幻灯片标题

作者姓名

作者机构

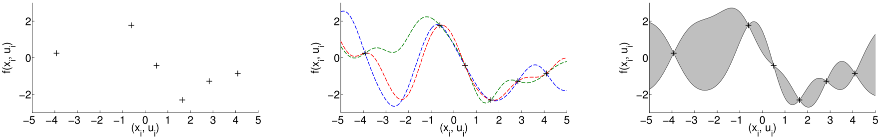
2020/01/01

Outline

- Introduction
- PILCO
- DeepPILCO
- PETS

Motivation

model-based 方法有着较好的 efficiency，但存在一定的局限性：model bias（对于小样本 & 无先验，这个问题更为严重），使用有 bias 的 model 去预测，进而会带来更大的误差。



Solution

学习一个 probabilistic dynamics model，这样的 model 能够 express uncertainty。在 PILCO 中，使用 GP regression 来学习这个 probabilistic dynamics model。

believe 刚才学到的 model，**不做 rollout，而是直接计算**

$$J^{\pi} = \mathbb{E}_{\mathbf{x}_t}\left[c\left(\mathbf{x}_t\right)\right] = \int c\left(\mathbf{x}_t\right)\mathcal{N}\left(\mathbf{x}_t \mid \mu_t, \Sigma_t\right)\mathrm{d}\mathbf{x}_t$$

之后采用基于梯度 ($\frac{\mathrm{d}J}{\mathrm{d}\theta}$) 的 Policy Search，找到最优参数，使得

$$\theta^* = \arg\min_{\theta} J^{\pi_\theta}$$

此时得到 $\pi^* = \pi_{\theta^*}$

Algorithm 1 PILCO

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

Improving PILCO with Bayesian neural network dynamics models

Motivation

GP regression 不适合对复杂环境（e.g. 高维度）建模。

Solution

主要考虑是将 regression 替换为 NN (Neural Network)，但需要解决带来的两个问题：Output Uncertainty & Input Uncertainty

1. Output Uncertainty

将 GP regression 替换为 NN，能够进行更复杂的建模，但一般的 NN 不能 express model uncertainty，考虑使用 BNN (Bayesian Neural Network) 。

2. Input Uncertainty

PILCO 因为使用了 GP regression，因此可以 analytically propagates state distributions through the dynamics model, i.e. 推导出分布 $p(X_0), \ldots, p(X_T)$，这些分布能够表示不确定的 input $X_0, \ldots, X_T$。

但若使用 BNN，他的输入层只能接受确定性的 input，无法表示所需要的不确定性，因此考虑使用 particle methods：

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

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models

Motivation

和上一篇一样在解决 PILCO 的 complexity 问题

Solution

整体的核心思想和上一篇 DeepPILCO 一样：用更复杂的 NN 来代替 GP regression，并解决所带来的 uncertainty 问题。不过这一篇的结果要比上一篇好很多。

1.PE (Probabilistic Ensemble)

1. aleatoric (inherent system stochasticity)

使用 probabilistic NN 来解决 aleatoric uncertainty（上一篇的 output uncertainty）

与 Bayesian NN 不同的是，这篇文章使用了 log prediction probability 来 express uncertainty：

$$\operatorname{loss}_{\mathrm{P}}(\theta)=-\sum_{n=1}^{N} \log \tilde{f}_{\theta}\left(s_{n+1} \mid s_{n}, a_{n}\right)$$

该 NN 的 input 是 s, a，output 为 $\tilde{f}_{\theta}$，是一个 parameterized distribution。

2) epistemic (subjective uncertainty, due to limited data)

使用 Ensemble 来解决 epistemic uncertainty

核心思想就是对多个 Probabilistic NN 进行 ensemble：consider ensembles of B-many bootstrap models, using θ_b to refer to the parameters of our b^{th} model $\tilde{f}_{\theta_b}$, then get $\tilde{f}_\theta = \frac{1}{B}\sum_{b=1}^{B} \tilde{f}_{\theta_b}$.

2.TS (trajectory sampling)

create P particles from the current state, $s_{t=0}^{p} = s_0 \forall p$，每个 particle 都服从分布进行传播：$s_{t+1}^{p} \sim \tilde{f}_{\theta_{b(p,t)}}\left(s_t^p, a_t\right)$，其中 particle 的选择是基于一个 bootstrap $b(p,t) \in \{1, \ldots, B\}$，最终将 P 组 Particles 采集到的结果进行 ensemble。

Algorithm 1 Our model-based MPC algorithm '*PETS*':

Initialize data $\mathbb{D}$ with a random controller for one trial.
for Trial $k = 1$ to K **do**
  Train a *PE* dynamics model $\tilde{f}$ given $\mathbb{D}$.
  for Time $t = 0$ to TaskHorizon **do**
    for Actions sampled $\boldsymbol{a}_{t:t+T} \sim \mathrm{CEM}(\cdot)$, 1 to NSamples **do**
      Propagate state particles $\boldsymbol{s}_\tau^p$ using *TS* and $\tilde{f} | \{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
      Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}_\tau^p, \boldsymbol{a}_\tau)$
      Update $\mathrm{CEM}(\cdot)$ distribution.
    Execute first action $\boldsymbol{a}_t^*$ (only) from optimal actions $\boldsymbol{a}_{t:t+T}^*$.
    Record outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t^*, \boldsymbol{s}_{t+1}\}$.