Assignment 4: Gaussian Process

Xiyu Wang

xw5638 xiyu_wang@utexas.edu

Apr. 12, 2020

1 Introduction

Gaussian process provides a non-parametric representation of data generation process. In the process, the data sets are assumed to be sampled from a multivariate Gaussian distribution. According to the features of Gaussian distribution and Gaussian process, given the first and second moments the process can be fully defined. Thus, only the mean value and the covariance can define the behavior of a given Gaussian process. The kernel function specifies how the covariance of the process gets generated as the prior of the process. The kernel functions include hyperparameters which would get optimized via fitting with the given data.

In this report, a set of motion data corresponding to the position of a certain part of the body in a coordinate, over time. The data was collected by a set of sensors as the subjects traced curves. First, a simple global kernel would be fitted with data from one of the sensors. Then, a special sliding-window algorithm is applied to fit different kernels when comes to different part of the data. After that, the performance would be compared between the global kernel and the local kernels. At last, the parameters' value would be plotted and analyzed based on the performance of the local kernels.

2 Methods

In GPR, we first assume a Gaussian process prior, which can be specified using a mean function, m(x), and covariance function, k(x, x') as follows:

$$f(x) \sim GP(m(x), k(x, x'))$$

More specifically, a Gaussian process is like an infinite-dimensional multivariate Gaussian distribution, where any collection of the labels of the dataset are joint Gaussian distributed. Within this GP prior, we can incorporate prior knowledge about the space of functions through the selection of the mean and covariance functions. We can also easily incorporate independently, identically distributed Gaussian noise to the labels by summing the label distribution and noise distribution:

$$y \sim GP(m(x), k(x, x') + \delta_{ij}\sigma_n^2)$$

From the Gaussian process prior, the collection of training points and test points are joint multivariate Gaussian distributed:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

Where K is the covariance kernel matrix where its entries correspond to the covariance function evaluated at observations.

The key point of fitting a Gaussian process is to define a kernel function. In this experiment, the squared exponential kernel is used. Squared exponential kernel is also known as Gaussian kernel or RBF kernel with the expression as follows:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j))$$

Where the length parameter l controls the smoothness of the function and σ_f is the vertical variation. Also, a constant kernel is composed together with the RBF kernel who also have a σ_f. In scikit-learn, we can chose from a variety of kernels and specify the initial value and bounds on their hyperparameters.

One of the popular approaches to tune the hyperparameters of the kernel function is to maximize the log marginal likelihood of the training data:

$$\log P(y|f,\theta) = 0.5 f^T K^{-1} f - 0.5 \log|K| - constant$$

A gradient-based optimizer is typically used for efficiency.

$$\frac{d}{d\theta}\log P(y|f,\theta) = -0.5 trace\left(K^{-1}\frac{dK}{d\theta}\right) + 0.5 f^T K^{-1}\frac{dK}{d\theta}K^{-1} f$$

Because the log marginal likelihood is not necessarily convex, multiple restarts of the optimizer with different initialization is used.

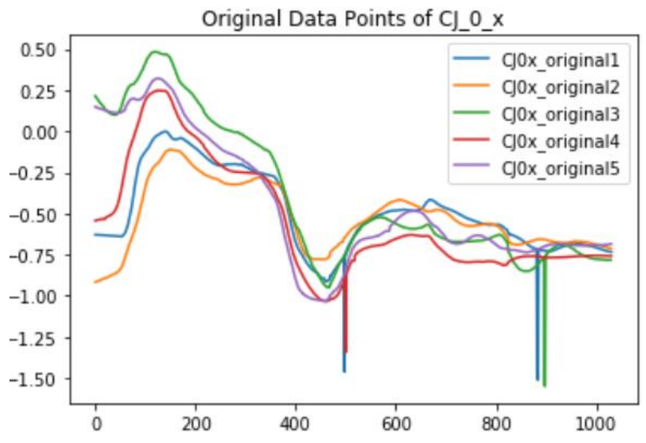
3 Results

The experiment is based on the dataset named “CJ” and do analyze based on the sensor 0 of all 50 using all five trails. For each sensor at each trail, there are 1030 data points within a time serie. With all three dimensions x, y and z, the experiment only consider x for example. Similar algorithm can be applied to y and z for sure. The programming process is made by python mainly with package GPR.

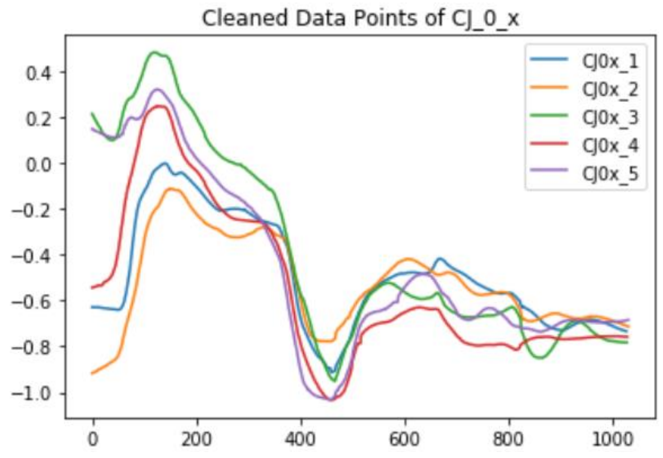
3 (1) Data Cleaning

The given data contians a parameter called “c” which indicates if the measurement is valid, by setting the value of c positive and negative. The point of data cleaning is to remove the invalid data point if they are not happening consecutively. Since the sampling of sensors are of high frequency, the missing of one data is possibily fine for the overall process. Another idea is to smooth the missing data with a linear or functional interpolation algorithm based on the nearest valid data points.

For plotting the data, do data cleaning for all the five trails of the target sensor. Before data cleaning, the data points distribute as follows:



We can find among the original data points there are several data points which are far away from the normal curve. Hopfully we can foresee those data points have the parameter c as negative.



Remove all the data points with neg ative c, the new data figures are shown:

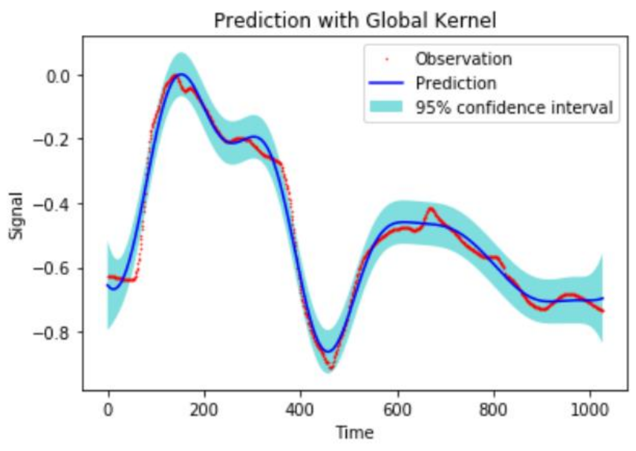
As we observed, the gaps are gone after the data cleaning. The rest of the data points looks similar among the five trails.

3 (2) Global Kernel Prediction

For the target sensor data set, apply one global kernel for Gaussian process regression.

In sklearn, define the kernel contains one ConstanKernel with initialized parameter of 1.0 and range with 0.1~1000, and one RBF kernel with parameter 10.0 and range with 0.1~1000. Train the Gaussian process regression with 10 times restart optimizers.

Fit the kernel with the data during the time series during time 0 to 1030. Generate the prediction data during the same time period and generate the parameters. Compare the prediction values and the real time observation. Also, plot the area of 95% confidence interval to show the sigma covariance value.



From the plot, it can be found that the prediction value is close to the observation, so the prediction accuracy is ok. The confidence interval is also close to the prediction value, so the confidence of the prediction is reliable.

After fitting the kernel, the parameters of the kernel is as shown:

	Length	Sigmal_f
Value of global kernel	93.8547	0.4836

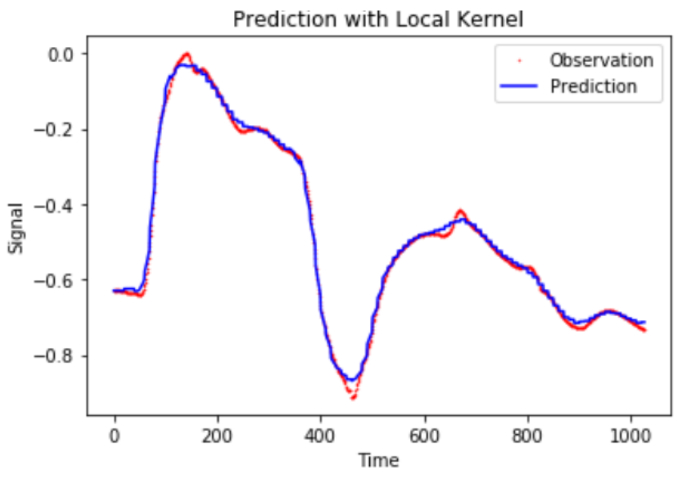
3 (3) Local Kernel Prediction

For the local kernel prediction, a special sliding window algorithm is adopted to separate the whole 1030 data into small sets with overlapping evenly. For each window, also called frame, one kernel will be fitted with its own parameters and make its own predictions. After sliding the window with a specific delta width of the data rows step by step, a series of kernels will be fitted, and predictions will be collected. After that, the final prediction result would be the mean value of the predictions from multiple kernels.

For this experiment, the window size is 50 and the delta width is 10. The kernels initialization is set the same with the global kernel example.

In the coding part, a while loop is set to do the sliding window until ran out of all the 1030 data. Within the loop, an array summarizes the predictions for each data observation time. Also, a counter array keeps counting the times the data get predicted for later use of getting the mean value. The length and sigma parameters are also recorded for further use.

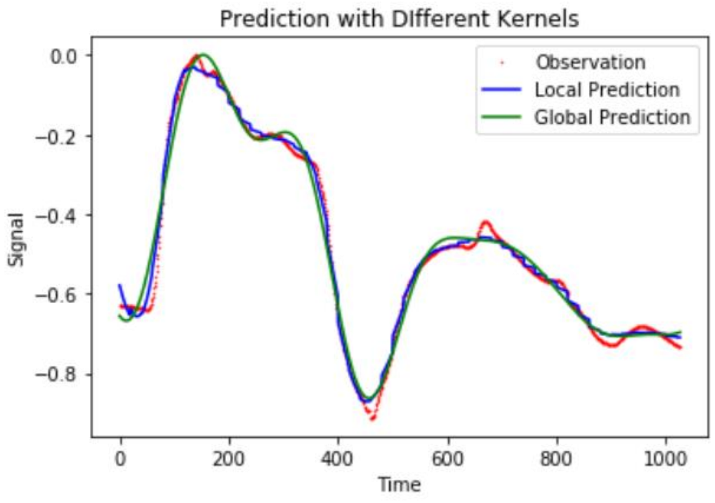
After the local kernel prediction, plot and compare the value of observation and the prediction as follows:



It’s shown that the prediction is quite close to the observation. Generally, compare with the global prediction, the accuracy looks higher in this case. In next part more comparison will be applied to the two algorithms.

3 (4) Comparison and Accuracy Evaluation

For comparison, plot the observation and the predictions in both global kernel and the local kernels at the same time:



In the smooth part, both kernels work great. In the curving part, the local prediction in blue curve matches the red dots observations better than the global prediction in green curve.

Numerically, the mean squared error (MSE) can represent the accuracy of the prediction:

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

The MSE for the two predictions are shown in the table:

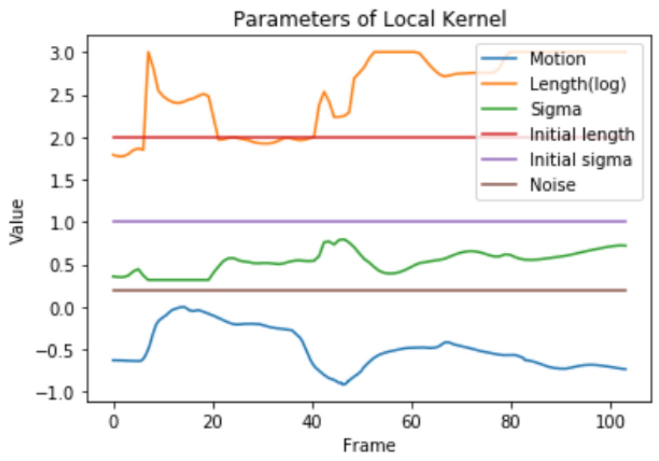
	Global Kernel	Local Kernels
MSE	1.0722e-3	0.16031e-3

Since the MSE of the local kernels is much smaller of the one of global kernel. The numerical result of comparison is the same with the plotting general comparison. The global kernel prediction is less accurate of the one from local kernels.

3 (5) Local Kernel Parameters

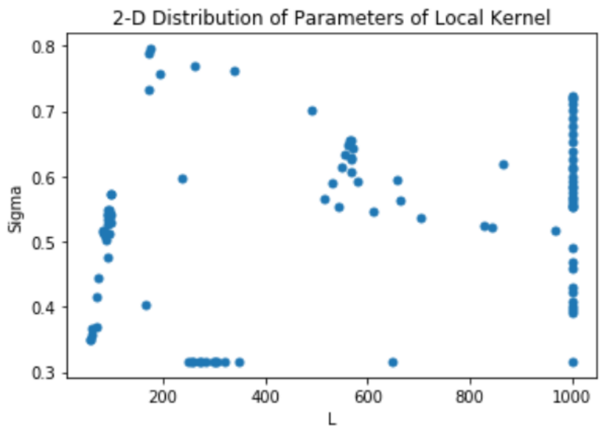
While observing the parameters, I notice that the length scale value (sigma_l) can be large when the training data points are not enough for this case. For instance, when the window size is smaller than 50, the parameter length can be around 100,000 and reach the scale boundary. But when the window size enlarged, the problem is getting released and disappears soon. In the case of window_size = 200, the length looks normal until the window reaches the end of the data set and the rest data points are not enough to fill one window.

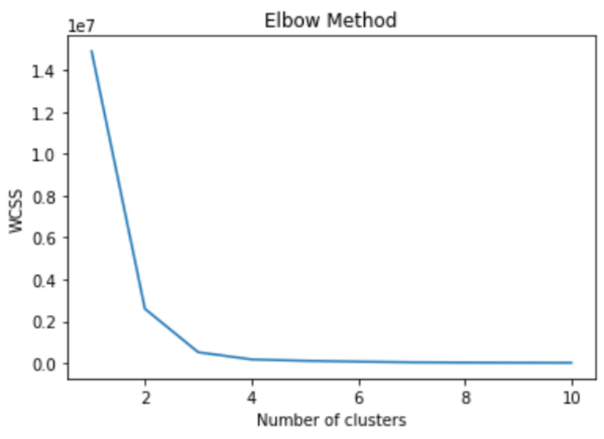
Comparing the initial value of length and sigma with the parameters of the local kernels during sliding, plot all the data as well as the motion as follows when window size is 200:



3 (6) Kernel Cluster

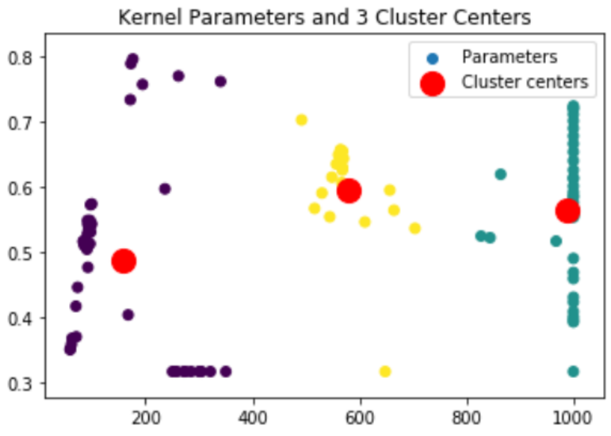
To observe the cluster of the kernels, first map the hyperparameters into a 2-D space and see the distribution.

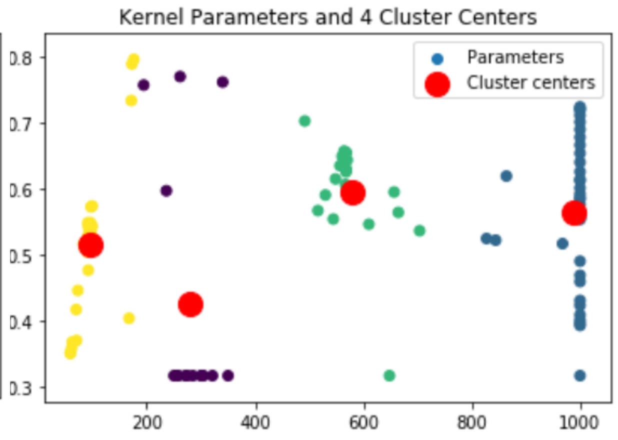




Adopt K-mean clustering algorithm to the kernels. Use the Elbow Method to determine the number of clusters. According to elbow method, 3 or 4 clusters would be enough for our kernel clustering.

For 3 and 4 clusters, the cluster centers are shown:





Observe the data and the centers. The three clusters model would be good enough and make sense according to the plot.

# of Clusters	C1	C2	C3	C4
3	38	19	46	~
4	38	19	31	15

As shown in the table, 4 clusters model divides one of the clusters of the 3 clusters model. In the plot figures, the length is making the leading role in deciding which cluster the kernel belongs to. For very small length and large length cases, the sigma values vary frequently but for middle size of length, the sigma values are quite in the same range around 0.5 ~ 0.7.

4 Summary

In this experiment, Gaussian process regression is adopted to modeling the time-varying x-coordinate of a sensor located on a subject's head during a tracing experiment. After regression fitting with data, the optimal hyperparameters of the kernel were determined and were used to predict the data by maximizing the probability of the points given the distribution fit to the observation data points. The traces produced by the sensor were shown to be quite similar across trails. For the target sensor case, the observed behavior trends change with time, and thus the time-varying hyperparameters realized by sliding window algorithm can improve the model fit compared with using the same kernel model for the global data.

The parameters of local kernels keep changing while the window sliding. The length scale keeps fluctuating while sigma varies mostly within 0~1. If we apply clustering algorithm to the kernels based on their hyperparameters, generally there would be three clusters. It also indicated that during fitting of the Gaussian process regression, there are some patterns when comes into different types of data.