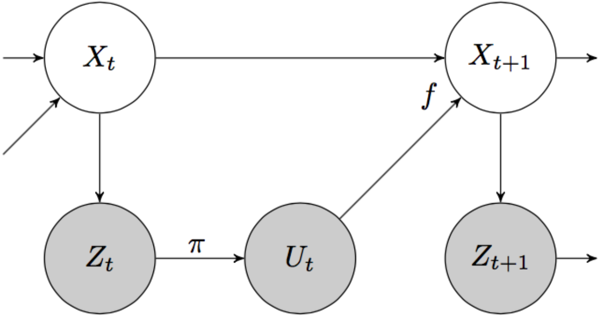
PILCO: A Model-Based and Data-Efficient Approach to Policy Search

(M.P. Deisenroth and C.E. Rasmussen)

CSC2541
November 4, 2016

PILCO Graphical Model

PILCO – Probabilistic Inference for Learning COntrol



- Latent states $\{X_t\}$ evolve through time based on previous states and controls
- Policy π maps Z_t, a noisy observation of X_t, into a control, U_t

PILCO Objective

Transitions follow dynamic system

$$x_t = f(x_{t-1}, u_{t-1})$$

where $x \in \mathbb{R}^D$, $u \in \mathbb{R}^F$ and f is a latent function.

Let π be parameterized by θ and $u_t = \pi(x_t, \theta)$. The objective is to find π that minimizes expected cost of following π for T steps

$$J^\pi(\theta) = \sum_{t=0}^{T} \mathbb{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)]$$

Cost function encodes information about a target state, e.g.,
$c(x) = 1 - \exp(-\|x - x_{\text{target}}\|^2 / \sigma_c^2)$

Algorithm

Algorithm 1 PILCO

1: *Define* policy's functional form: $\pi : z_t \times \psi \to u_t$.
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

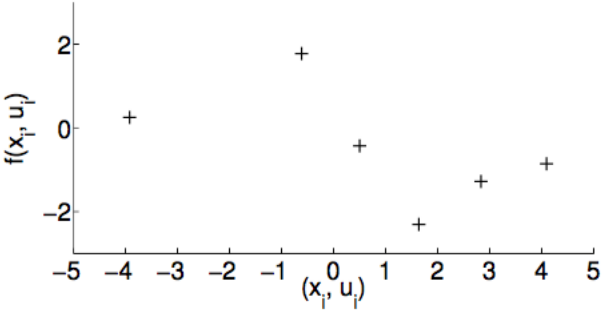
Algorithm

Algorithm 1 PILCO

1: *Define* policy's functional form: $\pi : z_t \times \psi \to u_t$.
2: *Initialise* policy parameters ψ randomly.
3: **repeat**
4: *Execute* system, record data.
5: $\boxed{\textit{Learn}\text{ dynamics model.}}$
6: *Predict* system trajectories from $p(X_0)$ to $p(X_T)$.
7: *Evaluate* policy:
$$J(\psi) = \textstyle\sum_{t=0}^{T} \gamma^t \mathbb{E}_X[\text{cost}(X_t)|\psi].$$
8: *Optimise* policy:
$$\psi \leftarrow \arg\min_{\psi} J(\psi).$$
9: **until** policy parameters ψ converge

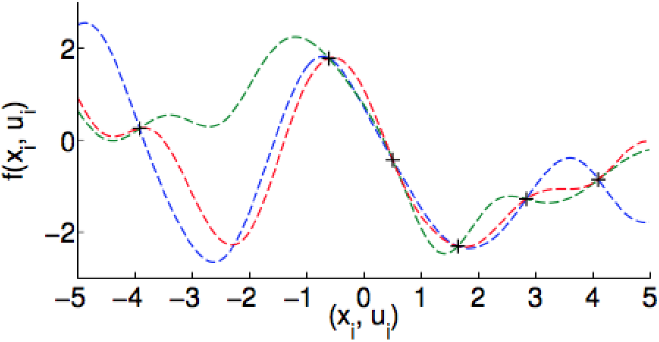
Dynamics Model Learning

Multiple plausible function approximators of f



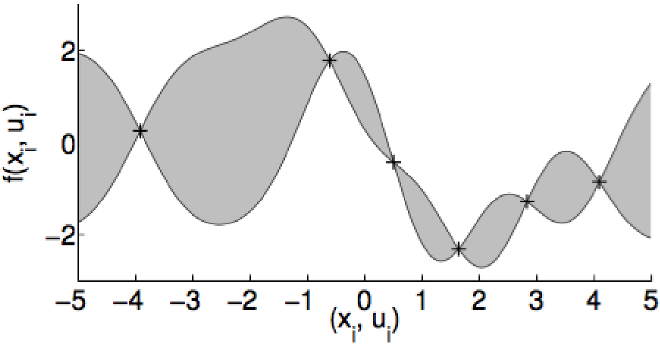
Dynamics Model Learning

Multiple plausible function approximators of f



Dynamics Model Learning

Define a Gaussian process (GP) prior on the latent dynamic function f



Dynamics Model Learning

Let the prior of f be $\mathcal{GP}(0, k(\tilde{x}, \tilde{x}'))$ where $\tilde{x} \triangleq [x^T u^T]^T$ and the squared exponential kernel is given by

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \alpha^2 \exp\big(-\tfrac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')^\top \mathbf{\Lambda}^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')\big)$$

Dynamics Model Learning

Let $\Delta_t = x_t - x_{t-1} + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \Sigma_\varepsilon)$ and $\Sigma_\varepsilon = \mathrm{diag}([\sigma_{\varepsilon_1}, \ldots, \sigma_{\varepsilon_D}])$. The GP yields one-step predictions (see Section 2.2 in reference 3)

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

Given n training inputs $\tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_n]$ and corresponding training targets $y = [\Delta_1, \ldots, \Delta_n]$, the posterior GP hyper-parameters are learned by evidence maximization (type 2 maximum likelihood).

Algorithm

Algorithm 1 PILCO

1: *Define* policy's functional form: $\pi : z_t \times \psi \to u_t$.
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

Policy Evaluation

In evaluating objective $J^{\pi}(\theta)$, we must calculate $p(x_t)$ since

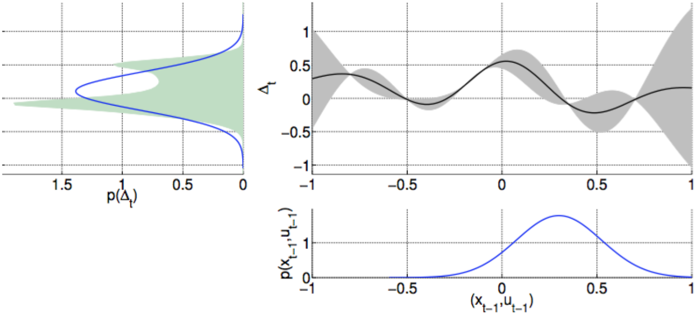
$$J^{\pi}(\theta) = \sum_{t=0}^{T} \mathbb{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)]$$

We have $x_t = x_{t-1} + \Delta_t - \varepsilon$, where in general, computing $p(\Delta_t)$ is analytically intractable.

Instead, $p(\Delta_t)$ is approximated with a Gaussian via moment matching.

Moment Matching

- Input distribution $p(x_{t-1}, u_{t_1})$ is assumed Gaussian
- When propagated through the GP model, we obtain $p(\Delta_t)$
- $p(\Delta_t)$ is approximated by a Gaussian via moment matching



Moment Matching

$p(x_t)$ can now be approximated with $\mathcal{N}(\mu_t, \Sigma_t)$ where

$$\mu_t = \mu_{t-1} + \mu_\Delta$$
$$\mathbf{\Sigma}_t = \mathbf{\Sigma}_{t-1} + \mathbf{\Sigma}_\Delta + \mathrm{cov}[\mathbf{x}_{t-1}, \Delta_t] + \mathrm{cov}[\Delta_t, \mathbf{x}_{t-1}]$$
$$\mathrm{cov}[\mathbf{x}_{t-1}, \Delta_t] = \mathrm{cov}[\mathbf{x}_{t-1}, \mathbf{u}_{t-1}]\mathbf{\Sigma}_u^{-1}\mathrm{cov}[\mathbf{u}_{t-1}, \Delta_t]$$

μ_Δ and Σ_Δ are computed exactly via iterated expectation and variance

$$\mu_\Delta^a = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}[\mathbb{E}_f[f(\tilde{\mathbf{x}}_{t-1})|\tilde{\mathbf{x}}_{t-1}]]$$

$$\sigma_{aa}^2 = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}\left[\mathrm{var}_f[\Delta_a|\tilde{\mathbf{x}}_{t-1}]\right] + \mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a^2] - (\mu_\Delta^a)^2$$
$$\sigma_{ab}^2 = \mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a \Delta_b] - \mu_\Delta^a \mu_\Delta^b\,, \quad a \neq b\,,$$

Algorithm

Algorithm 1 PILCO

1: *Define* policy's functional form: $\pi : z_t \times \psi \to u_t$.
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

Analytic Gradient for Policy Improvement

- Let $\mathcal{E}_t = \mathbb{E}_{x_t}[c(x_t)]$ so that $J^\pi(\theta) = \sum_{t=1}^{T} \mathcal{E}_t$.
- $\mathcal{E}_t$ depends on θ through $p(x_t)$, which depends on θ through $p(x_{t-1})$, which depends on θ through μ_t and Σ_t, ..., which depends on θ based on μ_u and Σ_u, where $u_t = \pi(x_t, \theta)$.
- Chain rule is used to calculate derivatives
- Analytic gradients allow for gradient-based non-convex optimization methods, e.g., CG or L-BFGS

Data-Efficiency

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

Advantages and Disadvantages

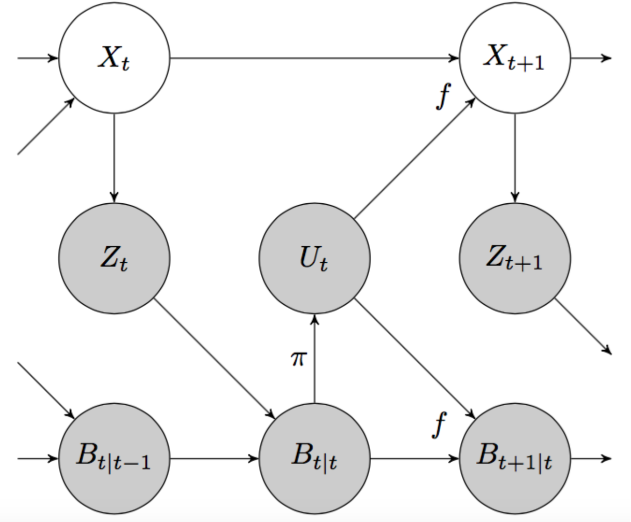
Advantages

- Data-efficient
- Incorporates model-uncertainty into long-term planning
- Does not rely on expert knowledge, i.e., demonstrations, or task-specific prior knowledge.

Disadvantages

- Not an optimal control method. If $p(X_i)$ do not cover the target region and σ_c induces a cost that is very peaked around the target solution, PILCO gets stuck in a local optimum because of zero gradients.
- Learned dynamics models are only confident in areas of the state space previously observed.
- Does not take temporal correlation into account. Model uncertainty treated as uncorrelated noise

Extension: PILCO with Bayesian Filtering



R. McAllister and C. Rasmussen, "Data-Efficient Reinforcement Learning in Coninuous-State POMDPs."
https://arxiv.org/abs/1602.02523

References

1. M.P. Deisenroth and C.E. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search" in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.
2. R. McAllister and C. Rasmussen, "Data-Efficient Reinforcement Learning in Coninuous-State POMDPs." `https://arxiv.org/abs/1602.02523`
3. C.E. Rasmussen and C.K.I. Williams (2006) *Gaussian Processes for Machine Learning*. MIT Press. `www.gaussianprocess.org/gpml/chapters`
4. C.M. Bishop (2006). *Pattern Recognition and Machine Learning* Chapter 6.4. Springer. ISBN 0-387-31073-8.