CS 287 Advanced Robotics – Fundamental Knowledge

Pieter Abbeel, Laura Smith, Ignasi Clavera, Harry Xu

November 9, 2019

This document contains all the possible exam questions and the answers for CS287 Fall 2019. The exam will be closed-book. There will be no clarification questions during the exam, because this handout already clarifies everything and is yours to study.

Happy studying!

1 Value Iteration

Q: For an MDP $(S, A, P(s'|s,a), R(s,a), \gamma, H)$, (a) Write the objective to be optimized; (b) Write out the Value Iteration algorithm; (c) Describe in words the meaning of $V_i^*(s)$ and $\pi_i^*(s)$; (d) Write out the Bellman equation, satisfied by V^* at convergence.

A:

(a)

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t) \mid \pi\right]$$

(b)

set $V_0^*(s) = 0$ for all states s

for $i = 1, \ldots, H$

for all states s:

$(V_i^*(s), \pi_i^*(s)) \leftarrow \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_{i-1}^*(s')\right]$

(c)

$V_i^*(s)$ = expected sum of rewards accumulated starting from state s, if acting optimally for i steps

$\pi_i^*(s)$ = optimal action when in state s and getting to act for i steps

(d)

$$V^*(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')\right]$$

2 Policy Iteration

Q: For an infinite horizon MDP $(S, A, P(s'|s,a), R(s,a), \gamma)$, (a) Write the objective to be optimized; (b) Write out the Policy Iteration algorithm; (c) Provide intuition why the policy improvement step indeed improves the policy.

A:

(a)

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid \pi\right]$$

(b)

randomly initialize the policy $\pi_0(s)$

for $k = 0, 1, \ldots$ (till convergence)

Evaluate the current policy π_k:

init $V^{\pi_k}(s)$ for all states s (any init OK, most efficient: $V^{\pi_{k-1}}$)

for $i = 1, 2, \ldots$ (till convergence)

for all states s:

$V^{\pi_k}(s) \leftarrow R(s, \pi_k(s)) + \gamma \sum_{s'} P(s'|s, a) V^{\pi_k}(s')$

Improve upon the current policy through one-step lookahead:

$\pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^{\pi_k}(s')$

(c)

First, let's consider the non-stationary policy which at the first time step follows π_{k+1} and then onwards follows π_k. This non-stationary policy is at least as good as π_k, because by definition it takes the optimal first action assuming using π_k then onwards. In contrast, π_k doesn't get to optimize that first action.

Now, since we have a stationary MDP, on the second time step, we are faced with the exact same problem as at the first time step. This means that at the second time step we're also better off executing π_{k+1} (rather than π_k).

This same reasoning applies to all future time steps, hence applying π_{k+1} at all time steps is better than π_k. (That is, until convergence has been achieved, at which point $\pi_{k+1} = \pi_k$.)

3 Max-Ent Value Iteration

Q: (a) Consider the 1-step max-ent problem:

$$V = \max_{\pi(a)} \ (\mathbb{E}\left[r(a)\right] + \beta\mathcal{H}(\pi(a)))$$

(i) Write out the dual formulation; (ii) Write out the optimality conditions and derive the solution for $\pi(a)$; (iii) Fill the solution for $\pi(a)$ into the objective and simplify to get the expression for max-ent V.

(b) Now consider max-ent value iteration:

$$V_k(s) = \max_{\pi} \ (\mathbb{E}\left[R(s,a) + V_{k-1}(s')\right] + \beta\mathcal{H}(\pi(a|s)))$$

Using the results from (a), write out a closed-form expression for $V_k(s)$ and for $\pi_k(a|s)$.

A:

(a)(i)

$$\max_{\pi(a)} \min_{\lambda} \ \mathcal{L}(\pi(a),\lambda) = \max_{\pi(a)} \min_{\lambda} \sum_a \pi(a)r(a) - \beta \sum_a \pi(a)\log\pi(a) + \lambda(\sum_a \pi(a) - 1)$$

(a)(ii)

$$\frac{\partial}{\partial\lambda}\mathcal{L}(\pi(a),\lambda) = 0$$
$$\textstyle\sum_a \pi(a) - 1 = 0$$

$$\frac{\partial}{\partial\pi(a)}\mathcal{L}(\pi(a),\lambda) = 0$$
$$\frac{\partial}{\partial\pi(a)} \textstyle\sum_a \pi(a)r(a) - \beta\sum_a \pi(a)\log\pi(a) + \lambda(\sum_a \pi(a) - 1) = 0$$
$$r(a) - \beta\log\pi(a) - \beta + \lambda = 0$$
$$\beta\log\pi(a) = r(a) - \beta + \lambda$$
$$\pi(a) = \exp\left[\tfrac{1}{\beta}(r(a) - \beta + \lambda)\right]$$
$$\pi(a) = \tfrac{1}{Z}\exp(\tfrac{1}{\beta}r(a))$$

$$Z = \sum_a \exp(\frac{1}{\beta}r(a))$$

(a)(iii)

$$\begin{aligned}
V = & \quad \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) r(a) - \beta \sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \log \left(\frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \right) \\
= & \quad \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \left(r(a) - \beta \log \left(\exp(\frac{1}{\beta} r(a)) \right) \right) - \beta \sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \log \frac{1}{Z} \\
= & \quad 0 - \beta \log \frac{1}{Z} \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \\
= & \quad -\beta \log \frac{1}{Z} \\
= & \quad \beta \log \textstyle\sum_a \exp(\frac{1}{\beta} r(a))
\end{aligned}$$

(b)

$$\begin{aligned}
V_k(s) = & \quad \max_\pi \mathbb{E} \left[r(s,a) + \beta \mathcal{H}(\pi(a|s) + V_{k-1}(s') \right] \\
= & \quad \max_\pi \mathbb{E} \left[Q_k(s,a) + \beta \mathcal{H}(\pi(a|s) \right]
\end{aligned}$$

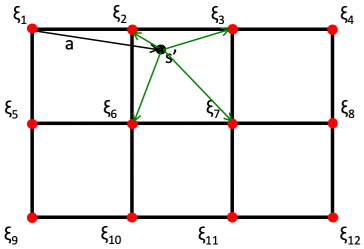
$$V_k(s) = \beta \log \sum_a \exp(\frac{1}{\beta} Q_k(s,a))$$

$$\pi_k(a|s) = \frac{1}{Z} \exp(\frac{1}{\beta} Q_k(s,a))$$

4 Solving Continuous MDPs with Discretization

Q: (a) Sketch how to solve Continuous MDPs with Discretization with stochastic mapping onto all direct neighbors. (b) Discuss why this often outperforms mapping onto the single nearest neighbor.

A:



Discrete states: $\{\xi_1, ..., \xi_{12}\}$

$$P(\xi_2 \mid \xi_1, a) = p_A$$
$$P(\xi_3 \mid \xi_1, a) = p_B$$
$$P(\xi_6 \mid \xi_1, a) = p_C$$
$$P(\xi_7 \mid \xi_1, a) = p_D$$
$$\text{s.t. } s' = p_A\xi_2 + p_B\xi_3 + p_C\xi_6 + p_D\xi_7$$

The two main challenges with single nearest neighbor are (i) that small actions can have zero effect on the discretized state; (ii) that solutions found in the discretized MDP might take advantage of the very particular deterministic properties it contains.

5 Cross Entropy Method

Q: Describe the Cross-Entropy Method to solve $max_x f(x)$, assuming $x \in \mathbb{R}^n$. Discuss what are the hyperparameters. Discuss how it can also be applied when $x \in \{0,1\}^n$.

A:

$$\max_x f(x)$$

CEM:

sample $\mu^{(0)} \sim \mathcal{N}(0, \sigma^2)$

for iter i = 1, 2, ...

for e = 1, 2. ...

sample $x^{(e)} \sim \mathcal{N}(\mu^{(i)}, \sigma^2)$

compute $f(x^{(e)})$

endfor

$\mu^{(i+1)} = \text{mean}\{x^{(e)} : f(x^{(e)}) \text{ in top } 10\%\}$

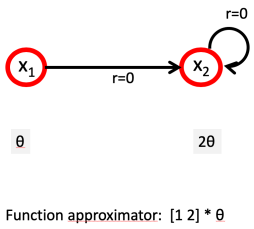
- sigma and 10% are hyperparameters
- can in principle also fit sigma to top 10% (or full covariance matrix if low-D)
- How about discrete action spaces?
  - Within top 10%, look at frequency of each discrete action in each time step, and use that as probability
  - Then sample from this distribution

6 Challenges with Approximate Value Iteration

Q: (a) Describe the Tsitsiklis & Van Roy 2-state example MDP showing that value iteration with least squares function approximation can diverge. (b) Draw a simple scenario of two operators, where the first operator (T_1) is a contraction in the infinity-norm, and the second operator (T_2) is a contraction in the two-norm, but alternating them leads to divergence.

A:

(a)



$$\bar{J}_\theta = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \theta$$

$$\begin{aligned} \bar{J}^{(1)}(x_1) &= 0 + \gamma \hat{J}_{\theta^{(0)}}(x_2) = 2\gamma\theta^{(0)} \\ \bar{J}^{(1)}(x_2) &= 0 + \gamma \hat{J}_{\theta^{(0)}}(x_2) = 2\gamma\theta^{(0)} \end{aligned}$$

Function approximation with least squares fit:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \theta^{(1)} \approx \begin{bmatrix} 2\gamma\theta^{(0)} \\ 2\gamma\theta^{(0)} \end{bmatrix}$$

Least squares fit results in:

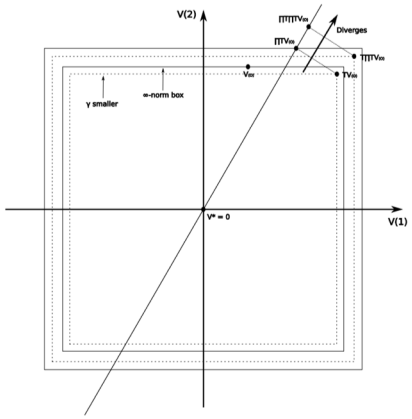
$$\theta^{(1)} = \frac{6}{5}\gamma\theta^{(0)}$$

Repeated back-ups and function approximations result in:

$$\theta^{(i)} = \left(\frac{6}{5}\gamma\right)^i \theta^{(0)}$$

which diverges if $\gamma > \frac{5}{6}$ even though the function approximation class can represent the true value function.]

(b)



7 LQR

Q: (a) Describe the LQR setting in standard notation, (b) Derive the LQR update equations

A:

(a)

The LQR setting assumes a linear dynamical system:

$$x_{t+1} = Ax_t + Bu_t,$$

x_t: state at time t
u_t: input at time t
It assumes a quadratic cost function:

$$g(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t$$

with $Q \succ 0, R \succ 0$.

(b)

Initialize $J_0(x) = x^\top P_0 x$.

$$\begin{aligned} J_1(x) &= \min_u \left[x^\top Q x + u^\top R u + J_0(Ax + Bu)\right] \\ &= \min_u \left[x^\top Q x + u^\top R u + (Ax + Bu)^\top P_0 (Ax + Bu)\right] \quad (1) \end{aligned}$$

To find the minimum over u, we set the gradient w.r.t. u equal to zero:

$$\nabla_u [\ldots] = 2Ru + 2B^\top P_0 (Ax + Bu) = 0,$$

$$\text{hence: } u = -(R + B^\top P_0 B)^{-1} B^\top P_0 A x \quad (2)$$

$$\begin{aligned} \text{(2) into (1): } J_1(x) &= x^\top P_1 x \\ \text{for: } P_1 &= Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1) \\ K_1 &= -(R + B^\top P_0 B)^{-1} B^\top P_0 A. \end{aligned}$$

8 Gradient Descent and Newton's Method

Q: (a) Describe the Gradient Descent algorithm ; (b) Discuss the notion of well-conditioned versus poorly-conditioned problems by considering gradient descent on the objective $f(x) = \frac{1}{2}\left(x_1^2 + \gamma x_2^2\right)$ for different choices of γ; (c) Derive the Newton Step; (d) Discuss how Newton's method fares on the objective from (b).
A: (a)

Initialize x

Repeat

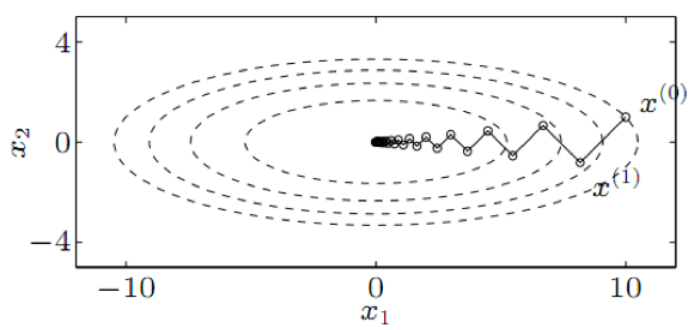
Compute the gradient $\nabla f(x)$

Line search: find a good step size $t > 0$

Update: $x \leftarrow x - t\nabla f(x)$

Until stopping criterion satisfied

(b)



This problem is well conditioned for $\gamma \approx 1$. This problem is poorly conditioned for $\gamma >> 1$ and for $0 < \gamma << 1$.

(c) Newton's method considers the local 2nd order Taylor approximation:

$$f(x + \Delta x) \approx f(x) + \nabla f(x)^\top \Delta x + \frac{1}{2}\Delta x^\top \nabla^2 f(x) \Delta x$$

Assuming $\nabla^2 f(x) \succ 0$ (which is true for convex f), the minimum of the second-order approximation is found at:

$$\Delta x = -(\nabla^2 f(x))^{-1} \nabla f(x)$$

which is called the Newton step.

(d) Since the objective under (b) is a quadratic objective, the Newton step will take us directly to the optimum, independent of the conditioning (i.e. independent of the value of γ).

9 Natural Gradient

Q: (a) Derive the natural gradient for a maximum likelihood problem

$$\max_\theta f(\theta) = \max_\theta \sum_i \log p(x^{(i)}; \theta)$$

and through your derivation show how it's different from the Hessian. (b) List the benefits

A:

(a)

- Gradient: $\frac{\partial f(\theta)}{\partial \theta_p} = \sum_i \frac{\partial \log p(x^{(i)};\theta)}{\partial \theta_p} = \sum_i \frac{\partial p(x^{(i)};\theta)}{\partial \theta_p} \frac{1}{p(x^{(i)};\theta)}$
- Hessian: $\frac{\partial^2 f(\theta)}{\partial \theta_q \partial \theta_p} = \sum_i \frac{\partial^2 p(x^{(i)};\theta)}{\partial \theta_q \partial \theta_p} \frac{1}{p(x^{(i)};\theta)} - \frac{\partial p(x^{(i)};\theta)}{\partial \theta_q} \frac{1}{p(x^{(i)};\theta)} \frac{\partial p(x^{(i)};\theta)}{\partial \theta_p} \frac{1}{p(x^{(i)};\theta)}$

  $\nabla^2 f(\theta) = \sum_i \frac{\nabla^2 p(x^{(i)};\theta)}{p(x^{(i)};\theta)} - \left(\nabla \log p(x^{(i)};\theta)\right)\left(\nabla \log p(x^{(i)};\theta)\right)^\top$
- Natural gradient: $= \left(\sum_i \left(\nabla \log p(x^{(i)};\theta)\right)\left(\nabla \log p(x^{(i)};\theta)\right)^\top\right)^{-1} \left(\sum_i \nabla \log p(x^{(i)};\theta)\right)$

(b)

1. faster to compute than Hessian (only gradients needed)
2. guaranteed to be negative definite
3. found to be superior to exact Hessian in various experiments
4. invariant to reparameterization of the distribution p

10 Constrained Optimization: Penalty Method

Q: Consider the constrained optimization problem:

$$\begin{aligned} \min_x \quad & g_0(x) \\ \text{s.t.} \quad & g_i(x) \leq 0 \ \forall i \\ & h_j(x) = 0 \ \forall j \end{aligned}$$

(a) Write out the penalty formulation and annotate what is the "merit function"; (b) Describe the general Penalty Method; (c) Describe the Penalty Method with Trust Region Inner Loop; (d) Describe Dual Descent

A:

(a)

$$\min_x \quad g_0(x) + \mu \sum_i |g_i(x)|^+ + \mu \sum_j |h_j(x)| = \min_x f_\mu(x)$$

$f_\mu(x)$ is the merit function

(b)

Inner loop: optimize merit function over x, which could be done with gradient descent, Newton or quasi-Newton, or trust region methods

Outer loop: increase μ

exit when constraints are satisfied after completion of inner loop

(c) The trust region inner loop will repeatedly solve:

$$\begin{aligned} \min_x \quad & g_0(\bar{x}) + \nabla_x g_0(\bar{x})(x - \bar{x}) + \mu \textstyle\sum_i |g_i(\bar{x}) + \nabla_x g_i(\bar{x})(x - \bar{x})|^+ + \mu \sum_j |h_j(\bar{x}) + \nabla_x h_j(\bar{x})(x - \bar{x})| \\ \text{s.t.} \quad & \|x - \bar{x}\|_2 \leq \varepsilon \end{aligned}$$

(d)

$$\max_{\lambda \geq 0, \nu} \min_x g_0(x) + \sum_i \lambda_i g_i(x) + \sum_j \nu_j h_j(x)$$

Dual Descent iterates over:

1. Optimize over x
2. Gradient descent step for λ and ν:

   $\lambda_i \leftarrow \lambda_i + \alpha g_i(x)$ (clip to zero to keep positive)
   $\nu_j \leftarrow \nu_j + \alpha h_j(x)$

11 Optimization for Control

Q: (a) Write out the breakdown of optimal control formulations across (i) open-loop vs. closed-loop; (ii) shooting vs. collocation in 2x2 table. (b) What is a key benefit of collocation? (c) What is a key benefit of shooting?

A:

(a)

	Return open-loop controls $u_0, u_1, \ldots, u_H$	Return feedback policy $\pi_\theta(\cdot)$ (e.g. linear or neural net)
shooting	$\min_{u_0,u_1,\ldots,u_H} c(x_0,u_0) + c(f(x_0,u_0),u_1) + c(f(f(x_0,u_0),u_1),u_2) + \ldots$	$\min_\theta c(x_0,\pi_\theta(x_0)) + c(f(x_0,\pi_\theta(x_0)),\pi_\theta(f(x_0,\pi_\theta(x_0)))) + \ldots$
collocation	$\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H} \sum_{t=0}^H c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t) \ \forall t$	$\min_{x_0,x_1,\ldots,x_H,\theta} \sum_{t=0}^H c(x_t,\pi_\theta(x_t))$ s.t. $x_{t+1} = f(x_t,\pi_\theta(x_t)) \ \forall t$ $\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H,\theta} \sum_{t=0}^H c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t) \ \forall t$, $u_t = \pi_\theta(x_t) \ \forall t$

(b) In shooting, the influence of the control inputs propagates directly over time, often making for a very poorly conditioned problem. In collocation, by having the state as optimization variable, the influence is decoupled into per-time-step influence, improving conditioning.

(c) Collocation can get stuck in infeasible local optima, whereas with shooting the solution is always something that can be executed.

12 Rapidly Exploring Random Trees

Q: Write out the RRT algorithm

A:

GENERATE_RRT($x_{init}, K, \Delta t$)
1 $\mathcal{T}$.init(x_{init});
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 $\mathcal{T}$.add_vertex(x_{new});
8 $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
9 Return $\mathcal{T}$

RANDOM_STATE(): often uniformly at random over space with probability 99%, and the goal state with probability 1%, this ensures it attempts to connect to goal semi-regularly
SELECT_INPUT(): often a few inputs are sampled, and one that results in x_new closest to x_rand is retained; sometimes optimization is run to find the best input

13 Bayes' Filters and Particle Filters

Q: (a) Write out the Bayes' filter Time Update and Measurement Update equations; (b) Write out the particle filter algorithm. In addition to the pseudo-code, include brief comments about what's happening in each line.

A:

(a) Time Update:

$$P(x_{t+1}|z_0,\ldots,z_t) = \sum_{x_t} P(x_{t+1}|x_t)P(x_t|z_0,\ldots,z_t)$$

Measurement Update:

$$P(x_{t+1}|z_0,\ldots,z_t,z_{t+1}) = \frac{1}{Z}P(x_{t+1}|z_0,\ldots,z_t)P(z_{t+1}|x_{t+1})$$

(b)

1. Algorithm **particle_filter**(S_{t-1}, u_t, z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1,\ldots,n$ — *Generate new samples*
4. Sample index $j(i)$ from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_t)$ using $x_{t-1}^{j(i)}$ and u_t
6. $w_t^i = p(z_t \mid x_t^i)$ — *Compute importance weight*
7. $\eta = \eta + w_t^i$ — *Update normalization factor*
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ — *Add to new particle set*
9. **For** $i = 1,\ldots,n$
10. $w_t^i = w_t^i / \eta$ — *Normalize weights*

14 POMDPs

Q: (a) Draw the POMDP light-dark domain; (b) Draw the state-space plan the certainty-equivalent policy would come up with (which plans purely in state space assuming the mean estimate is the actual state); (c) Draw what the optimal solution would do, which can be found by planning in belief space.

A:

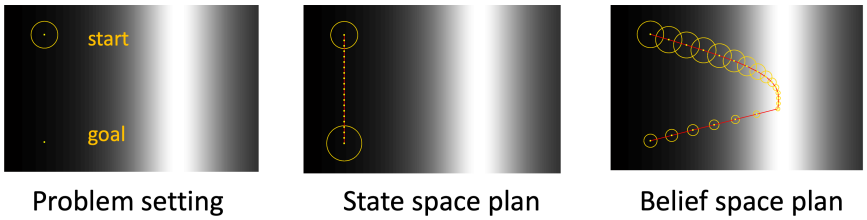


Figure 1: From left to right: (a), (b), (c)

15 Imitation Learning

Q: (a) Describe Behavioral Cloning; (b) Describe distribution shift, why it happens, and its effect; (c) Describe DAgger; (d) Describe the problem setting of Inverse Optimal Control

A:

(a) Given demonstration data (o_t, a_t) where o_t is the observation at time t and a_t the action taking by the demonstrator at time t, behavioral cloning runs supervised learning to learn the mapping from observation to action.

(b) Distribution shift refers to encountering a different distribution over observations at test time compared to training time. In behavioral cloning this happens because the learned policy doesn't perfectly match the demonstration policy. In turn, this means the learned policy is applied to observations it wasn't trained for, which can often be problematic.

(c) DAgger corrects for distribution shift by collecting expert data under the distribution over observations currently encountered by the learned policy.

(d)

Given:

State space, Action space

Transition model

Demonstrations (samples from π^*)

Goal:

Learn reward function $R(s, a)$ that best explaines the demonstrations

16 Policy Gradients

Q: (a) Derive the likelihood ratio policy gradient starting from the objective

$$U(\theta) = \sum_{\tau} P(\tau;\theta)R(\tau)$$

(b) Introduce the temporal aspect, to get an expression for $\nabla_\theta \log P(\tau^{(i)};\theta)$ that shows the dynamics model is not required to compute the policy gradient; (c) Write out the vanilla policy gradient expression with value function baseline.

A:

(a)

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \textstyle\sum_\tau P(\tau;\theta)R(\tau) \\
&= \textstyle\sum_\tau \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \textstyle\sum_\tau P(\tau;\theta)\frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)}R(\tau) \\
&= \textstyle\sum_\tau P(\tau;\theta)\nabla_\theta \log P(\tau;\theta)R(\tau)
\end{aligned}$$

$$\nabla_\theta U(\theta) \approx \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)};\theta)R(\tau^{(i)})$$

(b)

$$\begin{aligned}
\nabla_\theta \log P(\tau^{(i)};\theta) &= \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)}|s_t^{(i)},u_t^{(i)})}_{\text{dynamics model}} \cdot \underbrace{\pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{policy}}\right] \\
&= \nabla_\theta \left[\sum_{t=0}^{H} \log P(s_{t+1}^{(i)}|s_t^{(i)},u_t^{(i)}) + \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\right] \\
&= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \\
&= \sum_{t=0}^{H} \underbrace{\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{no dynamics model required!!}}
\end{aligned}$$

(c)

$$\nabla_\theta U(\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \left(\sum_{k=t}^{H-1} R(s_k^{(i)},u_k^{(i)}) - V_\phi(s_t)\right)$$

17 TRPO and PPO

Q: (a) Describe the TRPO surrogate objective; (b) Describe the PPO v1 surrogate objective; (c) Describe the PPO v2 surrogate objective

A:

(a)

$$\begin{aligned} \max_{\theta} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \\ \text{s.t.} \quad & \hat{\mathbb{E}}_t \left[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right] \leq \delta \end{aligned}$$

(b)

$$\min_{\beta \geq 0} \max_{\theta} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] - \beta \left(\hat{\mathbb{E}}_t \left[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right] - \delta \right)$$

Optimize by running dual descent, which alternatives over gradient steps for θ to maximize and gradient steps over β to minimize

(c) Let:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

Then Optimize the surrogate loss:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_t \right) \right]$$

18 Q-Learning (forthcoming)

Q: (a) Describe the tabular Q-learning algorithm; (b) Describe the main one-line change for Q-learning with function approximation.

A:

(a)

- Init $Q(s, a)$ [anything is fine, all 0 can make sense; optimistic can also make sense]
- Get initial state s
- Iterate (till convergence)
  - SAMPLE: sample action a, get reward r and next state s'
  - COMPUTE TARGET VALUE:
    - IF s' is terminal:
      - target $= r$
      - Sample new initial state s'
    - ELSE:
      - target $= r + \gamma \max_{a'} Q(s', a')$
  - UPDATE Q: $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha$ target
  - $s \leftarrow s'$

(b)

The UPDATE now becomes an update of a parameterized function Q_θ, often done in small batches. For a single sample we'd take one (or more) steps on this objective:

$$\min_\theta \left(Q_\theta(s, a) - \text{target}\right)^2$$

19 DDPG and SAC

Q: (a) Described DDPG; (b) Describe SAC

A:

(a)

- ROLL-OUTS: collect data under current policy, add data in replay buffer $\mathcal{D}$
- LEARNING UPDATES (loss functions on which to take a few steps):

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}} \left(r_t + \gamma Q_\theta(s_t, \pi_\phi(s_t)) - Q_\theta(s_t, a_t)\right)^2$$

$$J_\pi(\phi) = -\mathbb{E}_{s_t\sim\mathcal{D}} \left[Q_\theta(s_t, \pi_\phi(s_t, z_t)\right]$$

Note: in practice there are two Q functions for stabilization.

(b)

- ROLL-OUTS: collect data under current policy π_ϕ, add data in replay buffer $\mathcal{D}$
- LEARNING UPDATES (loss functions on which to take a few steps):

$$J_V(\psi) = \mathbb{E}_{s_t\sim\mathcal{D}} \left[\left(V_\psi(s_t) - \mathbb{E}_{a_t\sim\pi_\phi}[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t|s_t)]\right)^2\right]$$

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}} \left(r_t + \gamma V_\psi(s_{t+1}) - Q_\theta(s_t, a_t)\right)^2$$

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}} \left[D_{\mathrm{KL}} \left(\pi_\phi(\cdot|s_t) \| \frac{\exp(Q_\theta(s_t,\cdot)}{Z_\theta(s_t)}\right)\right]$$

Note: in practice there are two Q-functions and two V-functions for stabilization.

20 Model-based RL

Q: (a) Describe canonical model-based RL; (b) What is the model-bias problem in model-based RL?

A:

(a)

Iterate

- Collect data under current policy
- Learn dynamics model from all data collected so far
- Improve policy by using the learned dynamics model

(b) Policy optimization using the learned dynamics model can result in exploiting regions where insufficient data was available to support the learned dynamics model, which can lead to catastrophic failures when the policy that looks great in simulation gets deployed in real world.