Modeling Motion Data with Gaussian Processes

Daniel Alejandro Noble Hernandez
dan833
The University of Texas at Austin

---

1. Introduction

Gaussian processes are random process that allow one to extend the idea of a Gaussian distribution to a collection of random variables as a distribution over possible functions; they can be visualized as an infinite-dimensional generalization of a multivariate normal distribution. A Gaussian process utilizes a kernel function to specify the covariance of the prior – which will be used to update beliefs about the prediction – and that kernel function, or kernel functions, has certain hyperparameters that can be optimized to improve the prediction of the data.

The purpose of this project was to take five separate traces of a positional coordinate of a marker on a person's body over time and come up with a Gaussian Process with which to model it based on the traces. This was done in two different ways. The first was by coming up with a kernel using hyperparameters applied to the whole dataset, allowing for a general Gaussian Process to be fit to the data. The second involved using a sliding window method that came up with a local kernel, optimizing the hyperparameters for each particular window of the trace. These results could then be compared using a log marginal likelihood to determine which was the more effective method.

---

2. Methods

2.1 Prediction with global kernel

Before generating a global kernel for a prediction, the dataset had to be read in. This was done using pandas, which allowed for a coordinate from one of the markers to be read in across all five traces for a certain participant. The frame and desired position values were isolated from the rest of the dataset – discarding values that had a negative c value, which indicated an invalid data point – and position values were sampled at random across the five traces to generate a representative trace to which one would fit a Gaussian Process. This led to a 5150 x 2 array, indicative of 5 traces with 1030 time points each.

The global kernel and Gaussian Process were fitted and implemented respectively using the following steps:

1. Initialize the hyperparameters sigma_f, sigma_l, and sigma_n and generate a kernel using the following function:
   $Kernel = sigma_f^2 * RBF(lengthScale\ = sigma_l) + WhiteKernel(noiseLevel = sigma_n^2$
2. Fit the data using the GaussianProcessRegressor from scikit learn
3. The GaussianProcessRegressor optimizes the hyperparameters automatically, so find and save those.
4. Predict the Gaussian process and use the returned mean to graph the prediction, whilst using the covariance to plot the confidence interval.
5. Plot each of the traces simultaneously alongside the prediction and all the datapoints as shown in Figure 1.
6. Compute the log marginal likelihood of the general kernel.

The prediction is visualized alongside the rest of the data in Figure 1, and from then on, the remaining step is to compare the generated log marginal likelihood with that of the local kernel to determine a most efficient method.

2.2 Prediction with local kernels

The local kernel followed much the same steps as described above, with the following exceptions. A window size of 50, a step size of 10, and an initial position of 0 were specified before carrying out the implementation. The author used a while loop to move across the data, incrementing the window by the step size with each iteration, until the full dataset had been iterated through. For each step of the loop, a Gaussian Process was fitted to the current dataset and the mean value – to be used for the prediction – was used to come up with an average mean where the windows overlapped for different kernels.

As was done for the global kernel, a local log marginal likelihood was computed and the updated mean value was used to plot the five traces alongside the data points and the prediction. This result can be seen in Figure 2 and can be compared with the equivalent generated with the global kernel.

---

3. Results

3.1 Prediction with global kernel

The results from making a prediction with the global kernel can be seen below in Figure 1. The plot shows each of the traces alongside the prediction that uses the randomly sampled data and the data points.

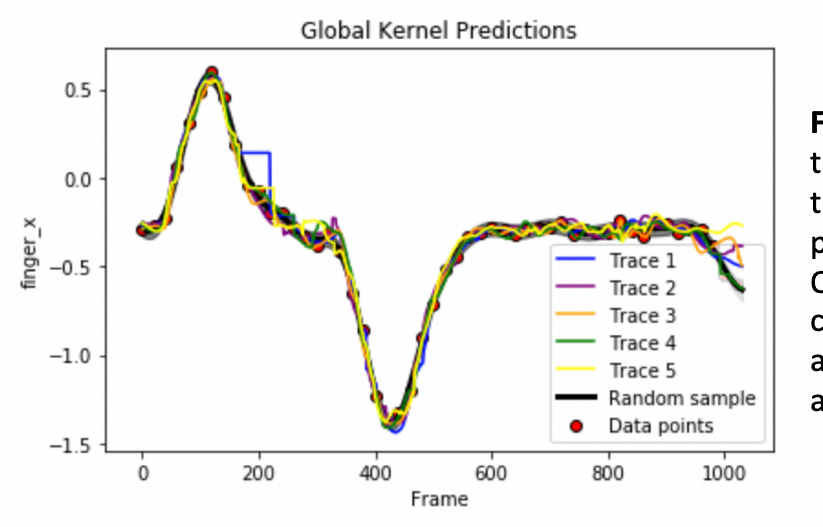

Figure 1: The figure shows the plots for each of the traces alongside the prediction and data points. One can see that the traces closely follow the prediction and data points, suggesting an efficient method.

Qualitatively, one can judge that the fit closely follows the data and thus appears to be an efficient method.

3.2 Prediction with local kernel

The results from the local kernel are displayed in Figure 2 below. It is carried out much the same way as Figure 1, but the main difference can be seen in the random sample. Unlike that in the global kernel, this one does not follow the data as closely, and can be qualitatively shown to be a less efficient method.

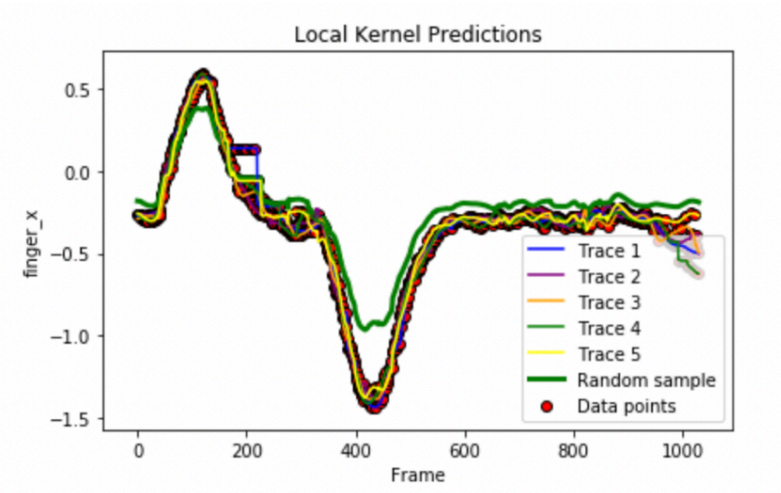

Figure 2: The plot shows the traces alongside the prediction after using the local kernel. The prediction can qualitatively be seen not to follow the data as closely, suggesting that it is not as efficient as the global kernel

3.3 Comparison of global and local kernels

Figure 3 shows the values of the log marginal likelihood for the global and local kernel methods, and the significantly larger value of that from the global kernel indicates quantitatively that this is a better method.

```
LML for global kernel = 49.48189249428893
LML for local kernel = -0.30546859981205376
```

Figure 3: The log marginal likelihood is higher for the global kernel than that the local kernel, confirming that the global kernel is more efficient

3.4 Hyperparameter values

Figure 4 shows the initial and changing hyperparameter values alongside the prediction made by the Gaussian Process. The problem with it is that the updated values of the hyperparameters are scaled incorrectly and so are shown compressed on the right hand side. The author ran out of time to correct this but the fluctuation of values is the same.

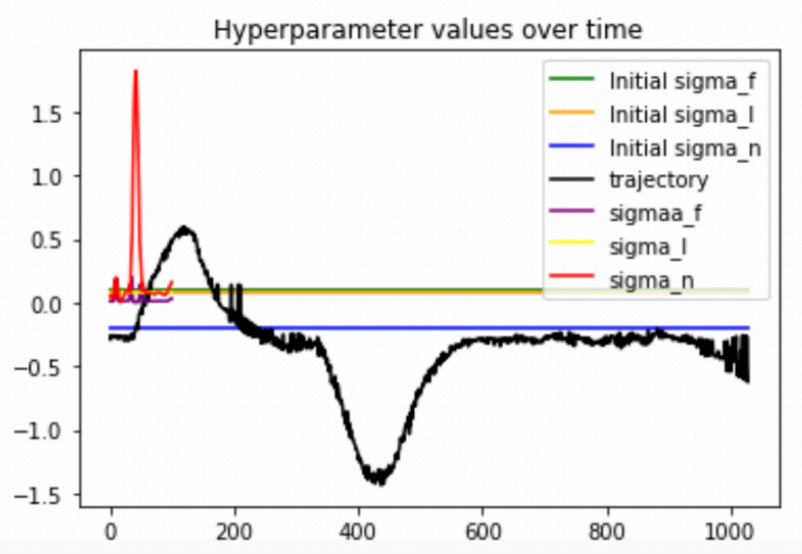

Figure 4: The figure shows the initial hyperparameter values alongside the prediction made of the data and the changing values of the hyperparameters.

One can see that sigma_f changes marginally throughout the course of the loop, but sigma_l does not change at all, and the majority of the variability is seen in sigma_n, which fluctuates significantly halfway through.

---

4. Conclusion

Gaussian processes were generated successfully using two separate methods: the first was done using a global kernel and global hyperparameters applied to the full dataset, and the second was carried out by using a sliding window that came up with local kernels for small sets of the data. This was done to determine if one method was better than the other, and one can draw the conclusion that the global kernel generated better results overall. There is one aspect that could have been implemented better; the changing values of the hyperparameters were not scaled correctly and thus appear skewed in Figure 4, but as the author noted earlier, this can be visualized by imagining the same data spread out across the x_axis. Overall, the general kernel displayed a significantly more efficient and accurate method.