Vivek k Khetan
UT-EID: vkk287

Gaussian Process

Assignment:5
Machine Learning

Introduction:

There are many popular approaches to generalized or to find the distribution of unseen data. One common approach to find the distribution of any hidden data is to use a supervised learning and then to tune the hyper-parameters of our assumed model so that it minimizes the error on the data. Instead of using these popular supervised methods, we can also assume that the data is approximately Gaussian distributed. The major benefits of this approach are that we only have two parameters to estimate, i.e., mean and median and we can assume any random distribution to be summation and product of many other Gaussian distributions.

Finding the combines distribution for lot of joint random distribution is very complicated but if we assume that all the individual random distribution is Gaussian then, their combined joint distribution will be a Gaussian Process and it will be such easier to estimate [1].

Problem:

For this assignment, we were provided sensor data of 12 different individuals tracing the curve in the space for 5 different times. For each user, for their each curve trace, 50 different sensors were capturing their motion data. We were also provided with data of the movement of curve in the space as well as user's finger and head. We also had data to check whether the marker's position is valid or not. The C-labeled data, next to each sensor data in each excel file means whether the marker position is valid or not. If the motion capture system was not able to get the marker positions, then the corresponding C label was given a negative value. In the excel files, the vertical frame is time. For each of 60 trace, the data was captured for around 20 seconds with a frequency of 60 Hz. There are also some missing data, it is due to sensor device. Sometimes the motion capture device was missing data and it was indicated with a blank space.

The goal of this assignment is to fit a Gaussian Process learning algorithm and to come with an optimized hyper parameter estimates for the fitted Gaussian Distribution that maximizes the posterior probability for the given data.

Method:

First step was to make sense of data, the data provided was very dense and varied. Therefore, I have to spent a lot of time analyzing and making sense of data. I made some plots to visualize the data and understand it using equations given in the class. Then, I implemented Gaussian process and obtained the optimal value of hyper-parameters.

Following are the steps I took for obtaining the optimum value of hyperparameters:

1. Get the data (x, xstar, y, y, star): I used data from 2 different csv files
2. Define the kernel function: I used square exponential function to define my kernel function
3. Compute k(x,x), k(x, xstar), k(xstar, xstar): using my defined kernel function I computed covariance for training data, test data as well as covariance of their interaction.
4. Compute Q: added random initial noise to the computer training covariance
5. Used numpy library to do cholesky decomposition of Q and solve linear matrix equation to obtain alpha
6. Using the output of cholesky decomposing, I also calculated predictive mean as well as predictive variance of the test data
7. Then I wrote a function to obtain negative log posterior probability
8. Then, another function to compute gradient of negative log posterior probability
9. I used Scipy.optimize.fmin_cg function to converge my negative log posterior and obtain the optimize value of hyper parameters. scipy.optimize.fmin_cg takes log-posterior probability, gradient of log posterior, hyper parameters as its input arguments and gives optimize hyper parameters as output.

These steps are also advised by Stephen Marsland in his Book[1].

input: X (inputs), $\mathbf{y}$ (targets), k (covariance function), σ_n^2 (noise level), $\mathbf{x}_*$ (test input)

$L := \text{cholesky}(K + \sigma_n^2 I)$

$\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$

$\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$ } predictive mean eq. (2.25)

$\mathbf{v} := L \backslash \mathbf{k}_*$

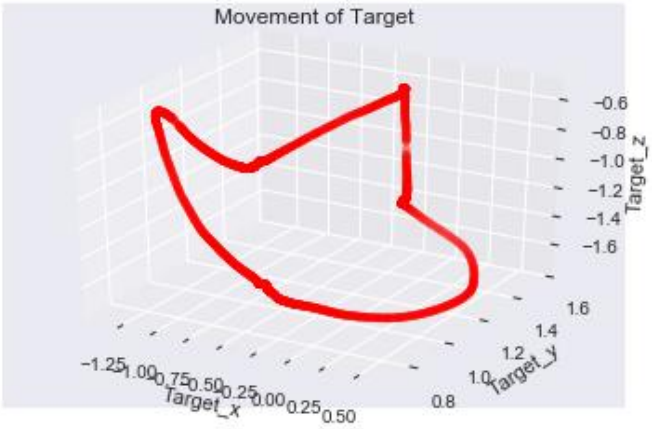
$\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$ } predictive variance eq. (2.26)

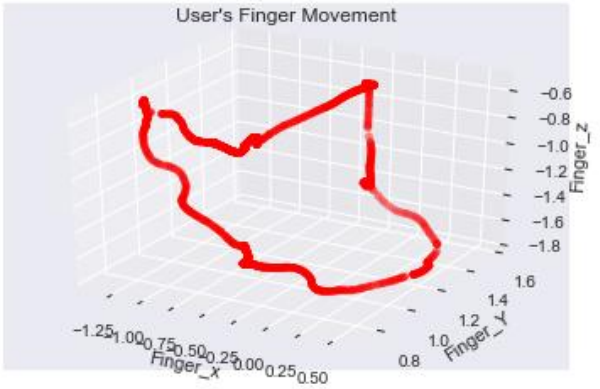
$\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi$ eq. (2.30)

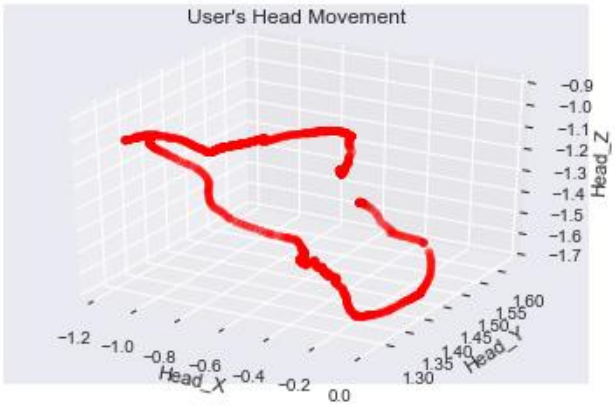
return: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)

Results:

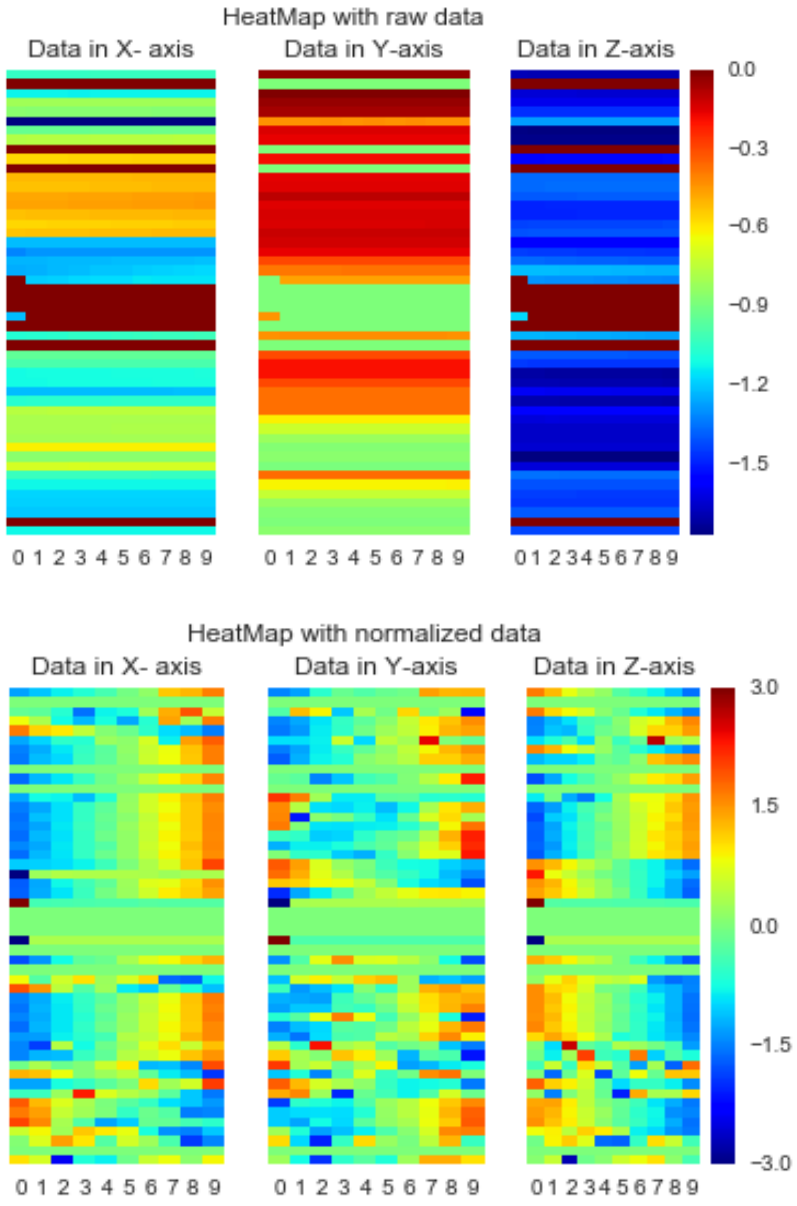
Visualizing and understanding the data was one of the most important and interesting part of this assignment. I made some 3-D plots to visualize the curve movement as well as the person's fingers and head movement in the space. I also created heatmaps for the provided sensor data.







It is interesting to see how user's finger are almost imitating the curve movement but head isn't. It seems reasonable as anyone will try to follow the curve with finger and observe it with head's movement.

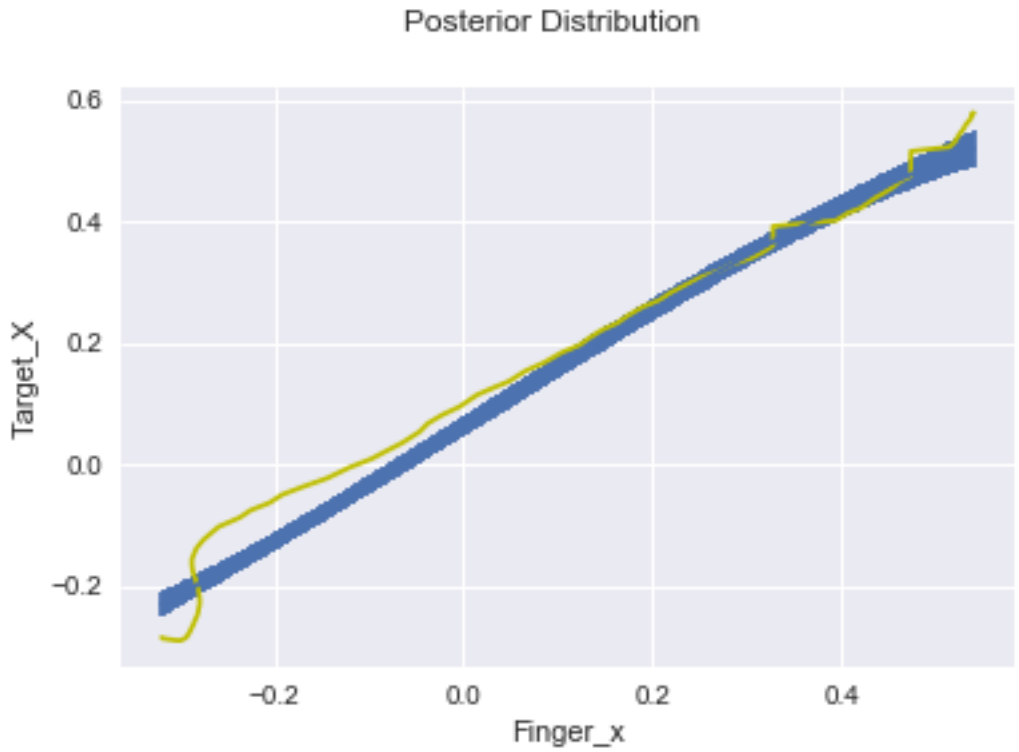


Above are the heat-map for the one trace of a single user. I plotted one heat map with the provided raw data, another with normalized data. I used sklearn.preprocessing.scale function to normalize the data and see the relationship between various sensors for a single trace. I did the heatmap plotting separately for x, y, z axis. It is interesting to see how the heatmaps are different in x,y and z axis for raw data but almost similar in normalized data. It is corroborated by the fact that whatever positions user has in x,y and z axis, its relative change in position for two sensors should be same .

While doing cholesky decomposition, I have some error like, " Matrix is not positive SemiDefinite". I was not able to understand the reason but use of np.squeeze for theta solved it for me.

For hyper parameter tuning, I tried my code with various combination of data points and starting value of theta (σf, l, σn). For 120 data points (corresponding to 2 second of data), and initial theta of [0.1,0.1,0.05]; I was getting an optimize theta = [0.098, 0.978, 0.05]. It was same scenario with various other combinations too. I therefore think the optimal value of hyper parameter for the user1 should be around this value only.

I have also plotted posterior distribution with optimized hyper parameters, predictive mean and predictive variance.



Discussion and Conclusions:

This assignment has many challenges, first one was dense size and scale and data. I had to visualize dataset to get the idea about what each column represents and how to use it in my analysis. Another one was to carefully slicing them to make 3-D plot of the trace, user's finger, user's head and heatmap for sensors data in all there axis(x,y,z). Writing the logPosterier function, gradlogPosterier function, Kernel function was also challenging task. The task of hyper parameter tuning was relatively easy as I used scipy api for that.

This assignment was a challenging as well as a learning experience. I have to do a lot of reading to understand and implement the Gaussian Process but it will help in taking courses related to robotics and self-driving cars. As the provided data was real world random raw data, this experience will help me analyze the sensor data from drones, robots, self-driving cars and advance manufacturing equipment.

References:

[1] Stephen Marsland, “Chapter 18: Gaussian Process”, in “Machine Learning: An Algorithmic Perspective”, 2nd Edition, Chapman & Hall/Crc Machine Learning & Pattern Recognition Series, Boca Raton, FL: Taylor & Francis Group LLC, 2014, pp. 395-413.

[2] Wikipedia, “Gaussian process”, https://en.wikipedia.org/wiki/Gaussian_process