University of BRISTOL

[2020-04-20 Mon]

# GPR Kernels

Mark Blyth

# This week's goals

- Rederive GPR for vector outputs
  - Turns out this is an open problem
  - Hard to do generally, but I've found a way to avoid needing this
- Get GPR to work
  - Some success
- Use it for a predictor-corrector
  - Not got this far yet

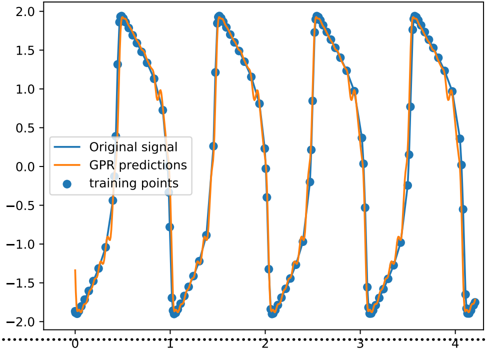
# Last time. . .

> University of BRISTOL
>
> In Silico CBC
>
> *[2020-04-15 Wed]*
>
> ## Period windowing (1)
>
> - We have a periodic signal $f(t')$, taken from our observed system output (here, neuron spikes)
> - We wish to split it into windows $f_1(t), f_2(t), \dots, t \in [0, 1]$, such that $f_i(1) = f_{i+1}(0)$ (periodicity)
> - Then $f_i(t)$ is a function representing the $i$'th period of the signal
>   - Eg. if $f(t') = \sin(\frac{t'}{2\pi})$ with $t \in [0, \infty)$, then $f_1(t) = \sin(\frac{t}{2\pi})$, $f_2(t) = \sin(\frac{t}{2\pi} + 2\pi)$, $f_3(t) = \sin(\frac{t}{2\pi} + 4\pi), \dots$, with $t \in [0, 1]$
>   - By periodicity we have $f_i(t) = f_j(t)$, and $f_i(1) = f_{i+1}(0)$
> - Fitting a model $t \to f_i(t)$ to these function observations gives us the periodic orbit model $f^*(t)$ at the current parameter value
> - It's hard to split data up into these periods!
>
> 7 / 13 bristol.ac.uk

Splitting data into periods turns out to be easy:

- Rescale time as $t = t'/T \mod T$
- Periodic kernels take care of this automatically
  - Choosing an appropriate kernel can make life much easier

# Model fitting – it sort of works!




- Trying to model a Fitzhugh-Nagumo output with a Gaussian process
- Some wiggliness between datapoints (more on that later!) but it generally works

# 30 seconds intro to Gaussian processes

- Here, takes time as an input, and outputs a Gaussian distribution at that time
- The Gaussian distribution is the probability distribution of our function value, at that time
- This works by maintaining a probability distribution over candidate functions
- Bayes' rule is to condition on the evidence, and form a posterior function distribution
- *Bayes' rule needs good priors!*

# GPR kernels

- They specify our prior distributions over functions
- Good kernels = good priors = good results
- (Kernels are interesting – they implicitly encode an infinite dimensional feature space)

# Periodic kernels

Data are periodic, so it makes sense to have a kernel that's periodic

- If we choose a periodic kernel then we're favouring periodic functions in our prior function distribution
- Periodic kernels give better fits on periodic data!
- But, if the period isn't specified correctly, they'll give big errors and be harder to optimise...

# Aperiodic kernels

To avoid period-errors, use an aperiodic kernel and overlay each period's data on top of each other

- Aperiodic kernels don't encode our prior beliefs about periodicity, so they're not going to give as good a fit to the data
- But, they have no period component, so they aren't sensitive to errors in the period
- This means that in practice they can actually still give reasonable fits to the data

# Hyperparameters

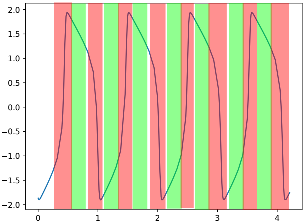
Both periodic and aperiodic kernels rely on hyperparameters; often. . .

- $l$: how similar nearby datapoints are
- $\sigma_f^2$: function amplitude
- $\sigma_n^2$: noise in the function observations

There's an interplay between kernel choice and hyperparameter selection:

- well-chosen kernels are easier to fit hyperparameters to; will still give good results with bad hyperparameters
- bad kernels give bad results unless the hyperparameters are perfect, which is hard!

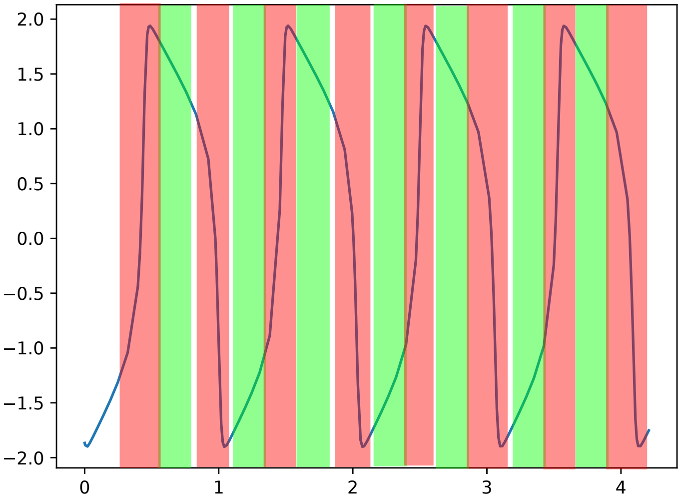
# Characteristic lengths



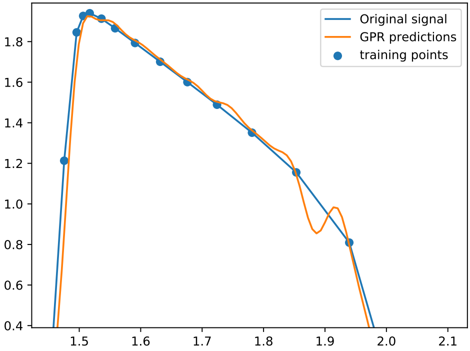

Bigger version on the next slide

- $l$ is the most interesting hyperparameter
- Measures how similar near-by datapoints are to each other
- Since neurons are a multiple-timescale system, this isn't trivial
- RED: points sampled close in time map to very different values, and are therefore dissimilar; small $l$
- GREEN: points sampled close in time map to similar values; big $l$

# Characteristic lengths



# The effects of $l$




- $l$ varies across the signal
- modelling with constant $l$ gives bad results

# Solution

There's kernels for modelling variable $l$, but…

- $l$ itself becomes a function of input variables
- No longer a single hyperparameter to fit, but an enitre hyperfunction
- Hyperparameter space goes from 3-dimensional to infinite!
- One approach models the $l$ function as a Gaussian process, and demonstrates an efficient / computationally tractable way of fitting it

# Generalised spectral mixture kernels

- Use GPR to generate a kernel for the specific input data
- Provides a tractable way of fitting this kernel
- Once fitted for one periodic orbit, it will still work well for the rest
- Automatically deals with periodicity, non-stationarity, so we resolve the periodic kernel dilemma!
- The paper is hard

Remes, Sami, Markus Heinonen, and Samuel Kaski. "Non-stationary spectral kernels." Advances in Neural Information Processing Systems. 2017.

# Next steps

- Work through the paper to understanding
  - Might take a while!
- Implement a GSMKernel
  - This should finish off the the GPR part
  - If GPR turns out to be a no-go, the rest of the predictor/corrector scheme will still work with another interpolating model, eg. periodic splines
- Code up a predictor
  - Should be trivial once GPR is sorted
- Code up a corrector
  - Should be interesting but very doable