CS-E4075 Special course on Gaussian processes: Session #11 Dynamical models

Harri Lähdesmäki

Aalto University

harri.lahdesmaki@aalto.fi

Monday 15.02.2021

Outline

- Motivation
- Multi-output Gaussian processes
- Discrete-time dynamical GP models
    - Application 1: Model-based reinforcement learning with GPs
- Discrete-time dynamical GPs models: noisy data
- Continuous-time dynamical models with kernels and GPs

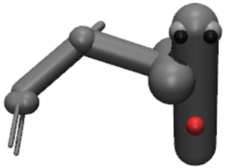
- Many real-world problems involve a dynamical system
- For many real-world systems the dynamics are unknown

→ We would like learn a dynamical system from observed data

- Dynamics of many real-world systems can be controlled

→ We would like to use our learned proxy dynamics to control the system

Robotics:



Video prediction:

Multi-output Gaussian processes

- In this lecture we will consider dynamical models in a high dimensional space ($d > 1$)
- Consider a multi-output (vector-valued) function $\mathbf{f} : \mathbb{R}^p \rightarrow \mathbb{R}^d$ where $d > 1$, denoted as

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_d(\mathbf{x}) \end{pmatrix}$$

- We denote that $\mathbf{f}$ follows a multi-output Gaussian process prior as

$$\mathbf{f}(\mathbf{x}) \sim GP(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta)),$$

  where $\boldsymbol{\mu}(\mathbf{x}) \in \mathbb{R}^d$ is the mean (which we assume as $\mathbf{0}$) and $\boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta)$ is a matrix-valued positive definite cross-covariance function (CCF)

- The (i, j) entry of the d-by-d matrix $\boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta)$ defines the covariance between the output dimensions (i, j) for any inputs $\mathbf{x}, \mathbf{x}'$

$$[\boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta)]_{i,j} = \mathrm{cov}(f_i(\mathbf{x}), f_j(\mathbf{x}')|\theta)$$

Multi-output Gaussian processes: CCFs

- The CCF can have a number of different structures
  - Independent
  - Implicit/Linear model of coregionalization
  - Full cross-covariance

- Here we assume multi-output GPs that factorize across the output dimensions, which is equivalent to diagonal CCF
$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta) = \mathrm{diag}\left(k_1(\mathbf{x}, \mathbf{x}'|\theta), \ldots, k_d(\mathbf{x}, \mathbf{x}'|\theta)\right)$$

- The dimension specific scalar-valued covariance functions $k_i(\mathbf{x}, \mathbf{x}'|\theta_i)$ can be any valid kernels and are often assumed to be shared across output dimensions

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta) = k(\mathbf{x}, \mathbf{x}'|\theta) \cdot I_d$$

- We assume (unless stated otherwise) the squared exponential (SE) covariance functions with input dimension specific length-scales (thus, $\theta = (\sigma_f, \ell_1, \ldots, \ell_d)$)

$$K(\mathbf{x}, \mathbf{x}'|\theta) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{d}\frac{(x_j - x_j')^2}{\ell_j^2}\right) \cdot I_d$$

Multi-output Gaussian processes: joint Gaussian

- Consider a finite collection of inputs $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$
- By the definition of GPs, the function values evaluated at X have a joint multivariate Gaussian distribution

$$\mathbf{f}(X) = \begin{pmatrix} \mathbf{f}(\mathbf{x}_1) \\ \vdots \\ \mathbf{f}(\mathbf{x}_N) \end{pmatrix} \in \mathbb{R}^{Nd} \quad \text{and} \quad p(\mathbf{f}(X)) = \mathcal{N}(\mathbf{f}(X) | \mathbf{0}, \boldsymbol{K}_{XX}(\theta))$$

where

$$\boldsymbol{K}_{XX}(\theta) = \begin{pmatrix} \boldsymbol{K}(\mathbf{x}_1, \mathbf{x}_1|\theta) & \boldsymbol{K}(\mathbf{x}_1, \mathbf{x}_2|\theta) & \cdots & \boldsymbol{K}(\mathbf{x}_1, \mathbf{x}_N|\theta) \\ \boldsymbol{K}(\mathbf{x}_2, \mathbf{x}_1|\theta) & \boldsymbol{K}(\mathbf{x}_2, \mathbf{x}_2|\theta) & \cdots & \boldsymbol{K}(\mathbf{x}_2, \mathbf{x}_N|\theta) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{K}(\mathbf{x}_N, \mathbf{x}_1|\theta) & \boldsymbol{K}(\mathbf{x}_N, \mathbf{x}_2|\theta) & \cdots & \boldsymbol{K}(\mathbf{x}_N, \mathbf{x}_N|\theta) \end{pmatrix} \in \mathbb{R}^{Nd \times Nd}$$

- For the multi-output GP prior that factorizes across dimensions, we can also write simply as

$$p(\mathbf{f}(X)) = \prod_{i=1}^{d} p(f_i(X))$$

Discrete-time dynamical models

- Consider a discrete-time, stochastic dynamical system with states $\mathbf{x}_t \in \mathbb{R}^d$, for $t = 0, 1, 2, \ldots$

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + \mathbf{w}_t,$$

  where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is an unknown transition function and $\mathbf{w}_t$ is the i.i.d. system noise $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$

- Alternatively we can write

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{f}(\mathbf{x}_t), \Sigma)$$

- The system is first-order Markovian

Discrete-time dynamical models

- Consider a discrete-time, stochastic dynamical system with states $\mathbf{x}_t \in \mathbb{R}^d$, for $t = 0, 1, 2, \ldots$

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + \mathbf{w}_t,$$

  where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is an unknown transition function and $\mathbf{w}_t$ is the i.i.d. system noise $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$

- Alternatively we can write

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{f}(\mathbf{x}_t), \Sigma)$$

- The system is first-order Markovian

- Assume a collection of N time-series trajectories of length $T+1$, $D = \{\mathbf{x}_{0:T}^{(n)}\}_{n=1}^{N}$, where the measurements are the true system states $\mathbf{x}$ without any measurement noise

- The data D can be presented as $N \cdot T$ state transition pairs $(\mathbf{x}_t^{(n)}, \mathbf{x}_{t+1}^{(n)})$ (or input-output pairs)

$$\begin{aligned}
X &= \left(\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_{T-1}^{(1)}, \ldots, \mathbf{x}_0^{(N)}, \ldots, \mathbf{x}_{T-1}^{(N)}\right) \in \mathbb{R}^{d \times NT} \\
\mathbf{y} &= \left(\mathbf{x}_1^{(1)T}, \ldots, \mathbf{x}_T^{(1)T}, \ldots, \mathbf{x}_1^{(N)T}, \ldots, \mathbf{x}_T^{(N)T}\right)^T \in \mathbb{R}^{dNT \times 1}
\end{aligned}$$

Discrete-time dynamical GP models

- We are interested in learning the underlying dynamical model which is completely unknown
  - We do not have a parametric form for $\mathbf{f}$
- We can assign the multi-output GP prior for $\mathbf{f}$

$$\mathbf{f}(\mathbf{x}) \sim GP(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta))$$

- We can learn an estimate of the unknown transition function from D by maximizing the marginal likelihood w.r.t. GP hyperparameters (and the system noise parameter Σ if unknown)

$$\ln p(\mathbf{y}|\theta) = -\frac{NT}{2} - \frac{1}{2}\ln|\boldsymbol{K}_{\mathbf{y}}| - \frac{1}{2}\mathbf{y}^T \boldsymbol{K}_{\mathbf{y}}^{-1}\mathbf{y},$$

where

$$\boldsymbol{K}_{\mathbf{y}} = \boldsymbol{K}_{XX}(\theta) + I_{NT} \otimes \Sigma$$

Discrete-time dynamical GP models

- We are interested in learning the underlying dynamical model which is completely unknown
  - We do not have a parametric form for **f**
- We can assign the multi-output GP prior for **f**

$$\mathbf{f}(\mathbf{x}) \sim GP(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta))$$

- We can learn an estimate of the unknown transition function from D by maximizing the marginal likelihood w.r.t. GP hyperparameters (and the system noise parameter Σ if unknown)

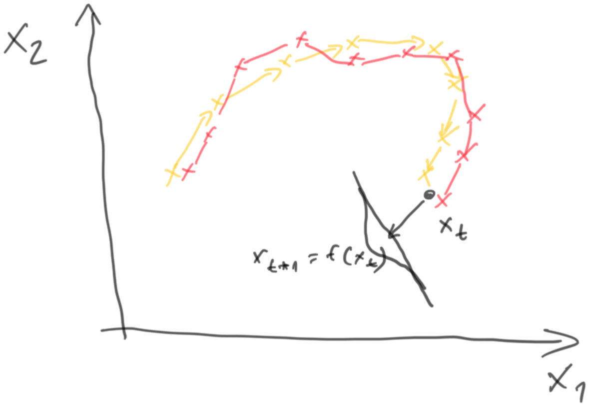
$$\ln p(\mathbf{y}|\theta) = -\frac{NT}{2} - \frac{1}{2}\ln|\boldsymbol{K}_{\mathbf{y}}| - \frac{1}{2}\mathbf{y}^T\boldsymbol{K}_{\mathbf{y}}^{-1}\mathbf{y},$$

  where

$$\boldsymbol{K}_{\mathbf{y}} = \boldsymbol{K}_{XX}(\theta) + I_{NT} \otimes \Sigma$$

- If the output dimensions of the GP are a priori independent, dimensions do not contain any shared parameters (i.e., $\boldsymbol{K}(\mathbf{x}, \mathbf{x}'|\theta) = \mathrm{diag}\,(k_1(\mathbf{x}, \mathbf{x}'|\theta_1), \ldots, k_d(\mathbf{x}, \mathbf{x}'|\theta_d))$), and the covariance of the system noise is also diagonal, then it can be seen that the learning factorizes across dimensions

Discrete-time dynamical GP models: illustration



Discrete-time dynamical GP models: predictions

- After learning the dynamics model with $(X, \mathbf{y})$, the standard GP predictive distributions can be used to compute the transition function prediction for a new input state $\mathbf{x}^*$

$$p(\mathbf{f}(\mathbf{x}^*)|X, \mathbf{y}, \mathbf{x}^*) = \mathcal{N}(\mathbf{f}(\mathbf{x}^*)|\boldsymbol{\mu}(\mathbf{x}^*), \Sigma(\mathbf{x}^*)),$$

where $\mu(\mathbf{x}^*)$ and $\Sigma(\mathbf{x}^*)$ are the standard prediction equations

$$\begin{aligned} \boldsymbol{\mu}(\mathbf{x}^*) &= \boldsymbol{K}_{\mathbf{x}^* X} \boldsymbol{K}_{\mathbf{y}}^{-1} \mathbf{y} \\ \Sigma(\mathbf{x}^*) &= \boldsymbol{K}_{\mathbf{x}^* \mathbf{x}^*} - \boldsymbol{K}_{\mathbf{x}^* X} \boldsymbol{K}_{\mathbf{y}}^{-1} \boldsymbol{K}_{X \mathbf{x}^*} \end{aligned}$$

- Thus, given a state at time t, $\mathbf{x}_t$, our estimate of the one time-step prediction $p(\mathbf{x}_{t+1}|X, \mathbf{y}, \mathbf{x}_t)$ is obtained by combining the above equations with the system noise $\mathbf{w}_t$:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = p(\mathbf{x}_{t+1}|X, \mathbf{y}, \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+1}|\boldsymbol{\mu}(\mathbf{x}_t), \Sigma(\mathbf{x}_t) + \Sigma)$$

Discrete-time dynamical GP models: long-term predictions

- Given a state distribution at time t, $p(\mathbf{x}_t)$, we are often interested in making long-term predictions for the state evolution $\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+H}$
- Given $p(\mathbf{x}_t)$, the prediction equation for a single time step can be written

$$p(\mathbf{x}_{t+1}) = \int p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{x}_t)d\mathbf{x}_t,$$

  where $p(\mathbf{x}_{t+1}|\mathbf{x}_t) = p(\mathbf{x}_{t+1}|X, \mathbf{y}, \mathbf{x}_t)$
- For long-term predictions $p(\mathbf{x}_{t+1}), \ldots, p(\mathbf{x}_{t+H})$, we can iteratively make one time-step predictions
- The above integral cannot be solved analytically but can be approximated by Monte Carlo sampling
  - Draw $\mathbf{x}_{t+1}$ from $p(\mathbf{x}_{t+1})$, then with $\mathbf{x}_{t+1}$ fixed draw $p(\mathbf{x}_{t+2}|X, \mathbf{y}, \mathbf{x}_{t+1})$, etc.
  - This will give a realization from $p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots, \mathbf{x}_{t+H})$

Discrete-time dynamical GP models: prediction illustration

- An illustration of long term predictions
- Uncertainty accumulates

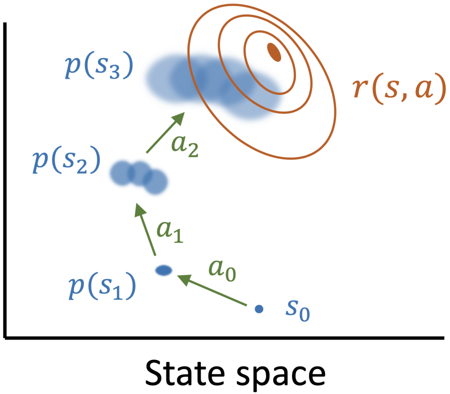


Figure from (Gadd et al, 2021)

Discrete-time dynamical GP models: alternative model variants

- Deterministic dynamic model

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t)$$

- Time differential model

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t) + \mathbf{w}_t,$$

  where the GP prior for **f** now has a zero mean

- Time differential model with irregular sampling times $(t_0, t_1, \ldots, t_N)$ (also called gradient matching)

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \Delta t_i \cdot \mathbf{f}(\mathbf{x}(t_i)) + \mathbf{w}(t_i),$$

  where $\Delta t_i = t_{i+1} - t_i$

Discrete-time dynamical GP models: alternative model variants

- Deterministic dynamic model

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t)$$

- Time differential model

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t) + \mathbf{w}_t,$$

  where the GP prior for **f** now has a zero mean

- Time differential model with irregular sampling times $(t_0, t_1, \ldots, t_N)$ (also called gradient matching)

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \Delta t_i \cdot \mathbf{f}(\mathbf{x}(t_i)) + \mathbf{w}(t_i),$$

  where $\Delta t_i = t_{i+1} - t_i$

- Control model with an external control variate $\mathbf{a}_t \in \mathbb{R}^k$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t, \mathbf{a}_t) + \mathbf{w}_t,$$

  where $\mathbf{f} : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$

- Deep GP model

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \underbrace{\mathbf{f}_L \circ \mathbf{f}_{L-1} \ldots \mathbf{f}_1(\mathbf{x}_t)}_{\text{deep GP with } L \text{ layers}} + \mathbf{w}_t$$

- Higher-order models

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t, \ldots, \mathbf{x}_{t-l}) + \mathbf{w}_t,$$

Application 1: Model-based reinforcement learning with GPs

- Reinforcement learning (RL) provides a principled framework for data-driven autonomous learning for control and sequencial decision making
- Through trial-and-error, controls are chosen with the goal of completing a task or maximizing a pre-defined objective
- An example: learn to control a robot via trial-and-error to make the robot to complete a task

Application 1: Model-based reinforcement learning with GPs

- Reinforcement learning (RL) provides a principled framework for data-driven autonomous learning for control and sequencial decision making
- Through trial-and-error, controls are chosen with the goal of completing a task or maximizing a pre-defined objective
- An example: learn to control a robot via trial-and-error to make the robot to complete a task
- Standard RL methods are typically implemented with neural networks that require lots of trial-and-errors to complete a task
- Model-based reinforcement learning (MBRL) provides a solution to achieve sample efficiency, i.e., learn an accurate model / complete a task with little data

$\rightarrow$ MBRL with GPs

MBRL with GPs: learning a dynamics model / emulator

- Assume the control model with an external control variate $\mathbf{a}_t \in \mathbb{R}^k$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t, \mathbf{a}_t) + \mathbf{w}_t$$

  and the GP prior for $\mathbf{f} : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$

- An estimate of the Markovian transition function **f** emulates the real-world system and can be used to predict the outcome / next state given the current state $\mathbf{x}_t$ and control action $\mathbf{a}_t$

MBRL with GPs: learning a dynamics model / emulator

- Assume the control model with an external control variate $\mathbf{a}_t \in \mathbb{R}^k$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t, \mathbf{a}_t) + \mathbf{w}_t$$

  and the GP prior for $\mathbf{f} : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$

- An estimate of the Markovian transition function **f** emulates the real-world system and can be used to predict the outcome / next state given the current state $\mathbf{x}_t$ and control action $\mathbf{a}_t$
- Assume that noise-free data has been collected from the real-world system, which can be presented as a collection of triplets (or input $\mathbf{x}_t, \mathbf{a}_t$ and output $\mathbf{x}_{t+1}$ pairs)

$$D = \{(\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1})\}_{t=0}^{T}$$

- The discrete-time dynamical GP model can be learned again by maximizing the marginal likelihood as described abnove

MBRL with GPs: objective function

- In RL setting we typically have a pre-defined reward function $r(\mathbf{x}, \mathbf{a})$
  - Reward is high for that part of the state space where we want the system to end-up, and low elsewhere
  - Reward may also penalize large control actions
- For brevity, denote the H-step ahead control actions as $(\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H}) = \mathbf{a}_{t:t+H}$
- The objective function can be defined as the expected reward over a time horizon H

$$R(\mathbf{a}_{t:t+H}) = \sum_{\tau=t}^{t+H} \int r(\mathbf{x}_{\tau+1}, \mathbf{a}_\tau) p(\mathbf{x}_{\tau+1} | \mathbf{x}_\tau, \mathbf{a}_\tau) d\mathbf{x}_{\tau+1},$$

where the prediction equation $p(\mathbf{x}_{\tau+1} | \mathbf{x}_\tau, \mathbf{a}_\tau)$ is now augmented with the control $\mathbf{a}_\tau$

MBRL with GPs: objective function

- In RL setting we typically have a pre-defined reward function $r(\mathbf{x}, \mathbf{a})$
  - Reward is high for that part of the state space where we want the system to end-up, and low elsewhere
  - Reward may also penalize large control actions
- For brevity, denote the H-step ahead control actions as $(\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H}) = \mathbf{a}_{t:t+H}$
- The objective function can be defined as the expected reward over a time horizon H

$$R(\mathbf{a}_{t:t+H}) = \sum_{\tau=t}^{t+H} \int r(\mathbf{x}_{\tau+1}, \mathbf{a}_\tau) p(\mathbf{x}_{\tau+1} | \mathbf{x}_\tau, \mathbf{a}_\tau) d\mathbf{x}_{\tau+1},$$

  where the prediction equation $p(\mathbf{x}_{\tau+1} | \mathbf{x}_\tau, \mathbf{a}_\tau)$ is now augmented with the control $\mathbf{a}_\tau$
- The integral cannot be solved in closed-form but can be approximated by sampling trajectories (=long-term predictions) from the dynamics model $\mathbf{f}$ with Monte Carlo
- The goal is to choose control actions $\mathbf{a}_{t:t+H}$ so that the objective function over a time horizon H is maximized

$$\hat{\mathbf{a}}_{t:t+H} = \arg\max R(\mathbf{a}_{t:t+H})$$

MBRL with GPs: illustration

- An illustration of MBRL with GPs: maximization of the reward / objective function

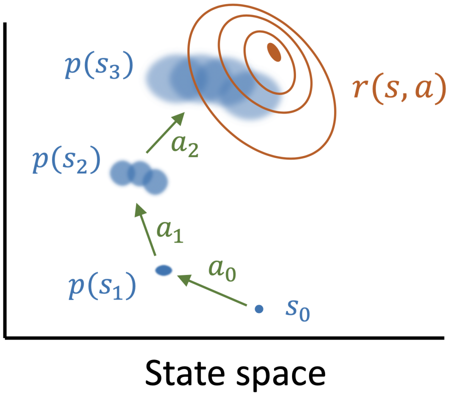


Figure from (Gadd et al, 2021)

MBRL with GPs: open-loop control

- How to find an optimal action sequence $\hat{\mathbf{a}}_{t:t+H}$?
- Cross-entropy method
  1. Initialize $\mathbf{m}$ and $\mathbf{v}$
  2. Sample $k = 1, \ldots, K$ action sequences
  $$\mathbf{a}_{t:t+H}^{(k)} \sim \mathcal{N}(\mathbf{m}, \operatorname{diag} \mathbf{v})$$
  3. For each action sequence $\mathbf{a}_{t:t+H}^{(k)}$ sample $p = 1, \ldots, P$ trajectories $\mathbf{x}_{t+1}^{(k,p)}, \ldots, \mathbf{x}_{t+H+1}^{(k,p)}$ using the GP surrogate of the dynamics model $\mathbf{f}$
  4. For each action sequence approximate the expected long term reward over horizon H as
  $$\hat{R}(\mathbf{a}_{t:t+H}^{(k)}) = \sum_{\tau=t}^{t+H} \frac{1}{P} \sum_{p=1}^{P} r(\mathbf{x}_{\tau+1}^{(k,p)}, \mathbf{a}_{\tau}^{(k,p)})$$
  5. Return the current best action sequence, or update $\mathbf{m}$ and $\mathbf{v}$ using the mean and variance of the top performing action sequences and go back to step 2

MBRL with GPs: model predictive control with feedback

- MBRL with GPs is typically implemented as a model predictive control (MPC) method with feedback (closed-loop) control
- Once the optimal control sequence at time t is found $\hat{\mathbf{a}}_{t:t+H}$, only the first control action $\hat{\mathbf{a}}_t$ is applied to the real-world system
- The system transitions from the current state $\mathbf{x}_t$ to a state $\mathbf{x}_{t+1}$
- After a single step ahead we update the data $D \leftarrow D \cup \{(\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1})\}$ and re-train the GP dynamics model
- The MPC then re-optimizes the actions $\hat{\mathbf{a}}_{t+1:t+H+1}$ and this closed-loop MPC continues by iteratively learning dynamics and re-optimizing the action sequence

MBRL with GPs: CartPole example

- CartPole example (with a wall on right)
- Measurements of the system state include
  - Position and velocity of the cart
  - Angle and angular velocity of the pole
- Desired states are on right with the pole at up-right position
- Visualizations of the GPs based learning at an early and a later learning stage

Figure from (Gadd et al, 2021)

MBRL with GPs: CartPole example (2)

- Comparison of shallow and deep GP model variants with $L = 1, \ldots, 3$ on the (modified) CartPole example

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \underbrace{\mathbf{f}_L \circ \mathbf{f}_{L-1} \ldots \mathbf{f}_1(\mathbf{x}_t, \mathbf{a}_t)}_{\text{deep GP with } L \text{ layers}} + \mathbf{w}_t$$

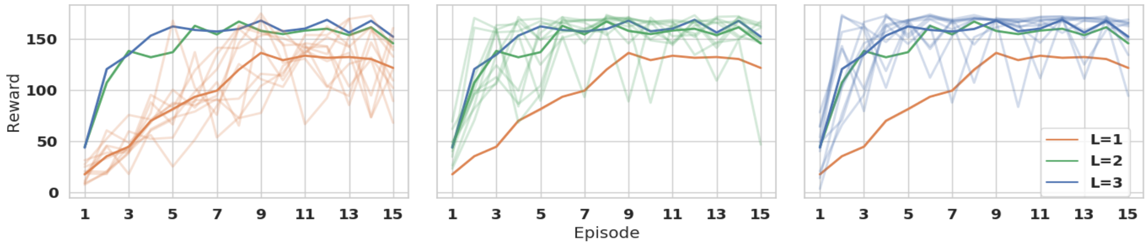


Figure from (Gadd et al, 2021)

Dynamical GP models with explicit basis functions

Application 2: Gene regulatory network inference with GPs

Discrete-time dynamical GP models: noisy data

- Previous models / methods assumed noise-free data
- If the states $\mathbf{x}_t$ cannot be measured exactly, then our model should account for measurement uncertainty: e.g. $\mathbf{y}_t = \mathbf{x}_t + \mathbf{n}_t$, where $\mathbf{n}_t$ denotes additive measurement noise, e.g. $\mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{y}}^2 I_d)$

Discrete-time dynamical GP models: noisy data

- Previous models / methods assumed noise-free data
- If the states $\mathbf{x}_t$ cannot be measured exactly, then our model should account for measurement uncertainty: e.g. $\mathbf{y}_t = \mathbf{x}_t + \mathbf{n}_t$, where $\mathbf{n}_t$ denotes additive measurement noise, e.g. $\mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{y}}^2 I_d)$
- More generally, we can consider a model where the dynamics $\mathbf{x}_t$ are embedded in a low-dimensional latent space and possibly high-dimensional observations $\mathbf{y}_t$ are conditional on $\mathbf{x}_t$

$$
\begin{aligned}
\mathbf{x}_t &= \mathbf{f}(\mathbf{x}_{t-1}) + \mathbf{w}_t \\
\mathbf{y}_t &= \mathbf{g}(\mathbf{x}_t) + \mathbf{n}_t,
\end{aligned}
$$

  where $\mathbf{g} : \mathbb{R}^d \rightarrow \mathbb{R}^D$ and $\mathbf{n}_t \sim N(\mathbf{0}, \sigma_{\mathbf{y}}^2 I_D)$

- Traditional auto-regressive methods assume linear mappings for $\mathbf{f}$ and $\mathbf{g}$
- If the underlying model is non-linear but unknown, we can assign multi-output GP prior for both $\mathbf{f}$ and $\mathbf{g}$

$\rightarrow$ We retrieve a model that is called Gaussian process dynamical model (GPDM) (Wang et al., 2005)

Gaussian process dynamical model: inference

- Gaussian process dynamical model (GPDM) can be understood as a GPLVM model where the latent variables evolve according to discrete-time dynamical GP model
- Fitting the GPDM involves simultaneously solving the GPLVM as well as inferring smooth dynamical GP model for the latent GPLVM embeddings

Gaussian process dynamical model: inference

- Gaussian process dynamical model (GPDM) can be understood as a GPLVM model where the latent variables evolve according to discrete-time dynamical GP model
- Fitting the GPDM involves simultaneously solving the GPLVM as well as inferring smooth dynamical GP model for the latent GPLVM embeddings
- Lets denote data $\boldsymbol{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_T]^T$, latent embedding $\boldsymbol{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]^T$, and hyperparameters of the latent GP ($\mathbf{f}$) and embedding GP ($\mathbf{g}$) as $\theta_\mathbf{f}$ and $\theta_\mathbf{g}$
- The learning involves maximizing

$$p(\boldsymbol{X}, \theta_\mathbf{f}, \theta_\mathbf{g} | \boldsymbol{Y}) \propto \underbrace{p(\boldsymbol{Y} | \boldsymbol{X}, \theta_\mathbf{g})}_{\text{latent embedding}} \underbrace{p(\boldsymbol{X} | \theta_\mathbf{f})}_{\text{dynamics}} p(\theta_\mathbf{f}) p(\theta_\mathbf{g})$$

- Derivation of the exact expression for $p(\boldsymbol{X}, \theta_\mathbf{f}, \theta_\mathbf{g} | \boldsymbol{Y})$ is similar with the derivation of the GPLVM (see (Wang et al., 2005) for details)

Gaussian process dynamical model: comparison to other methods

- An illustration of the GPDM and comparison against other methods

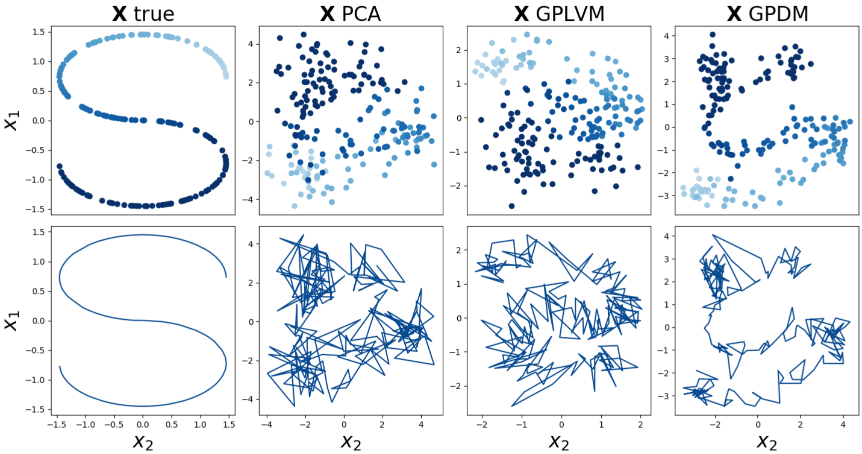


Figure from http://gregorygundersen.com/blog/2020/07/24/gpdm/

Gaussian process dynamical model: illustration on CMU mocap data

- An illustration of the GPDM on the high-dimensional CMU mocap walking data

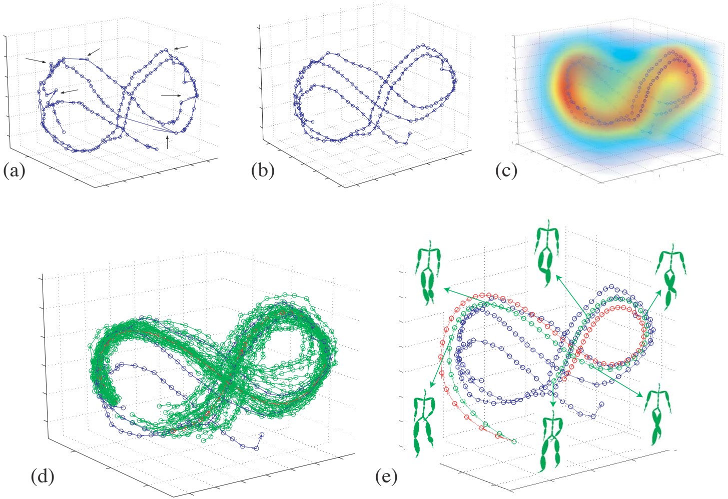


Figure from (Wang et al., 2005)

Updated motivation

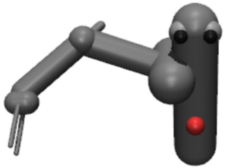
- Many real-world problems involve a continuous-time dynamical system
- For many real-world systems the dynamics are unknown

→ We would like learn a continuous-time dynamical system from observed data

- Dynamics of many real-world systems can be controlled

→ We would like to use our learned continuous-time proxy dynamics to control the system

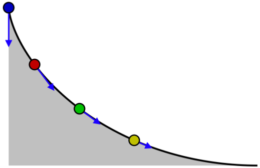
Robotics:



Video prediction:

Updated motivation

Some ODE models can be built from first principles (e.g. Leibniz at 1690)

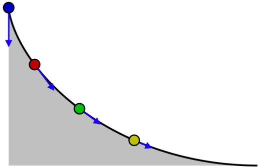


$$y(s) = s^2$$
$$dy^2 = 4y(dx^2 + dy^2)$$
$$\frac{dx}{dy} = \frac{\sqrt{1-4y}}{2\sqrt{y}}$$

https://en.wikipedia.org/
wiki/File:Tautochrone_curve.gif

Updated motivation

Some ODE models can be built from first principles (e.g. Leibniz at 1690)



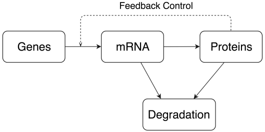
$$y(s) = s^2$$

$$dy^2 = 4y(dx^2 + dy^2)$$

$$\frac{dx}{dy} = \frac{\sqrt{1-4y}}{2\sqrt{y}}$$

https://en.wikipedia.org/wiki/File:Tautochrone_curve.gif

Gene regulatory network models

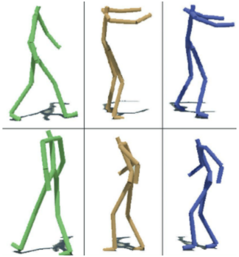


$$dr/dt = f(p) - Vr$$

$$dp/dt = Lr - Up$$

$$f(p) = f(p_0) + \frac{df(p)}{dp}|_{p_0}(p - p_0)$$

Motion capture (100's of joints)



Video prediction (1M pixels over 100K frames)

Ordinary differential equations (ODEs)

- An ordinary differential equation (ODE) system defined by differential field / drift function

$$\frac{d\mathbf{x}_t}{dt} := \dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t)$$

where

$$\dot{\mathbf{x}}_t, \mathbf{x}_t \in \mathbb{R}^D, \qquad \mathbf{f} : \mathbb{R}^D \to \mathbb{R}^D$$

Ordinary differential equations (ODEs)

- An ordinary differential equation (ODE) system defined by differential field / drift function

$$\frac{d\mathbf{x}_t}{dt} := \dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t)$$

  where

$$\dot{\mathbf{x}}_t, \mathbf{x}_t \in \mathbb{R}^D, \qquad \mathbf{f} : \mathbb{R}^D \to \mathbb{R}^D$$

- Given an initial state $\mathbf{x}_0$ and (possible) parameters θ, ODE solution $\mathbf{x}_t := \mathbf{x}(t, \theta, \mathbf{x}_0)$ indexed by $t \in \mathcal{T} = \mathbb{R}_+$ is

$$\mathbf{x}_t = \mathbf{x}_0 + \int_0^t \dot{\mathbf{x}}_\tau d\tau = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}_\tau) d\tau$$

- Interested in cases where $\boldsymbol{f}$ is completely unknown and is to be estimated from noisy data

Black-box ODEs

- We are interested in cases where $\boldsymbol{f}$ is completely unknown and we are given only noisy observations at $T = (t_1, \ldots, t_N)$:

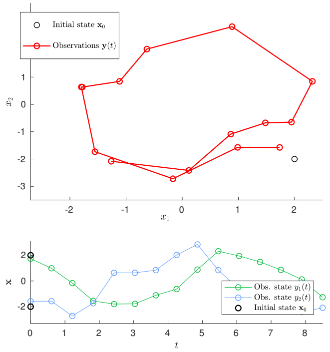
$$
\begin{aligned}
\boldsymbol{y}_t &= \boldsymbol{x}_t + \varepsilon_t \\
\varepsilon_t &\sim \mathcal{N}(\mathbf{0}, \Omega) \\
\Omega &= \mathrm{diag}(\omega_1^2, \ldots, \omega_D^2)
\end{aligned}
$$

Black-box ODEs

- We are interested in cases where $\boldsymbol{f}$ is completely unknown and we are given only noisy observations at $T = (t_1, \ldots, t_N)$:

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{x}_t + \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(\mathbf{0}, \Omega) \\ \Omega &= \operatorname{diag}(\omega_1^2, \ldots, \omega_D^2) \end{aligned}$$

- Input data

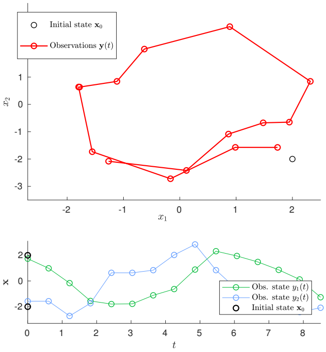


Black-box ODEs

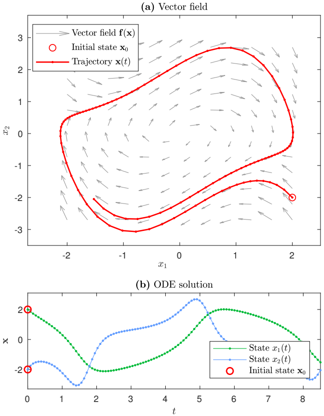
- We are interested in cases where $\boldsymbol{f}$ is completely unknown and we are given only noisy observations at $T = (t_1, \ldots, t_N)$:

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{x}_t + \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(\mathbf{0}, \Omega) \\ \Omega &= \mathrm{diag}(\omega_1^2, \ldots, \omega_D^2) \end{aligned}$$

- Input data



- Inference / The true system



Nonparametric ODE (npODE) Model (Heinonen et al., 2018)

- As before, we set a vector-valued Gaussian process (GP) prior over the D-dimensional vector field

$$\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{GP}(\mathbf{0}, K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')), \qquad K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{D}\frac{(x_j - x_j')^2}{\ell_j^2}\right) \cdot I_D$$

with kernel parameters $\boldsymbol{\theta} = (\sigma_f, \ell_1, \ldots, \ell_D)$ that defines prior mean and covariance

$$\begin{aligned} \mathbb{E}[\mathbf{f}(\mathbf{x})] &= \mathbf{0} \\ \mathrm{cov}[\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')] &= K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') \end{aligned}$$

Nonparametric ODE (npODE) Model (Heinonen et al., 2018)

- As before, we set a vector-valued Gaussian process (GP) prior over the D-dimensional vector field

$$\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{GP}(\mathbf{0}, K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')), \qquad K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{D} \frac{(x_j - x_j')^2}{\ell_j^2}\right) \cdot I_D$$

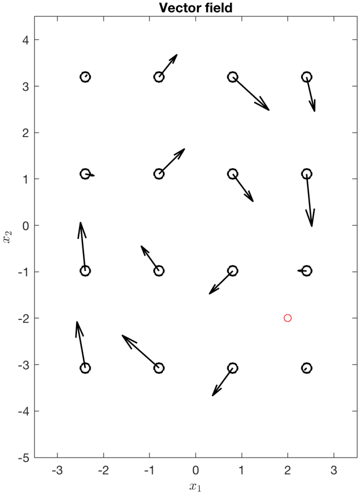
with kernel parameters $\boldsymbol{\theta} = (\sigma_f, \ell_1, \ldots, \ell_D)$ that defines prior mean and covariance

$$\begin{aligned} \mathbb{E}[\mathbf{f}(\mathbf{x})] &= \mathbf{0} \\ \mathrm{cov}[\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')] &= K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') \end{aligned}$$

- By GP definition

$$\begin{aligned} X &= (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T && \in \mathbb{R}^{n \times D} \\ \mathbf{f}(X) &= (\boldsymbol{f}(\boldsymbol{x}_1)^T, \ldots, \boldsymbol{f}(\boldsymbol{x}_n)^T)^T && \in \mathbb{R}^{nD \times 1} \\ p(\mathbf{f}(X)) &= \mathcal{N}(\mathbf{f}(X) | \mathbf{0}, \mathbf{K}_{\boldsymbol{\theta}}(X, X)) \\ \mathbf{K}_{\boldsymbol{\theta}}(X, X) &= (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^{n} && \in \mathbb{R}^{nD \times nD} \end{aligned}$$

Inducing points, kernel interpolation, integration



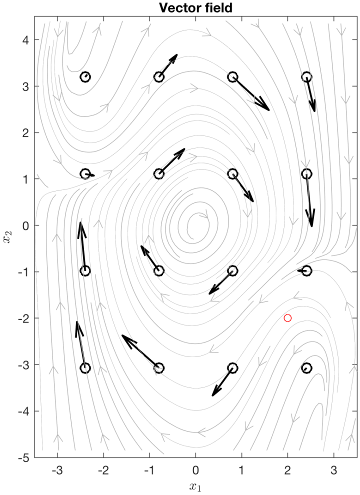
Introduce:

- Inducing points and vectors

$$Z = (\mathbf{z}_1, \ldots, \mathbf{z}_M)^T \in \mathbb{R}^{M \times D}$$

$$U = (\mathbf{u}_1, \ldots, \mathbf{u}_M)^T = (\mathbf{f}(\mathbf{z}_1), \ldots, \mathbf{f}(\mathbf{z}_M))^T \in \mathbb{R}^{M \times D}$$

Inducing points, kernel interpolation, integration



Introduce:

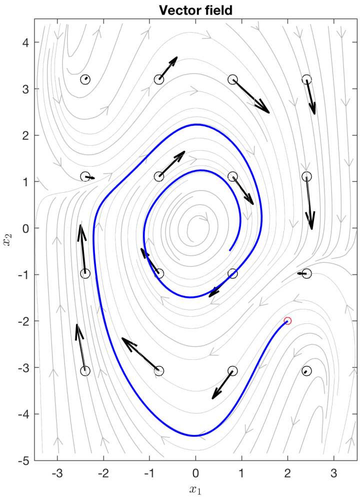
- Inducing points and vectors

$$Z = (\mathbf{z}_1, \ldots, \mathbf{z}_M)^T \in \mathbb{R}^{M \times D}$$
$$U = (\mathbf{u}_1, \ldots, \mathbf{u}_M)^T = (\mathbf{f}(\mathbf{z}_1), \ldots, \mathbf{f}(\mathbf{z}_M))^T \in \mathbb{R}^{M \times D}$$

- For any $\mathbf{x} \in \mathbb{R}^D$, we obtain vector field by GP "posterior" predictions

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}|Z, U) \triangleq \mathbf{K}_\theta(\mathbf{x}, Z)\mathbf{K}_\theta(Z, Z)^{-1}\mathrm{vec}(U)$$

Inducing points, kernel interpolation, integration



Introduce:

- Inducing points and vectors

$$Z = (\mathbf{z}_1, \ldots, \mathbf{z}_M)^T \in \mathbb{R}^{M \times D}$$
$$U = (\mathbf{u}_1, \ldots, \mathbf{u}_M)^T = (\mathbf{f}(\mathbf{z}_1), \ldots, \mathbf{f}(\mathbf{z}_M))^T \in \mathbb{R}^{M \times D}$$

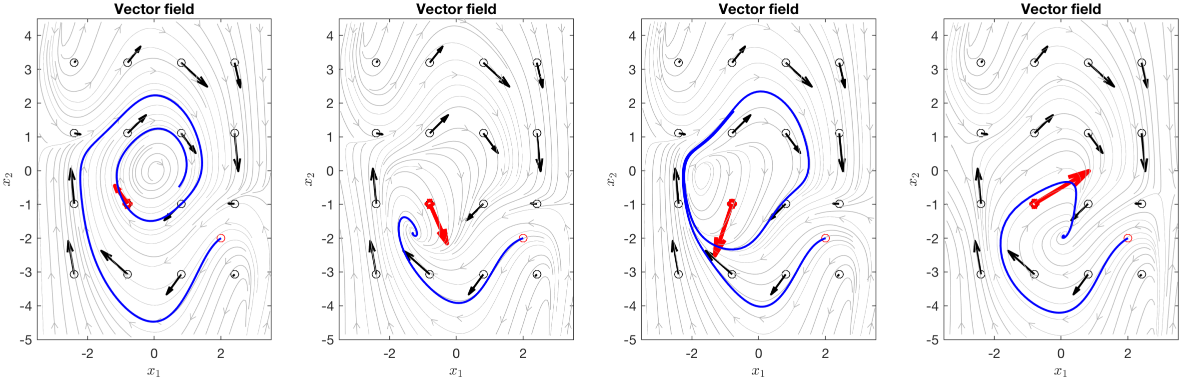
- For any $\mathbf{x} \in \mathbb{R}^D$, we obtain vector field by GP "posterior" predictions

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}|Z, U) \triangleq \mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}, Z)\mathbf{K}_{\boldsymbol{\theta}}(Z, Z)^{-1}\mathrm{vec}(U)$$

- We can integrate $\mathbf{x}_t = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}_\tau|Z, U)d\tau$

Changing an inducing vector

- Inducing vectors and kernel hyperparameters completely specify the vector field / initial value problem



Posterior

- The posterior is then

$$p(U, \boldsymbol{x}_0, \boldsymbol{\theta}, \Omega | Y, Z) \propto \underbrace{p(Y | U, Z, \boldsymbol{x}_0, \boldsymbol{\theta}, \Omega)}_{\text{likelihood}} \underbrace{p(U | Z, \boldsymbol{\theta})}_{\text{GP prior}} p(\boldsymbol{\theta}) p(\Omega) = \mathcal{L},$$

Posterior

- The posterior is then

$$p(U, \boldsymbol{x}_0, \boldsymbol{\theta}, \Omega | Y, Z) \propto \underbrace{p(Y | U, Z, \boldsymbol{x}_0, \boldsymbol{\theta}, \Omega)}_{\text{likelihood}} \underbrace{p(U | Z, \boldsymbol{\theta})}_{\text{GP prior}} p(\boldsymbol{\theta}) p(\Omega) = \mathcal{L},$$

where

$$\begin{aligned} p(Y | U, Z, \boldsymbol{x}_0, \boldsymbol{\theta}, \Omega) &= \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i | \boldsymbol{x}_{t_i}, \Omega) \\ &= \prod_{i=1}^{N} \mathcal{N}\Big(\mathbf{y}_i \,\Big|\, \underbrace{\mathbf{x}_0 + \int_0^{t_i} \mathbf{f}_U(\mathbf{x}_\tau) d\tau}_{\mathbf{x}_U(t_i)}, \Omega\Big) \end{aligned}$$

$$p(U | Z, \boldsymbol{\theta}) = \mathcal{N}(vec(U) | \mathbf{0}, \boldsymbol{K}_{\boldsymbol{\theta}}(Z, Z))$$

Remark: $\Omega = \mathrm{diag}(\omega_1^2 \ldots, \omega_D^2)$

Model estimation with gradients

- We can seek the MAP solution

$$U_{\text{MAP}}, \boldsymbol{x}_{0,\text{MAP}}, \boldsymbol{\theta}_{\text{MAP}}, \Omega_{\text{MAP}} = \underset{U, \boldsymbol{x}_0, \boldsymbol{\theta}, \Omega}{\operatorname{argmax}} \log \mathcal{L}$$

  or aim sampling the posterior
- Gradient descent or HMC sampling both need computing the gradients of the likelihood

$$\frac{dp(y_i|\mathbf{x}_0, U, \Omega)}{dU} = \underbrace{\frac{d\mathcal{N}(\mathbf{y}_i|\mathbf{x}_U(t_i), \Omega)}{d\mathbf{x}_U(t_i)}}_{\text{easy}} \underbrace{\frac{d\mathbf{x}_U(t_i)}{dU}}_{\text{hard}}$$

  which requires computing sensitivities

$$\frac{d\mathbf{x}_U(t)}{dU} = \frac{d}{dU}\left(\mathbf{x}_0 + \int_0^t \mathbf{f}_U(\mathbf{x}(\tau))d\tau\right) \equiv S(t)$$

Sensitivities

- Lets consider the time derivative of $S(t)$

$$\dot{S}(t) = \frac{d}{dt}\frac{d\mathbf{x}_U(t)}{dU} = \frac{d}{dU}\overbrace{\frac{d\mathbf{x}_U(t)}{dt}}^{\dot{\mathbf{x}} \triangleq \mathbf{f}} = \frac{d\mathbf{f}(\mathbf{x}_U(t), U)}{dU}$$

[1] Recall that the derivative of a composite function $f(g(x))$ is $f'(g(x))g'(x)$

- Lets consider the time derivative of $S(t)$

$$\dot{S}(t) = \frac{d}{dt}\frac{d\mathbf{x}_U(t)}{dU} = \frac{d}{dU}\overbrace{\frac{d\mathbf{x}_U(t)}{dt}}^{\dot{\mathbf{x}} \triangleq \mathbf{f}} = \frac{d\mathbf{f}(\mathbf{x}_U(t), U)}{dU}$$

- Total derivative of the right hand side[1]

$$\overbrace{\frac{d}{dt}\frac{d\mathbf{x}_U(t)}{dU}}^{\dot{S}(t)} = \overbrace{\frac{\partial \mathbf{f}(\mathbf{x}_U(t), U)}{\partial \mathbf{x}}}^{J(t)}\overbrace{\frac{d\mathbf{x}_U(t)}{dU}}^{S(t)} + \overbrace{\frac{\partial \mathbf{f}(\mathbf{x}_U(t), U)}{\partial U}}^{R(t)}$$

[1] Recall that the derivative of a composite function $f(g(x))$ is $f'(g(x))g'(x)$

Sensitivities

- Lets consider the time derivative of $S(t)$

$$\dot{S}(t) = \frac{d}{dt}\frac{d\mathbf{x}_U(t)}{dU} = \frac{d}{dU}\overbrace{\frac{d\mathbf{x}_U(t)}{dt}}^{\dot{\mathbf{x}} \triangleq \mathbf{f}} = \frac{d\mathbf{f}(\mathbf{x}_U(t), U)}{dU}$$

- Total derivative of the right hand side[1]

$$\overbrace{\frac{d}{dt}\frac{d\mathbf{x}_U(t)}{dU}}^{\dot{S}(t)} = \overbrace{\frac{\partial \mathbf{f}(\mathbf{x}_U(t), U)}{\partial \mathbf{x}}}^{J(t)}\overbrace{\frac{d\mathbf{x}_U(t)}{dU}}^{S(t)} + \overbrace{\frac{\partial \mathbf{f}(\mathbf{x}_U(t), U)}{\partial U}}^{R(t)}$$

- Sensitivities form another ODE system!

[1] Recall that the derivative of a composite function $f(g(x))$ is $f'(g(x))g'(x)$

Sensitivities

- Lets consider the time derivative of $S(t)$

$$\dot{S}(t) = \frac{d}{dt}\frac{d\mathbf{x}_U(t)}{dU} = \frac{d}{dU}\overbrace{\frac{d\mathbf{x}_U(t)}{dt}}^{\dot{\mathbf{x}} \triangleq \mathbf{f}} = \frac{d\mathbf{f}(\mathbf{x}_U(t), U)}{dU}$$

- Total derivative of the right hand side[1]

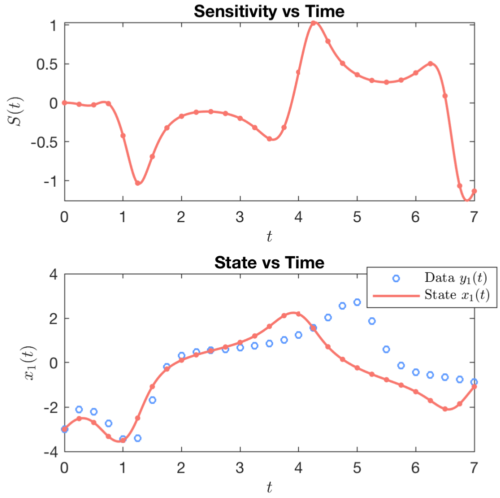
$$\overbrace{\frac{d}{dt}\frac{d\mathbf{x}_U(t)}{dU}}^{\dot{S}(t)} = \overbrace{\frac{\partial \mathbf{f}(\mathbf{x}_U(t), U)}{\partial \mathbf{x}}}^{J(t)}\overbrace{\frac{d\mathbf{x}_U(t)}{dU}}^{S(t)} + \overbrace{\frac{\partial \mathbf{f}(\mathbf{x}_U(t), U)}{\partial U}}^{R(t)}$$

- Sensitivities form another ODE system!
- Analytical forms for $J(t)$ and $R(t)$ are available (recall: $\mathbf{f}(x) = \mathbf{K}_\theta(\mathbf{x}, Z)\mathbf{K}_\theta(Z, Z)^{-1}\mathrm{vec}(U)$)

$$J(t) = \frac{\partial \mathbf{K}_\theta(\mathbf{x}, Z)}{\partial \mathbf{x}}\mathbf{K}_\theta(Z, Z)^{-1}\mathrm{vec}(U) \qquad R(t) = \mathbf{K}_\theta(\mathbf{x}, Z)\mathbf{K}_\theta(Z, Z)^{-1}$$

[1] Recall that the derivative of a composite function $f(g(x))$ is $f'(g(x))g'(x)$

Efficient integration in parallel



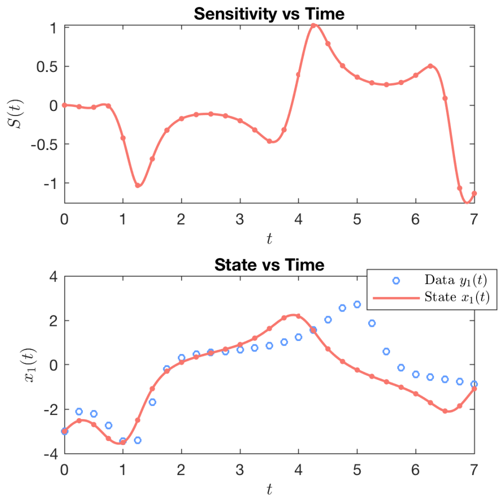
- We solve two ODE systems **efficiently** in parallel

$$S(t) = S_0 + \int_0^t \Big(J(\tau)S(\tau) + R(\tau) \Big) d\tau$$

$$\mathbf{x}_t = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}_\tau) d\tau$$

where $S(0) = 0$ and $\mathbf{x}_0 = \hat{\mathbf{x}}_0$.

Efficient integration in parallel



- We solve two ODE systems **efficiently** in parallel

$$S(t) = S_0 + \int_0^t \Big(J(\tau)S(\tau) + R(\tau) \Big) d\tau$$

$$\mathbf{x}_t = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}_\tau) d\tau$$

where $S(0) = 0$ and $\mathbf{x}_0 = \hat{\mathbf{x}}_0$.

- Partial derivative wrt. σ_f (finite diff.) and Ω (easy); $(\ell_1, \ldots, \ell_D)$ as part of model selection

Noncentral Parameterisation

- Latent re-parameterisation of the posterior using Cholesky decomposition:

$$\boldsymbol{L}_{\boldsymbol{\theta}} \boldsymbol{L}_{\boldsymbol{\theta}}^T = \boldsymbol{K}_{\boldsymbol{\theta}}(Z, Z)$$

$$U = \boldsymbol{L}_{\boldsymbol{\theta}} \widetilde{U}$$

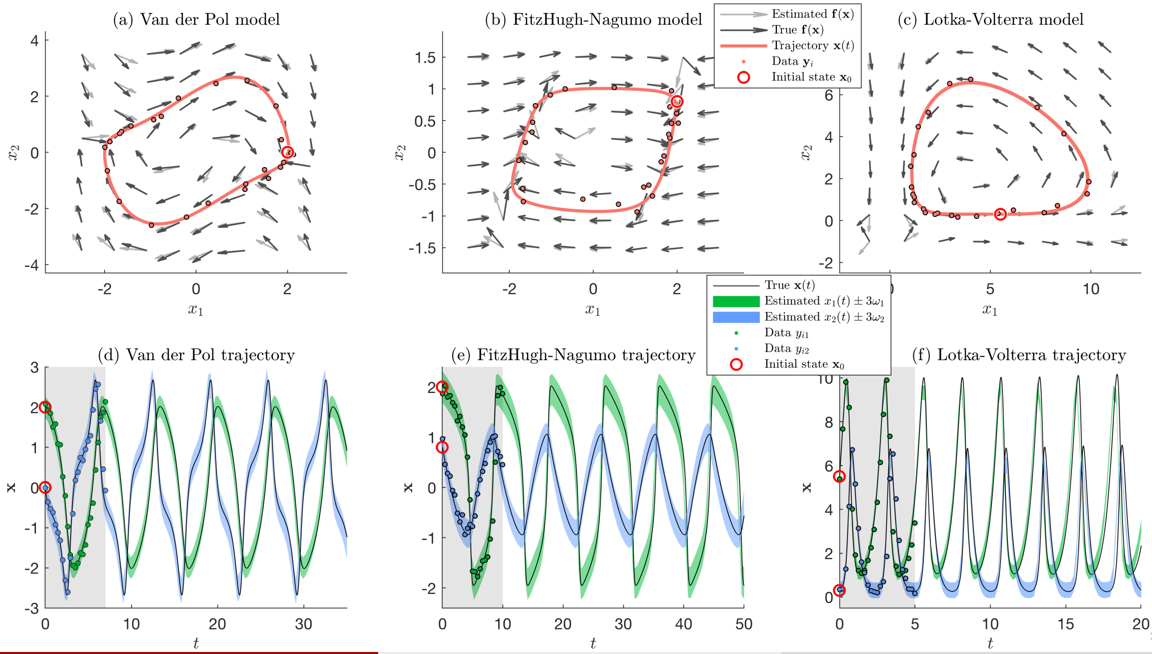
$$\widetilde{U} \sim \mathcal{N}(\mathbf{0}, I)$$

$$\nabla_{\widetilde{U}} \log \mathcal{L} = \boldsymbol{L}_{\boldsymbol{\theta}}^T \nabla_U \log \mathcal{L}$$

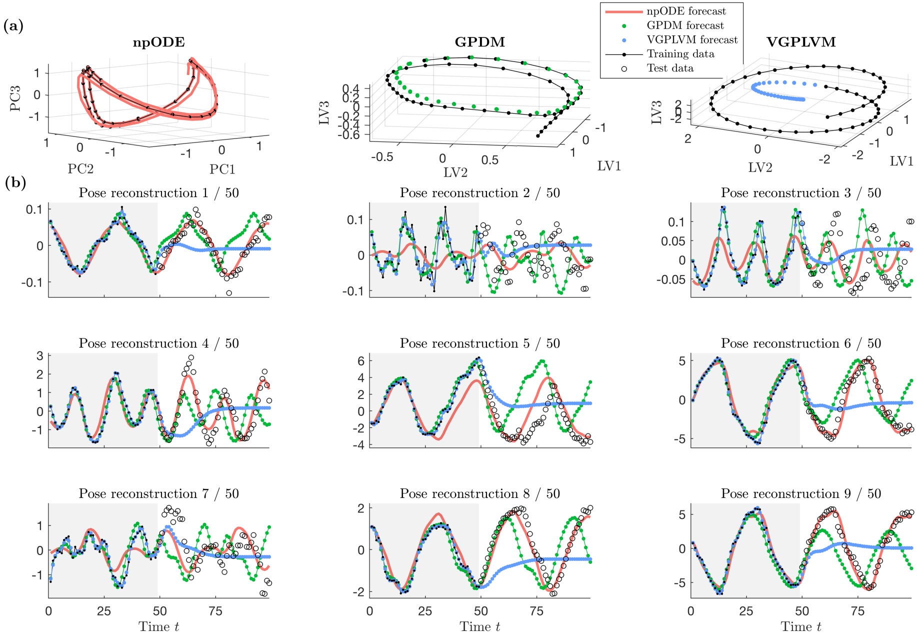
Simulated Dynamics

- Three simulated differential systems:
  - Van der Pol (VDP)
  - FitzHugh-Nagumo (FHN), and
  - Lotka-Volterra (LV) oscillators
- Data specs:
  - 5 time series for training
  - 25 data points in each time series
  - 1 cycle of VDP&FHN, 1.7 cycle of LV
  - Added noise variance: 0.1^2

Model fit and predictions



npODE of the CMU mocap walking data



References

- Gadd C, Heinonen M, Lähdesmäki, Kaski S, Sample-efficient reinforcement learning using deep Gaussian processes, https://arxiv.org/abs/2011.01226
- Heinonen M, Yildiz C, Mannerström H, Intosalmi J, Lähdesmäki H, Learning unknown ODE models with Gaussian processes, In *Proceedings of the 35th International Conference on Machine Learning (ICML), PMLR* 80:1959-1968, 2018.
- Wang JM, Fleet DJ, Hertzmann A, Gaussian process dynamical models, In Proc. *NIPS*, pp. 1441-1448, 2005.