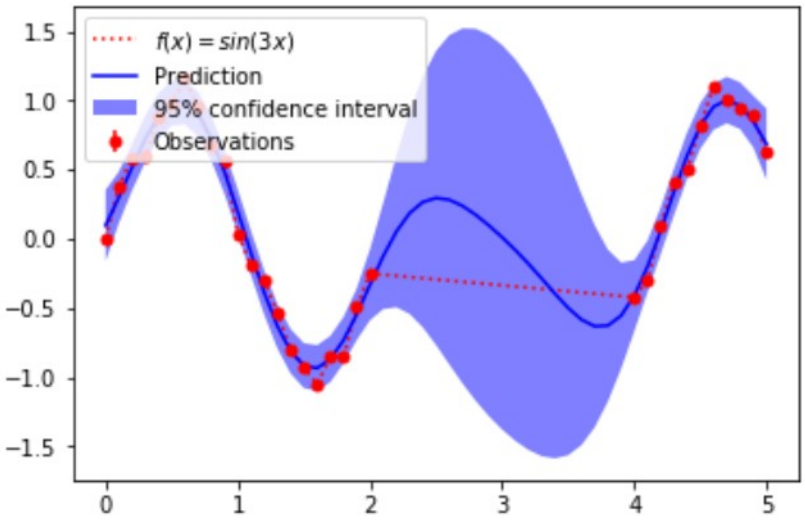
Our second covariance function is **Kernel 2 = C1*RBF1+RBF2**
Kernel 2 = 31.62 * RBF(length_scale=88) * RBF(length_scale=86.8) * RBF(length_scale=59) * RBF(length_scale=5.09) + RBF(length_scale=0.0374)**

Our third covariance function is **Kernel 3 = C1*RQ1 + RBF2**
Kernel 3 = 31.62 * RationalQuadratic(alpha=0.00319, length_scale=2.85) + RBF(length_scale=0.0277)**

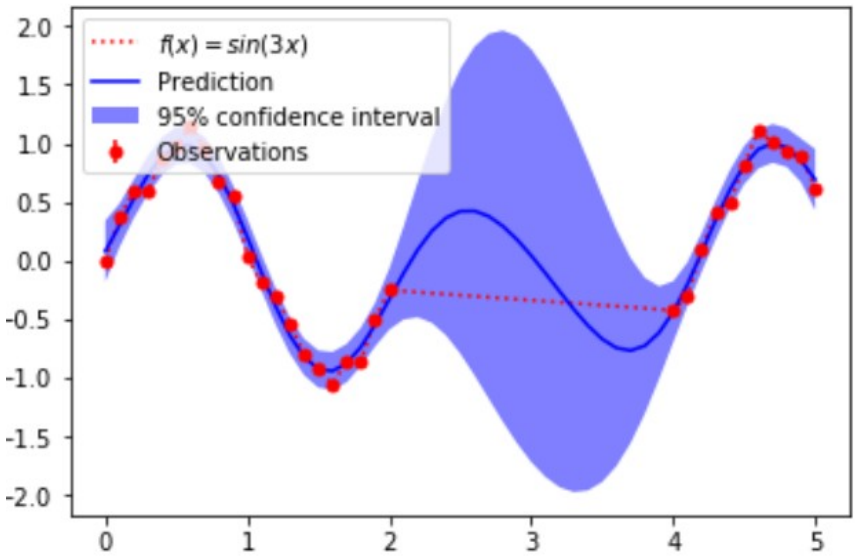
PLOT

For problem 4a)

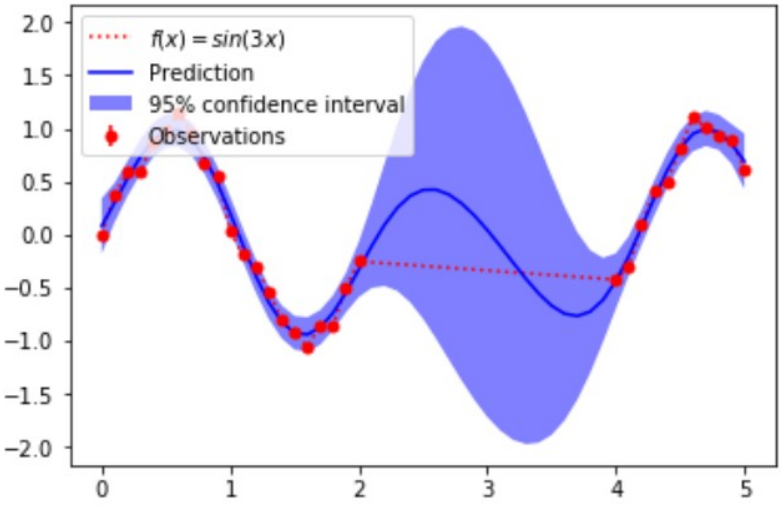
Kernel 1



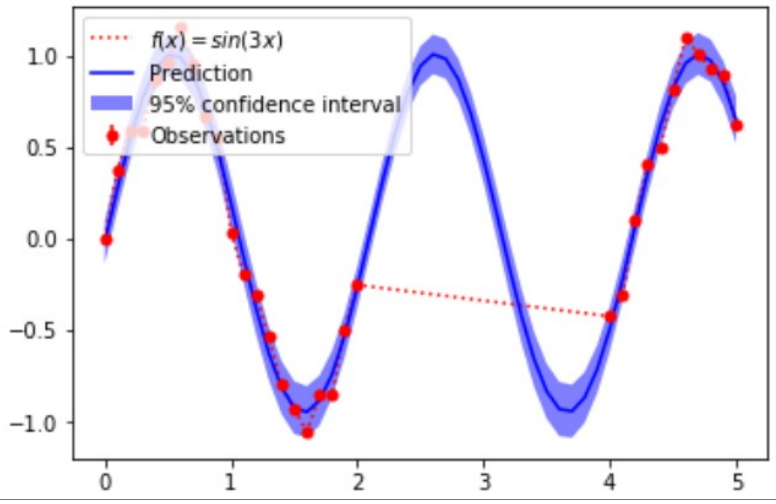
KERNEL 2



KERNEL 3



KERNEL 4



- Since problem 4b is 4D problem, we cannot plot it as it cannot be visualized, hence we rely on our Mean Squared Error to check fit.

MEAN SQUARED ERROR (MSE)

Mean Squared Error is an estimator measures the average of error squares i.e. the average squared difference between the estimated values and true value.
It is defined as -

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

For Problem 4a)

MSE for kernel 1 is 0.07321080101813422
MSE for kernel 2 is 0.04592746345311912
MSE for kernel 3 is 0.0459274680372526
MSE for kernel 4 is 0.0011681027509283575

For Problem 4b)

MSE for kernel 1 is 269.14713567903657
MSE for kernel 2 is 30.636923584683218
MSE for kernel 3 is 10.291429676381382

SOME NOTES

The code for the project was done on Jupyter Notebooks, the code files were then converted to .py, which is why it might generate some error on running them.

The jupyter notebooks were exported as PDFs and are attached for viewing.
Since Code for Problem 4b) takes a while to run, esp given that I am using a combination of multiple kernels and fitting the data to all 3 kernels, the outputs have not been processed in the jupyter notebook pdf. However, on running the actual code file, we will see the outputs converge to the MSE as seen above.