Alex Navarro ajn853 Machine Learning Assignment 3

**Gaussian Processes**

**Introduction**

Regression is a useful tool in the world of machine learning. Using regression, whether linear or otherwise, it is possible to make informed estimates of the output of a process or function given enough sample data. The most common type of regression is linear regression in which the data is modeled as a function of the form , and the regression process creates estimates for the parameters *m* and *c*. However, another more powerful form of regression is known as a Gaussian Process.

When using Gaussian Processes, or GP’s, there is no assumption made on the model of the data. Each data point is assumed to be normally distributed around some mean function value, and the regression curve or function is created based on its proximity to nearby datapoints. Using this method, it is possible to not only create an estimate of the function value at arbitrary test points but to also determine the uncertainty of that estimate based on the uncertainties and relative spread of the data points.

**The Data**

Provided were motion capture datasets from 12 individuals. Each individual was asked to trace their finger according to a certain target trajectory while data was recorded. The data itself consists of 50 different tracker trajectories in the X, Y, and Z directions. In addition to each of the 50 trackers, the resolved trajectory for the individual’s head and finger trajectories are included in the data.

**Chart

Description automatically generated**

Figure 1: Target trajectory for the individuals to follow

Each individual performed the motion 5 times, giving a total of 60 trajectories with 50 timeseries of over a thousand data-points each. Not all of the data is used for this assignment. In particular, the 5 runs of a single individual are used, the first four to generate mean and standard deviations for the movements, and the fifth as a test data set to quantify the validity of our gaussian regression.

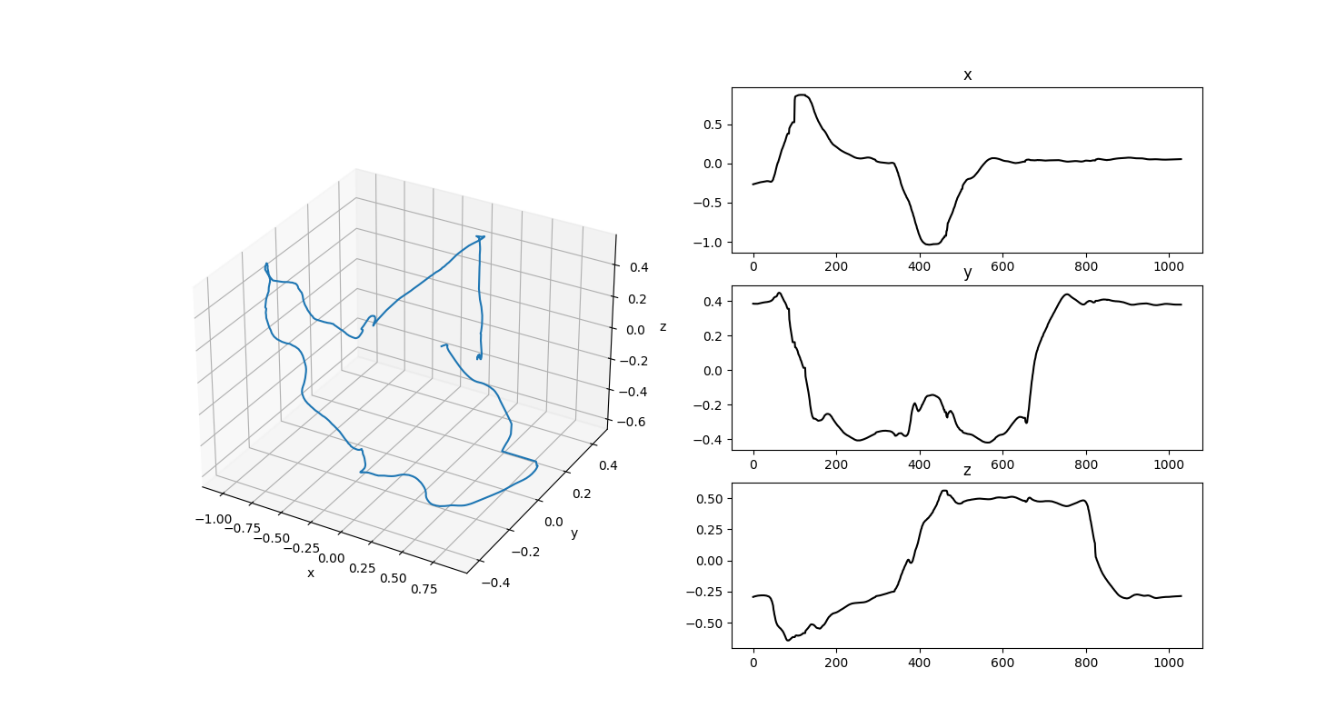


Figure : Average trajectory of the first individual

**Method**

Regression

The GP regression algorithm is built around the kernel function chosen. In our case, we are using the squared exponential kernel function

where and are hyperparameters to be determined. Additionally, there is always noise in real world data, and so the kernel function should also incorporate that. This is done with addition of an additional noise hyperparameter . In this way, the kernel function ends up looking like

Now consider a case in which we have a data vector **y** along with its independent variable **x**. Let us also assume that we have the variance data for the **y** vector such that for a given datapoint “” in the **y** vector, . Then, if we want to determine the value of y at some arbitrary test point , we know that the underlying distribution is joint Gaussian of the form

where K is a matrix of the form

and additionally and .

And since we have the entire **y** dataset, we can condition the probability of based on the data present in **y**. This gives use the final form of the gaussian regression equation

Hyperparameters

The hyperparameters and are used to characterize the data of the Gaussian regression. The term characterizes the variance of the data, while , the length parameter, is a measure of how close two data points have to be in order to have a significant effect on the estimates of one another. The term is the noise term and accounts for the variability due to noise. It has a very similar effect on the fit to , but unlike its counterpart, can be measured if the variability of the input data is known.

Other than , these hyperparameters cannot be known in advance and so it is necessary to calculate them. They can, of course, be fit manually, but that is a long and tedious process that gives suboptimal results. The better course of action is the solve the maximization problem on the log likelihood of the data values conditioned on the independent variable **x**

This maximization problem was solved with the use of the scipy.optimize library function minimize, which uses the BFGS multivariable optimization algorithm. In order to convert the problem from a maximization problem to a minimization problem, the sign of the objective function was simply switched, such that the final problem was of the form

**NB:** This function is much reduced compared to the original function as constant terms and scaling factors have been removed.

**Results**