[2020-04-15 Wed]

In Silico CBC

Mark Blyth

Week's goal

- Redraft paper
- Work towards an *in silico* single-cell CBC experiment

Week’s activities

- Haven’t touched the paper yet
- Making steady progress towards CBC simulations
  - So far, the discretisation-free method looks like it will work
  - There’s some interesting problems for learning periodic orbit models, which have been my main focus
- Code committed so far is available on GitHub
  `https://github.com/MarkBlyth/SingleCellCBC`

Coding

Coded up various bits:

- Model class, for running parameterised simulations
- Controller class, for adding controllers onto models
- Simple PID, efficient PD, full state-feedback control schemes
- Multiple-input-single-output (MISO) GPR scheme
- Abstract kernel class and square-exponential kernel instance
- Two frequency estimation algorithms

Yet to code

- Multiple-input-multiple-output (MIMO) GPR
- Period windowing
- Hyperparameter optimisation
- Periodic orbit prediction, correction

MIMO GPR

- Either a MISO GPR for each output dimension, or...
- Rederive GPR equations for MIMO, and alter MISO GPR class for the new MIMO behaviours

Former works only if each output dimension is assumed to be statistically independent.

- There's python libraries for GPR, but hyperparameter optimisation is particularly important for this application, so it's going to be easier to just code up a custom one

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 $f_1(t), f_2(t), \ldots, t \in [0,1]$, 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 $f_1(t) = \sin(\frac{t}{2\pi})$, $f_2(t) = \sin(\frac{t}{2\pi} + 2\pi)$, $f_3(t) = \sin(\frac{t}{2\pi} + 4\pi), \ldots$, 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!

Period windowing (2)

Current windowing method:

1. Use autocorrelation methods to estimate fundamental frequency
2. Use nonlinear least squares to refine this estimate
3. Use the fundamental frequency estimate to partition data into cycles of period $1/f_0$

Issue:

- Fundamental freqency estimation is subject to fairly large numerical errors (c. 1%)
- Any numerical errors will cascade, so that we end up with $f_1(t+\phi), f_2(t+2\phi), \ldots$, which we can't accurately learn a model from

Period windowing (3)

Solution: since we can't accurately split up data, do it approximately and use the hyperparameter optimisation to refine it:

- Say data x_i was observed at time t'_i
- Rescale t' to $t = at' + b$, so that...
  - $t_i = 0$ when x_i is the first datapoint in the period
  - $t_i = 1$ when x_i is the last datapoint in the period
- For accurate f_0 estimation, let $T = 1/f_0$; then for the k'th period,
  - $a_k = 1/T = f_0$ and $b_k = kT = k/f_0$, so
  - $t = t'/T + kT$

But since the f_0 estimate isn't accurate, take these values of a_k, b_k as an initial estimate, then refine them by optimising alongside the other hyperparameters

Hyperparameter optimisation

- Log-marginal-likelihood, $p(y|X)$, gives the probability of seeing the outputs, given only the inputs
- Describes how well the class of model fits the data, independently of how well the model is actually fitted
- To optimise the hyperparameters, maximise this
  - For a SE kernel, hyperparameters are signal noise σ_n^2, signal variance σ_f^2, characteristic lengths (decorrelation distances) l, and windowing parameters a_k, b_k

We can leverage periodicity to force a faster fit, by maximising the performance index

$$K = p(y|X) - \lambda \|f(1) - f(0)\|^2$$

for fitted model f, where λ determines the significance of periodicity.

Gaussian process priors

- The GPR kernel (covariance function) encodes our prior assumptions about the model
- Better priors give better posteriors
- Since we know we're modelling a periodic function, we can build better models by encoding this information in the prior
- We want a kernel that expands into periodic basis functions
- I haven't read it yet, but Rasmussen ch.4 should contain enough information to figure out how to do this

Open problems

- Rederive GPR equations for MIMO, and implement
- Come up with an appropriate kernel for periodic functions
- Build a hyperparameter optimisation scheme
- Use it to implement periodic orbit segmentation
- Put it all together to make a nice periodic orbit learning scheme
- Use the learning scheme in a predictor-corrector

Next steps

- Finish the bits mentioned previously
- Write up some bits about how I did it and why (I'll forget otherwise!)
- Re-code it up in C++?
  - Mainly just an excuse for me to learn / practice C++ for scientific computing, but also...
  - C++ is faster and more efficient, which would be useful for speeding up actual experiments
  - Low-level language, which makes it easier to run on embedded devices
- Package everything up as a python library?
  - Written to be very general, extensible, and well-documented code, so this shouldn't be a difficult step
  - Might make things easier for other people to test out CBC ideas