Detecting Muscle Cocontraction Through Sliding Window Gaussian Processes

Abrar Anwar
aa76875
University of Texas at Austin
abraranwar123@gmail.com

Abstract

Muscle cocontraction has always used electromyographic data to detected in muscles. Previous works have given multiple intuitions on when cocontraction occurs. The use of a sliding window to create a collection of overlapping Gaussian processes are proposed, where the similarity/differences in the data are represented by nearby kernel hyperparameter values. Given a dataset of motion capture data of a subject doing a variety of traces, this approach is then analyzed and shown to be an effective means of detecting muscle cocontraction in a human elbow joint.

1. Introduction

Muscle cocontraction is defined as the simultaneous contraction of two groups of muscles (agonist and antagonist muscles) around a joint in order to stabilize one's limbs. The detection of muscle cocontraction is typically discovered through electromyographic (EMG) data by attaching surface electrodes to subjects who then perform tasks. This motivates the creation of an alternative approach for detecting cocontraction. A collection of Gaussian processes trained on a keypoints in a dataset of human joint poses over multiple time intervals is explored as an approach to detect potential muscle cocontraction.

1.1. Muscle Cocontraction

Muscle cocontraction is typically seen as inefficient [2]; however, one belief is that cocontraction can be optimal in uncertain (non-deterministic) environments [4]. To test their predictive gait simulation model for it, a swing-up problem was simulated, where the arm moved from a downwards to an upwards position with the goal of optimizing energy usage. Their simulated model minimized muscle activation to make trajectory optimizations of human movements more accurate.

In an experiment to detect cocontraction during arm movement accuracy tests, it has been observed that there exists an inverse relationship between target size and cocontraction in the arm when attempting to point towards a target [1]. Furthermore, it has also been seen that muscle cocontraction is related to movement velocity [9]. It has also been seen in some muscle groups that muscle cocontraction occurred during fast movements for young subjects, while it was was more frequent during the deceleration phase for elderly subjects [3]. While also investigating human arm impedance effects against a robot arm, [6] concluded with statistically significant results that muscle cocontraction induces an anisotropic change of arm stiffness, which means muscle cocontraction is related to arm stiffness across specific axes.

Previous research allows me to hypothesize that Gaussian processes fit on specific coordinates may be able to detect muscle cocontraction, as this model could potentially encode changes in velocity and direction.

1.2. Gaussian Processes

A Gaussian process (GP) is a distribution over an infinite number of possible functions. GPs take an input x and predict some output y.

$$y = f(x) + \epsilon \tag{1}$$

where ϵ is Gaussian distributed noise. We will say the Gaussian process is:

$$f(x) \sim GP(m(x), k(x, x')) \tag{2}$$

where $m(x)$ is a mean function, which is usually set to zero, and the latter is the covariance function. A GP is a nonparametric prior over functions which are given by the mean and covariance functions. The covariance function used in this paper is the squared-exponential covariance function (3), also known as the radial basis function (RBF). The kernel has three hyperparameters σ_f, σ_n and ℓ. σ_f refers to the signal variance, while σ_n is the noise variance. ℓ is the length-scale that defines the RBF function.

$$k(x, x') = \sigma_f^2 exp(\frac{1}{2\ell^2}(x - x')^2) + \sigma_n^2 \delta \tag{3}$$

Given a data matrix D, we can build a covariance matrix $K_{ij} = k(x_i, x_j)$ where $x_i, x_j \in D$. With this covariance matrix, we can build a GP posterior for a test input X_*, training inputs/outputs X, f, and an estimated output f_* :

$$p(f_*|X_*, X, f) = \mathcal{N}(f|\mu, \Sigma) \tag{4}$$

where $\Sigma = K_{**} - K_*^T * K^{-1} K_*$.

Algorithm 1 Gaussian Process Fit/Predict

1: **Input:** Training input X, training output y, covariance function $k(x, x')$, noise level σ_n^2, and test input X_*
2: $L = cholesky(K + \sigma_n^2 I)$
3: $\alpha = L^T \setminus (L \setminus y))$
4: $\mu = K_*^T \alpha$
5: $v = L \setminus K_*$
6: $Var(\mu) = K_{**} - v^T v$
7: $\log p(y|X) = -\frac{1}{2} y^T \alpha - \sum_i \log L_{ii} - \frac{n}{2} log 2\pi$
8: μ is the mean of the prediction, $Var(\mu)$ is the variance of the prediction, and $\log p(y|X)$ is the marginal log likelihood

The common algorithm for calculating GPs is seen in Algorithm 1[1] [7]. The algorithm shows to be quite simple. A Cholesky decomposition of the covariance K gives us a lower triangular matrix such that $LL^T = K$. This is meant to handle the case of matrix inversion used in the equation for Σ above. A Cholesky decomposition is faster and more numerically stable than matrix inversion. An additional bonus of using Cholesky is the simplicity to calculate the log determinant, as seen in (5), which is required as the second term in the marginal log likelihood. Cholesky decomposition is also where GPs main issue arises: matrix inversion is $O(n^3)$; however, this will not be an issue in our small dataset. In addition, we add the noise parameter σ_n^2 to the observations in order to turn a potentially singular matrix to be nonsingular and tractable [5].

$$\log|A| = 2 \sum_{i=1}^{n} \log L_{ii} \tag{5}$$

Now that we have an algorithm for calculating GPs, the next step is to optimize the hyperparameters of the GP; in our case it's $(\sigma_n, \sigma_f, \ell)$. This is done by minimizing the marginal log likelihood function.

An assumption a GP makes is that the similarity of the output of a function increases as the similarity between two solutions increases [8]. This implies a kernel function can encode the similarity between the two solutions using its hyperparameters. Through this reasoning, I believe a sliding window approach of a collection of multiple GPs should be able to show where muscle cocontraction is occurring.

[1]I initially implemented this algorithm, but once we were allowed to use scki-kit learn, it has more stability than the usual algorithm used for GPs, thus provides better results

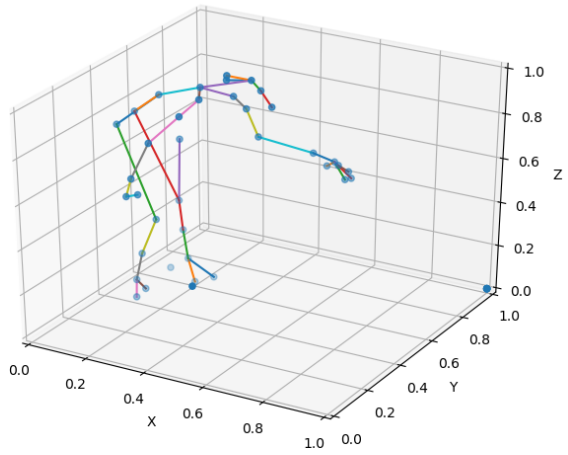


Figure 1. The 570th frame of subject AG's first trace overlayed on a skeleton structure (inverted)

2. Method

2.1. Dataset

The given dataset is of 3D keypoints generated through a subject wearing a motion capture suit and performing multiple tracing tasks. The motion capture suit gathers 50 keypoints on a subject as seen in Figure 2 located on several joints on the human body. Twelve subjects repeat a specific trace five times. Each trace consists of 1030 frames, sampled at a rate of approximately .011 seconds per frame; however, there are some skips of varying times between frames. This causes drift between traces where times don't match consistently. This causes a drift over the trace, where the end goal is as much as .15 seconds off. As such, the input X for the Gaussian process will be the time, not the frame number. If we look at Figure 1, we can see a skeleton frame based on the trace data. It becomes apparent that at multiple times throughout the trace some points are obstructed thus are empty points in the data. Also, the image is inverted. It appears as though the left arm is tracing, but it is actually the right arm.

2.2. Data Selection

Of the data provided, unique markers need to be chosen to do analysis with. Subject AG was chosen, who does a square trace. Marker 8 was chosen as the marker to analyze, as it is the right elbow, so it should be expected to detect muscle cocontraction since the right arm performs the trace. Ideally it will be seen during the upward motion as mentioned in [4]. In Figure 3, the trajectories of three of the five available traces for marker 8 can be seen in 3D.

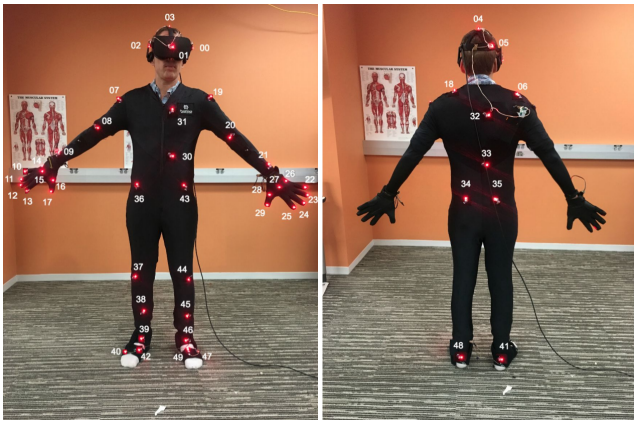


Figure 2. Example of the motion capture data provided by Dr. Ballard

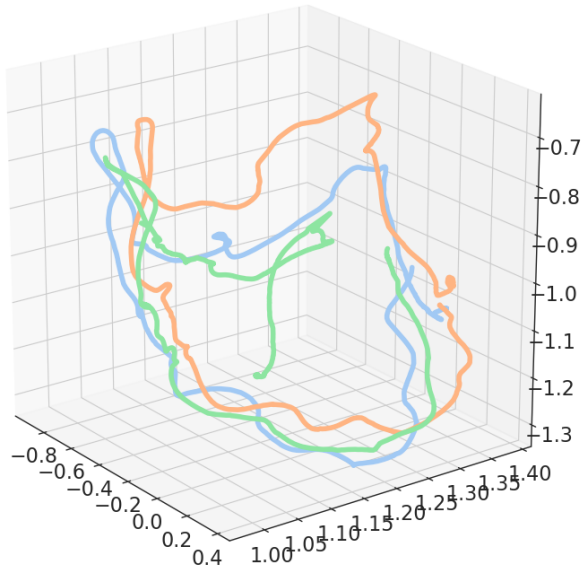


Figure 3. 3D trajectories of marker 8 of subject AG

Overall, 6.25% of the data is sampled to fit the GP, which comes to be a satisfactory number that still maintains speed and accuracy.

2.3. Global Kernel vs Sliding Window

A typical Gaussian process would use a single kernel function to represent the entire time series. A global kernel is decent for most cases, and appears to have a competitive sum of squared residuals compared to having more windows represent the data, as seen in Table 1. The windows in the table refer to how many windows the time were split up into using even numbers of sliding window instances for the dataset. The training was on four of the five tests, and then tested on the other one to calculate the residual error. Although the global kernel cannot tell us much about muscle cocontraction through its single set of hyperparameters,

Table 1. Residuals for multiple windows on marker 8_x

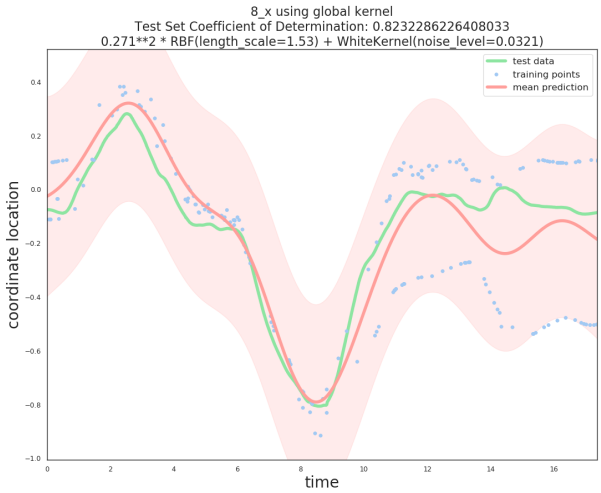
Windows	Sum of Squared Residuals
1	71.2489
2	70.0175
4	69.7980
7	74.3624
10	69.2276
16	66.9623
25	69.5048

it's able to compete in modelling human motion data to a set of many kernels. The lowest residual error is relatively significant at 16 windows, which will be discussed later.

2.4. 1D vs 3D Coordinate

The Gaussian process can predict either one coordinate, namely the x-coordinate of marker 8, or predict all of the coordinates of a given marker at once using the same kernel. A potential advantage in using 3D with a collection of overlapping kernels is the potential anisotropic nature of muscle cocontraction [6] could be detected in the 3D case.

3. Results



3.1. Global Fitting

A sample fit of a GP on the data for marker 8 can be seen on Figure 4 and 5. The shaded areas represent a 95% confidence interval from the mean, which is the colored line. Having a single kernel function represent a single dimension, the standard deviations are able to represent most of the training data on the interval. The 3D global kernel is unable to represent the outliers in the case of the orange line in the middle, as we see some of the training data is left behind.

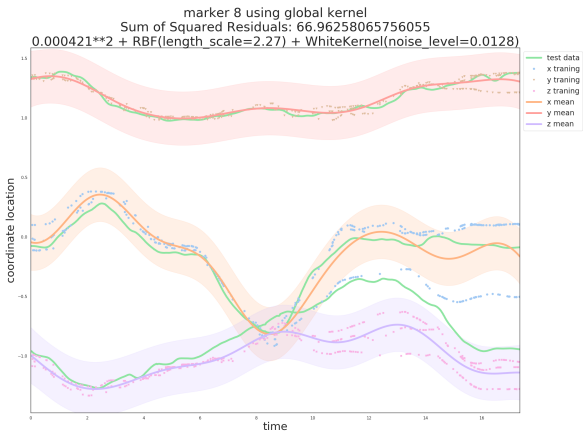


Figure 5. Global kernel used for the 3D marker 8 (xyz)

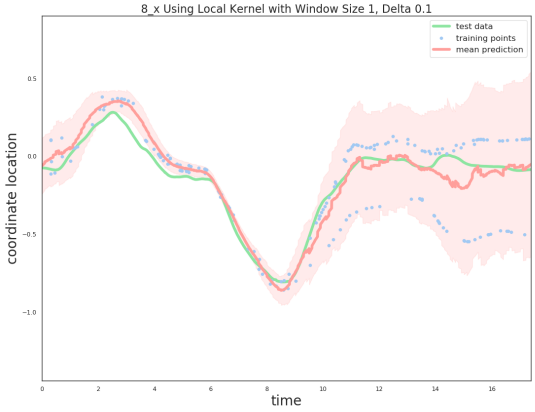


Figure 6. Local kernels with a window size of 1 and a delta of .1 for 1D data

3.2. Sliding Window Fitting

A sliding window in the 1D and 3D cases provide interesting results when looking at a qualitative visualization on what encapsulates the data the best. Figure 6 shows the sliding window in the case of marker 8. For any given point in time, there exists multiple GPs that can give an output. The average mean and standard deviation is represented in the visualization. Due to this, you can see there are some discontinuities, but it should not be much of a problem. The fit is good using a temporal window size of 1 seconds with a delta of .1 seconds. This means any given time point is fit with 20 different kernel functions. The average of these across multiple scales give us good predictions. As the delta is increased, it does not do much other than make the function smoother, as seen in Figure 7. The additional smoothness will cause issues when analyzing the change in hyperparameters, so it would be preferable to choose a slightly coarse approximation.

In the 3D case, the same as above applies. In Figure 8, we can see that the sliding window fits are significantly better than the use of a global kernel in both the 1D and 3D cases. This is important because in some motions, humans start off with strong cocontraction, but as time goes on, variance begins to show and muscle cocontraction decreases [1]. Due to this, the use of a sliding window has advantages for modeling human motion as it can better model that variance compared to a global kernel.

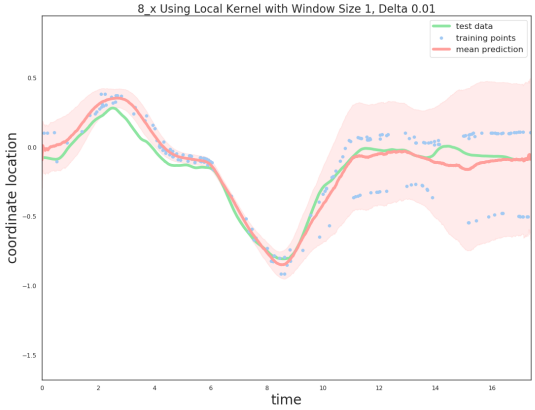


Figure 7. Local kernels with a window size of 1 and a delta of .01 for 1D data

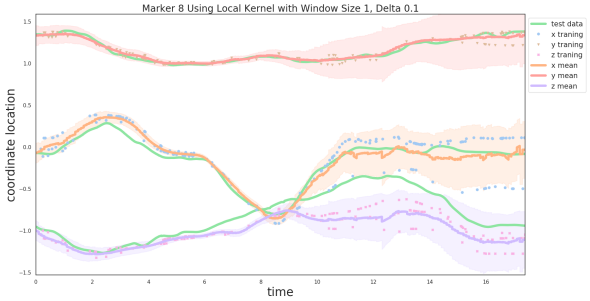


Figure 8. Local kernels with a window size of 1 and a delta of .1 for 3D data

3.3. Hyperparameter Comparison for Muscle Cocontraction Detection

In the 1D case, Figure 9 and 10 shows the 3 hyperparameters for a temporal window size of 1 with a delta of .1 and .01. As seen previously during the mean prediction, the data is smoother with a smaller delta. The red line represents one of the trajectories in the dataset (the prediction was not placed in there as it looks slightly discontinuous, but the analysis will end up being the same). We can tell during changes in motion, the sigma_l hyperparameter changes. When the motion of the x axis changes, a slight bump is vis-

Table 2. Time intervals where hyperparameters begin to shift for the 3D case

Approximate Time	Close Hyperparameters
4-6	sigma_f, sigma_l
7-10	sigma_n
11-12.5	sigma_l

ible during the motion. These times roughly match up with those in Table 2, which describe the 3D data's hyperparameters. However, there does exist some variations that don't seem to line up with any changes in our data, namely after time 12.5, where sigma_f is slowly increasing. These variations in the kernel function could be used to detect muscle cocontraction.

More interestingly, in the 3D case, Figure 11 shows some more interesting results. The shifts in the hyperparameters for the 3D case are put in Table 2. During the time interval of approximately 4 seconds, sigma_f and sigma_l's values cluster together to be higher, specifically during the deceleration period during the movement of making the shape, which is in line with findings of previous works [3] (although this is under the assumption it holds for the elbow joint). Although it is impossible to tell whether true muscle cocontraction is occurring as we do not have a ground truth dataset, it definitely provides intuition on whether it is occurring or not.

Figure 12 shows n sliding windows evenly split between the data for marker 8along with their corresponding hyperparameters in Figure 13[2]. These figures show the center of each window rather than the mean of multiple kernels' outputs. Due to this, there is more discontinuity in the graph, but it should be fine. Furthermore, the hyperparameters are split over windows rather than the average hyperparameters at a given timeframe. This way it's easier to see differences between the windows. Irregularities in the hyperparameter space are denoted by stars on Figure 12. It is evident that lower amounts of windows to cover a time series has little variation in the kernel hyperparameters. In this formulation of the sliding window, the usage of 16 windows which has the lowest squared residual error as seen back in Table 1, as well as align with where I believe muscle cocontraction is occurring. This also lines up with the 3D data's approximate jumps in the hyperparameters as seen in Table 2.

4. Summary

This paper explored the usage of a collection of overlapping Gaussian processes through sliding windows in order to detect muscle cocontraction. Although there does not exist any ground truth data, the use of intuitions and findings from previous works in muscle cocontraction allows the hyperparameter space of a collection of GPs to be a plausible method to detect cocontraction.

[2]These graphs looks as such as it was made before the TA provided the rubric and stated how the data was meant to be presented. I believe these still provide useful information for detecting muscle cocontraction

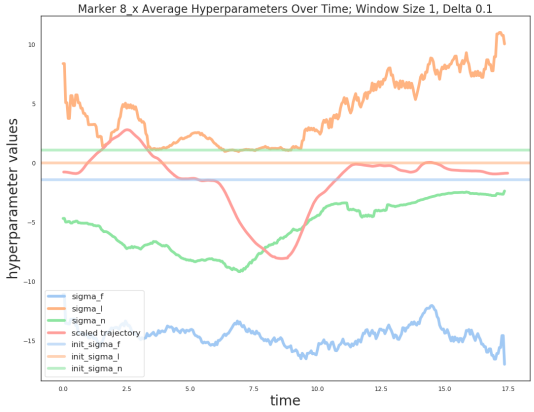


Figure 9. Hyperparameters for local kernels with a window size of 1 and a delta of .1 for marker 8_x

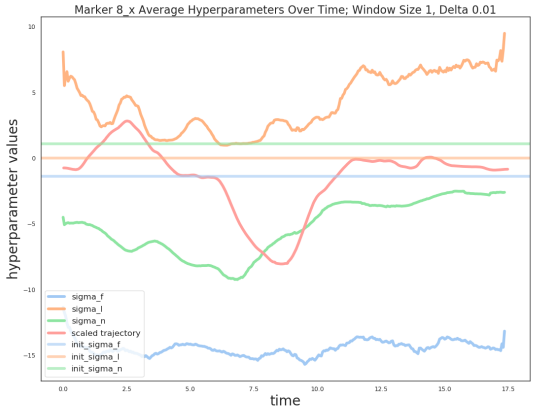


Figure 10. Hyperparameters for local kernels with a window size of 1 and a delta of .1 for marker 8_x

References

[1] P. L. Gribble, L. I. Mullin, N. Cothros, and A. Mattar. Role of cocontraction in arm movement accuracy. *Journal of Neurophysiology*, 89(5):2396–2405, May 2003.

[2] N. Hogan. Impedance control: An approach to manipulation: Part ii—implementation. 1985.

[3] Y. Iwamoto, M. Takahashi, and K. Shinkoda. Differences of muscle co-contraction of the ankle joint between young and elderly adults during dynamic postural control at different speeds. *Journal of Physiological Anthropology*, 36(1), Aug. 2017.

[4] A. D. Koelewijn and A. J. [van den Bogert]. A solution method for predictive simulations in a stochastic environment. *Journal of Biomechanics*, page 109759, 2020.

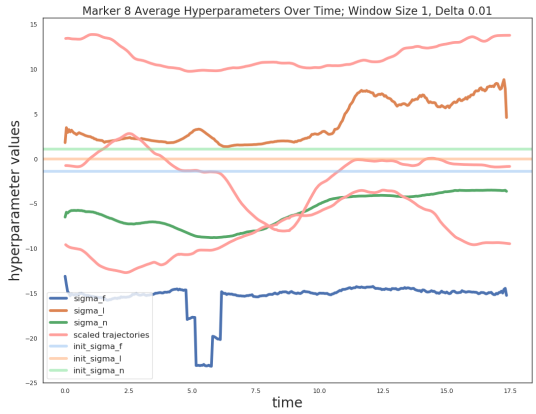


Figure 11. Hyperparameters for local kernels with a window size of 1 and a delta of .01 for 3D marker 8

[5] A. McHutchon and C. E. Rasmussen. Gaussian process training with input noise. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 1341–1349, Red Hook, NY, USA, 2011. Curran Associates Inc.

[6] H. Patel, G. O'Neill, and P. Artemiadis. On the effect of muscular cocontraction on the 3-d human arm impedance. *IEEE Transactions on Biomedical Engineering*, 61(10):2602–2608, 2014.

[7] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[8] I. Roman, R. Santana, A. Mendiburu, and J. A. Lozano. Evolving gaussian process kernels from elementary mathematical expressions, 2019.

[9] M. Suzuki, D. M. Shiller, P. L. Gribble, and D. J. Ostry. Relationship between cocontraction, movement kinematics and phasic muscle activity in single-joint arm movement. *Experimental Brain Research*, 140(2):171–181, July 2001.

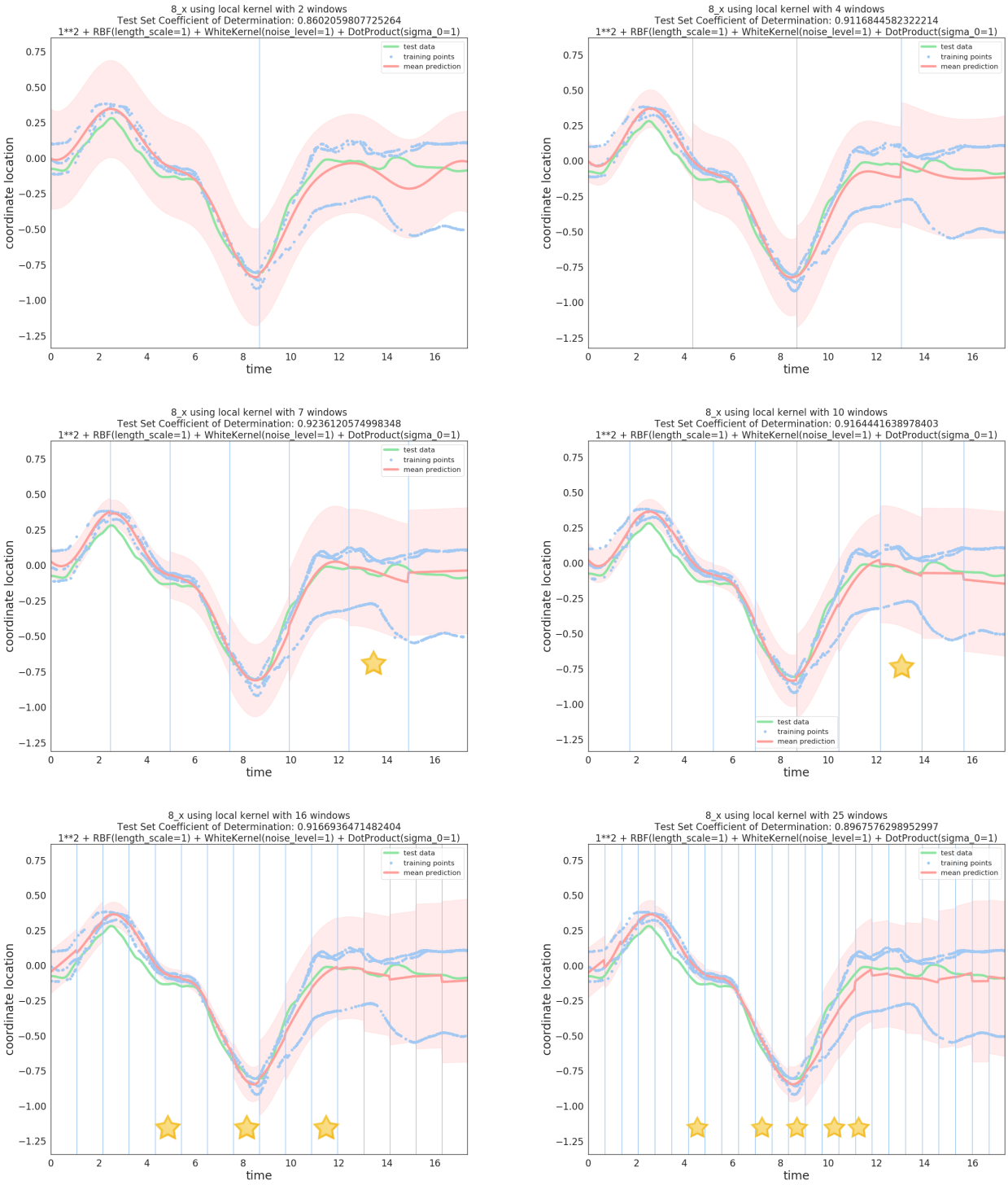


Figure 12. Various local kernel window sizes split up for marker 8_x

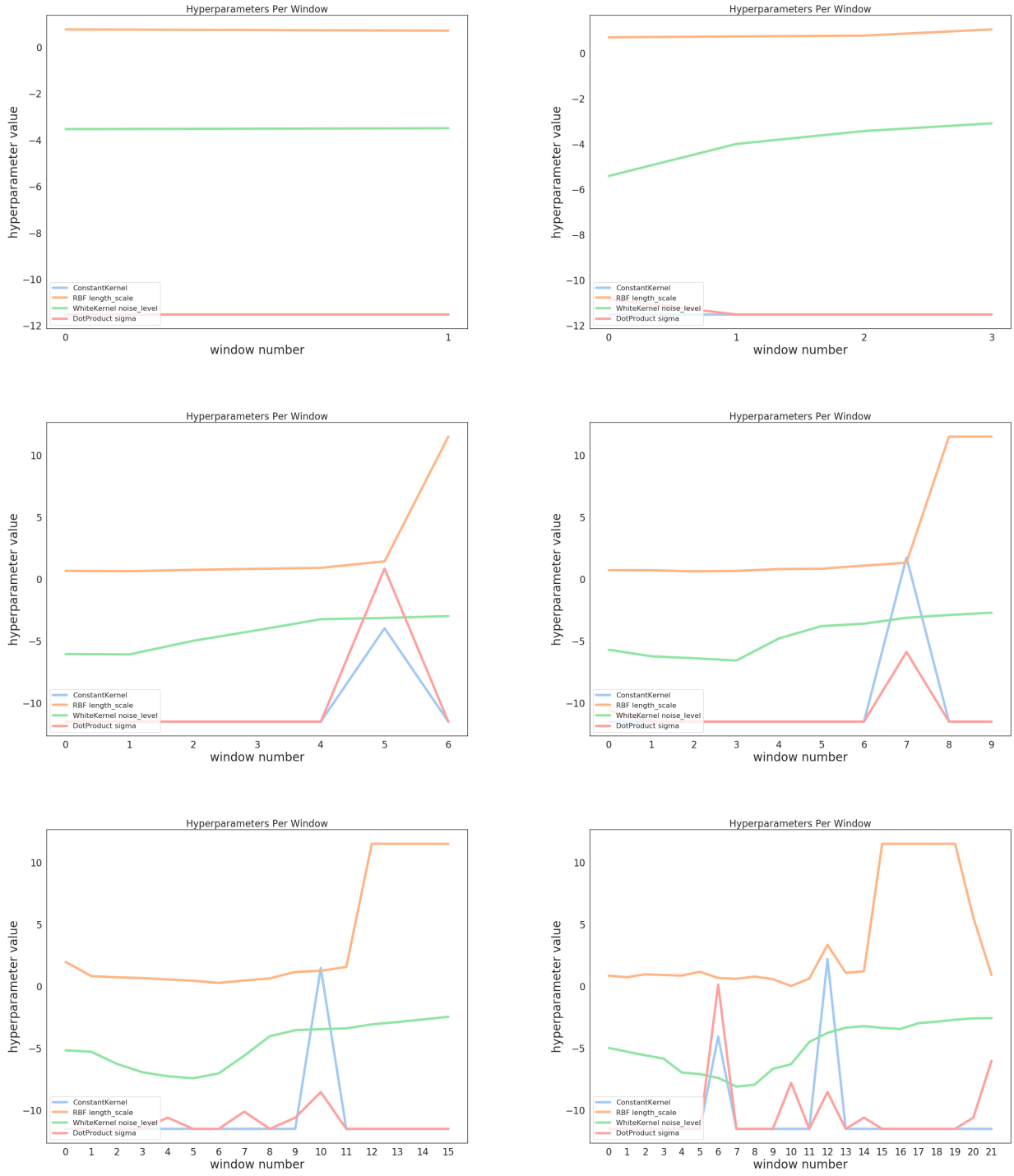


Figure 13. Hyperparameters for various local kernels split up for marker8_x