PILCO: A Model-Based and Data-Efficient Approach to Policy Search

Marc Peter Deisenroth, Carl Edward Rasmussen

Topic: Model-Based RL

Presenter: Parth Jaggi

Model-Based and Data-Efficient Approach to Policy Search

Motivation and Main Problem

- What is the problem being solved?
  - Model-based RL's key problem is model bias
  - This is more pronounced with a lack of data samples

- Bad sample data efficiency renders these methods unusable for lower cost mechanical systems

Motivation and Main Problem

- Why is increasing sample data efficiency hard?
  - Requires informative prior knowledge
  - Extracting more information from available data

- Can we increase data efficiency without assuming any expert knowledge?

PILCO Contributions

1. PILCO is model-based policy search method that reduces Model bias.

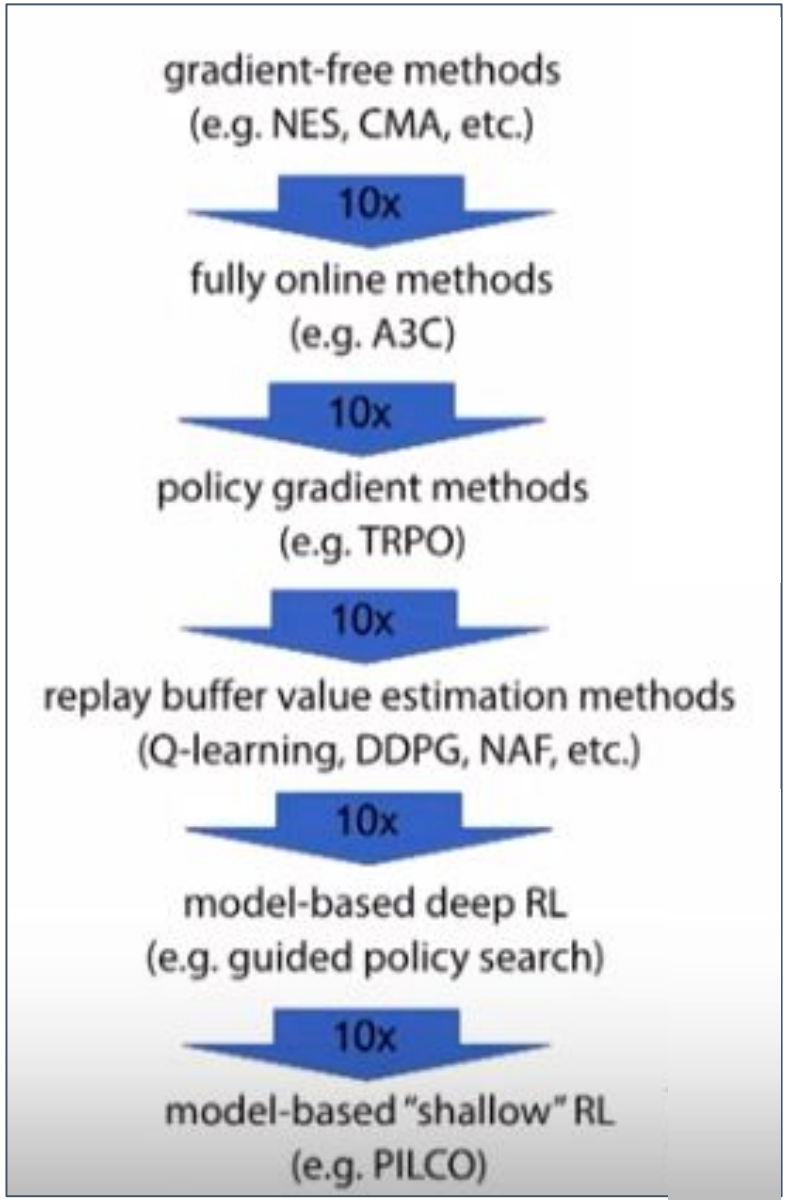
2. Learns Probabilistic Dynamics model and incorporates model uncertainty into planning.

    - This facilitates learning from very few trials (some cases <20 secs)

3. Computes policy gradients analytically.

Model-Based RL Motivation

- Sample efficiency
- Transferability and Generality



Model-Based vs Model-Free

MB Upsides:

- Efficiently extract valuable information from available data
- Performs much better than MF where there is lack of sample data

Model-Based vs Model-Free

MB Upsides:

- Efficiently extract valuable information from available data

- Performs much better than MF where there is lack of sample data

MB Downsides:

- Lower overall reward with respect to Model-Free Methods
(if sufficient time provided for Model-Free method)

- Model Bias: assumes that learned dynamics accurately resembles the real environment

Model-Based vs Model-Free

MB Upsides:

- Efficiently extract valuable information from available data

- Performs much better than MF where there is lack of sample data

MB Downsides:

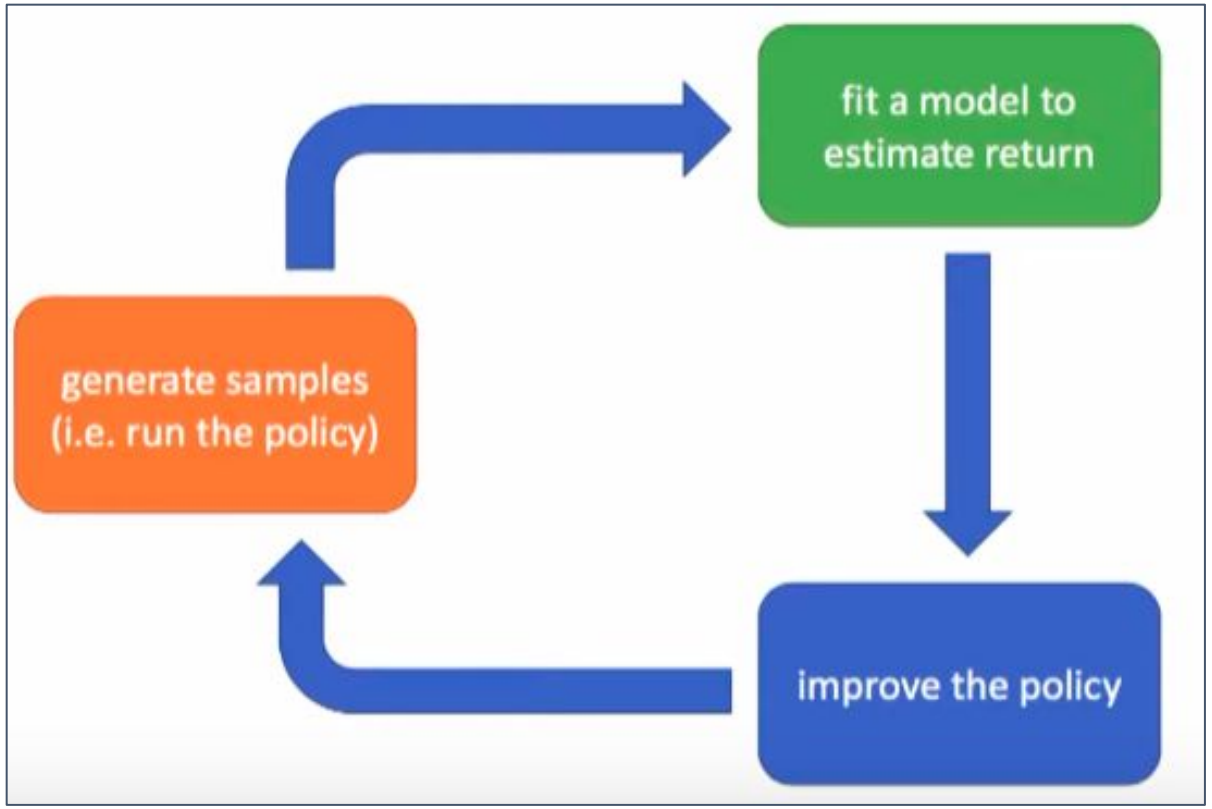
- Lower overall reward with respect to Model-Free Methods
(if sufficient time provided for Model-Free method)

- Model Bias: assumes that learned dynamics accurately resembles the real environment

    - What can this lead to? Optimizer's Curse

Vanilla Model-Based Algorithm

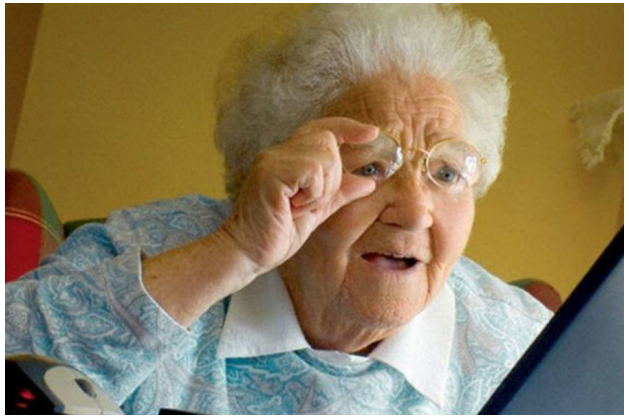
But what kind of model should we learn?



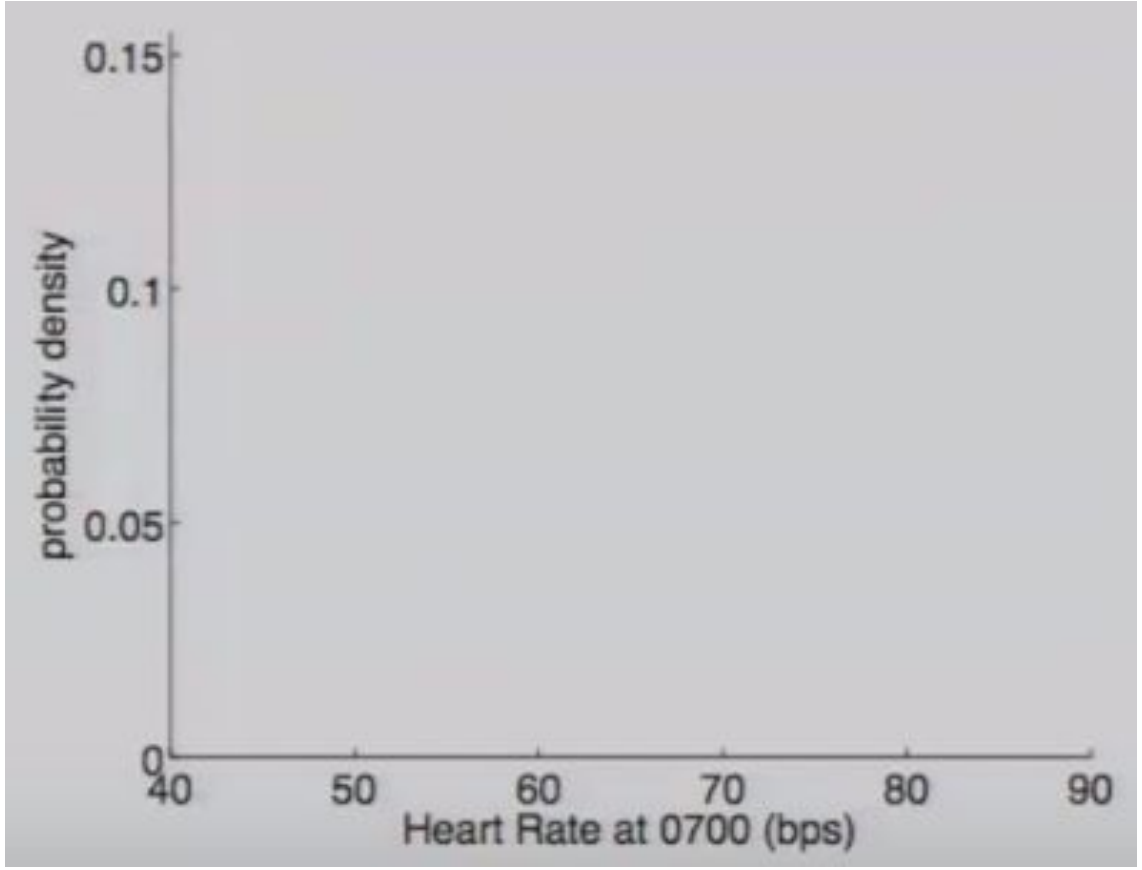
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model $f_\phi(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$
3. backpropagate through $f_\phi(\mathbf{s}, \mathbf{a})$ into policy to optimize $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

Gaussian Process

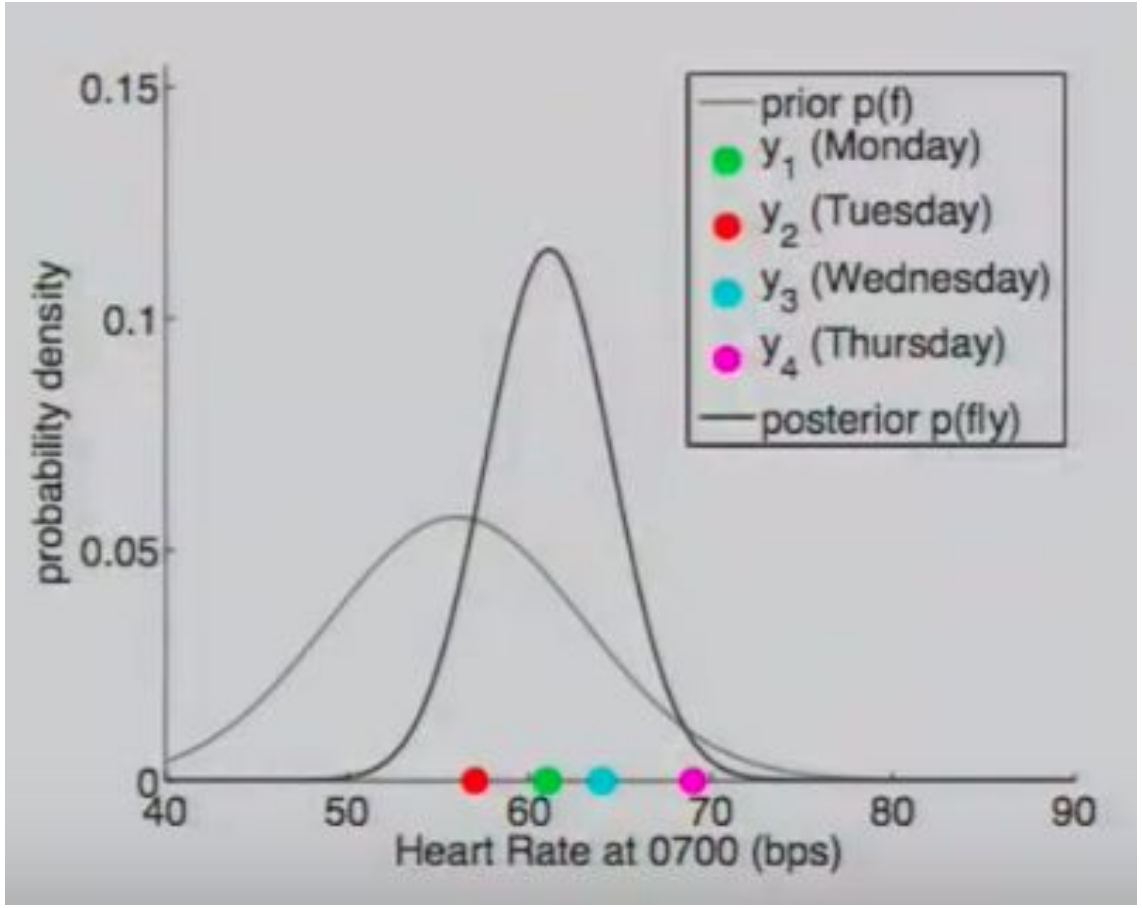
Gaussian process is a stochastic process (a collection of random variables indexed by time or space), such that every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed.



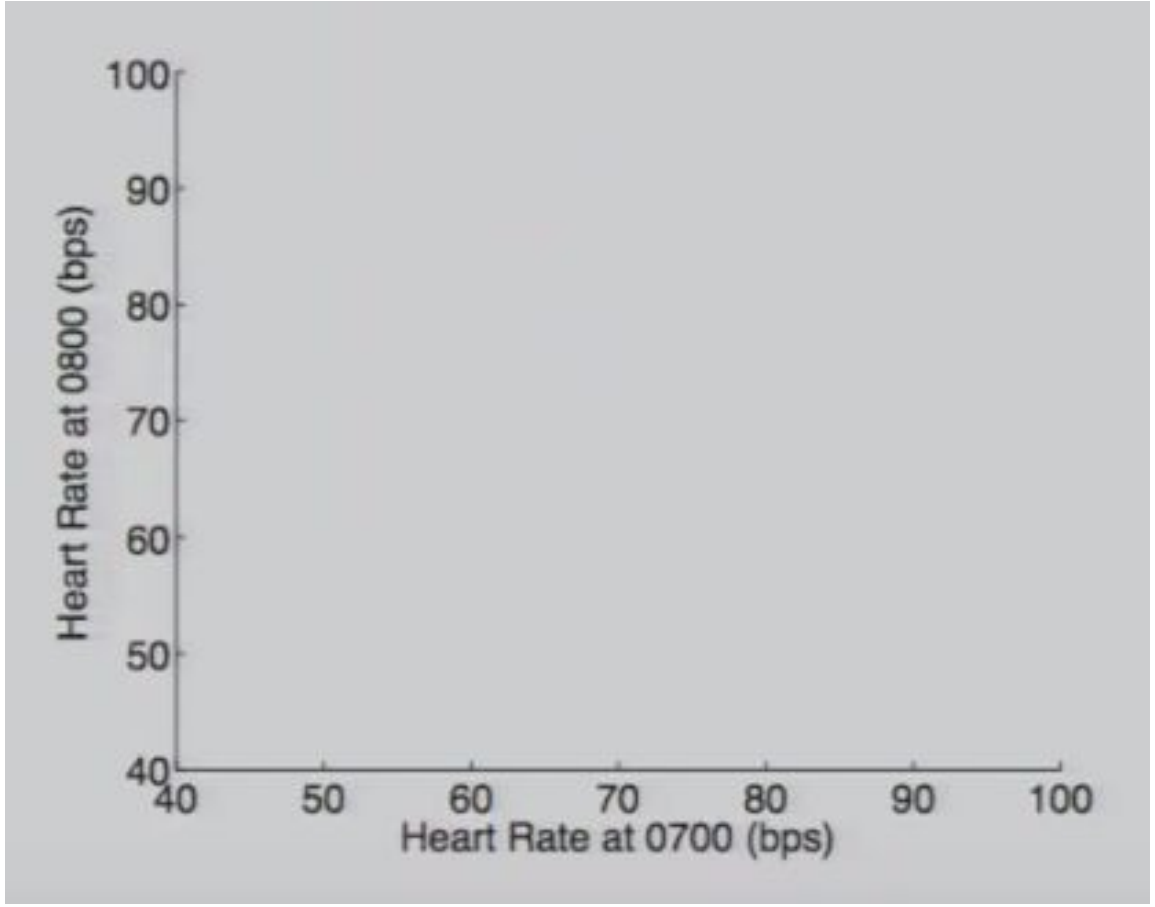
Gaussian Process Intuition



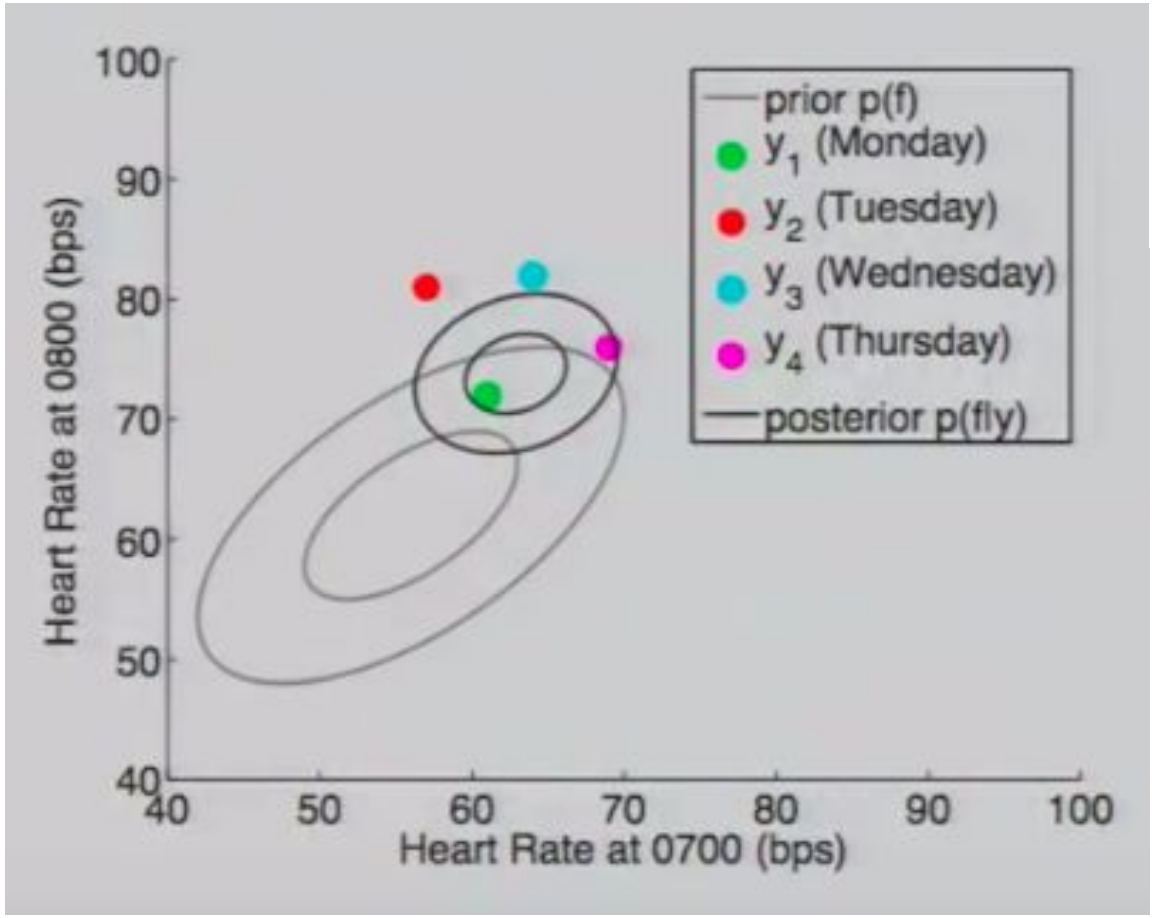
Gaussian Process Intuition

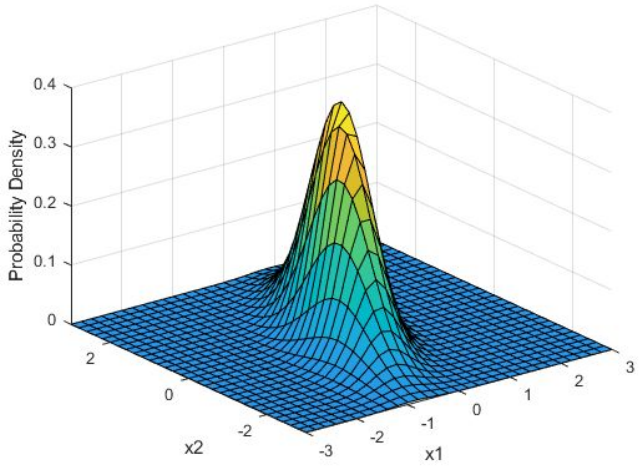


Gaussian Process Intuition

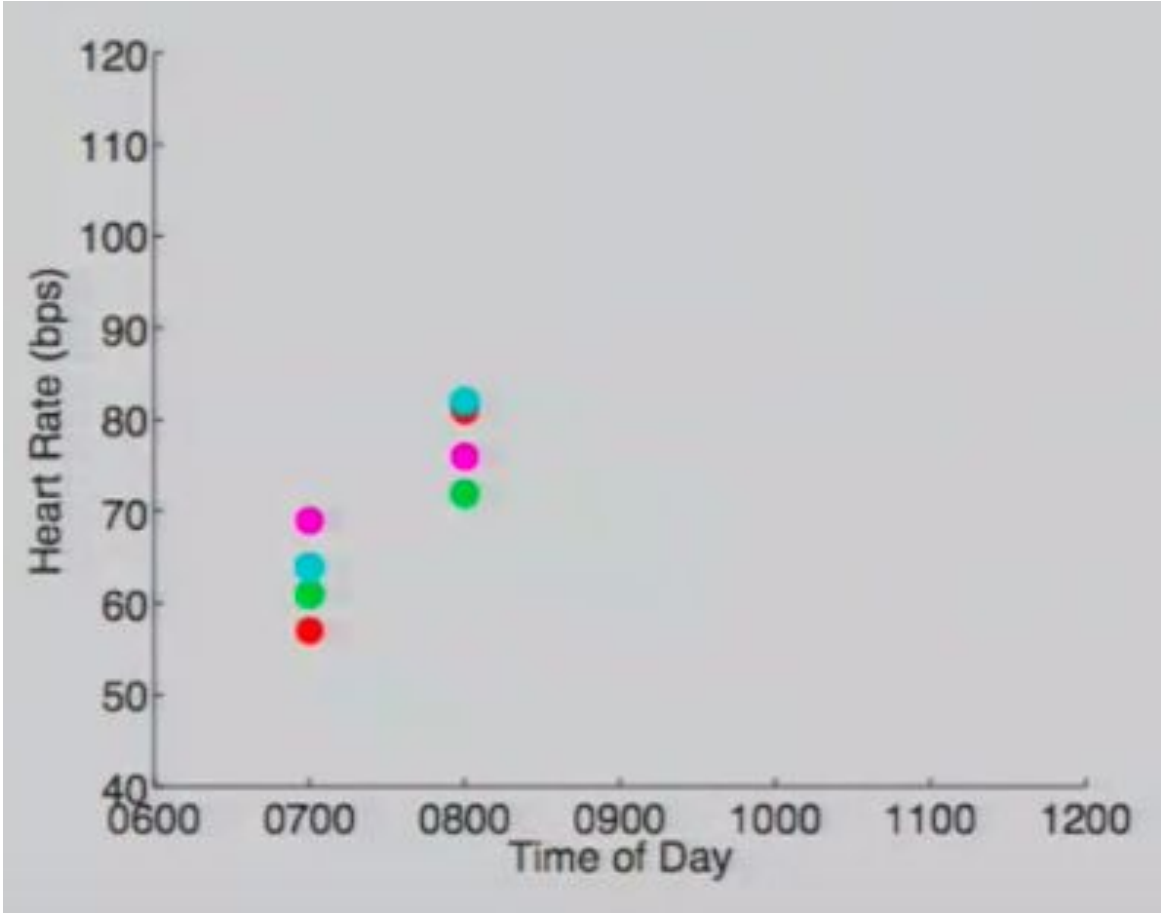


Gaussian Process Intuition

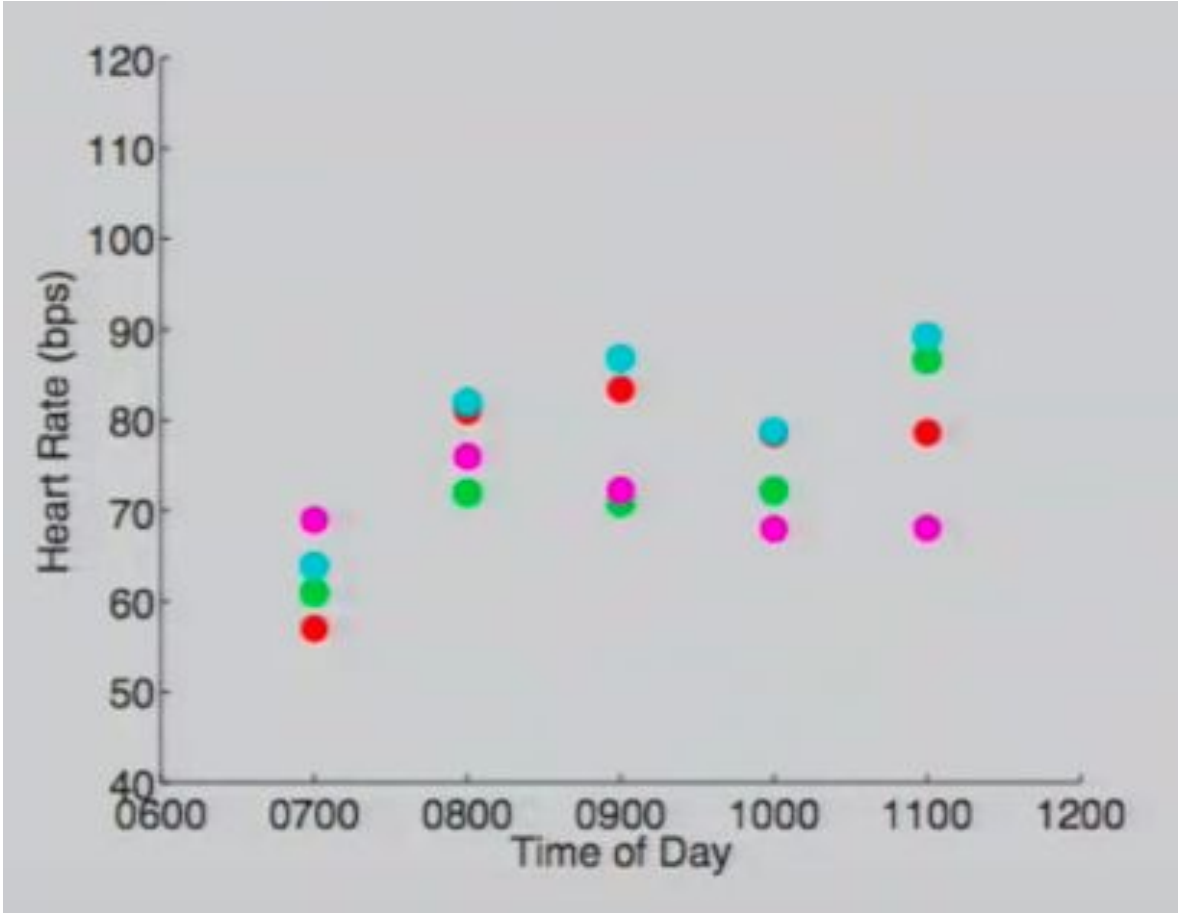




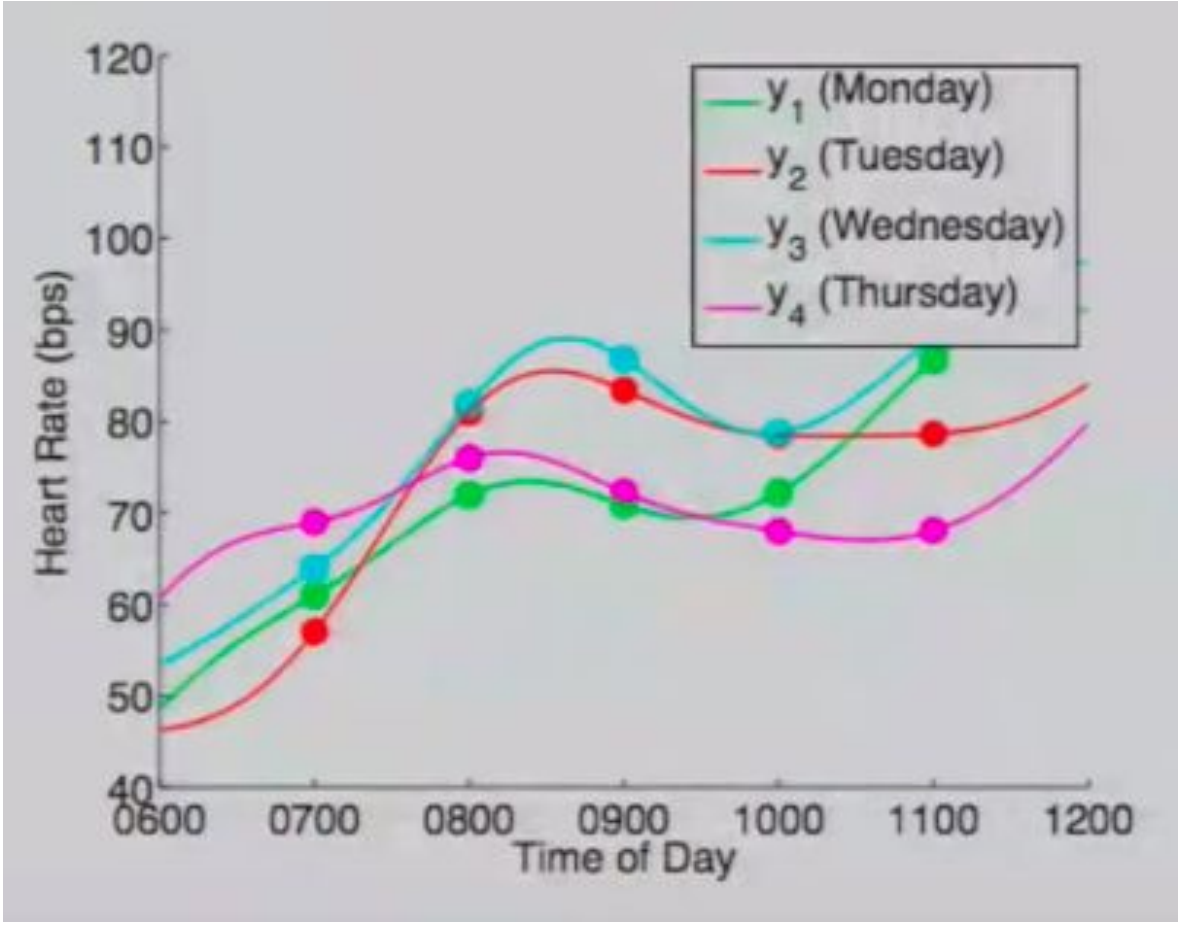
Gaussian Process Intuition



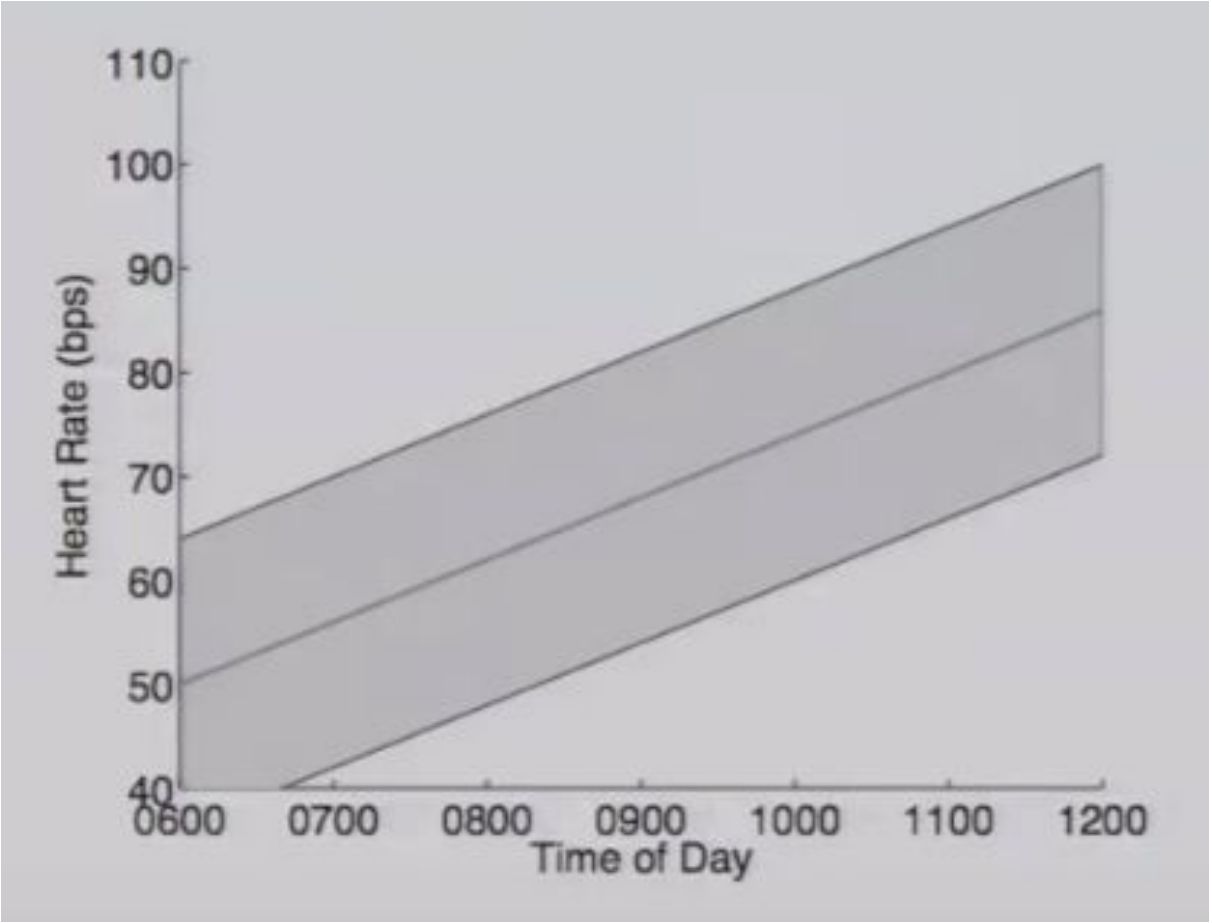
Gaussian Process Intuition



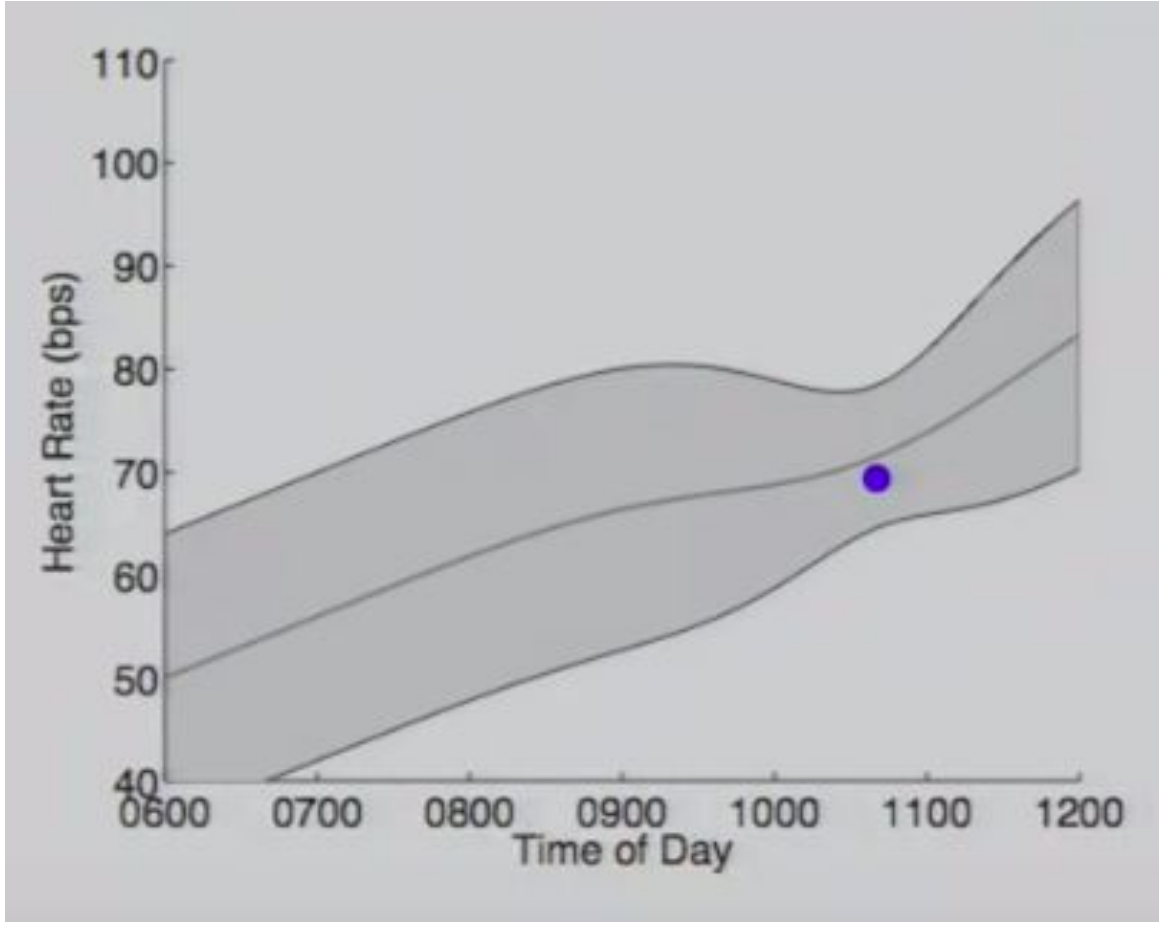
Gaussian Process Intuition



Gaussian Process Intuition

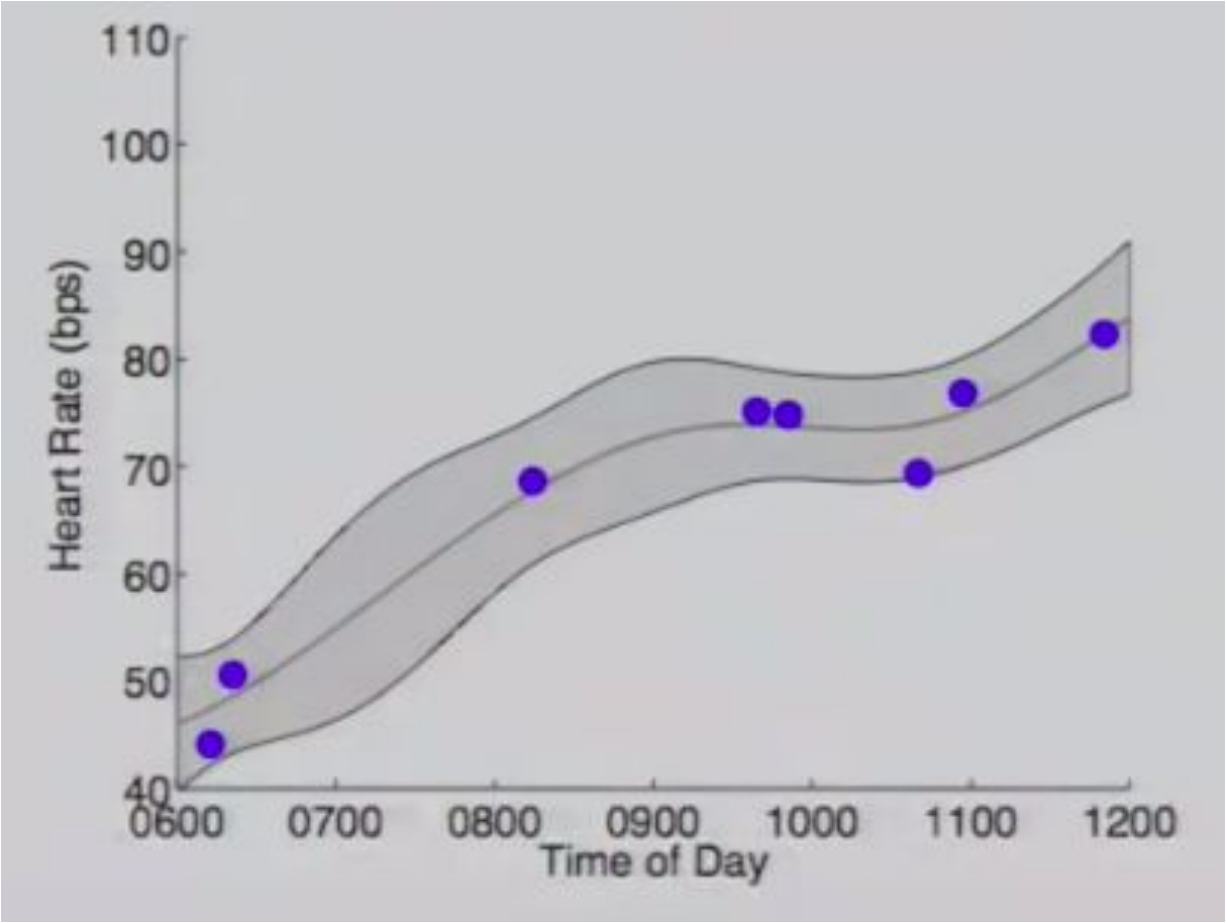


Gaussian Process Intuition



Gaussian Process Intuition

Can this do 2D?



Gaussian process is a stochastic process such that every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed.

Approach

$$J^{\pi}(\theta) = \sum_{t=0}^{T} \mathbb{E}_{\mathbf{x}_t}\left[c(\mathbf{x}_t)\right], \quad \mathbf{x}_0 \sim \mathcal{N}(\mu_0, \mathbf{\Sigma}_0)$$

$c(\mathbf{x}_t)$ is the cost (negative reward) of being in state x at time t

We are minimizing the expected return.

1. Dynamics Model Learning
2. Policy Evaluation
3. Analytic Gradients for Policy Improvement

Dynamics Model Learning - Using GP

Training inputs: $\text{tuples } (\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \in \mathbb{R}^{D+F}$

Training targets: $\Delta_t = \mathbf{x}_t - \mathbf{x}_{t-1} + \varepsilon \in \mathbb{R}^D$

Where: $\varepsilon \sim \mathcal{N}(0, \mathbf{\Sigma}_\varepsilon)$, $\mathbf{\Sigma}_\varepsilon = \text{diag}([\sigma_{\varepsilon_1}, \ldots, \sigma_{\varepsilon_D}])$

One steps predictions from the GP are:

$$p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t \mid \mu_t, \mathbf{\Sigma}_t\right),$$
$$\mu_t = \mathbf{x}_{t-1} + \mathbb{E}_f[\Delta_t],$$
$$\mathbf{\Sigma}_t = \text{var}_f[\Delta_t].$$

Policy Evaluation

$$p(\Delta_t) = \int p(f(\tilde{\mathbf{x}}_{t-1})|\tilde{\mathbf{x}}_{t-1}) p(\tilde{\mathbf{x}}_{t-1})\, \mathrm{d}\tilde{\mathbf{x}}_{t-1}$$

$$\tilde{\mathbf{x}} := [\mathbf{x}^\top \mathbf{u}^\top]^\top$$

Having the mean μ_Δ and the covariance Σ_Δ of the predictive distribution $p(\Delta_t)$, the Gaussian approximation to the desired distribution $p(x_t)$ is given as $N(x_t | \mu_t, \Sigma_t)$ with:

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{\mu}_\Delta$$

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{t-1} + \boldsymbol{\Sigma}_\Delta + \mathrm{cov}[\mathbf{x}_{t-1}, \Delta_t] + \mathrm{cov}[\Delta_t, \mathbf{x}_{t-1}]$$

$$\mathrm{cov}[\mathbf{x}_{t-1}, \Delta_t] = \mathrm{cov}[\mathbf{x}_{t-1}, \mathbf{u}_{t-1}] \boldsymbol{\Sigma}_u^{-1} \mathrm{cov}[\mathbf{u}_{t-1}, \Delta_t]$$

$$\mathbb{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)] = \int c(\mathbf{x}_t) \mathcal{N}\left(\mathbf{x}_t \,|\, \mu_t, \boldsymbol{\Sigma}_t\right) \mathrm{d}\mathbf{x}_t$$

Gradients for Policy Improvement

$$\mathrm{d}J^{\pi} / \mathrm{d}\theta$$

$$\frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}\theta} = \frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}p(\mathbf{x}_t)} \frac{\mathrm{d}p(\mathbf{x}_t)}{\mathrm{d}\theta} := \frac{\partial \mathcal{E}_t}{\partial \mu_t} \frac{\mathrm{d}\mu_t}{\mathrm{d}\theta} + \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{\Sigma}_t} \frac{\mathrm{d}\boldsymbol{\Sigma}_t}{\mathrm{d}\theta}$$

$$\mathcal{E}_t := \mathbb{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)],$$

$$\frac{\mathrm{d}\mu_t}{\mathrm{d}\theta} = \frac{\partial \mu_t}{\partial \mu_{t-1}} \frac{\mathrm{d}\mu_{t-1}}{\mathrm{d}\theta} + \frac{\partial \mu_t}{\partial \boldsymbol{\Sigma}_{t-1}} \frac{\mathrm{d}\boldsymbol{\Sigma}_{t-1}}{\mathrm{d}\theta} + \frac{\partial \mu_t}{\partial \theta}$$

$$\frac{\mathrm{d}p(\mathbf{x}_t)}{\mathrm{d}\theta} = \frac{\partial p(\mathbf{x}_t)}{\partial p(\mathbf{x}_{t-1})} \frac{\mathrm{d}p(\mathbf{x}_{t-1})}{\mathrm{d}\theta} + \frac{\partial p(\mathbf{x}_t)}{\partial \theta},$$

$$\frac{\partial p(\mathbf{x}_t)}{\partial p(\mathbf{x}_{t-1})} = \left\{ \frac{\partial \mu_t}{\partial p(\mathbf{x}_{t-1})}, \frac{\partial \boldsymbol{\Sigma}_t}{\partial p(\mathbf{x}_{t-1})} \right\}.$$

Both μ_t and Σ_t are functionally dependent on the mean μ_u and the covariance Σ_u of the control signal (and θ) through μ_{t-1} and Σ_{t-1}

$$\frac{\partial \mu_t}{\partial \theta} = \frac{\partial \mu_\Delta}{\partial p(\mathbf{u}_{t-1})} \frac{\partial p(\mathbf{u}_{t-1})}{\partial \theta} = \frac{\partial \mu_\Delta}{\partial \mu_u} \frac{\partial \mu_u}{\partial \theta} + \frac{\partial \mu_\Delta}{\partial \boldsymbol{\Sigma}_u} \frac{\partial \boldsymbol{\Sigma}_u}{\partial \theta}$$

Algorithm

Policy Evaluation

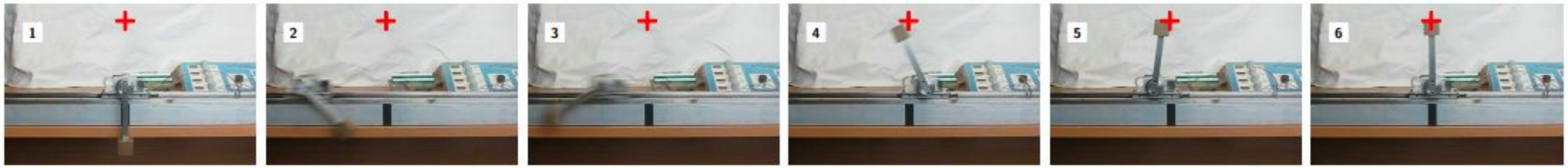
Algorithm 1 PILCO

1: **init:** Sample controller parameters $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
Apply random control signals and record data.
2: **repeat**
3: Learn probabilistic (GP) dynamics model, see Sec. 2.1, using all data.
4: Model-based policy search, see Sec. 2.2–2.3.
5: **repeat**
6: Approximate inference for policy evaluation, see Sec. 2.2: get $J^\pi(\theta)$, Eqs. (10)–(12), (24).
7: Gradient-based policy improvement, see Sec. 2.3: get $\mathrm{d}J^\pi(\theta)/\mathrm{d}\theta$, Eqs. (26)–(30).
8: Update parameters θ (e.g., CG or L-BFGS).
9: **until** convergence; **return** θ^*
10: Set $\pi^* \leftarrow \pi(\theta^*)$.
11: Apply π^* to system (single trial/episode) and record data.
12: **until** task learned

Experimental Results

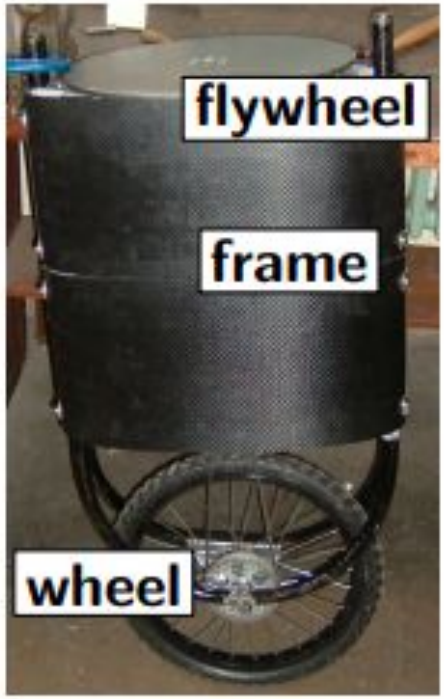
Table 1. PILCO's data efficiency scales to high dimensions.

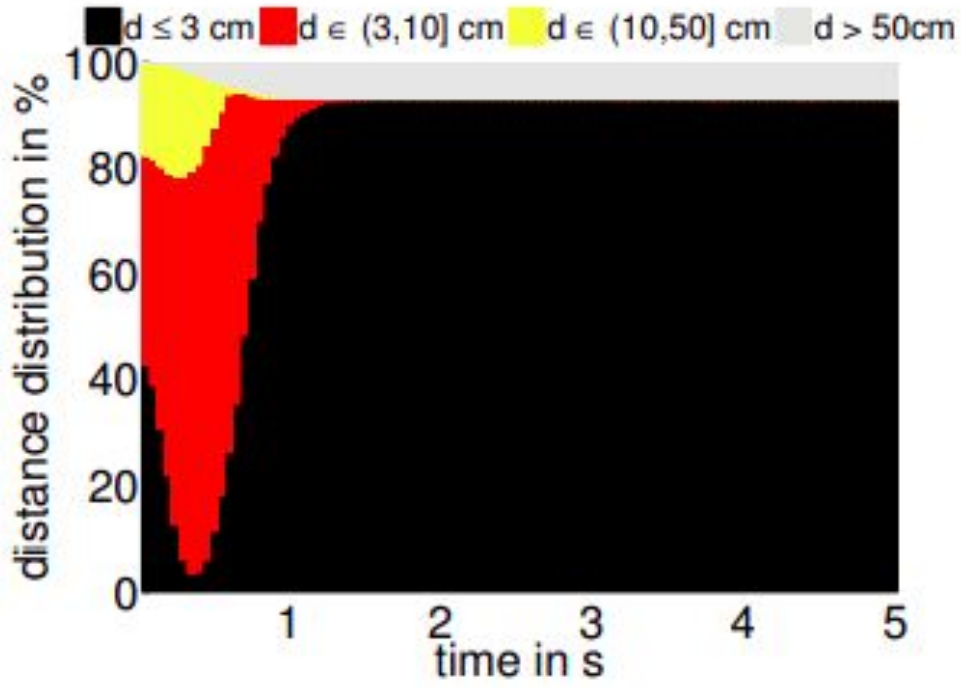
	cart-pole	cart-double-pole	unicycle
state space	$\mathbb{R}^4$	$\mathbb{R}^6$	$\mathbb{R}^{12}$
# trials	≤ 10	20–30	≈ 20
experience	$\approx 20\,\mathrm{s}$	$\approx 60\,\mathrm{s}$–$90\,\mathrm{s}$	$\approx 20\,\mathrm{s}$–$30\,\mathrm{s}$
parameter space	$\mathbb{R}^{305}$	$\mathbb{R}^{1816}$	$\mathbb{R}^{28}$



Real cart-pole system. Snapshots of a controlled trajectory of 20 s length after having learned the task. To solve the swing-up plus balancing, pilco required only 17.5 s of interaction with the physical system.

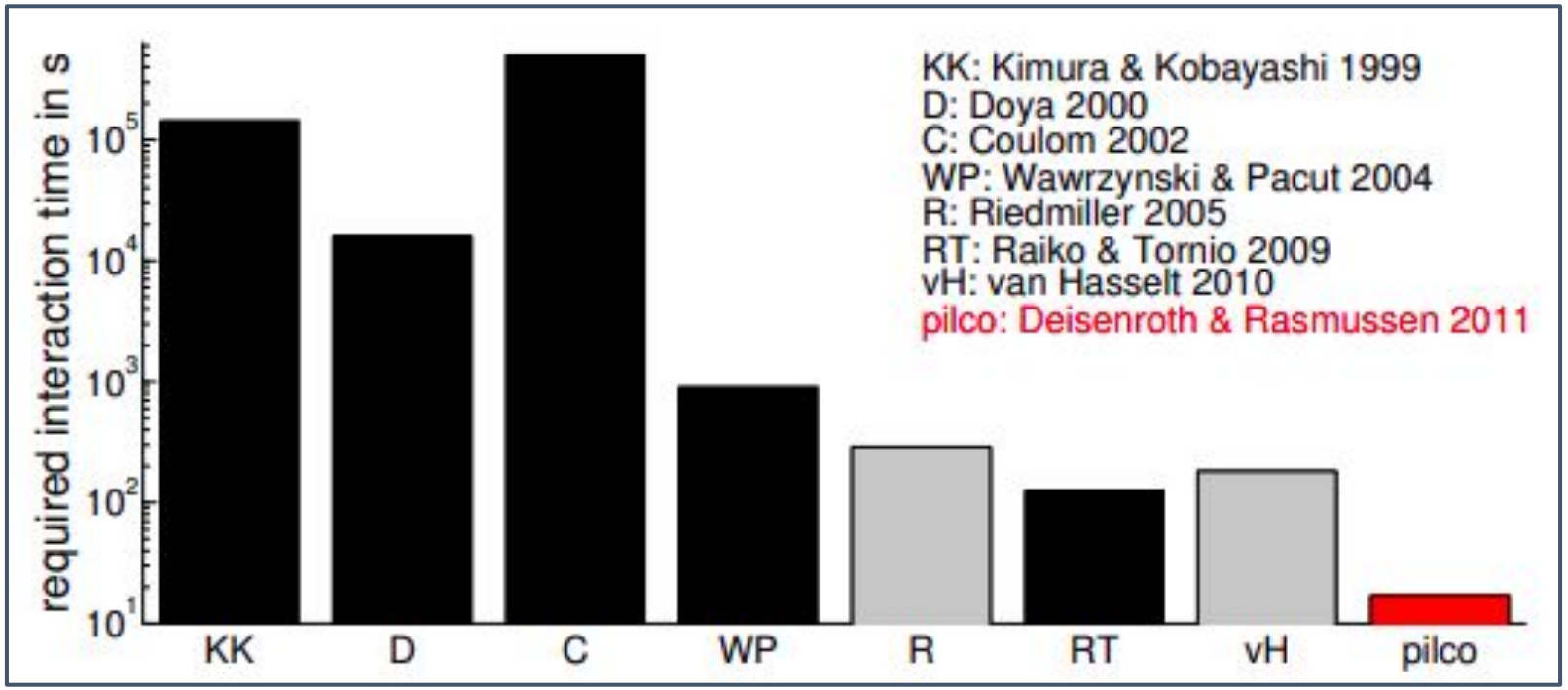
Experimental Results





Robotic unicycle. Histogram (after 1,000 test runs) of the distances of the flywheel from being upright.

Experimental Results



Critiques and Limitations

1. Approximated $p(\Delta_t)$ which could be a multi-modal distribution by a simple Gaussian distribution.
2. Environments covered had simple dynamics models
    a. GPs are computationally expensive. Cannot handle large number of samples.

Contributions (Recap)

- Problem: Model Bias
- Why is it important: Incorrect estimation of future states and confidence in prediction leads to poor results
- Key Insight:
  - Use probabilistic dynamics model to estimate certainty in future predictions and cascade of predictions

DeepPILCO: Improving PILCO with Bayesian Neural Network Dynamics Models

Yarin Gal and Rowan Thomas McAllister and Carl Edward Rasmussen

Topic: Model-Based RL

Presenter: Parth Jaggi

Motivation and Main Problem

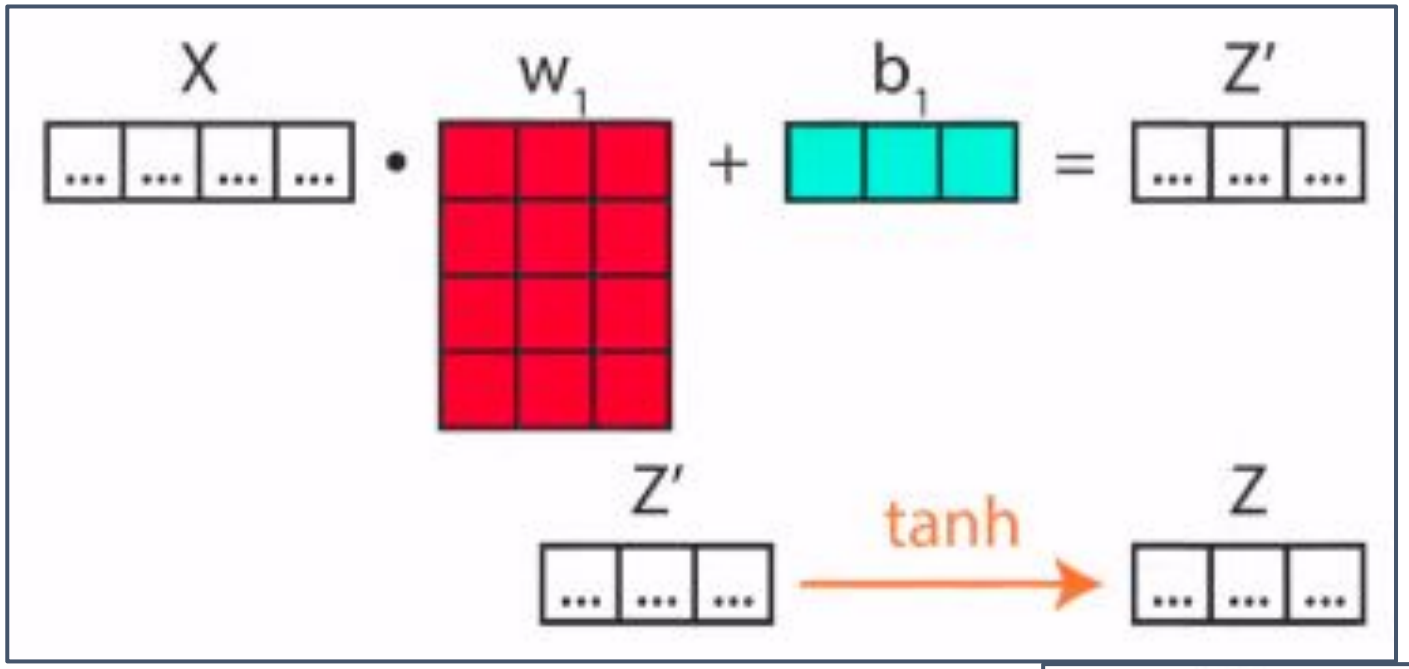
- What is the problem being solved?
  - GPs cannot be used for problems that need larger number of trials
    - GPs scale cubically with number of trials
  - PILCO does not consider temporal correlation in model uncertainty between successive state transitions, resulting in underestimation of state uncertainty at future time steps

DeepPILCO Contributions

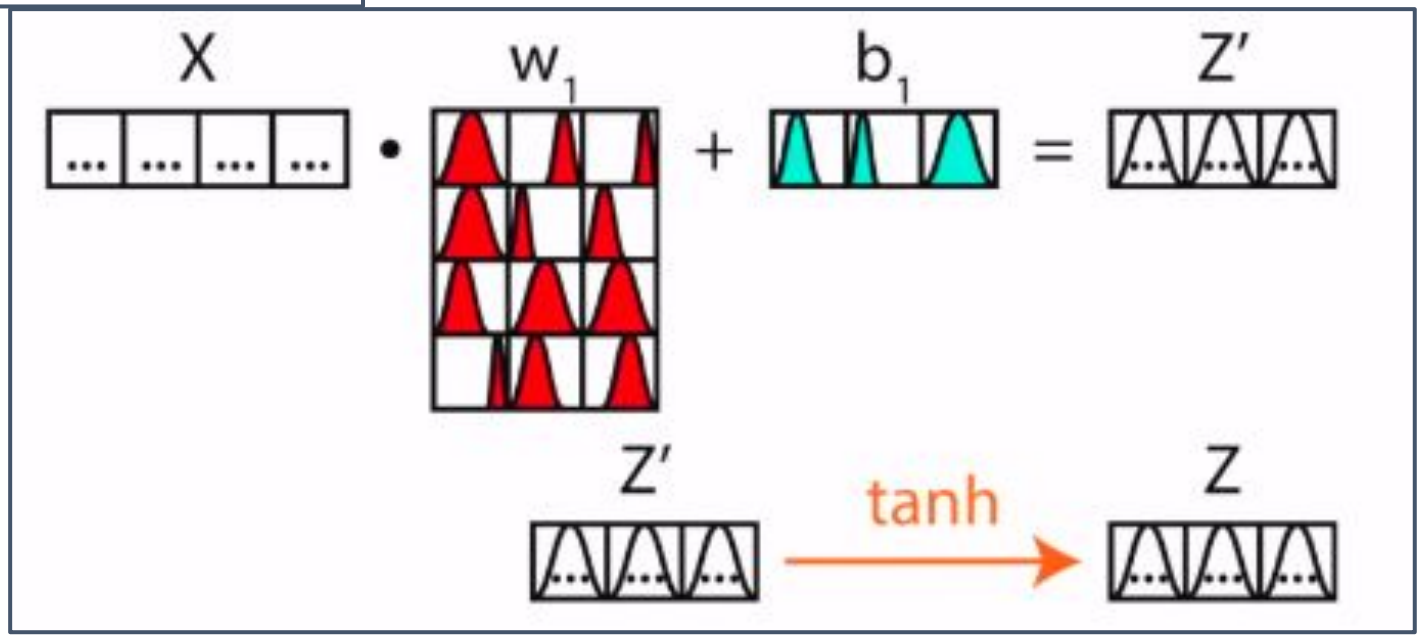
1. Replaced GP with a Bayesian deep dynamics model (BNN) while maintaining data-efficiency.

2. Used BNN with approximate variational inference allowing it to scale linearly with number of trials.

3. Used particle methods to sample dynamics function realisations and obtain lower cumulative cost than PILCO.



Bayesian Deep Learning



Approach

1. Output uncertainty

    Bayesian Neural Network. True posterior is intractably complex.

    Use Variational Inference (Dropout) to find distribution that minimizes KL divergence with true Posterior.

2. Input uncertainty

    Model must pass uncertain dynamics outputs from time step t as uncertain input into the dynamics model time step t+1.

    Particle Methods

3. Sampling functions from the dynamics model

    Sampling individual functions from the dynamics model and following a single function throughout an entire trial.

Approach - Output Uncertainty

1. Require output uncertainty from dynamics model to gain data-efficiency.
   Simple NN models cannot express output model uncertainty so BNN is used.
2. a) True posterior of a BNN is intractably complex
   b) Variational Inference (Dropout) is used to find distribution that minimizes KL divergence with true Posterior.
3. Uncertainty in the weights induces prediction uncertainty

Approach - Input Uncertainty

1. Propagate state distributions through dynamics model in the next time step. Cannot be done analytically for NNs.
2. Particle methods used to feed a distribution into the dynamics model.
    a. Sample set of particles from input distribution
    b. Pass these particles through the BNN dynamics model
    c. Yields an output distribution of particles.
3. Fitting a Gaussian distribution to output state distribution (also in PILCO) at each time step is critical
    a. Forces a unimodal fit which penalizes policies cause the predictive states to bifurcate (often precursor to a loss of control).

Approach - Sampling Functions

1. This approach allows following a single sampled function throughout an entire trial.
    a. Function weights are sampled once for the dynamics model and used at all timesteps
    b. Repeated application of the BNN model can be seen as a simple Bayesian RNN
2. PILCO does not consider such temporal correlation in model uncertainty between successive state transitions
    a. PILCO underestimates state uncertainty at future timesteps

Algorithm

Algorithm 1 PILCO

1: *Define* policy's functional form: $\pi : z_t \times \psi \rightarrow u_t$.
2: *Initialise* policy parameters ψ randomly.
3: **repeat**
4: *Execute* system, record data.
5: *Learn* dynamics model.
6: *Predict* system trajectories from $p(X_0)$ to $p(X_T)$.
7: *Evaluate* policy:

$$J(\psi) = \sum_{t=0}^{T} \gamma^t \mathbb{E}_X[\text{cost}(X_t)|\psi].$$

8: *Optimise* policy:

$$\psi \leftarrow \arg\min_{\psi} J(\psi).$$

9: **until** policy parameters ψ converge

Which point is changed for DeepPILCO?

Algorithm

Algorithm 2 Step 6 of Algorithm 1: *Predict* system trajectories from $p(X_0)$ to $p(X_T)$

Define time horizon T.
Initialise set of K particles $x_0^k \sim P(X_0)$.
for $k = 1$ to K **do**
    Sample BNN dynamics model weights W^k.
end for
for time $t = 1$ to T **do**
    for each particle x_t^1 to x_t^K **do**
        Evaluate BNN with weights W^k and input particle x_t^k, obtain output y_t^k.
    end for
    Calculate mean μ_t and standard deviation σ_t^2 of $\{y_t^1, ..., y_t^K\}$.
    Sample set of K particles $x_{t+1}^k \sim \mathcal{N}(\mu_t, \sigma_t^2)$.
end for

Results

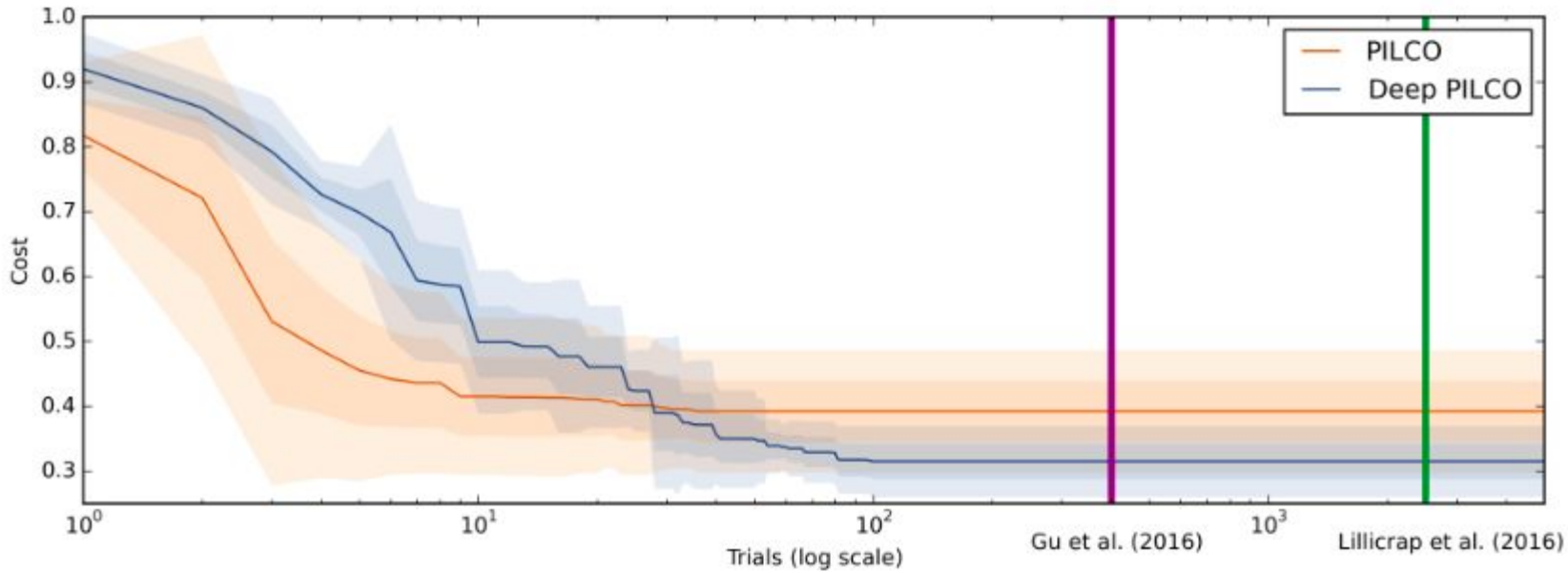


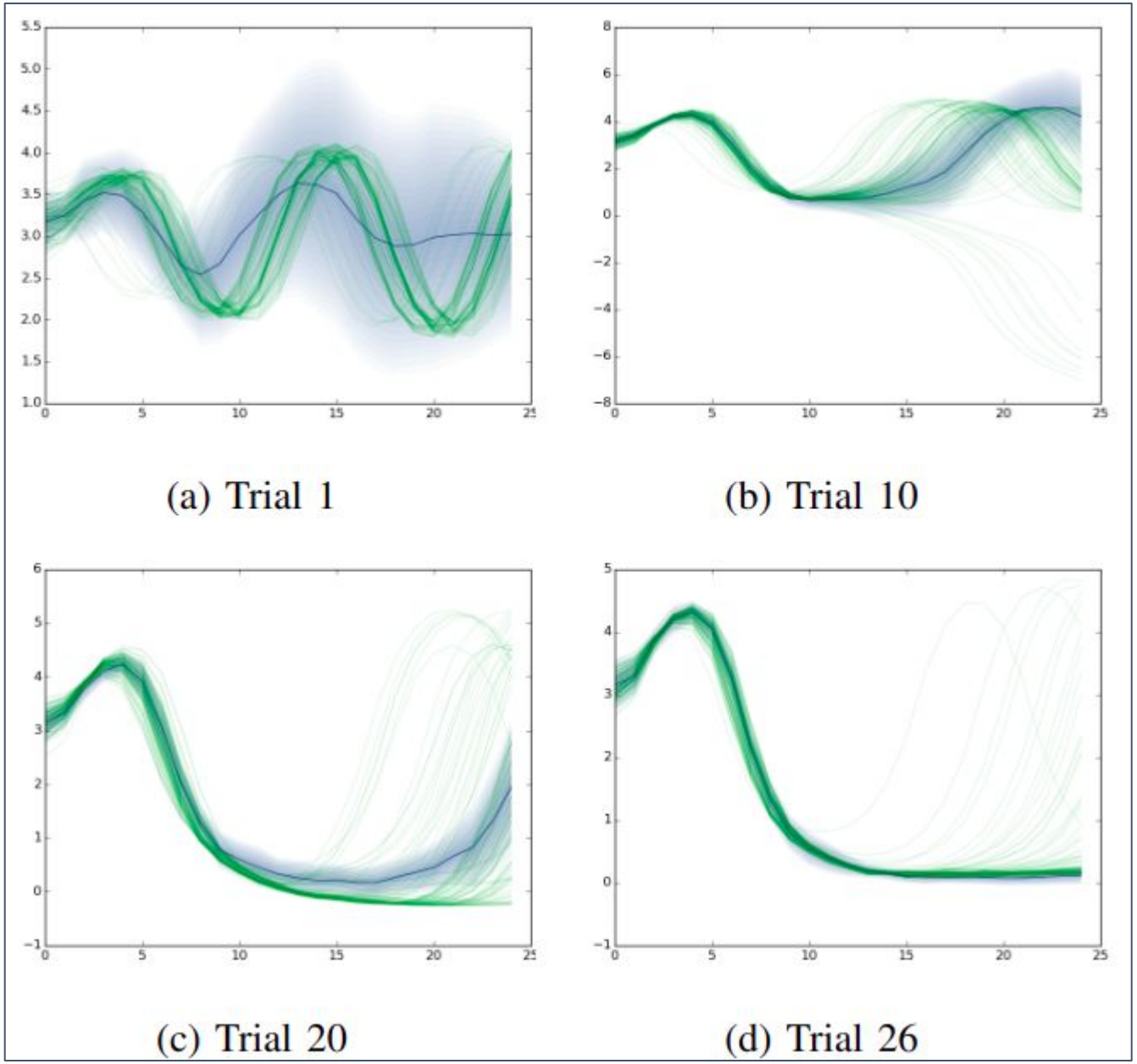
Fig. 3: Cost per trial (on *log scale*) for PILCO and Deep PILCO for the cartpole swing-up task. Vertical lines show estimates of number of trials required for model convergence for [9] (purple, requiring $\sim$ 400 trials) and [8] (green, requiring $\sim 2,500$ trials).

Progression of model fitting and controller optimisation as more trials of data are collected.

Each x-axis is timestep t, and each y-axis is the pendulum angle in radians.

The goal is to swing the pendulum up such that $\text{mod}(\theta, 2\pi) \approx 0$.

The green lines are samples from the ground truth dynamics. The blue distribution is our Gaussian-fitted predictive distribution of states at each timestep.



Contributions (Recap)

- Problem: Using a NN as probabilistic dynamics model
- Why is it important: GPs are very computationally expensive when working with large number of samples
- Why is it hard: Cannot be done analytically. Need approximation techniques
- Key Insight:
  - Prefer probabilistic dynamics model especially when optimizing data-efficiency
  - Using variational inference and particle methods techniques to use neural networks as probabilistic dynamics models