CS 295: Optimal Control and Reinforcement Learning

Winter 2020

Lecture 10: Model-Based Methods

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Today's lecture

- Motivating model-based methods
- Model-free RL with a model
- Optimal exploration for model learning
- Issues with approximate models
- Fitting models locally

What are model-based methods?

- Any method where we "explicitly" maintain an estimator of the dynamics p
- In table representation: just count parameters
  - Model-free is $O(|S| \cdot |A|)$ while stochastic model-based is $\Omega(|S|^2 \cdot |A|)$
- If features in a Q-network are informative of next state, is that "model-free"
- Not to be confused with ML terminology calling anything learned a "model"

Why model-based methods?

- Dynamics has more parameters, isn't it harder to learn? Usually, no
  - Dynamics can have simpler form and generalize better; and
  - Learned locally, unlike policy or value which encode global knowledge
- Model-based methods produce transferable knowledge
  - If only the task changes, i.e. r changes but not p
  - Can generalize across environment changes, e.g. friction or arm length

How to learn a model

- Interact with environment to get trajectory data — can be off-policy!
  - Often random policy is used
- Deterministic dynamics / reward: MSE loss

$$\mathcal{L}_\phi(s, a, r, s') = \|s' - f_\phi(s, a)\|_2^2 + (r - r_\phi(s, a))^2$$

- Stochastic dynamics: NLL loss

$$\mathcal{L}_\phi(s, a, s') = -\log p_\phi(s'|s, a)$$

- Another possibility discussed later

How to use a learned model

- As a fast simulator
- As an arbitrary-reset simulator
- As a differentiable model

Policy Gradient through the model

- Model is often learned with SGD — must be differentiable

$$\hat{\mathcal{J}}_\theta = \sum_t \gamma^t \hat{c}(x_t, u_t) = \sum_t \gamma^t \hat{c}(\hat{f}(\cdots \hat{f}(x_0, \pi_\theta(x_0)) \cdots, \pi_\theta(x_{t-1})), \pi_\theta(x_t))$$

- This loss function is ill-conditioned for SGD
  - Actions should ideally be coordinated across time steps
  - Perturbing one action individually may change $\hat{\mathcal{J}}_\theta$ unreasonably little / much
    - Vanishing / exploding gradients
  - Second-order methods can help, but for the same reason the Hessian is nasty

PG with a model

- Luckily, we have the Policy Gradient Theorem

$$\nabla_\theta \hat{\mathcal{J}}_\theta = \mathbb{E}_{\xi \sim p_\theta}\left[\sum_t \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{Q}_\theta(s_t, a_t)\right]$$

- Using the model just to compute $\hat{Q}_\theta(s_t, a_t)$, e.g. by MC
  - Avoids complications of gradients through the model

How to use a learned model

- As a fast simulator
- As an arbitrary-reset simulator
- As a differentiable model

Model-free RL with a model

- General scheme:

collect data
train model $\hat{p}, \hat{r}$
repeat
  sample s from the replay buffer
  sample $a|s$ from the learner's policy (or anything else)
  simulate $r = \hat{r}(s, a)$ and $s'|s, a \sim \hat{p}$
  perform model-free RL with (s, a, r, s')

Model-free RL with a model

- n-step on-policy version:

  collect data

  train model $\hat{p}, \hat{r}$

  repeat

  sample s from the replay buffer

  roll out the learner's policy for n steps in the simulator

  perform n-step model-free RL

Dyna

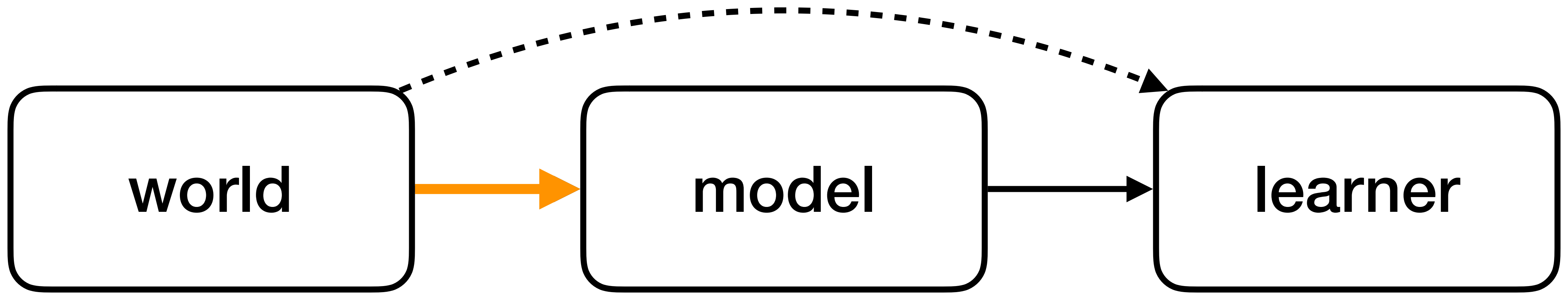
$$
\begin{aligned}
&\text{collect data} \\
&\text{train model } \hat{p}, \hat{r} \\
&\textbf{repeat} \\
&\qquad \text{sample } (s,a) \text{ from the replay buffer} \\
&\qquad \Delta Q(s,a) \leftarrow \hat{r}(s,a) + \gamma\, \mathbb{E}_{s'|s,a\sim\hat{p}}[\max_{a'} Q(s',a')]
\end{aligned}
$$

- Usually a fraction of samples taken from exploration in the real environment
  - Originally: to train the model as we go
  - With function approximation: to feed the replay buffer and reduce covariate shift

Why be model-free if we have the model?

- Learning to control is inherently model-free
  - Remember imitation learning?
  - As opposed to *planning*
- The model still gives benefits
  - It can diversify the experience data, like a replay buffer but more so
  - Incidental: generalization, transfer

Optimal exploration for model learning



- How to explore optimally for learning the model?
- Explicit Explore or Exploit (E[3]):
  - Maintain set $\mathcal{S}_k$ of sufficiently explored states
  - The model $\hat{\mathcal{M}}$ has the empirical transitions and rewards on $\mathcal{S}_k$
  - Other states collapsed to single absorbing state with reward r_{max}
- Principle of **optimism under uncertainty**

Explicit Explore or Exploit (E³)

$\mathcal{S}_k \leftarrow \varnothing$
repeat
    $\pi \leftarrow$ plan in $\hat{\mathcal{M}}$
    if $\Pr(\pi$ reaches absorbing state$) < \epsilon$ **then**
        terminate
    Otherwise
        execute π
        if $s \notin \mathcal{S}_k$ reached **then**
            take least tried action
            if each action tried K times **then**
                empirically estimate $\hat{p}(\cdot|s,\cdot)$, $\hat{r}(s,\cdot)$
                add s to $\mathcal{S}_k$

- When probability to explore is low, optimal policy in $\hat{\mathcal{M}}$ is truly near-optimal
- For provable guarantees, ϵ and K can be determined from $|\mathcal{S}|$
    - Or updated every time the number of visited states is doubled

R-max

- The model $\hat{\mathcal{M}}$ has all states, plus an optimistic absorbing state
- Sufficiently explored states have empirical transitions and rewards
- Others lead w.p. 1 and reward r_{max} to the absorbing state

$$
\begin{array}{l}
\text{mark all states } \textit{unknown} \\
\textbf{repeat} \\
\quad \pi \leftarrow \text{plan in } \hat{\mathcal{M}} \\
\quad \text{execute } \pi \\
\quad \text{record } (s, a, r, s') \text{ in } \textit{unknown} \text{ states} \\
\quad \textbf{if } N(s) = K \textbf{ then} \\
\qquad \text{empirically estimate } \hat{p}(\cdot|s, \cdot), \hat{r}(s, \cdot) \\
\qquad \text{mark } s \textit{ known}
\end{array}
$$

- **Implicit** explore or exploit

Issues with approximate models (1)

- In large state / action spaces, we can only approximate the dynamics
- No guarantees outside of training distribution
  - As in model-free RL, we can't be too far off-policy
- Solution: keep interacting using learner policy and updating the model

Issues with approximate models (2)

- Model inaccuracy accumulates

- If $|p_\phi(s'|s,a) - p(s'|s,a)|_1 \leqslant \epsilon$

  then $|p_\phi(s_t) - p(s_t)|_1 \leqslant \epsilon t$

- We have to plan far enough ahead to realize the consequences of actions

- But we don't have to *execute* those plans far ahead!

- Model Predictive Control (MPC):

$$
\begin{array}{l}
\mathcal{D} \leftarrow \text{collect data} \\
\textbf{repeat} \\
\quad \hat{\mathcal{M}} \leftarrow \text{train model } \hat{p}, \hat{r} \text{ from } \mathcal{D} \\
\quad \textbf{repeat} \\
\quad\quad \pi \leftarrow \text{plan in } \hat{\mathcal{M}} \text{ from current state } s \text{ to horizon } H \\
\quad\quad \text{take } \textit{one action } a \text{ according to } \pi \\
\quad\quad \text{add empirical } (s, a, r, s') \text{ to } \mathcal{D}
\end{array}
$$

How to use a learned model

- As a fast simulator
- As an arbitrary-reset simulator
- As a differentiable model

Local models

- Can we use a learned model for iLQR?
  - Option 1: learn global model, linearize locally — wasteful
  - Option 2: directly learn local linearizations:

initialize a policy $\pi(u_t|x_t)$
repeat
    roll out π to horizon T for N trajectories
    fit $p(x_{t+1}|x_t, u_t)$
    plan new policy π

How to fit local dynamics

- Option 1: linear regression
  - Find $(A_t, B_t)_{t=0}^{T-1}$ such that $x_{t+1} \approx A_t x_t + B_t u_t$
  - Do we care about error / noise?
    - If we assume it's Gaussian, doesn't affect policy; but could help evaluate the method
- Option 2: Bayesian linear regression
  - Use global model as prior
  - More data efficient across time steps and across iterations

How to plan with local models

- Option 1: as in iLQR, find optimal control sequence $\hat{u}$
  - Problem: model errors will cause actual trajectory to diverge
- Option 2: execute the optimal policy $\hat{L}_t \delta x_t + \hat{\ell}_t + \hat{u}_t$ directly in the world
  - Problem: need spread for linear regression, dynamics may be too deterministic
- Option 3: make control stochastic $\hat{L}_t \delta x_t + \hat{\ell}_t + \hat{u}_t + \epsilon_t$
  - Idea: have $\epsilon_t \sim \mathcal{N}(0, R^{-1})$
    - Optimal for the incurred costs, not for the spread needed for regression

Recap

- Roughly two schemes:
  - Plan in a learned model
  - Improve model-free RL using a learned model
- Good theory for how to explore optimally for learning a model