CS 391L Machine Learning HW4: Gaussian Process

Yuege Xie, EID: yx4256, Email: yuege@ices.utexas.edu[*]

1 Introduction

Gaussian Process (GP) is a stochastic process that every finite collection of the infinite random variables has a multivariate normal distribution. Gaussian Process Regression (GPR) is a non-parametric, Bayesian approach to regression based on the assumption that the data are from GP. The benefit of GPR is that it works well on small dataset and provides uncertainty measurements on the predictions. The Radial Basis Function (RBF) kernel is $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}\right)$. In this assignment, we use GPR with RBF kernel and optimize the log posterior likelihood with gradient fitting algorithm to fit the motion data captured by sensors on different objects. We choose one coordinate of a marker of one object motion at a time and utilize all 5 trials. The example of sampled data are shown in green points ("observations") in Figure 1. We use both a global uniform kernel and local kernels of the motion data, compare the performance, plot the trend of parameters, make a table to find clusters of similar parameters, and get the state of muscle co-contraction.

2 Method

In this report, I focus on using RBF kernel in Gaussian Process Regression. I fitted the data with global kernel and local kernels, and the detailed steps are as follows.

1. **Prepare data:**
   - **Read data:** I load the data from "CT" file folder, choose 15_x as my target coordinate and keep data with the label $15\_c > 0$. In my case, all the data are effective.
   - **Sample data:** For each frame, I randomly sample a data from a trial (there are five trials in total) and store in a dictionary with their frame and elapsed time. I use frame as input and the coordinate as output.
2. **Implement global fitting using GPR:**
   - **Initialize a prior kernel:** I initialize the kernel as "Prior (kernel: 1**2 * RBF(length_scale=100) + WhiteKernel(noise_level=0.001)". The predictions of the prior kernel are in Figure 1, which does not fit anything.
   - **Fit a global kernel:** I use GPR in sklearn to fit the data I prepared with the default optimizer "fmin_l_bfgs_b", which is a gradient based optimizer. The predictions of the posterior kernel are in Figure 2.
3. **Implement local fitting using GPR based on sliding windows:**
   - **Set sliding windows:** I choose sliding window size $s = 100$, and stride $\delta = 10$, i.e. the intervals are $[0, 100), [10, 110), \ldots [930, 1030)$.
   - **Fit local kernels:** I use the same initial kernels as above, and train the parameters with each 100 points and record the parameters σ_f, σ_l and σ_n from gpr.kernel_.theta, which are log values of each value.
   - **Predict using local kernels:** I predict the motion data using a local kernel in the corresponding interval. The final prediction results in Figure 3 is based on the mean of local predictions. For example, For interval $[10, 20)$, the result is mean of predictions from $[0, 100)$ and $[10, 110)$, i.e. $\text{pred}([10, 20)) = \frac{\text{pred}_{[0,100)}([10,20)) + \text{pred}_{[10,110)}([10,20))}{2}$. Another example is $\text{pred}([100, 110)) = \frac{\sum_{i=1}^{10} \text{pred}_{[i*10,100+i*10)}([100,110))}{10}$. I implement this

[*]Oden Institute for Computational Engineering and Sciences, UT Austin.

in diving a vector in the form $[1, \dots, 1, 2, \dots, 2, \dots, 10, \dots, 10, 9 \dots 9, \dots, 1 \dots, 1]^T$ (see details in the code).

- **Plot the parameters in each frame and Make the table of each local interval:** The log-scale value of parameters σ_f, σ_l and σ_n are in Figure 4.

3 Results

The code and plots are in the appendix "hw4-GP.ipynb". I show and analyze the plots and table in the section.

Comparison of Global Kernel and Local Kernels The prediction of motion data using a global kernel are in Figure 1 (prior) and Figure 2 (posterior); The prediction of motion data using local kernels is in Figure 3. Comparison is as follows.

- The 95% confidence interval (grey interval) is tighter in the plot using local kernels, especially in the intervals $[150, 250$ and $[600, 800)$.
- More data are contained in the 95% confidence interval (grey interval) using local kernels, especially in the interval $[400, 450)$.
- The prediction (dark line) is smoother using a global kernel since results of local kernels are the mean of several local kernels.
- In general, using local kernels seem to fit the motion data better.

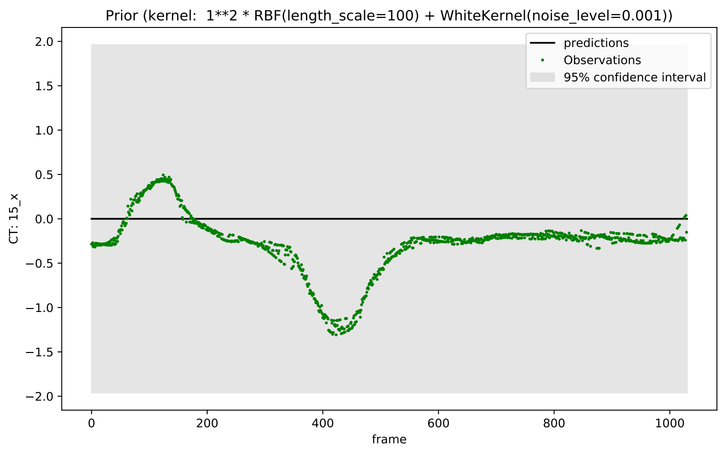

Figure 1: The prediction of motion data using a initial global kernel.

Plots and Table of Parameters' value of local kernels The plot of the parameters' values of local kernels at each frame is in Figure 3, and the table of the values is in Table 1.

- **Plots:** There are intervals with stable parameters (also can be found in highlighted parts of Table 1.) The are also some peaks of the parameters, where we can see variations of parameters with the oscillation of motion data. This means local kernels make sense to fit motion data "more locally", compared to global kernel. the sudden variation of the values of parameters can be used to analyze muscle co-contraction.
- **Table:** I highlight the distinct intervals where the kernels cluster with similar values in yellow and green. (The colors do not have any meaning, which only uses to help distinguish different clusters.) As we can see from the highlighted part, these are 7 clusters: $[0, 130), [70, 230), [330, 450), [480, 590), [560, 680), [710, 830), [820, 940)$. Between them, we can see the sudden variation of the values of parameters, where we can take it as muscle co-contraction.

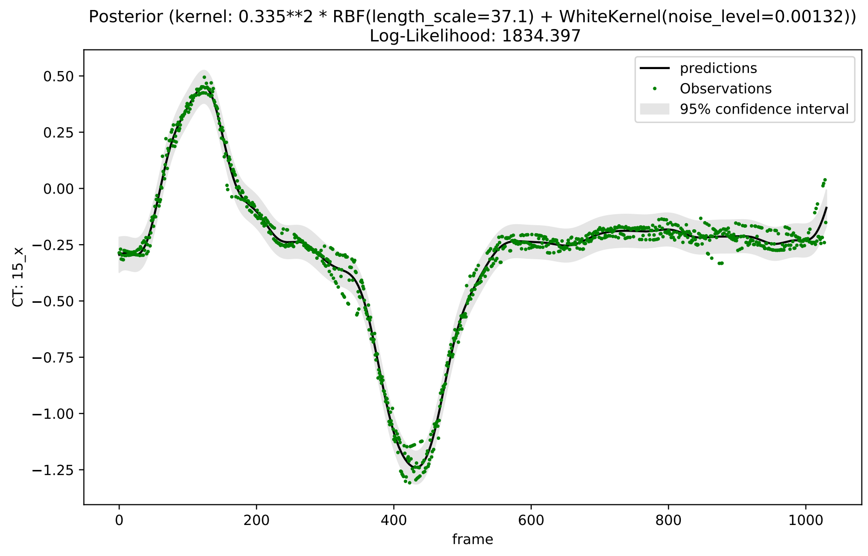

Figure 2: The prediction of motion data using a fitted global kernel.

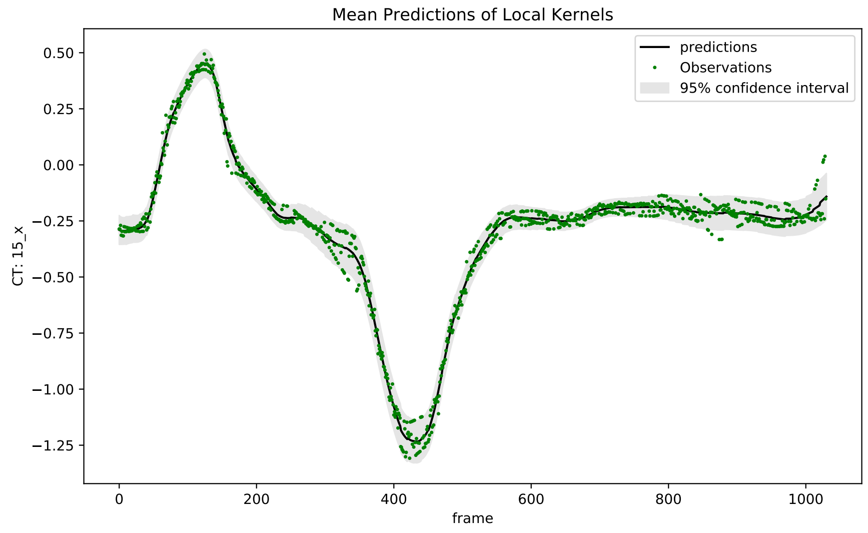

Figure 3: The prediction of motion data using fitted local kernels. Initialization of each local kernel is "Prior (kernel: 1**2 * RBF(length_scale=100) + WhiteKernel(noise_level=0.001)". Sliding window size $s = 100$, and stride $\delta = 10$.

4 Summary

In this project, use both a global uniform kernel and local kernels of the motion data, compare the performance, plot the trend of parameters, make a table to find clusters of similar parameters, and get the state of muscle co-contraction. Comparing the prior and posterior predictions, the learned model fit motion data very well and predict good confidence interval. Comparing a global kernel and local kernels, the local kernels fit more subtle details and can be used to analyze muscle co-contraction, while the global kernel fits a more smoother model. From the local parameters table, we can see distinct intervals where the kernels cluster with similar values and the sudden variations between them, where we can use to analyze muscle co-contraction as well.

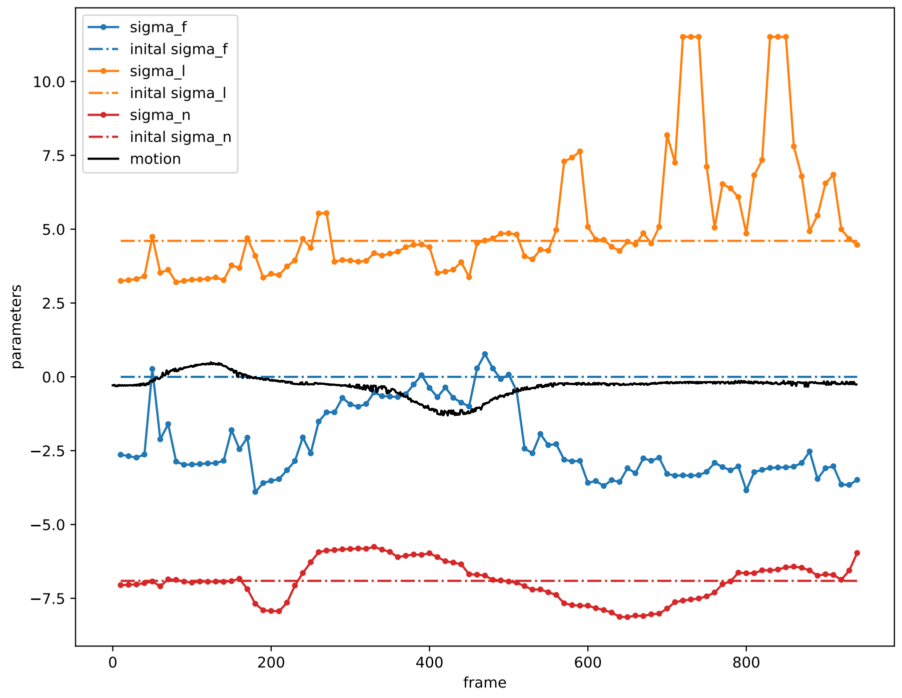

Figure 4: The parameters' values of local kernels at each frame. Initialization of each local kernel is "Prior (kernel: 1**2 * RBF(length_scale=100) + WhiteKernel(noise_level=0.001)". Sliding window size $s = 100$, and stride $\delta = 10$.

interval	σ_f	σ_l	σ_n	interval	σ_f	σ_l	σ_n
$[0, 100)$	-2.64	3.25	-7.05	$[470, 570)$	0.28	4.69	-6.87
$[10, 110)$	-2.68	3.28	-7.04	$[480, 580)$	-0.08	4.85	-6.89
$[20, 120)$	-2.73	3.31	-7.03	$[490, 590)$	0.08	4.86	-6.93
$[30, 130)$	-2.63	3.41	-6.98	$[500, 600)$	-0.47	4.82	-6.97
$[40, 140)$	0.27	4.74	-6.93	$[510, 610)$	-2.43	4.08	-7.08
$[50, 150)$	-2.11	3.53	-7.09	$[520, 620)$	-2.59	3.98	-7.20
$[60, 160)$	-1.60	3.63	-6.86	$[530, 630)$	-1.93	4.31	-7.20
$[70, 170)$	-2.87	3.20	-6.87	$[540, 640)$	-2.31	4.28	-7.29
$[80, 180)$	-2.98	3.25	-6.93	$[550, 650)$	-2.28	4.97	-7.38
$[90, 190)$	-2.97	3.29	-6.97	$[560, 660)$	-2.81	7.29	-7.67
$[100, 200)$	-2.95	3.30	-6.93	$[570, 670)$	-2.86	7.43	-7.73
$[110, 210)$	-2.93	3.32	-6.94	$[580, 680)$	-2.85	7.63	-7.75
$[120, 220)$	-2.92	3.37	-6.93	$[590, 690)$	-3.59	5.08	-7.75
$[130, 230)$	-2.84	3.27	-6.94	$[600, 700)$	-3.53	4.65	-7.83
$[140, 240)$	-1.80	3.78	-6.92	$[610, 710)$	-3.69	4.64	-7.90
$[150, 250)$	-2.45	3.69	-6.84	$[620, 720)$	-3.50	4.40	-7.98
$[160, 260)$	-2.06	4.70	-7.19	$[630, 730)$	-3.56	4.26	-8.13
$[170, 270)$	-3.90	4.10	-7.68	$[640, 740)$	-3.09	4.58	-8.14
$[180, 280)$	-3.60	3.36	-7.91	$[650, 750)$	-3.26	4.48	-8.08
$[190, 290)$	-3.52	3.49	-7.93	$[660, 760)$	-2.76	4.86	-8.10
$[200, 300)$	-3.46	3.45	-7.94	$[670, 770)$	-2.84	4.52	-8.04
$[210, 310)$	-3.16	3.74	-7.65	$[680, 780)$	-2.74	5.08	-8.02
$[220, 320)$	-2.85	3.94	-7.07	$[690, 790)$	-3.29	8.18	-7.85
$[230, 330)$	-2.05	4.67	-6.64	$[700, 800)$	-3.35	7.25	-7.63
$[240, 340)$	-2.59	4.37	-6.28	$[710, 810)$	-3.33	11.51	-7.58
$[250, 350)$	-1.51	5.53	-5.94	$[720, 820)$	-3.35	11.51	-7.54
$[260, 360)$	-1.20	5.54	-5.88	$[730, 830)$	-3.33	11.51	-7.50
$[270, 370)$	-1.20	3.90	-5.86	$[740, 840)$	-3.22	7.12	-7.43
$[280, 380)$	-0.71	3.96	-5.84	$[750, 850)$	-2.91	5.05	-7.30
$[290, 390)$	-0.93	3.94	-5.83	$[760, 860)$	-3.05	6.53	-7.02
$[300, 400)$	-1.01	3.90	-5.81	$[770, 870)$	-3.17	6.38	-6.93
$[310, 410)$	-0.92	3.93	-5.82	$[780, 880)$	-3.03	6.09	-6.63
$[320, 420)$	-0.53	4.19	-5.76	$[790, 890)$	-3.84	4.85	-6.65
$[330, 430)$	-0.65	4.11	-5.85	$[800, 900)$	-3.23	6.83	-6.65
$[340, 440)$	-0.67	4.18	-5.93	$[810, 910)$	-3.15	7.34	-6.55
$[350, 450)$	-0.68	4.24	-6.10	$[820, 920)$	-3.08	11.51	-6.55
$[360, 460)$	-0.59	4.39	-6.06	$[830, 930)$	-3.07	11.51	-6.52
$[370, 470)$	-0.26	4.47	-6.01	$[840, 940)$	-3.06	11.51	-6.45
$[380, 480)$	0.06	4.48	-6.03	$[850, 950)$	-3.04	7.80	-6.43
$[390, 490)$	-0.38	4.39	-5.97	$[860, 960)$	-2.91	6.79	-6.46
$[400, 500)$	-0.69	3.52	-6.10	$[870, 970)$	-2.53	4.93	-6.55
$[410, 510)$	-0.36	3.56	-6.24	$[880, 980)$	-3.46	5.46	-6.73
$[420, 520)$	-0.71	3.63	-6.29	$[890, 990)$	-3.09	6.55	-6.68
$[430, 530)$	-0.87	3.88	-6.35	$[900, 1000)$	-3.03	6.85	-6.71
$[440, 540)$	-1.00	3.37	-6.68	$[910, 1010)$	-3.65	4.99	-6.87
$[450, 550)$	0.29	4.53	-6.70	$[920, 1020)$	-3.66	4.67	-6.56
$[460, 560)$	0.77	4.62	-6.73	$[930, 1030)$	-3.49	4.47	-5.96

Table 1: Parameters. The highlight parts are distinct intervals where the kernels cluster with similar values.