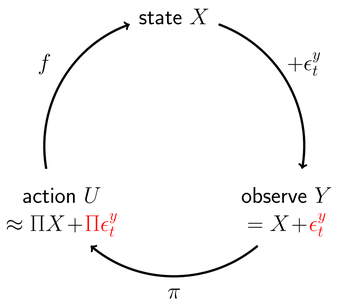
Data Efficient RL for Gaussian POMDPs

Rowan McAllister and Carl Rasmussen {rtm26,cer54}@cam.ac.uk

UNIVERSITY OF CAMBRIDGE

Two Problems

In many tasks, data efficiency is **critical**



Observation noise is **amplified** by high-gain controllers, creating cycles of increasing state variance

Data Efficient Reinforcement Learning

PILCO = data efficient RL framework exploiting **probabilistic dynamics models**

- **Modeling** observed dynamics helps **generalise** to unobserved dynamics
- But selecting a **single model** from a large plausible set → **model bias** (state prediction after many time steps ≈ random noise)
- Uncertainty helps **focus** policy optimisation towards **likely** improvements
- Model prediction using **all plausible dynamics functions** avoids model bias

PILCO Algorithm:

1: *Define* policy's functional form: $\pi : y \times \theta \to u$
2: *Initialise* policy parameters θ randomly
3: **repeat**
4: *Execute* system, record data
5: *Learn* dynamics model f
6: *Simulate* system trajectories from $p(X_0)$ to $p(X_T)$
7: *Evaluate* policy: $J(\theta) = \sum_{t=0}^{T} \mathbb{E}_X[\text{cost}(X_t)|\theta]$
8: *Improve* policy: $\theta \leftarrow \text{argmin}_\theta J(\theta)$
9: **until** policy parameters θ converge

But... whilst PILCO is data-efficient, it fails under significant observation noise.

- Could partial observability be incorporated into the algorithm?
- How might distributions over belief-states be modeled?

Dealing with Observation Noise

Our approach:

1. **Simulate in belief space** B instead of state space X (Figure 2)
2. Use PILCO's **Gaussian analytic framework** with moment matching:
   (a) Represent the current belief as Gaussian (Figure 3)
   (b) Represent the multiple future beliefs as **hierarchical-Gaussian** (Figure 4)

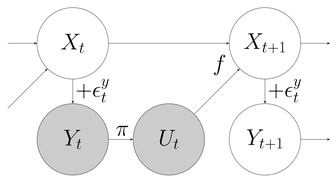


FIGURE 1: Unfiltered control

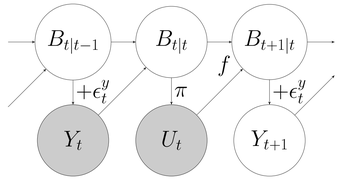


FIGURE 2: Filtered control

$$u_t = \pi(y_t) = \pi(x_t + \epsilon_t^y)$$

$$u_t = \pi(\mathbb{E}\left[b_{t|t}\right]) = \pi(W_b \mathbb{E}[b_{t|t-1}] + W_y y_t) = \pi(W_b \mathbb{E}[b_{t|t-1}] + W_y x_t + W_y \epsilon_t^y)$$
$$W_y \doteq \mathbb{V}[b_{t|t-1}] \cdot (\mathbb{V}[b_{t|t-1}] + \mathbb{V}[\epsilon_t^y])^{-1}$$

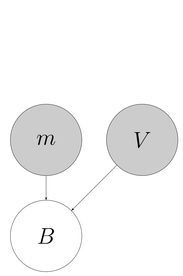


FIGURE 3: Belief in execution

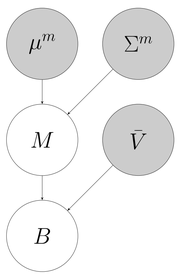


FIGURE 4: Belief in simulation

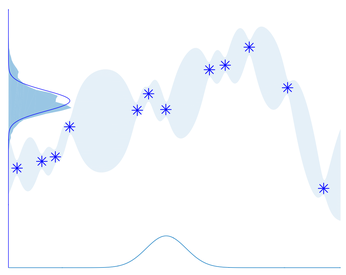


FIGURE 5: Filtering predict-step in execution

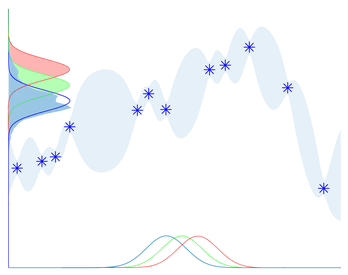
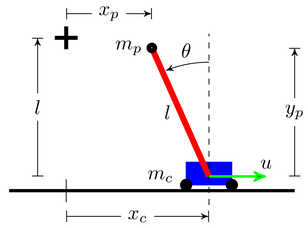


FIGURE 6: Filtering predict-step in simulation

Proof of Concept

- Evaluated the ideas above on the **cart-pole swing-up** benchmark



- A popular control benchmark with nonlinear dynamics and requires nonlinear control
- More data-efficient than PILCO under noise
- Finds better policies than Dallaire 2009, Deisenroth 2011, and Deisenroth 2012

FIGURE 7: Cart-pole system

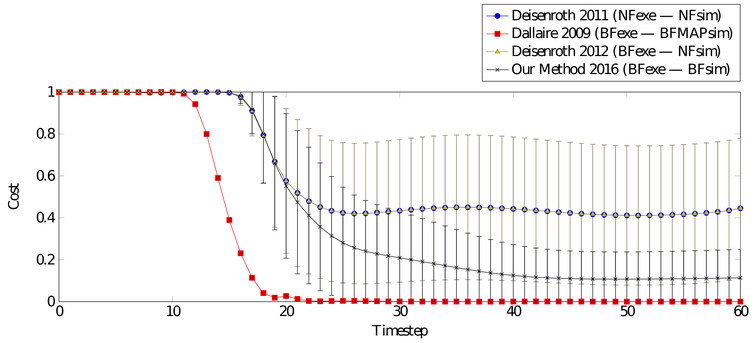


FIGURE 8: Predictive cost per time step

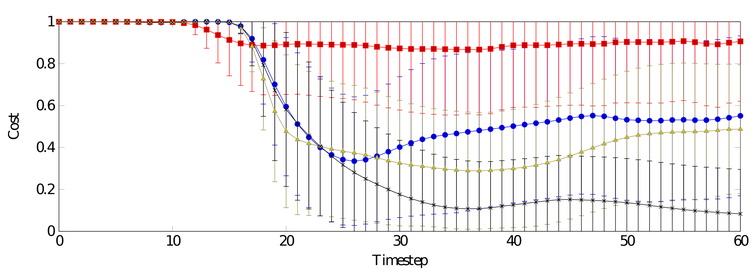


FIGURE 9: Empirical cost per time step

What's Next?

Current model:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + \epsilon_t^x \\ y_t &= x_t + \epsilon_t^y \\ u_t &= \pi(m_{t|t}) \\ c_t &= \text{cost}(x_t) \end{aligned}$$

Future model:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, t) + \epsilon_t^x \\ y_t &= Cx_t + Du_t + \epsilon_t^y \\ u_t &= \pi(m_{t|t}, V_{t|t}, t) \\ c_t &= \text{cost}(x_t, u_t, x_{t+1}, t) \end{aligned}$$

Paper: http://mlg.eng.cam.ac.uk/rowan/files/nips2017.pdf