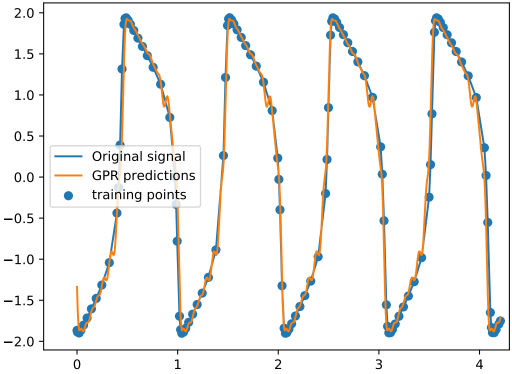
GPR Kernels

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

Model fitting – it sort of works!



30 seconds intro to Gaussian processes

- Gives us a Gaussian distribution at any given time
- That Gaussian distribution is the PDF of our function value at that time
- Works by maintaining a probability distribution over candidate functions
- Constructed by using Bayes' rule 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

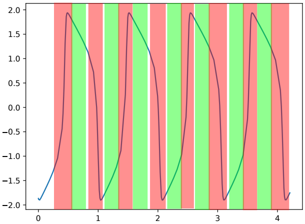
Both periodic and aperiodic kernels rely on hyperparameters; often…

- l: how similar nearby datapoints are
- σ_f^2: function amplitude
- σ_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!

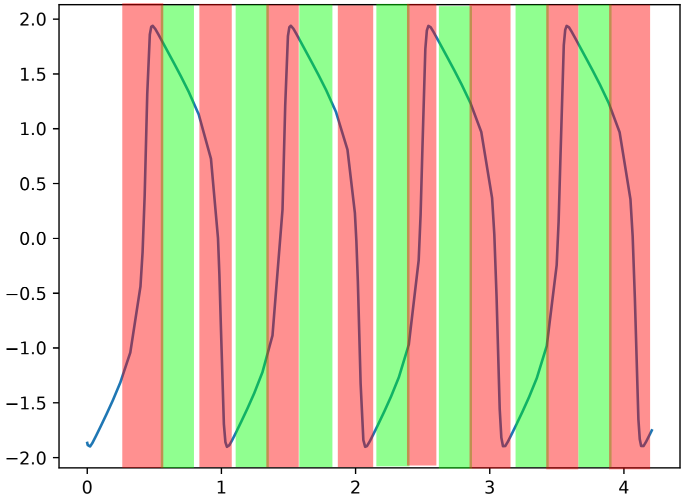
Characteristic lengths



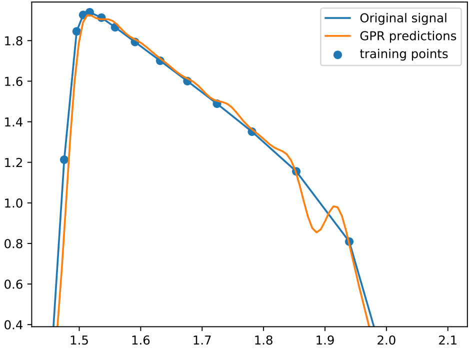
Bigger version on the next slide

- 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 becomes a function in space
- 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
- The paper is hard

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!

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