CS 277: Control and Reinforcement Learning

Winter 2021

Lecture 10: Model-Based Methods

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

Logistics

assignments

- Assignment 3 to be published this week
  - Due next Friday
- Assignments 1 + 2 to be graded this weekend

Learning vs. planning

- Model = dynamics + reward function
  - Planning = finding a good policy with access to a model
- Learning = improving performance using data
  - Are rollouts from the model considered "data"?
    - If yes, planning can involve learning
- Model-based learning = methods that explicitly learn the model
  - Unlike planning, access to a model is not given; it is learned

Model-based learning

- Is learning algorithm $\mathscr{A}$ model-based?
- In tabular representation — just count parameters:
  - Model-free = $O(|\mathcal{S}| \cdot |\mathscr{A}|)$ (to represent $\pi(a|s)$ or $Q(s,a)$)
  - Model-based = $\Omega(|\mathcal{S}|^2 \cdots |\mathscr{A}|)$ (to represent $p(s'|s,a)$)
- Not always clear-cut:
  - If intermediate features of DQN $Q_\theta(s,a)$ are informative of s', is this model-free?
- Not to be confused with ML terminology calling anything learned a “model”

Model-based learning: benefits

- Dynamics p has more parameters than $\pi \implies$ harder to learn? Usually, easier
  - p can have simpler form and generalize better to unseen states and actions and
  - p can be learned locally; π or Q encode global knowledge (long-term planning)
- Model-based methods produce transferable knowledge
  - Useful if MDP changes only slightly / partially
    - E.g. only the task changes, i.e. r changes but not p
    - Can generalize across environment changes, e.g. friction or arm length
    - Can help transfer learning in an inaccurate simulator to the real world (sim2real)

How to learn a model

- Interact with environment to get trajectory data
  - Deterministic continuous dynamics / reward: minimize MSE loss

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

  - Stochastic dynamics: minimize NLL loss

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

- Data can be off-policy $\implies$ unbiased estimate, but with covariate shift
  - Random policy is often used
- Another possibility discussed later

How to use a learned model

- Recall how planning benefitted from access to a model:
  - As a fast simulator
  - As an arbitrary-reset simulator
  - As a differentiable model

How to use a learned model

- Recall how planning benefitted from access to a 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

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

- Just do Policy Gradient over $\hat{\mathcal{J}}_\theta$?
  - Chain rule $\Longrightarrow$ back-propagation through time
- Sadly, $\hat{\mathcal{J}}_\theta$ is ill-conditioned for SGD
  - Perturbing one action individually may change $\hat{\mathcal{J}}_\theta$ unreasonably little / much
    - Vanishing / exploding gradients
  - Second-order methods can help, but Hessian is even nastier — for the same reason

PG with a model

- Luckily, we have the Policy Gradient Theorem

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

- Idea: use the model as a fast simulator just to estimate $\hat{Q}_{\bar{\theta}}(s_t, a_t)$
  - E.g., by Monte Carlo
  - Avoids complications of gradients through the model
    - Only backprop through single-step $\log \pi_\theta(a_t | s_t)$

How to use a learned model

- Ways 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 for using a model for model-free RL:

collect data ← interaction with environment (random policy)

train model $\hat{p}, \hat{r}$ ← supervised learning

repeat

    sample s from the replay buffer ← seeded by initial interaction; may interact more as learner improves

    sample $a|s$ from the learner's policy (or anything else)

    simulate $r = \hat{r}(s, a)$ and $s'|s, a \sim \hat{p}$ ← use model as simulator

    perform model-free RL with (s, a, r, s')

- Benefit: get diverse off-policy s, and fresh on-policy a

Model-free RL with a model

- On-policy actions $\implies$ allows n-step estimation without bias:

$$\begin{aligned}
&\text{collect data} \\
&\text{train model } \hat{p}, \hat{r} \\
&\textbf{repeat} \\
&\quad \text{sample } s \text{ from the replay buffer} \\
&\quad \text{roll out the learner's policy for } n \text{ steps in the simulator} \\
&\quad \text{perform } n\text{-step model-free RL}
\end{aligned}$$

- $\hat{r}(s_t, a_t) + \gamma \hat{r}(\hat{s}_{t+1}, a_{t+1}) + \cdots + \gamma^{n-1} \hat{r}(\hat{s}_{t+n-1}, a_{t+n-1})$ is unbiased
  - Except for model inaccuracy

Dyna

collect data

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

repeat

sample (s, a) from the replay buffer

$$\Delta Q(s,a) \leftarrow \hat{r}(s,a) + \gamma\, \mathbb{E}_{s'|s,a\sim\hat{p}}[\max_{a'} Q(s',a')] \quad - Q(s,a)$$

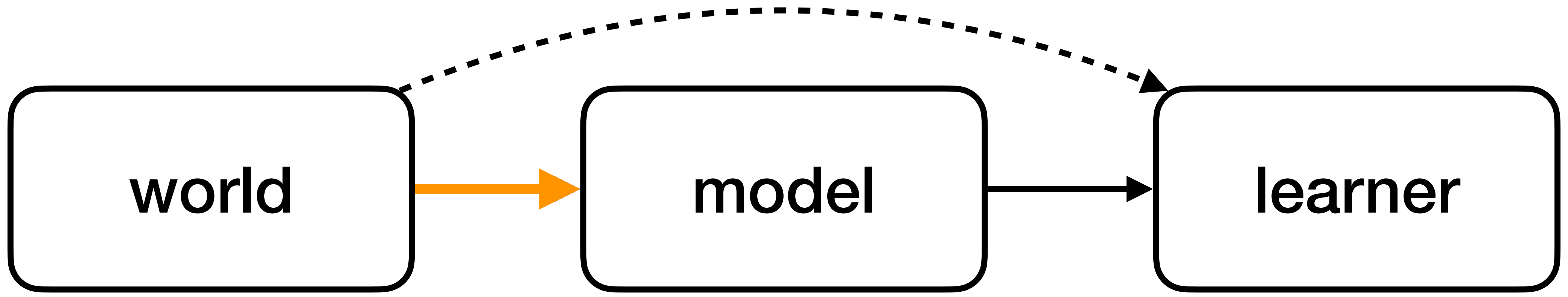
use model as simulator to estimate

- Improvement: mix in samples generated from learner interactions
  - Original benefit: keep training the model to be good for states that learner sees
  - With function approximation: feed the replay buffer and reduce covariate shift

Wait... Model-free RL... *with* a model?

- Why be model-free if we have the model?
- Learning to control is inherently model-free
    - Policy gradient is 0 for the $\log p(s' | s, a)$ term of $\log p_\theta(\xi)$
    - Same in Imitation Learning: optimize NLL $\mathscr{L}_\theta(s_t, a_t) = -\log \pi_\theta(a_t | s_t)$
    - As opposed to planning, which requires averaging over futures
- 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 (E³):
  - 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)

$\mathcal{S}_k \leftarrow \emptyset$
repeat
  $\pi \leftarrow$ plan in $\hat{\mathcal{M}}$
  if $\Pr(\pi \text{ reaches absorbing state}) < \epsilon$ **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 $\hat{\mathscr{M}}$ has all states, plus an optimistic absorbing state
- Sufficiently explored states have empirical transitions and rewards
- Others lead with probability 1 and reward $r_{\max}$ to the absorbing state

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

- 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 $|p_\phi(s'|s,a) - p(s'|s,a)|_1 \leq \epsilon$ then $|p_\phi(s_t) - p(s_t)|_1 \leq \epsilon t$
  - 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):

$\mathcal{D} \leftarrow$ collect data
repeat
  $\hat{\mathcal{M}} \leftarrow$ train model $\hat{p}, \hat{r}$ from $\mathcal{D}$
  repeat
    $\pi \leftarrow$ plan in $\hat{\mathcal{M}}$ from current state s to horizon H
    take *one action* a according to π
    add empirical (s, a, r, s') to $\mathcal{D}$

How to use a learned model

- Recall how planning benefitted from access to a 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 $\implies$ wasteful
  - Option 2: directly learn local linearizations:

$$
\begin{aligned}
&\text{initialize a policy } \pi(u_t|x_t) \\
&\textbf{repeat} \\
&\quad \text{roll out } \pi \text{ to horizon } T \text{ for } N \text{ trajectories} \\
&\quad \text{fit } p(x_{t+1}|x_t, u_t) \\
&\quad \text{plan new policy } \pi
\end{aligned}
$$

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