Gaussian Process Analysis report

DIYUAN DAI, EID: DD33653,

Department of Computer Science, The University of Texas at Austin

1 1. INTRODUCTION

The Gaussian process (GP) is a powerful model that can be used to represent the distribution of functions. At present, the common practice of machine learning is to parameterize the function and then use the generated parameter modeling to avoid the distribution representation (such as the weight of linear regression). But GP is different, and it directly generates non-parametric models for function modeling. An outstanding advantage of this is that it can simulate any black-box function and uncertainty.

This assignment is to find best sets of parameters by a sliding window method of Gaussian processes to predict the positions of markers on a subject tracing a curve.

2 METHOD

2.1 Basic Data Processing

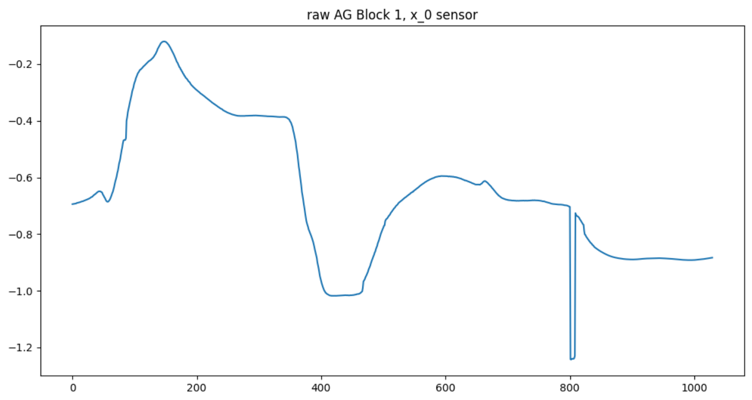
- Firstly, a set of data (each length of 1030 frames) from sensor samples is chosen and transformed for later use. To reduce the complexity, I personally chose the set AG, sensor 8, and ignored other dimension of data except for x coordinates.
- I used pandas package to do load data into a numpy array.

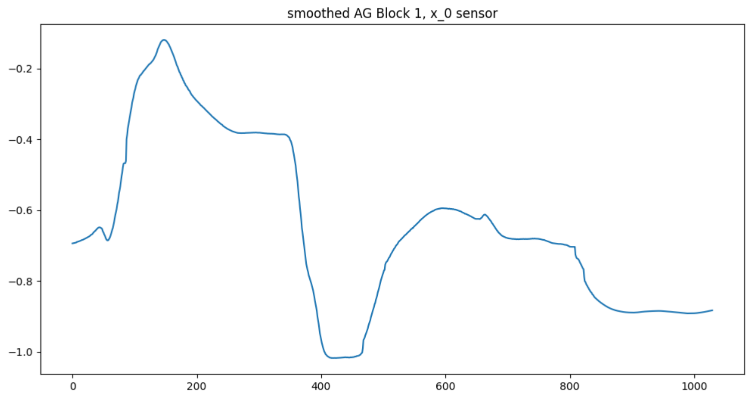
2.2 Data cleanse

- According to the instructions provided, the data contains an additional variable, C - labels, which indicate whether the input signal was good or not.

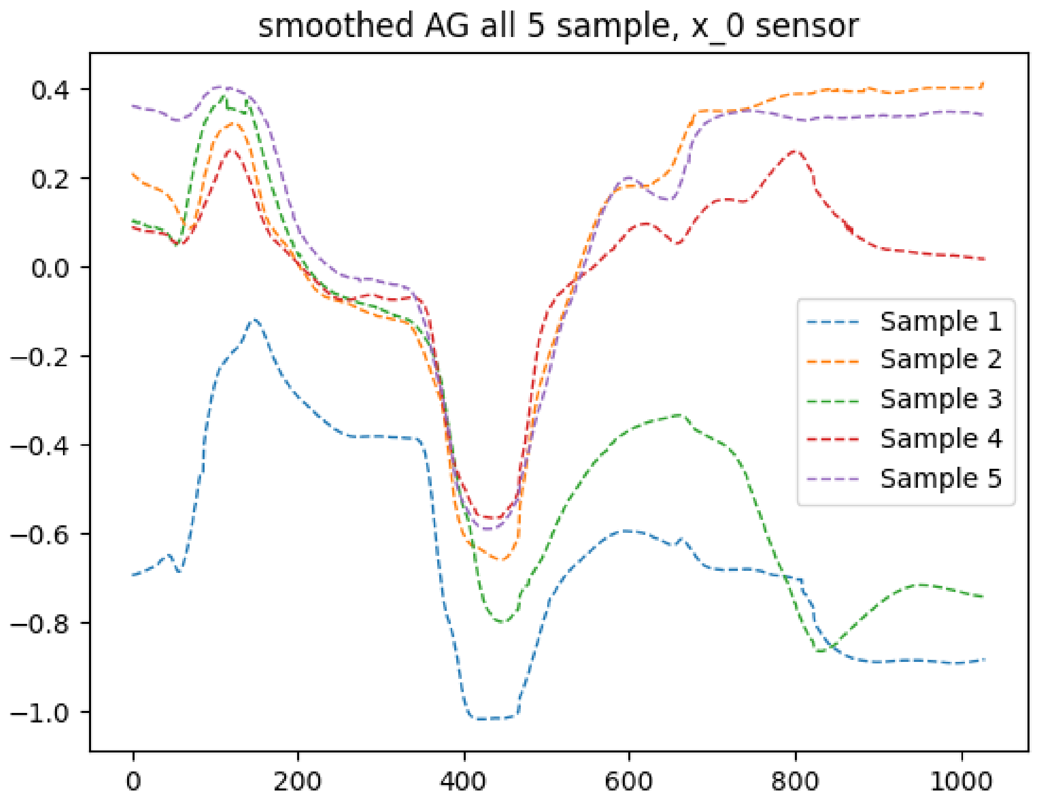
- Iterate through all the data and check the validity. If C value is negative, the data will be dropped and smoothed with a function:

$$data[i] = data[i-1] + 0.5 * (data[i-1] - data[i-2])$$





The data will be ready to use after this process.



2.3 Gaussian Process

- We start with a Gaussian with Mean and covariance It turns that a general covariance matrix will have a Cholesky decomposition

$$E(z) = \mu$$

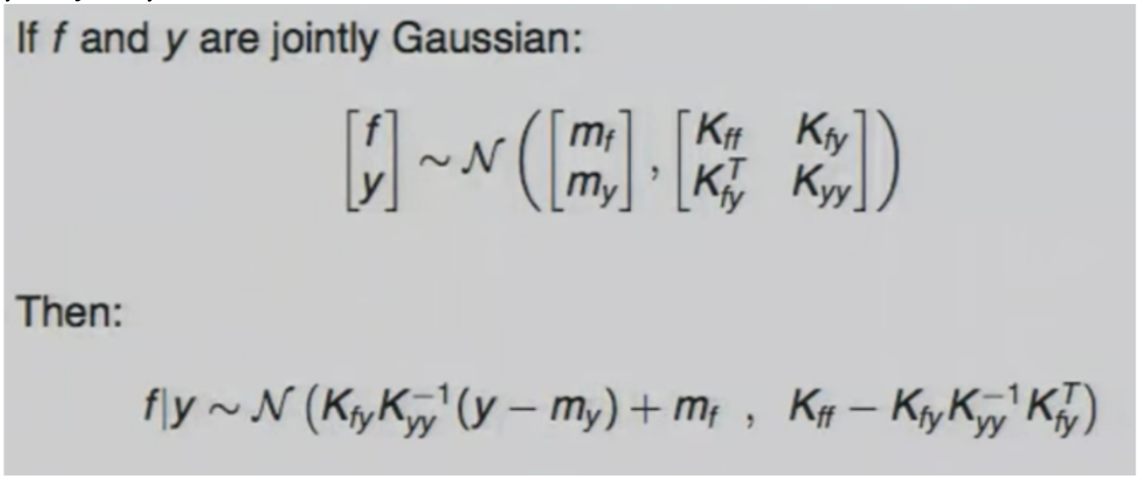
$$\Sigma = LL^T$$

- If z has zero mean and I is the covariance then any

$$y = \mu + L_z$$

  will have mean μ and covariance Σ

- Gaussian Process Regression (GPR) takes in data from multivariate Gaussian. If f and y are jointly Gaussian:



If f and y are jointly Gaussian:

$$\begin{bmatrix} f \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_f \\ m_y \end{bmatrix}, \begin{bmatrix} K_{ff} & K_{fy} \\ K_{fy}^T & K_{yy} \end{bmatrix}\right)$$

Then:

$$f|y \sim \mathcal{N}\left(K_{fy}K_{yy}^{-1}(y - m_y) + m_f \ , \ K_{ff} - K_{fy}K_{yy}^{-1}K_{fy}^T\right)$$

  So we can choose some kernel with hyper-parameters.

- For learning some parameters, the conditional probability formula is:

$$P(\mathbf{f}|\mathbf{X}) = (2\pi)^{\frac{n}{2}} |\mathbf{K}|^{\frac{1}{2}} \exp(-\frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f})$$

- Another feature that Gaussian Process has is to simulate noise, we can include some noise and then we have :

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I)$$

$$\log p(\mathbf{y}|X) = -\tfrac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} - \tfrac{1}{2}\log|K + \sigma_n^2 I| - \tfrac{n}{2}\log 2\pi$$

- Set up a gradient descent methods to optimize hyper parameters and use exp(parameter) to make sure they are positive:

$$\log P(\mathbf{t}|\mathbf{x}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{t}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{t} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{N}{2}\log 2\pi$$

$$\frac{\partial k}{\partial \sigma_f} = k'$$

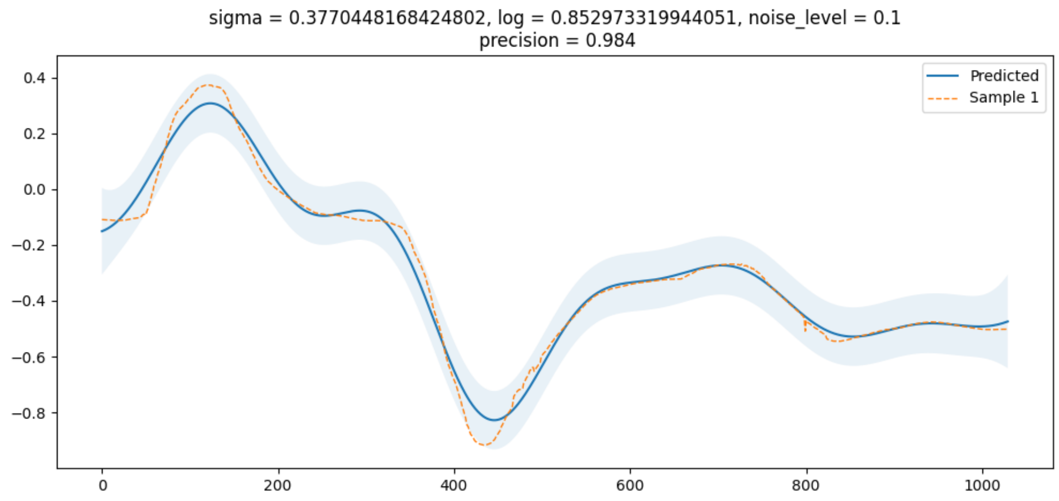
$$\frac{\partial k}{\partial \sigma l} = k' \times \left(-\frac{1}{2}\exp(\sigma_l)|\mathbf{x} - \mathbf{x}'|^2\right)$$

$$\frac{\partial k}{\partial \sigma_n} = \exp(\sigma_n)\mathbf{I}$$

- Here, I borrowed the learning function from scipy package. The learning process can be done by simply calling GaussianProcessRegression function. The learning function will output the optimal parameter after learning automatically. The last step will be set up initial parameters.

3 RESULTS

3.1 prediction of motion data with global kernel

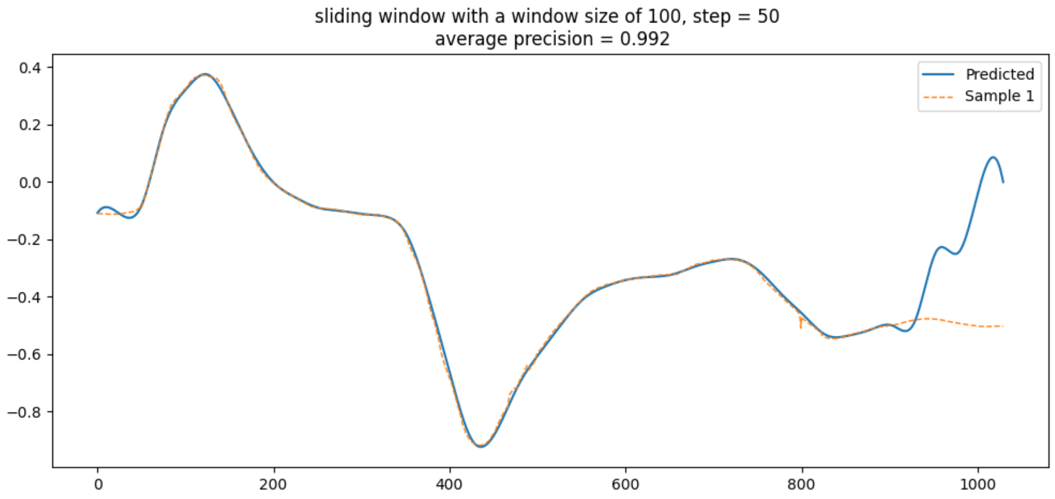


This figure shows the predicted signals, We can obviously find that the model can predict recognizable signals. This result is given out under initial parameters with sigma, log, noise = [0.5, 1.0, 0.01]. global kernel converge at 0.377044563894303 85.29727281700298, with a precision of 0.9839, in AG Block3 8_x

3.2 prediction of motion data with local kernel

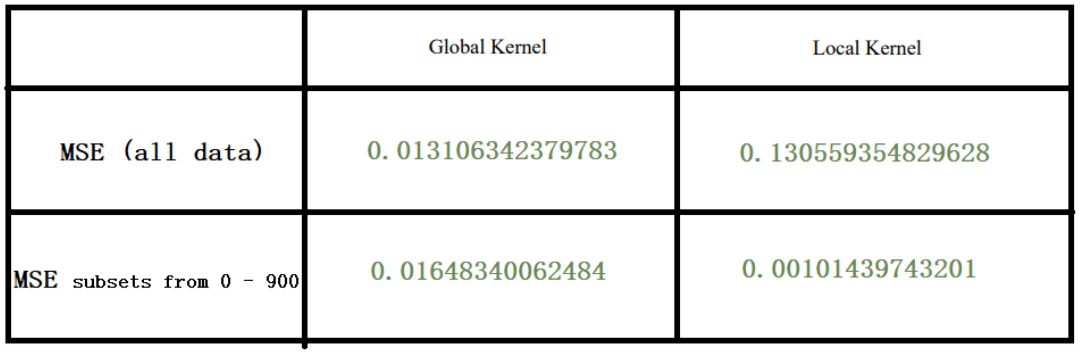
In this session, we choose same source to conduct the experiment. A sliding window method is implemented, which works as follows:
Choose an window length and starting from the initial point for one of the coordinates, determine the kernel parameters using the gradient descent algorithm. Next advance the start coordinate by a delta and refit the kernel parameters. As long as the parameters are stable, continue, but if they change determine a new start point at the stage and start the

setting process. It took much longer time to obtain a convergence.



3.3 Comparison of performances, global kernel vs. local kernel

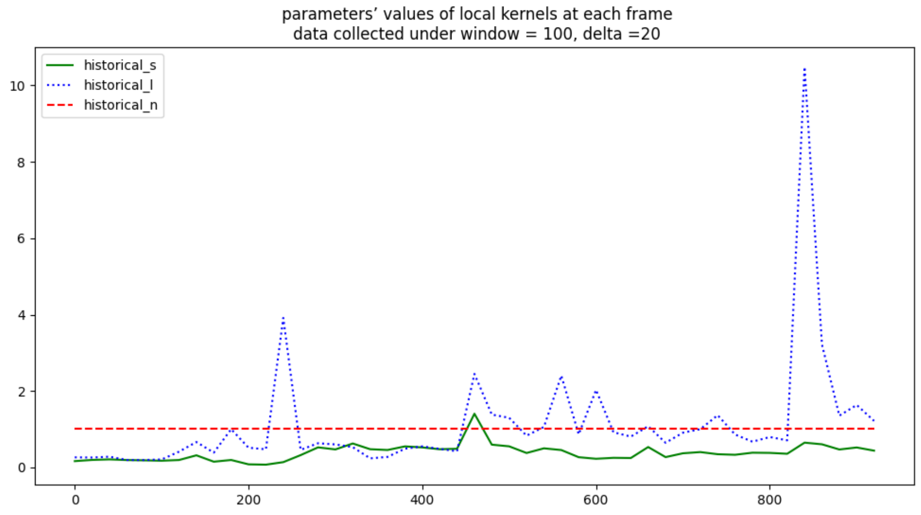
From two figures above, we can see that local kernel performs slightly better than global kernel, except for the last proportion of the data. It is hypothesised that the bad performance at the end is due to the design of sliding window encountering the edges where the data might be biased. Here, I borrowed MeanSquaredError from sklearn to have a more straightforward indication for precision of two kernels. I performed the calculation once taking all performance and once dropping out the outliers (900-1200), we can see local kernel outplays global kernel.



	Global Kernel	Local Kernel
MSE (all data)	0. 013106342379783	0. 130559354829628
MSE subsets from 0 - 900	0. 01648340062484	0. 00101439743201

3.4 parameters' values of local kernels at each frame

In the experiment, I record the changes of parameters while learning local kernels:



In this experiment with data in AG Block3 $8_x, thelength, intermsof \log_n$ is changing dramatically on the subset 800 to 900

We can also see that these subset of frames have the similar parameters:

$$0 - 150; 250 - 450; 700 - 800$$

4 SUMMARY

The quantification of uncertainty is essential. For example, when we can request more data, relying on the Gaussian process, we can explore the most unlikely data areas for efficient training.

To obtain a better training effect, we often need to do more adjustment calculations in the implementation. You may have noticed that GP contains two fundamental parameters: , and l. If you try to change them when collecting samples before, you will find magical changes in the vertical and horizontal directions of the image. For example, if we expect a more comprehensive output range, we need to enlarge the parameter accordingly. In fact, as with all methods that use the kernel function, we can even change the kernel function entirely if necessary.