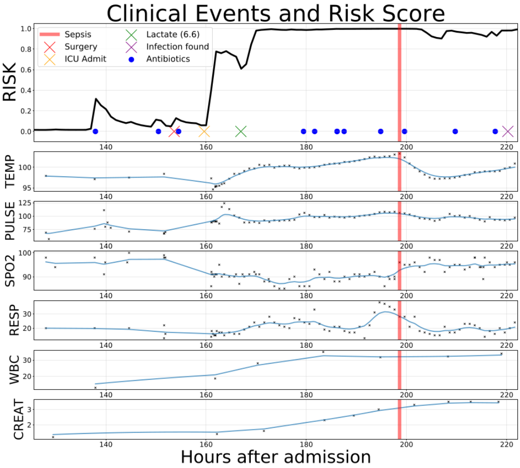
Tutorial: Machine Learning in Intensive Care Data Analysis

Lorenz Wernisch, Kevin Kunzmann

MRC-Biostatistics Unit Cambridge, UK

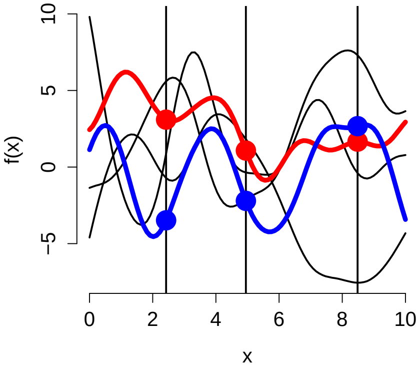
22 June 2018

Prediction in intensive care



Sepsis in ICU following cardiac surgery, J. Futoma et al., Improved Multi-Output Gaussian Process RNN with Real-Time Validation for Early Sepsis Detection, 2017

Gaussian process prior



Family of functions via covariance K on input points x

$y \sim N(0, K_{xx})$

Prediction for x^* from (x, y)

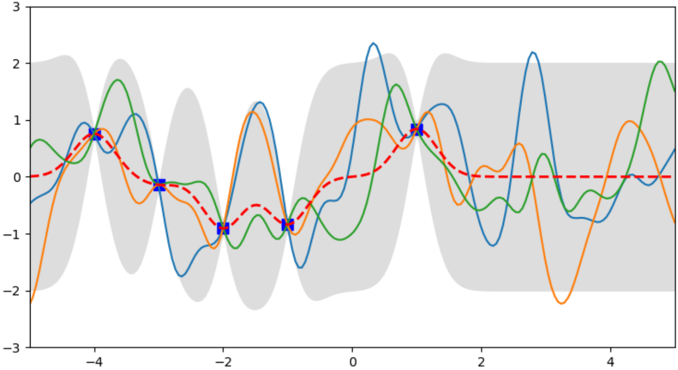
$y^* \sim N(K_{x^*x} K_{xx}^{-1} y, \Sigma)$

$\Sigma = K_{x^*x^*} - K_{x^*x} K_{xx}^{-1} K_{xx^*}$

Gaussian $\mathrm{cov}(x, x^*) = \theta_1 \exp(-\theta_2 (x - x^*)^2)$

Matern $\mathrm{cov}(x, x^*) = \theta_1 (1 + \theta_2 |x - x^*|) \exp(-\theta_2 |x - x^*|)$

GP uncertainty and samples

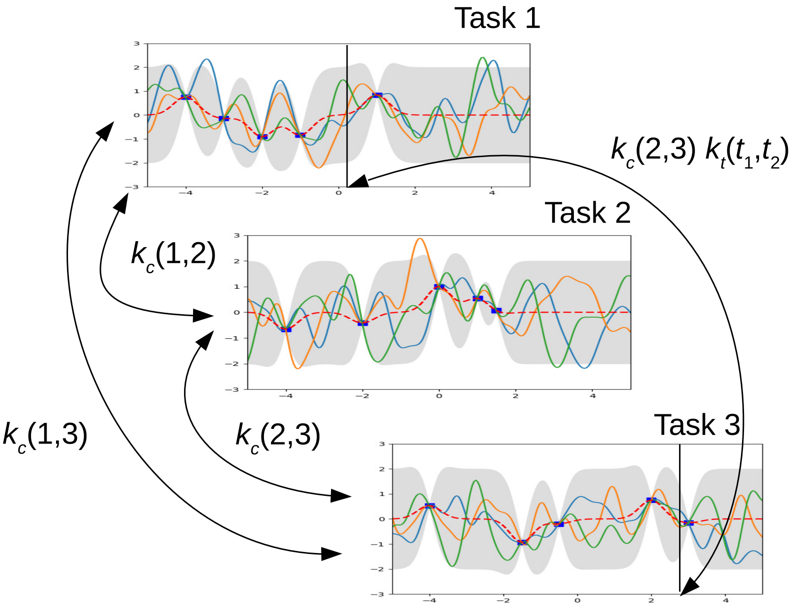


Regions of uncertainty, amount controlled by θ_1

Smoothness controlled by lengthscale $1/\theta_2$

Estimate by ML or MAP

Multitask GP



Kernels for multitask GP

Kernel for stacked time input vectors from each task

$$k_{\mathrm{MGP}}(l_1, l_2, t_1, t_2) = k_c(l_1, l_2)\, k_t(t_1, t_2)$$
$$K_{\mathrm{MGP}} = K_c(L, \theta_c) \otimes K_t(T, \theta_t)$$

Problem: same time parameters for all tasks

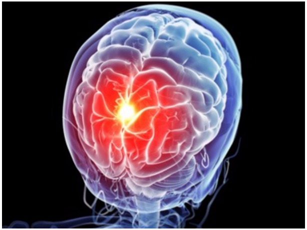
Compromise: convolution kernel between tasks

$$k(l_1, l_2, t_1, t_2) = \sqrt{\frac{2\theta_L^{(1)}\theta_L^{(2)}}{(\theta_L^{(1)})^2 + (\theta_L^{(2)})^2}}\, \exp\left(\frac{-(t_1 - t_2)^2}{(\theta_L^{(1)})^2 + (\theta_L^{(2)})^2}\right)$$

Kernel construction: must be positive semidefinite

Traumatic brain injury, ICU

CENTER-TBI consortium
David Menon

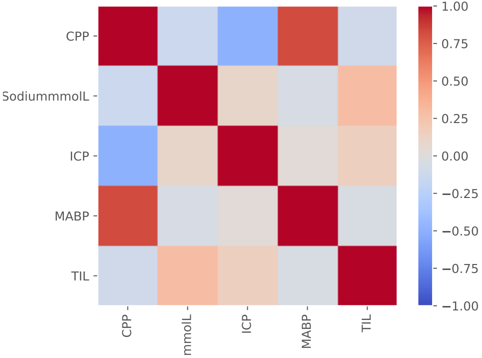


6000 patients: detailed medical records, follow up assessment, imaging data

1500 with ICU data:

Blood pressure (MAP)
Intracranial pressure (ICP)
Sodium lactate (infection)
Treatment intensity level score

Correlation component k_c of kernel



Cerebral perfusion pressure

Sodium DL lactate

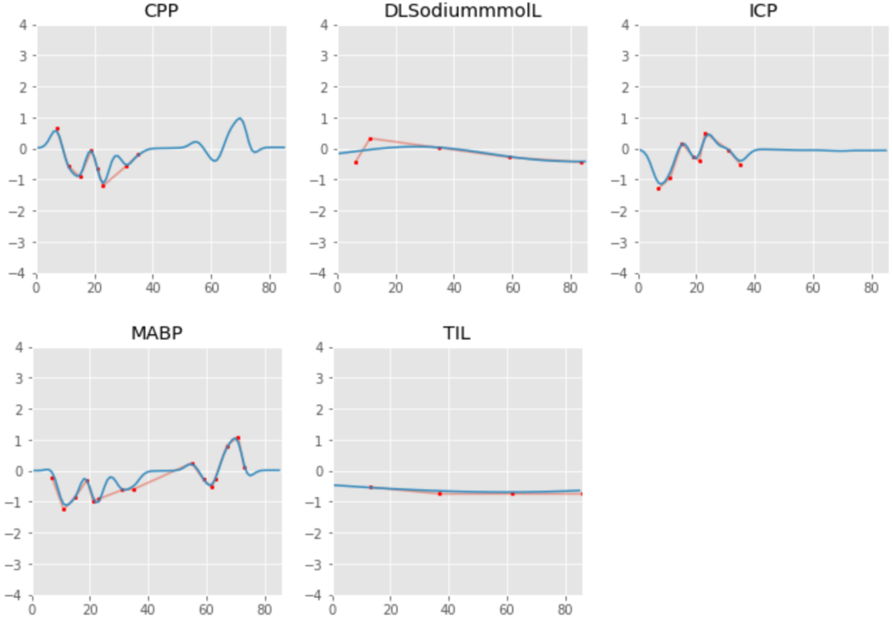
Intracranial pressure

Mean arterial pressure

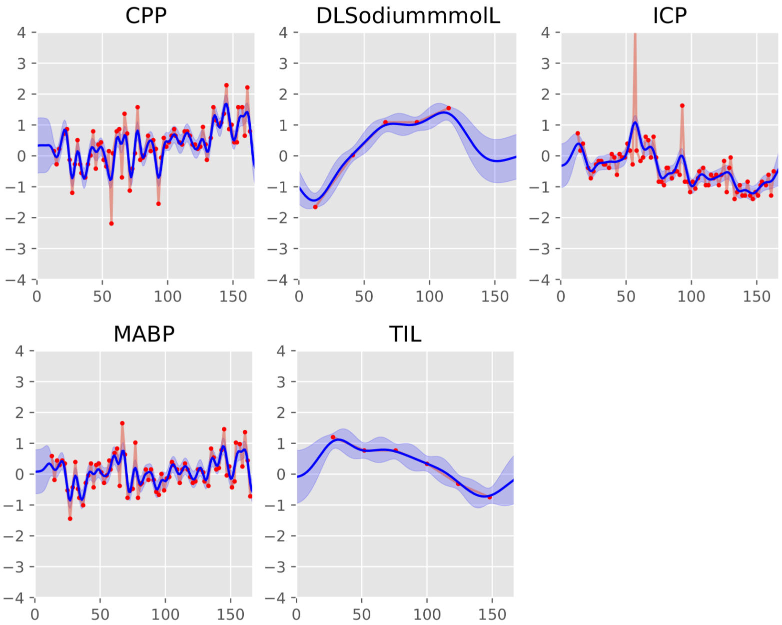
Treatment intensity level

Correlation sodium lactate - treatment intensity not seen in standard analysis

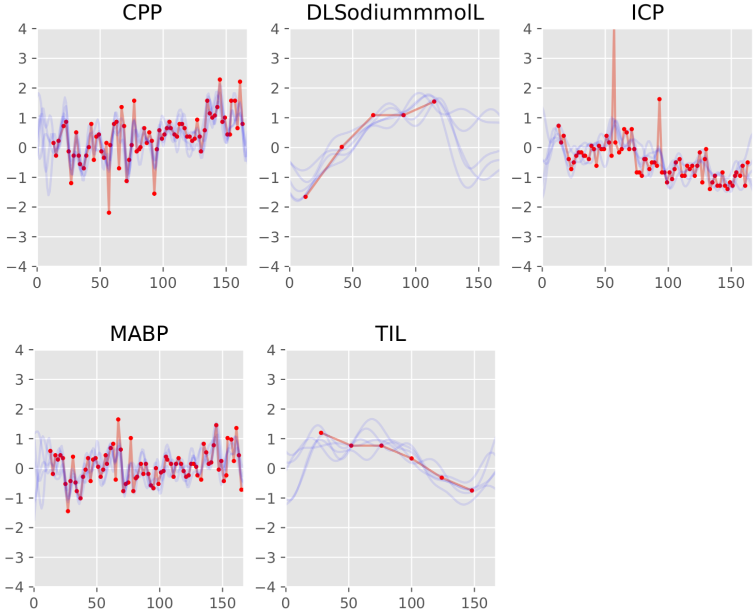
Effect of correlation in Multitask-GP



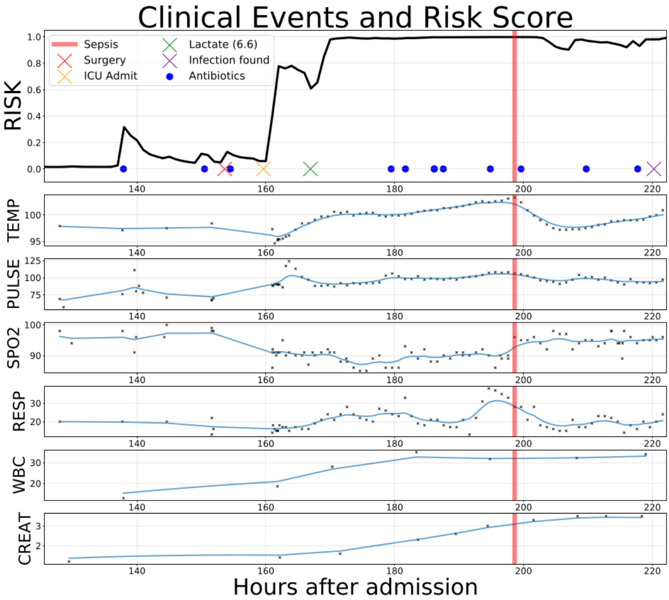
Posterior distribution of Multitask-GP



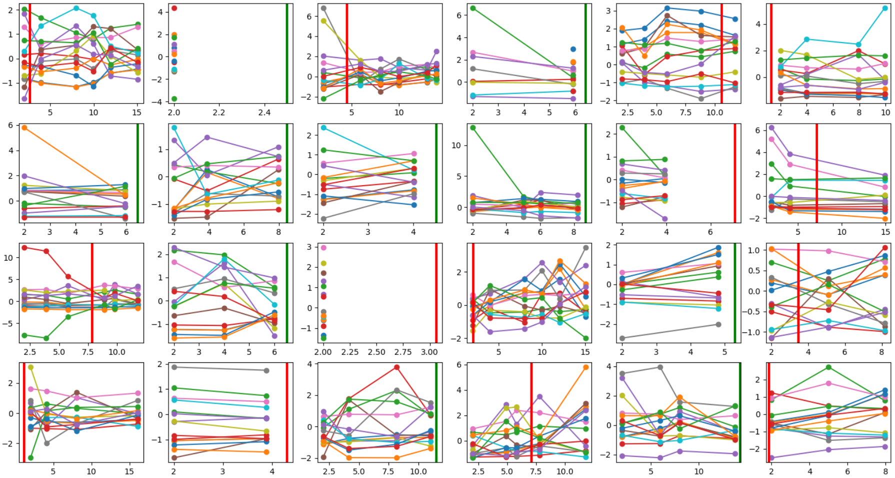
Posterior samples from Multitask-GP



Computing risk score from time series

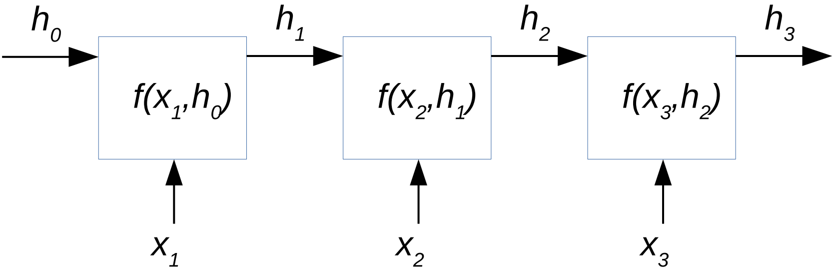


Secondary infection in ICU



Cellular markers for risk of secondary infection
Andrew Morris (School of Clinical Medicine)

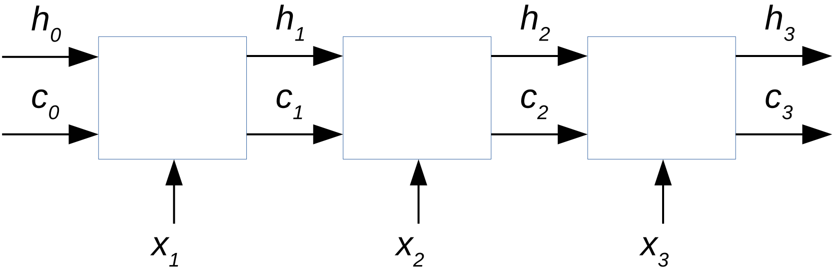
Recurrent neural network RNN



Autoregressive model fine for Markovian processes

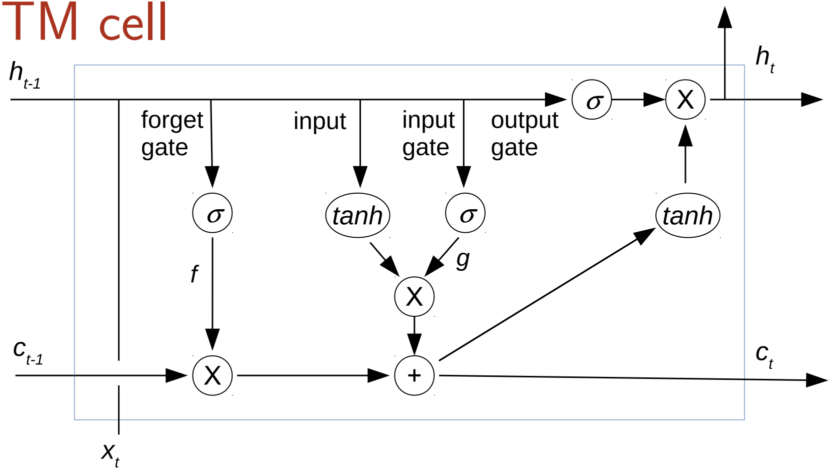
Problematic for long term effects:
Chinese whisper erosion

Long-short term memory LSTM RNN



In addition to latent state h_t carry cell state c_t protected from Chinese whisper erosion

LSTM cell

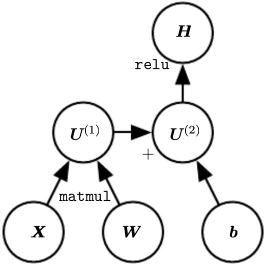


eg $f(h_{t-1}, x_t) = \sigma(b_0 + W_o x_t + V_0 h_{t-1})$

sigmoid $\sigma(x) \in [0, 1]$, $\tanh(x) \in [-1, 1]$

long term memory via internal state c_t and $+$

Computational graphs



TensorFlow works with *stateful dataflow graphs*

Algorithms for analysis of huge data sets (also Multicore, GPU)

Nodes represent eg: arithmetic operations, control clauses (if else), matrix manipulations, random number generators

Automatic differentiation, gradients easy to obtain
Efficient optimisation

Training the MGP-RNN

M-dimensional Multitask GP

Sample S trajectories from MGP: $x^{(s,i)} \in R^M$ for each patient i

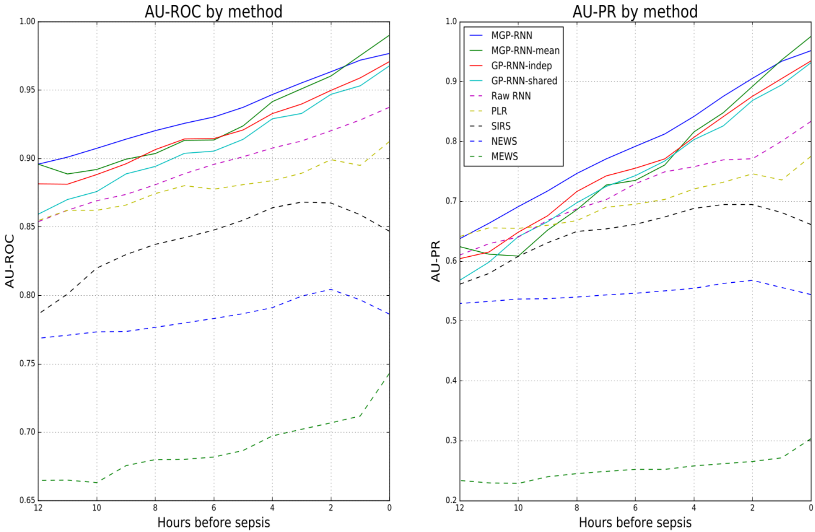
Loss compared to real outcome $y \in \{0, 1\}$:
$l(x, y) = y \log \mathrm{smax}(h_T) + (1-y) \log(1 - \mathrm{smax}(h_T))$

Minimize expected loss (over GP uncertainty):

$$L = \sum_i \frac{1}{S} \sum_s l(x^{(s,i)}, y^{(i)})$$

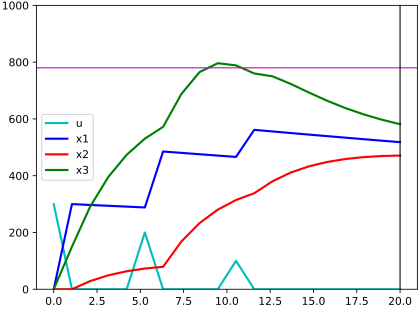
[softmax $\mathrm{smax}((h_0, h_1)) = e^{h_0}/(e^{h_0} + e^{h_1})$]

Comparison prediction of sepsis



J. Futoma et al., 2017, https://arxiv.org/pdf/1708.05894.pdf

Dynamic control

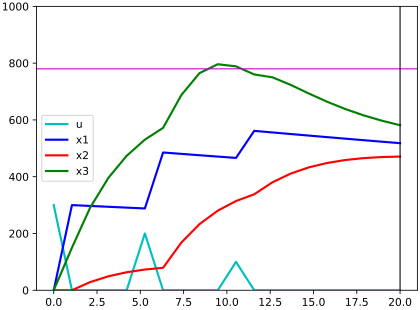


Inverventions u to move vital/lab measurements x_1, x_2, x_3 in certain direction: x_3 (green) to pink target

System unknown (black box)

Reinforcement learning:
(i) explore system, (ii) optimise towards desired outcome

Toy abstraction

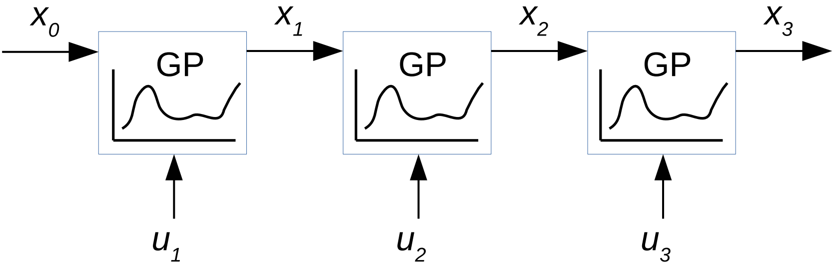


System of three variables $x_{1,t}, x_{2,t}, x_{3,t}$ measured daily over 20 days

We control input u_t to push $x_{3,20}$ to a target value on day 20

$$\begin{pmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{pmatrix} = \begin{pmatrix} x_{1,t-1} \\ x_{2,t-1} \\ x_{3,t-1} \end{pmatrix} + \begin{pmatrix} f_1(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \\ f_2(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \\ f_3(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \end{pmatrix}$$

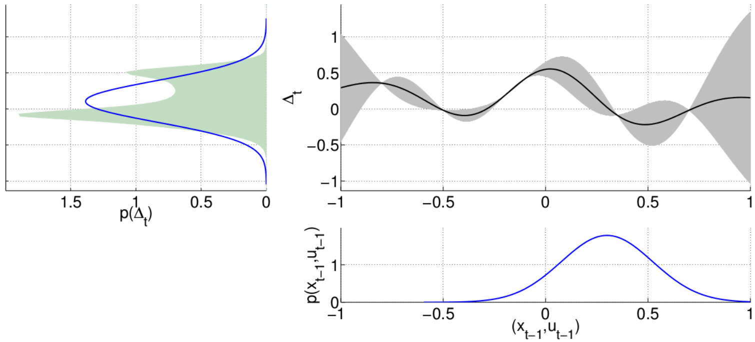
Reinforcement learning with GPs



x_t time series (ICU) data

Control input u_t to steer system in desired direction

Transmitting uncertainty



Gaussian uncertainty in *inputs*: non-Gaussian output

Pilco: approximate by Gaussian via moment matching
Deisenroth and Rasmussen, 2011

Optimize u using GP approximation

Minimize cost $c(u)$, eg for trajectory $x_3(u_0, \ldots, u_{19})$

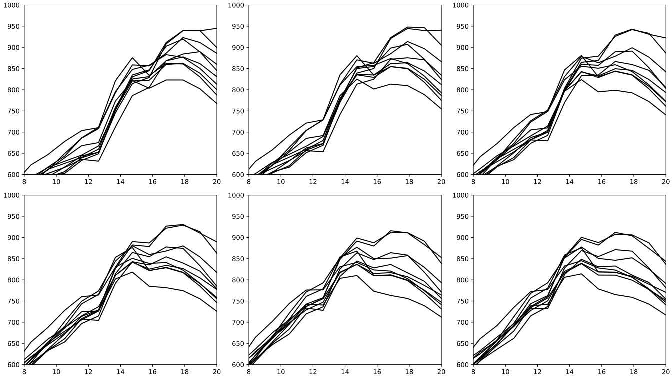
$$c(u) = (x_{3,20}(u_0, \ldots, u_{19}) - x_{\text{target}})^2 + \lambda \sum_t |u_t|$$

How to define trajectory using GPs?

Bad idea: use means of GP for $f_i(x_{t-1}, u_{t-1})$

Better idea: sample several *random* trajectories using GP uncertainty, minimise *expected* loss

Random trajectories



Expected loss $C(u) = 1/m \sum_k c(x^{(k)})$

Reparametrisation trick $p(x) = g(u, \epsilon)$:
$\nabla_u C(u) = 1/m \sum_k \nabla_u c(g(u, \epsilon_k))$
for a fixed sample $\epsilon_k \sim N(0, I)$

Reparameterisation for Gaussian

$p_N(z \mid \mu(u), \Sigma(u))$

Choleski factorisation $\Sigma(u) = C(u)C(u)^T$

With $\epsilon \sim N(0, I)$

$$z(u) = g(u, \epsilon) = g(\mu(u), \Sigma(u), \epsilon) = \mu(u) + C(u)\epsilon$$

$\mu(u), C(u)$: GPs trained on data and test input u

Dynamical system optimisation via GP

Reinforcement learning loop

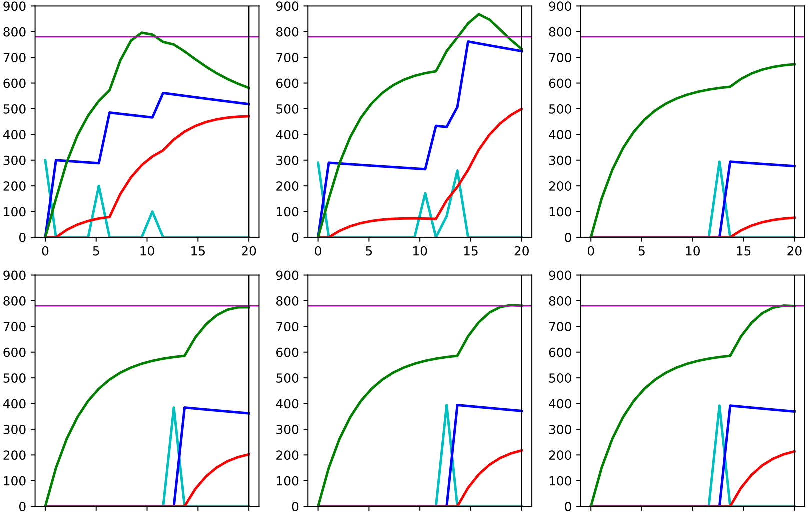
- Get experimental data with random control input
- Model data via recurrent GP
- Optimise u using expected loss
- Iterate

Online version: optimise policy $u_t = \phi_\omega(x_{t-1})$

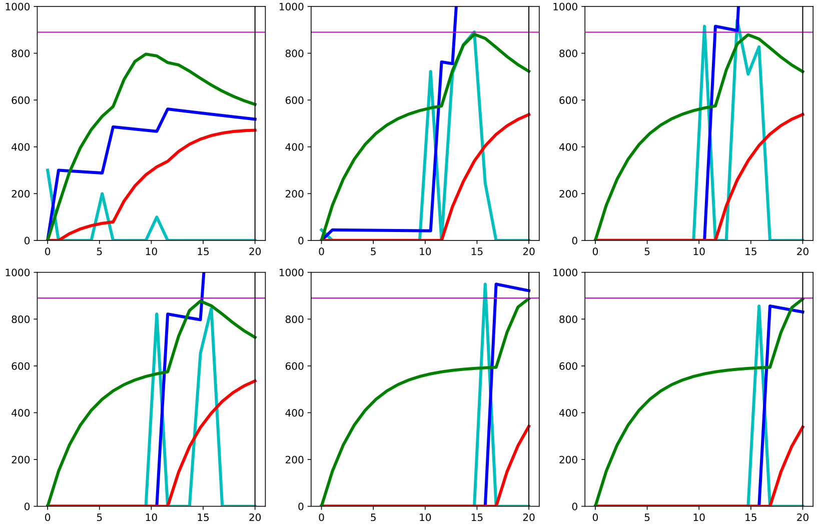
Markovian assumption

Advantage over RNN: structured GP kernel, incorporate uncertainty

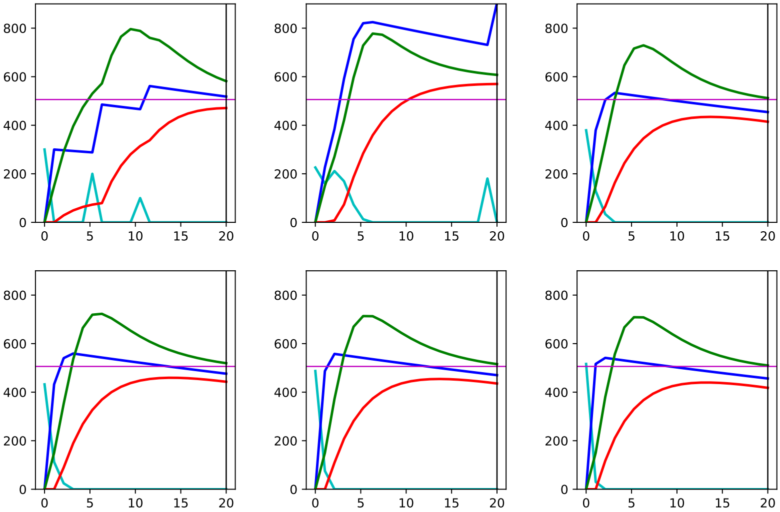
Aim: Green at 780 with few inputs



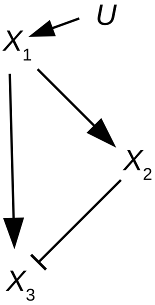
Zoom in: Green at maximum

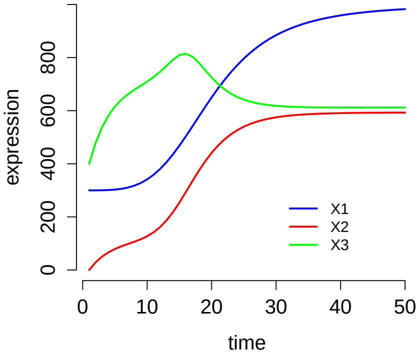


Aim: Green at minimum with few inputs



Incoherent or-feedforward loop





$$X_1(t) = (1 - \lambda_1)X_1(t-1) + U_{\mathrm{activate}}(t)$$
$$X_2(t) = (1 - \lambda_2)X_2(t-1) + h^{+}(X_1(t-1))$$
$$X_3(t) = (1 - \lambda_3)X_3(t-1) + h^{+}(X_1(t-1)) + h^{-}(X_2(t-1))$$

Thoughts

Machine learning algorithms: flexible, modular

Computational frameworks: very efficient, large data sets, support optimisation

Probabilistic modeling: ML frameworks allow Bayesian inference, probabilistic nodes, priors, HMC sampling

Gaussian processes: uncertainty, choice and flexibility in kernels, feature selection (ARD)

Intensive care: emphasis on control, real-time prediction to guide decisions