HW4 CS391L Gaussian Processes

Avani Agarwal, EID: aa88539
Department of Computer Science
The University of Texas at Austin
avaniagarwal@utexas.edu

1 OVERVIEW

The aim of this assignment is to learn hyper-parameters for 3D x,y,z position co-ordinates using Gaussian Processes. In total we have 50 markers (sensors), each marker has positions (x, y, z), so the marker positions are presented as (markerIndex _x, markerIndex _, markerIndex _z).

In this assignment we aim to tune the hyper-parameters for a global and local kernel for a Gaussian process (GP) to predict the motion of a sensor. We use GP to fit target data as they can give a non parametric representation of data processes if we assume that data is sampled from a multivariate GP. Moreover only the first and second moments of a GP are required to define its behaviour. Using different hyper-parameters we can define kernel functions and we can optimise these parameters to fit the data. The kernel function defines the co-variance and how it acts prior to the GP.

The data given to us comprises 50 sensors that were used to monitor 3D human movements. I chose the AG data set and the sensor 15 due to computational constraints. The data set contains the motion data for a certain part of the body for a given space time. The goal is to represent a subjects motion as a GP. I have chosen the 15th marker placed on the right hand due to computational constraints.

2 METHODS

I used the RBF kernel which can be given as Squared Exponential Kernel (Gaussian/RBF): $k(x, x') = \exp(\frac{-(x-x')^2}{2\gamma^2})$ where γ is the length scale of the kernel. Here x and x' are time vectors and thus, for given time vectors of length N, the kernel function produces an NxN covariance matrix.

The probability of a prediction of data at some input can be conditioned on the distribution of the observations.For training data of the form (y, t) hyper-parameters have to be learnt before predictions can be made, which can be achieved by the log-likelihood function of the Gaussian process model, and maximizing with respect to the hyperparameters.

The three hyper-parameters for a GP σ_f, σ_l and σ_n can be optimised by minimising the log likelihood of the prior.

I used sensor 15 for subject AG for this assignment.

2.1 Cleaning the data

The data provided contains an additional variable c, which indicates whether the input signal was good or not. To clean the data, missing points will be predicted using previous data points. For $C > 0$ data was good and for $C < 0$ data was bad.

2.2 NumPy Implementation

I was able to code kernel and posterior distribution for the GP. I first defined the range t from 0 to 1030 and calculated the zero mean vector and co-variance matrix required for calculating the prior distribution.

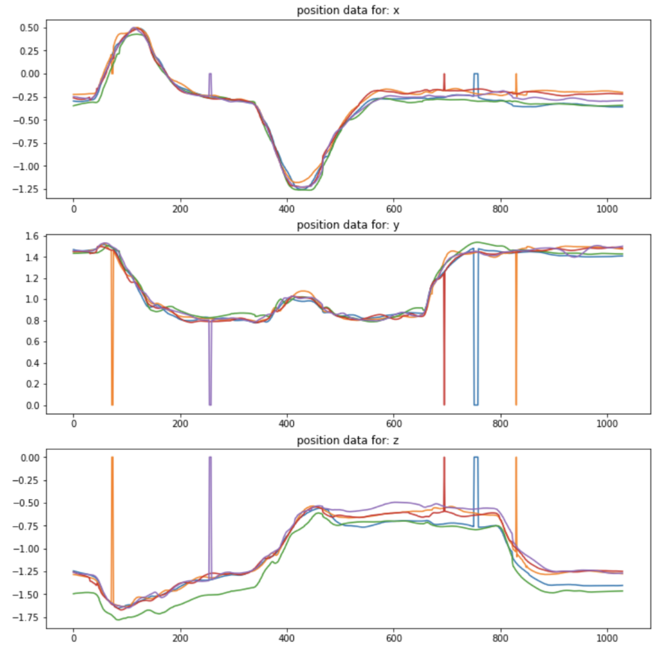

Figure 1: Displaying position data for x,y and z co-ordinate

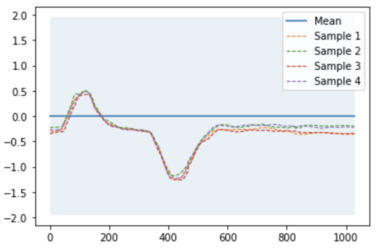

Figure 3: GP mean for four sample x-trajectories (blocks 1-4 for subject AG)

I then obtained training points from the last block at intervals of 30 to calculate the posterior distribution as shown in Figure 4.

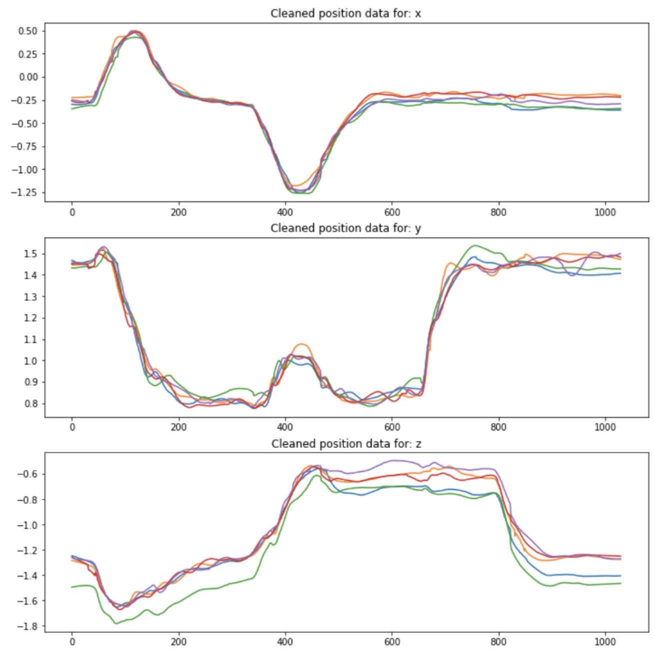

Figure 2: Displaying cleaned position data for x,y and z co-ordinate

```
Text(0.5, 1.0, 'sigma_f = 0.4, sigma_l = 20, sigma_n = 0.4')
```

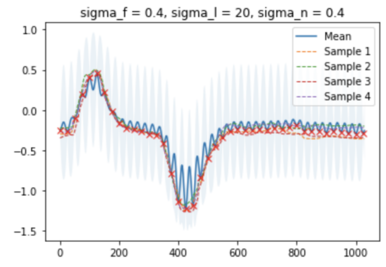

Figure 4: Mean and co-variance of the posterior predictive distribution

I also train the GP on noisy data with noise = 0.5 and sample from last block at intervals of 25 to calculate the posterior distribution.

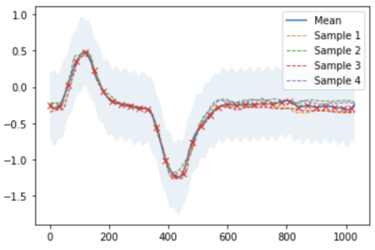

Figure 5: Mean and co-variance of the posterior predictive distribution with noise

2.3 Optimisation of Hyper-parameters

The following steps were performed for optimising the global window for GP.

- The x,y and z trajectory for AG for sensor 15 are used at intervals of 25.
- $.fit()$ function is used to fit the GP with initial parameters $\sigma_f = 1$, $\sigma_l = 1$ and $\sigma_n = 0.5$.
- These are used as input to the RBF kernel which can be given as Squared Exponential Kernel (Gaussian/RBF): $k(x, x') = \exp(\frac{-(x-x')^2}{2\gamma^2})$ where γ is the length scale of the kernel.
- $predict()$ is used to predict the three trajectories that are displayed below and $.get_params()$ is used to store optimised hyper-parameters. We then just have to sample from Gp to predict the trajectory.

3 RESULTS

3.1 Global Window

Below the optimal parameters are shown in Figure 7 and sub-optimal parameters in Figure 6 for global window.

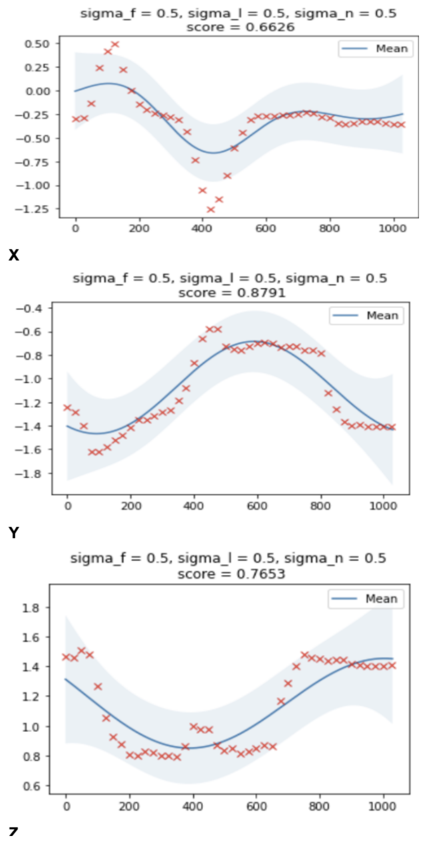

Figure 6: Sub Optimal Hyper-paramters for global window

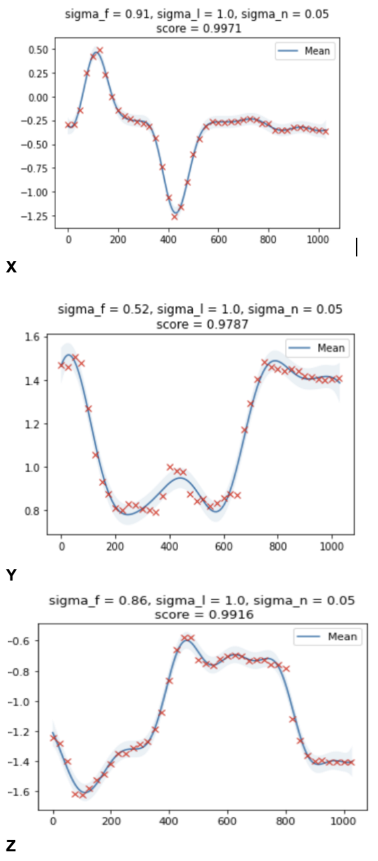

Figure 7: Optimal Hyper-paramters for global window

Figure 8 represents the actual and predicted trajectory.

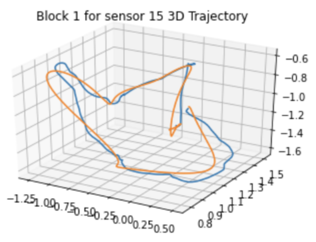

Figure 8: Actual vs predicted trajectory

Figure 9 depicts actual and Predicted x-Trajectory for Block 1 Sensor 15 (right hand).

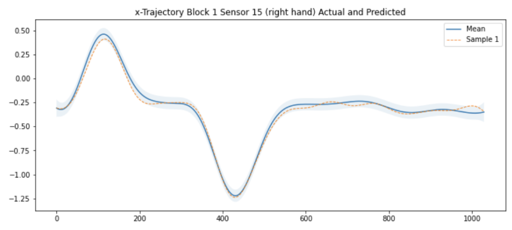

Figure 9: Actual and Predicted x-Trajectory for Block 1 Sensor 15 (right hand)

Figure 10 depicts actual and Predicted y-Trajectory for Block 1 Sensor 15 (right hand).

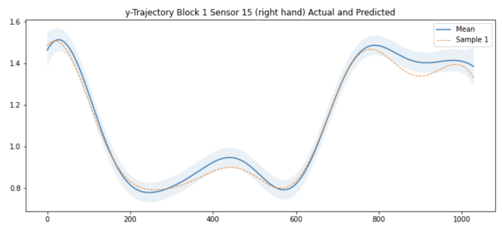

Figure 10: Actual and Predicted y-Trajectory for Block 1 Sensor 15 (right hand)

Figure 11 depicts actual and Predicted z-Trajectory for Block 1 Sensor 15 (right hand).

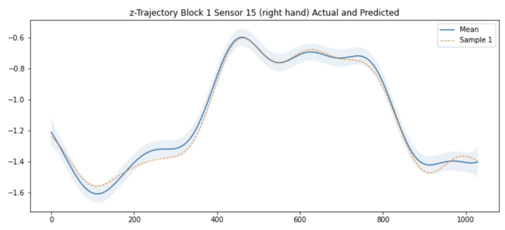

Figure 11: Actual and Predicted z-Trajectory for Block 1 Sensor 15 (right hand)

3.2 Sliding Window

I used a window size of 102 on a range defined by $t = (0, 1030)$ with $\delta = 51.18$ windows were trained and the average and co-variances for the overlapping areas was calculated.The table below indicates the average hyper-parameter values for the 18 windows for the x-y-z trajectories.

	σ_f	σ_l	σ_n
AG X block 1	0.3149	58.0564	0.05
AG Y block 1	0.8013	41.373	0.05
AG Z block 1	0.7601	41.9166	0.05

Below are the plots for sliding window for x,y and z co-ordinate trajectory.

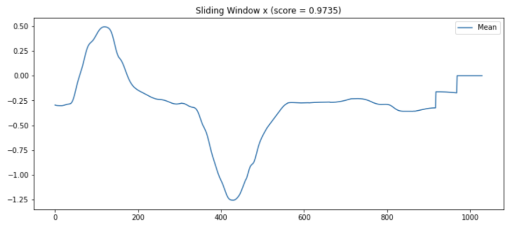

Figure 12: Predicted x-Trajectory for Block 1 Sensor 15 (right hand)

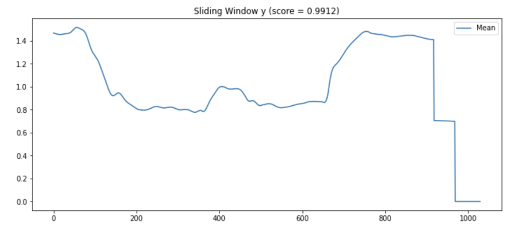

Figure 13: Predicted y-Trajectory for Block 1 Sensor 15 (right hand)

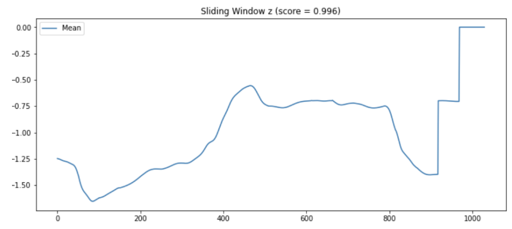

Figure 14: Predicted z-Trajectory for Block 1 Sensor 15 (right hand)

As the window size is uneven the mean is inaccurate towards the end.

4 CONCLUSIONS

The drawbacks are listed in section 5.1 which has been inspired by CMU lecture 21. GP, though effective to predict the 3D trajectory, does suffer from scalability as the number of observations drastically increases. The hyper-parameters do affect the results as σ_f contributed towards variation along the vertical axis and σ_l was used to adjust the smoothness. The average score for GP was 0.98913 for global window and average score of 0.9869 for sliding window was observed . These are very comparable score but global window does give a bit better performance. Maybe the lower score for sliding window was due to uneven window size.

5 NOTES

Please note that the below notes have been acquired from Carnegie Mellon University Machine Learning course lecture 21 notes. I have copy pasted it as part of a special NOTES section as I felt I could not have summarised it better. I understood and learnt a lot from lecture 21 and thought it would be a good idea to include in the report.

A gaussian process can be thought of as a gaussian distribution over functions (thinking of functions as infinitely long vectors containing the value of the function at every input). Formally let the input space $\mathcal{X}$ and $f : \mathcal{X} \to \mathbb{R}$ a function from the input space to the reals, then we say f is a gaussian process if for any vector of inputs $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ such that $x_i \in \mathcal{X}$ for all i, the vector of output $f(\mathbf{x}) = [f(x_1), f(x_2), \dots, f(x_n)]^T$ is gaussian distributed.

The gaussian process is specified by a mean function $\mu : \mathcal{X} \to \mathbb{R}$, such that $\mu(x)$ is the mean of $f(x)$ and a covariance/kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $k(x, x')$ is the covariance between $f(x)$ and $f(x')$. We say $f \sim GP(\mu, k)$ if for any $x_1, x_2, \dots x_n \in \mathcal{X}$, $[f(x_1), f(x_2), \dots, f(x_n)]^T$ is gaussian distributed with mean $[\mu(x_1), \mu(x_2), \dots, \mu(x_n)]^T$ and $n \times n$ covariance/kernel matrix $K_{\mathbf{xx}}$:

$$K_{\mathbf{xx}} = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \dots & k(x_2,x_n) \\ \dots & \dots & \dots & \dots \\ k(x_n,x_1) & k(x_n,x_2) & \dots & k(x_n,x_n) \end{bmatrix}$$

The kernel function must be symmetric and positive definite. That is $k(x, x') = k(x', x)$, and the kernel matrix K induced by k for any set of input is a positive definite matrix. Example of some kernel functions are given below:

- Squared Exponential Kernel (Gaussian/RBF): $k(x, x') = \exp(\frac{-(x-x')^2}{2\gamma^2})$ where γ is the length scale of the kernel.
- Laplace Kernel: $k(x, x') = \exp(\frac{-|x-x'|}{\gamma})$.
- Indicator Kernel: $k(x, x') = I(x = x')$, where I is the indicator function.
- Linear Kernel: $k(x, x') = x^T x'$.

More complicated kernels can be constructed by adding known kernel functions together, as the sum of 2 kernel functions is also a kernel function.

5.1 Drawbacks

GP does not perform good with an increase in number of observations N. Solving for the coefficients α defining the mean function requires $O(N^3)$ computations. Note that bayesian linear regression, which can be seen as a special case of GP with the linear kernel, has complexity of only $O(d^3)$ to find the mean weight vector, for d the dimension of the input space $\mathcal{X}$. Finally to make a prediction at any point, Gaussian Process requires $O(N\hat{d})$ (where $\hat{d}$ is the complexity of evaluating the kernel) while BLR only requires $O(d)$ computations.

6 ACKNOWLEDGEMENT

Thanks to Prof. Dana Ballard and Jirau Song (TA) for their ideas. I would also like to add that section 5 has been taken from lecture 21 notes for Machine learning taught at CMU.

REFERENCES