CS391L Machine Learning HW4: Gaussian Process

Zeyuan Hu, iamzeyuanhu@utexas.edu

EID:zh4378 Fall 2017

1 Introduction

In this task, we study the human movement position data captured by the motion capture system. We want to measure the stiffness of the movement, which happens when humans co-contract opposing muscles to move. One hypothesis is that if a joint controller is stiff, it will produce reliable repeated movements, whereas if it is very loose, there will be more variation in repeated movements. We use the Gaussian Process (GP) to test this hypothesis and we find out that GP in general is a good model to measure the human stiffness and thus we are able to confirm the hypothesis.

This writeup is organized in the following way: In the first section, I will describe the steps to finish the task. Next, I will talk about the experimentation result. Lastly, I will give a breif summary of the whole work.

2 Method

2.1 Gaussian Process

In GP, we put a probability distribution over the space of functions, which can be specified by the mean and covariance functions. Concretely, GP is described as a function $G(k(\boldsymbol{x}, \boldsymbol{x'}))$ that models some underlying functin $f(\boldsymbol{x})$. The covariance function $k(\boldsymbol{x}, \boldsymbol{x'})$ gives us the expected covariance matrix between the values of f at $\boldsymbol{x}$ and $\boldsymbol{x'}$. In this task, we use the squared exponential covariance matrix (also called Gaussian kernel or RBF kernel).

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \exp(\sigma_f) \exp(-\frac{1}{2} \exp(\sigma_l) \left|\boldsymbol{x} - \boldsymbol{x'}\right|^2) + \exp(\sigma_n)\boldsymbol{I}$$

where the signal variance σ_f^2 controls the overall variance of the function, and the length scale σ_l changes the degree of smoothing, trading it off against how well the curve matches

the training data. σ_n expresses how much noise is expected in the data points. Those are the hyperparameters that we will explore in detials in the experiment section.

To apply GP to the gression task, we perform the following steps: we compute the covariance matrix of the training data, and also the covariances between the training and test data, and the test data alone. Then we compute the mean and covariance of the posterior distribution and sample from it. The detailed steps is the following:

For given training data $(\boldsymbol{X}, \boldsymbol{t})$, where $\boldsymbol{x}$ is the vector of training time points and $\boldsymbol{t}$ is the vector of corresponding training function values at the respective time points. We have test data $\boldsymbol{x^*}$, covariance function $k()$, and hyperparameters $\boldsymbol{\theta} = (\sigma_f^2, \sigma_l^2, \sigma_n^2)$:

- Compute the covariance matrix $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n \boldsymbol{I}$
- Compute the covariance matrix $\boldsymbol{k^*} = k(\boldsymbol{X}, \boldsymbol{x^*})$
- Compute the covariance matrix $k^{**} = k(\boldsymbol{x^*}, \boldsymbol{x^*})$
- The mean of the process is $\boldsymbol{k^{*T}} \boldsymbol{K}^{-1} \boldsymbol{t}$
- The covariance is $k^{**} - \boldsymbol{k^{*T}} \boldsymbol{K}^{-1} \boldsymbol{k^*}$

The new point x_i^* will be interpolated with a gussian with mean m_i and variance $\text{Cov}_{i,i}$. Thus, if we want to build a 95% confidence interval over the interpolated m_i value for x_i^*, we will have an upper bound of $m_i + 1.96\sqrt{(\text{Cov}_{i,i}}$ and a lower bound of $m_i - 1.96\sqrt{\text{Cov}_{i,i}}$, where 1.96 is the z-score for 95% confidence.

As we will see in the experiment section, hyperparameters $\boldsymbol{\theta} = (\sigma_f^2, \sigma_l^2, \sigma_n^2)$ will have a significant effect on the shape of the resulting output curve. Thus, finding the correct hyperparameters is very important. One way is to choose the hyperparameters that maximize the log probability of seeing the data, which is specified by

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

We can use the gradient descent method to minimize this log likelihood. The only thing we need is the gradient. We let $\boldsymbol{Q} = (\boldsymbol{K} + \sigma_n^2\boldsymbol{I})$ and since $\boldsymbol{Q}$ is a function of all of the hyperparameters $\boldsymbol{\theta}_i$. Then the gradient is computed as

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log P(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{t}^T\boldsymbol{Q}^{-1}\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{\theta}}\boldsymbol{Q}^{-1}\boldsymbol{t} - \frac{1}{2}\text{trace}(\boldsymbol{Q}^{-1}\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{\theta}})$$

Since we are using the squared exponetial covariance function, then $\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{\theta}}$ is equal to

$$\begin{aligned}
\frac{\partial k}{\sigma_f} &= k' \\
\frac{\partial k}{\sigma_l} &= k' \times (-\frac{1}{2}\exp(\sigma_l)\left|\boldsymbol{x}-\boldsymbol{x'}\right|^2) \\
\frac{\partial k}{\sigma_n} &= \exp(\sigma_n)\boldsymbol{I}
\end{aligned}$$

where $k(x, x') = k' + \exp(\sigma_n)\boldsymbol{I}$.

2.2 Work with the data

In this task, we have 12 subjects' motion capture data and thus there are 12 directories (i.e., AG, CJ, CT, etc.). Each directory contains five subdirectories, which means that each subject traced a curve 5 times. In each subdirectory, there is a csv file. Totally there are 50 markers (sensors), and each marker has position (x,y,z). Thus, the marker positions are presented as (markerIndex_x, markerIndex_y, markerIndex_z). For example the position of marker 6 is (5_x, 5_y, 5_z). There are C-labeled data associated with each marker indicating whether the marker position is valid or not. If the motion capture system cannot get the marker positions, then the corresponding C-label value is negative. For example, if the (5_x, 5_y, 5_z, 5_c) is (1, 1.2, 1.3, 3), then the recording position is useful. However, if the tuple is (1, 1.2, 1.3, -1), then the recording position is useless.

2.3 Implementation

We utilize the scikit-learn package to perform the gaussian process hyperparameters learning and regression [1]. The whole implementation can be break down into two parts: the data processing and the GP. For the data processing part, we use `generate_observed_data_list` to read in the csv files under the `'data_GP/'` directory. Here we use `one_subject` variable to control whether we want to read all the csv files associated with one specific subject or we want to read in all the csv files. Once we read in the csv files, we invoke `read_in_data` function to process the csv files by formating all the x-coordinate values, y-coordinate values, and z-coordinate values into matrices with dimension 50×1030. Then we check the C-label

value to remove invalid data entries and append the result values as a list into `x_clean`, `y_clean`, and `z_clean` correspondingly. The dimension of all those three variables are $50 \times$ number of timeframe with valid data.

For the GP part, the key is to construct the kernel with the hyperparameters we discuss previously and whether we fix our hyperparameters or not. We use `ConstantKernel` (i.e., ConstantKernel API[1]) with parameter `constant_value` to control σ_f. We use `RBF` (i.e., RBF API[2]) with `length_scale` to control σ_l. Lastly, we `WhiteKernel` (i.e., WhiteKernel API[3]) to control σ_n.

3 Results

We first plot the GP model fitting one marker of one trail. We pick x_0 marker in the first movement data of the subject "AG". We make a comparison between fitting one marker of one trail with fixed hyperparameters $\sigma_f = 1.0$, $\sigma_l = 10$, and $\sigma_n = 10^{-2}$ and the fitting with the learned hyperparameters. In this case, the hyperparameters are optimized as $\sigma_f = 0.487^2$, $\sigma_l = 41$, and $\sigma_n = 10^{-10}$. Figure 1 and Figure 2 show the comparison result.

Next, we try to vary the hyperparameters across the data traces. Our hypothesis is that if a joint controller is stiff, it will produce reliable repeated movements, whereas if it is very loose, there will be more variation in repeated movements. Table 1 summarizes the optimized hyperparameters for the selected subjects. Figure 4 and Figure 5 plot all five repeated movements data for subject "AG" and subject "YY". I randomly select one movement data of all five momement data of a subject to fit the gaussian process and see if the resulting model can describe all five movement data relative well.

It can be seen that the variation across all repeated movements for both "AG" and for "YY" are relative small. This indicates that joint controller for both "AG" and "YY" is stiff.

Lastly, I fit the trails over all the subjects. The result is shown in Figure 5.

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.ConstantKernel.html`

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RBF.html`

[3] `http://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.WhiteKernel.html`

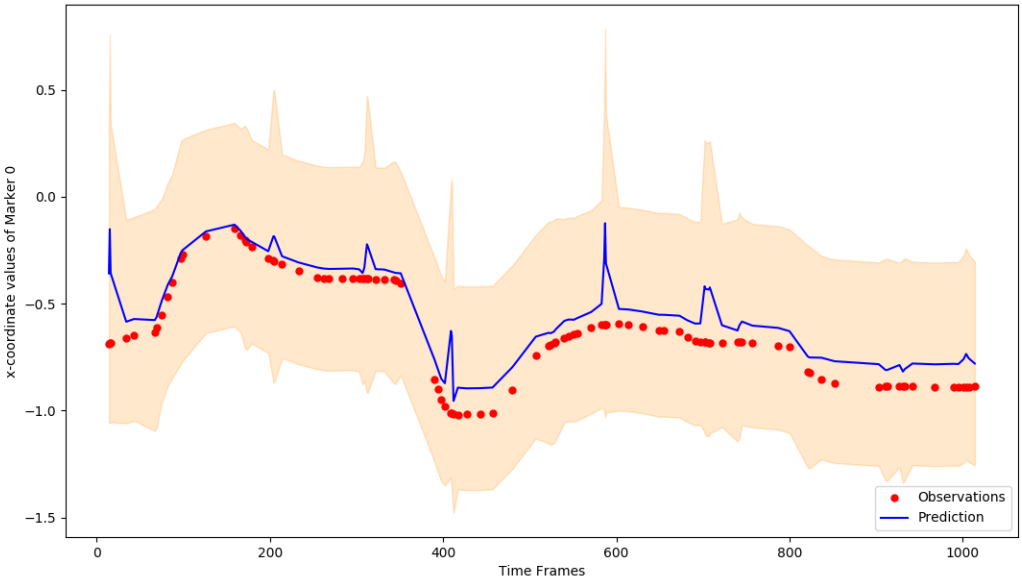

Figure 1: Initial, bad hyperparameters

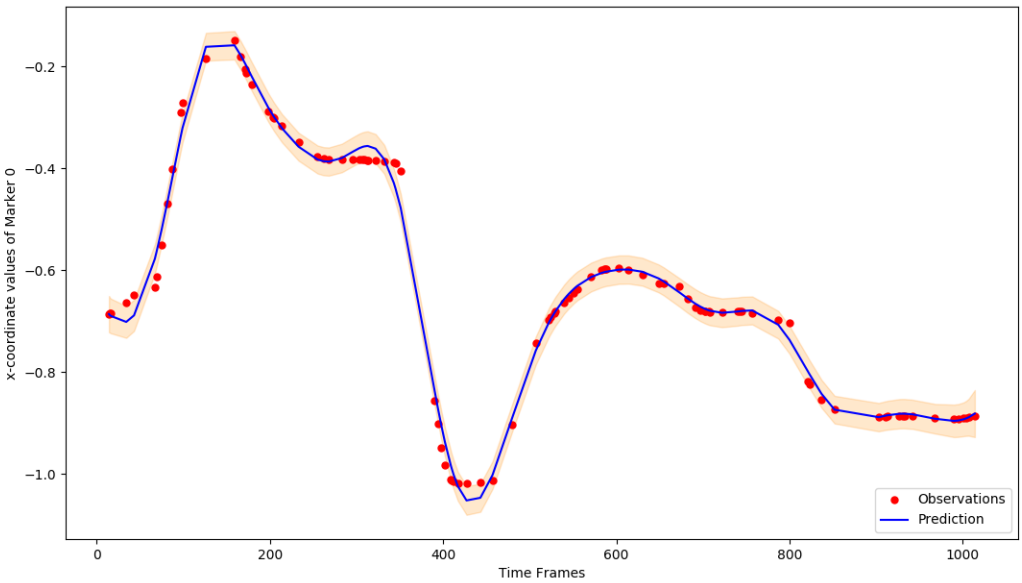

Figure 2: Initial, bad hyperparameters

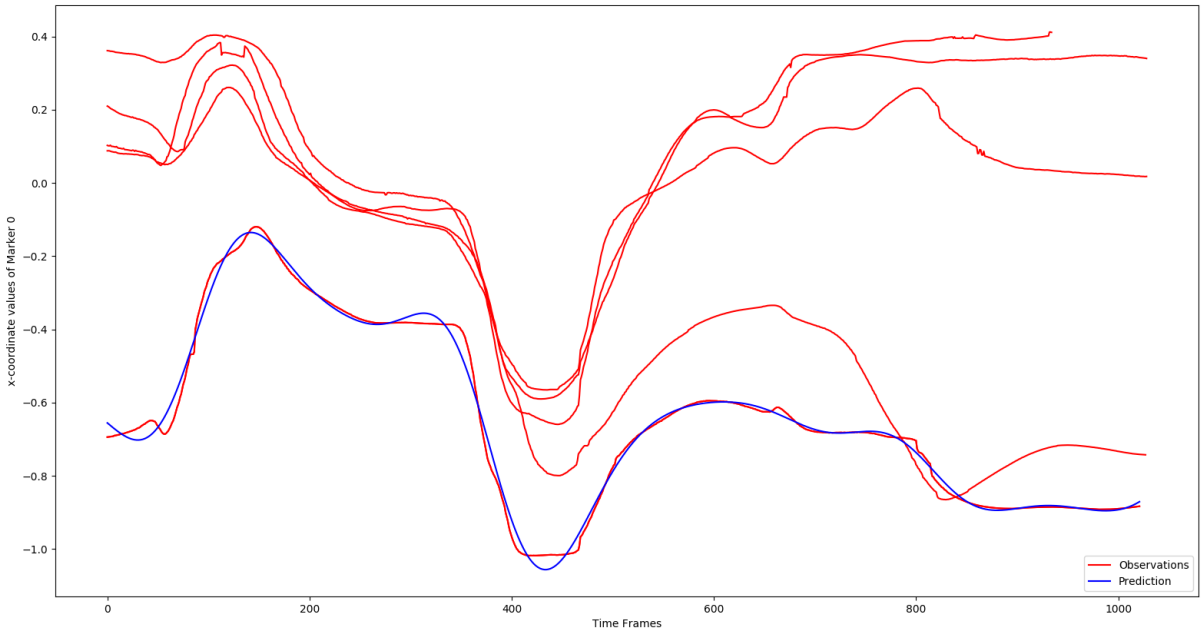

Figure 3: Plot of movement data of subject "AG"

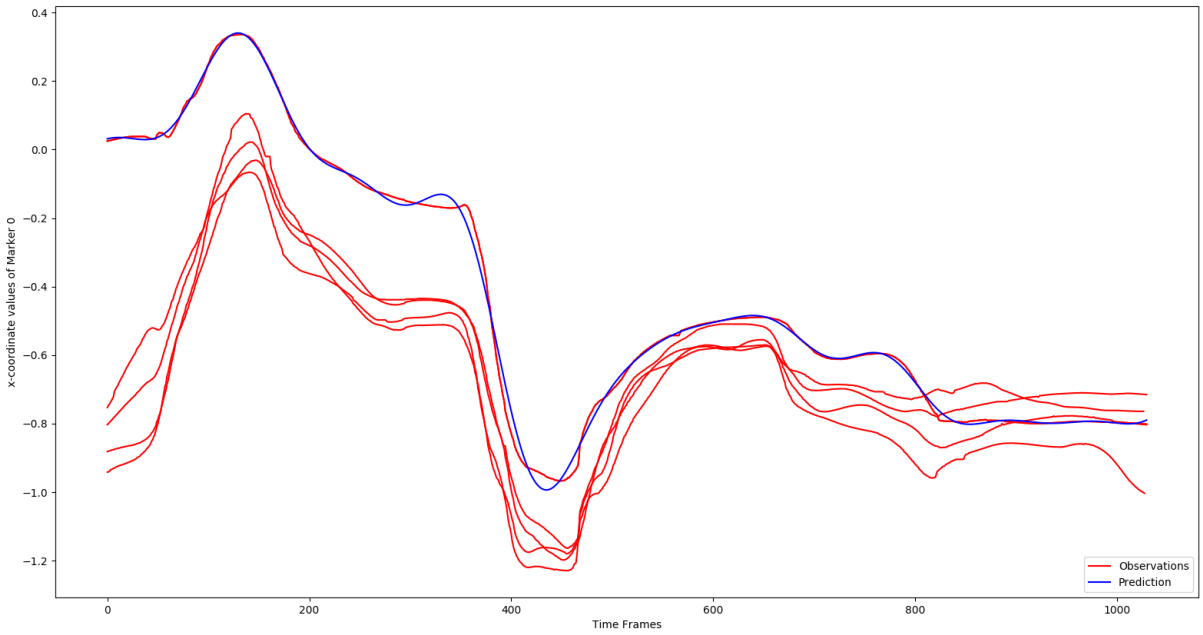

Figure 4: Plot of movement data of subject "YY"

Table 1: Optimized hyperparameters for selected subjects

Subject	σ_f	σ_l	σ_n
YY	0.404^2	36.7	10^{-10}
AG	0.487^2	41	10^{-10}
CJ	0.393^2	40.1	1.15×10^{-10}
CT	0.206^2	20.1	10^{-10}
EK	0.257^2	62	1.6×10^{-10}

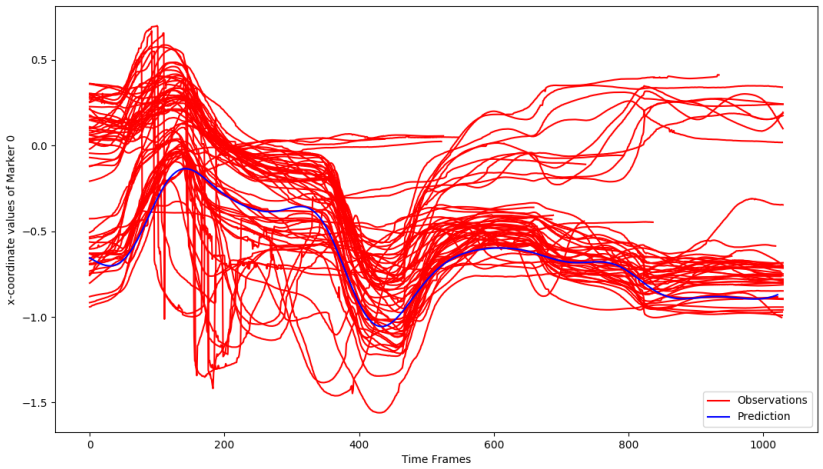

Figure 5: use the GP fitting trails over subjects on Marker 0_x

4 Summary

In this task, I apply the gaussian process regression to the human movement position data and find out that the joint controller for the subjects are stiff. In addition, I tune the hyperparameters in order to get the good fit of the model to the data.

Appendices

A How to run the code

To run my code, unzip the `hw4.zip` and put the directory named `data_GP` that contains data files under the same directory with `gp.py`. Then, you can run the code with the following commands:

- `python gp.py S` Run the gaussian process regression on a single trail (i.e., x_0)
- `python gp.py M` Run the gaussian prcoess regression on all the trails of all the subjects
- `python gp.py SN` Run the gaussian process regression on a single trail (i.e., x_0) and fix the hyperparameters
- `python gp.py SUB` Run the gaussian process regression on all trails of a single subject

My code is heavily commented and please take a look if there are any type of questions.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.