Sequential Decision Making

Lecture 5 : Beyond Value-Based Methods

Emilie Kaufmann

Ecole Centrale Lille, 2021/2022

Reminder

Until now we have seen Value-Based methods , that learn

$$Q(s, a)$$

an estimate of the optimal Q-Value function

$$\begin{aligned} Q^\star(s, a) &= \max_\pi Q^\pi(s, a) \\ &= \max_\pi \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \middle| s_1 = s, a_1 = a \right] \end{aligned}$$

➜ our guess for the optimal policy is then $\pi = \text{greedy}(Q)$:

$$\pi(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \; Q(s, a)$$

(a deterministic policy)

Outline

Optimizing over policies ?

We could try to

$$\underset{\pi \in \Pi}{\text{argmax}} \ \mathbb{E}^{\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \middle| s_1 \sim \rho\right]$$

where

$$\Pi = \{\text{stationary, deterministic policies } \pi : \mathcal{S} \to \mathcal{A}\}$$

and ρ is a distribution over first states.

➜ intractable !

Idea : relax this optimization problem by searching over a (smoothly) parameterized set of stochastic policies.

A new objective

- parametric family of stochastic policies $\{\pi_\theta\}_{\theta \in \Theta}$
- $\pi_\theta(a|s)$: probability of choosing a in s, given θ
- $\theta \mapsto \pi_\theta(a|s)$ is assumed to be differentiable

Goal : find θ that maximizes

$$J(\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \middle| s_1 \sim \rho\right]$$

over the parameter space Θ.

Idea : use gradient ascent

➜ How to compute the gradient $\nabla_\theta J(\theta)$?

➜ How to estimate it using trajectories ?

Warm-up : Computing gradients

- $f : \mathcal{X} \to \mathbb{R}$ is a (non differentiable) function
- $\{p_\theta\}_{\theta\in\Theta}$ is a set of probability distributions over $\mathcal{X}$

$$J(\theta) = \mathbb{E}_{X\sim p_\theta}\left[f(X)\right]$$

Proposition

$$\nabla_\theta J(\theta) = \mathbb{E}_{X\sim p_\theta}\left[f(X)\nabla \log p_\theta(X)\right]$$

Exercice : Prove it !

Outline

Finite-Horizon objective

$$J(\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \middle| s_1 \sim \rho\right]$$

for some $\gamma \in (0, 1]$.

- $\tau = (s_1, a_1, s_2, a_2, \dots, s_T, a_T)$ trajectory of length T
- π_θ induces a distribution p_θ over trajectories :

$$p_\theta(\tau) = \rho(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

- cumulative discounted reward over the trajectory :

$$R(\tau) := \sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)$$

Finite-Horizon objective

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[R(\tau)\right]$$

for some $\gamma \in (0, 1]$.

- $\tau = (s_1, a_1, s_2, a_2, \dots, s_T, a_T)$ trajectory of length T
- π_θ induces a distribution p_θ over trajectories :

$$p_\theta(\tau) = \rho(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

- cumulative discounted reward over the trajectory :

$$R(\tau) := \sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)$$

Computing the gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[R(\tau)\nabla_\theta \log p_\theta(\tau)\right]$$

and

$$\begin{aligned}
\nabla_\theta \log p_\theta(\tau) &= \nabla_\theta \log\left(\rho(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)\right)\\
&= \nabla_\theta \sum_{t=1}^{T}\left(\log\rho(s_1) + \log p(s_{t+1}|s_t,a_t) + \log \pi_\theta(a_t|s_t)\right)\\
&= \sum_{t=1}^{T}\nabla_\theta \log\pi_\theta(a_t|s_t)
\end{aligned}$$

Hence,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\sum_{t=1}^{T} R(\tau)\nabla_\theta \log\pi_\theta(a_t|s_t)\right]$$

The baseline trick

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\sum_{t=1}^{T} R(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

In each step t, we may substrack a baseline function $b_t(s_1, a_1, \dots, s_t)$, which depends on the beginning of the trajectory (up to s_t), i.e.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\sum_{t=1}^{T} \left(R(\tau) - b_t(s_1, a_1, \dots, s_t)\right)\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

Why ?

$$\begin{aligned}
&\mathbb{E}_{\tau \sim p_\theta}\left[b_t(s_1, a_1, \dots, s_t)\nabla_\theta \log \pi_\theta(a_t|s_t) | s_1, a_1, \dots, s_t\right] \\
&= b_t(s_1, a_1, \dots, s_t) \sum_{a \in \mathcal{A}} \pi_\theta(a|s_t)\nabla_\theta \log \pi_\theta(a|s_t) \\
&= b_t(s_1, a_1, \dots, s_t) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a|s_t) \\
&= b_t(s_1, a_1, \dots, s_t)\nabla_\theta \underbrace{\left(\sum_{a \in \mathcal{A}} \pi_\theta(a|s_t)\right)}_{=1} = 0
\end{aligned}$$

Choosing a baseline

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\sum_{t=1}^{T} \left(R(\tau) - b_t(s_1, a_1, \dots, s_t)\right) \nabla_\theta \log \pi_\theta(a_t | s_t)\right]$$

A common choice is

$$b_t(s_1, a_1, \dots, s_t) = \sum_{i=1}^{t-1} \gamma^{t-1} r(s_i, a_i)$$

which leads to

$$\begin{aligned} R(\tau) - b_t(s_1, a_1, \dots, s_t) &= \sum_{i=t}^{T} \gamma^{i-1} r(s_i, a_i) \\ &= \gamma^{t-1} \underbrace{\sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)}_{\text{discounted sum of rewards starting from } s_t} \end{aligned}$$

Policy Gradient Theorem

Using this baseline, we obtain

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=1}^{T}\gamma^{t-1}\left(\sum_{i=t}^{T}\gamma^{i-t}r(s_i,a_i)\right)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \\ &= \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{T}\gamma^{t-1}Q_t^{\pi_\theta}(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right]\end{aligned}$$

where

$$Q_t^{\pi}(s,a) = \mathbb{E}^{\pi}\left[\sum_{i=t}^{T}\gamma^{i-t}r(s_i,a_i)\middle| s_t = s, a_t = a\right]$$

Policy Gradient Theorem : Infinite Horizon

$$J(\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \middle| s_1 \sim \rho\right]$$

(taking the limit when $T \to \infty$ of the previous objective)

Policy Gradient Theorem [Sutton et al., 1999]

$$\nabla_\theta J(\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(s_t|a_t)\right]$$

where $Q^\pi(s, a)$ is the usual Q-value function of policy π.

Remark : sometimes written

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{(s,a)\sim d^\pi}\left[Q^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(s|a)\right]$$

with $d^\pi(s, a) = (1-\gamma)\sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{P}_\pi(S_t = s, A_t = a)$.

Outline

Recap : Exact gradients

- **Finite horizon**

$$\nabla_\theta J(\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} Q_t^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(s_t|a_t)\right]$$

- **Infinite horizon**

$$\nabla_\theta J(\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(s_t|a_t)\right]$$

➜ simple formulations to propose unbiaised estimates of the gradients based on trajectories (almost unbiaised for infinite horizon)

REINFORCE

- Initialize θ arbitrarily
- In each step, generate N trajectories of length T under π_θ

  $$(s_1^{(i)}, a_1^{(i)}, r_1^{(i)}, \dots, s_T^{(i)}, a_T^{(i)}, r_T^{(i)})_{i=1,\dots,N}$$

  compute a Monte-Carlo estimate of the gradient

  $$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^t G_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

  with $G_t^{(i)} = \sum_{s=t}^{T} \gamma^{s-t} r_s^{(i)}$.
- Update $\theta \leftarrow \theta + \alpha \widehat{\nabla_\theta J(\theta)}$

(one may use $N = 1$, and T large enough so that $\gamma^T / (1 - \gamma)$ is small)

REINFORCE

- Initialize θ arbitrarily
- In each step, generate N trajectories of length T under π_θ

$$(s_1^{(i)}, a_1^{(i)}, r_1^{(i)}, \dots, s_T^{(i)}, a_T^{(i)}, r_T^{(i)})_{i=1,\dots,N}$$

  compute a Monte-Carlo estimate of the gradient

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^t G_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

  with $G_t^{(i)} = \sum_{s=t}^{T} \gamma^{s-t} r_s^{(i)}$.
- Update $\theta \leftarrow \theta + \alpha \widehat{\nabla_\theta J(\theta)}$

(one may use $N = 1$, and T large enough so that $\gamma^T/(1-\gamma)$ is small)

Choosing the policy class

A common choice when $\mathcal{A}$ is finite is a softmax policy

$$\forall a \in \mathcal{A},\ \pi_\theta(a|s) = \frac{\exp(\kappa f_\theta(s,a))}{\sum_{a' \in \mathcal{A}} \exp(\kappa f_\theta(s,a'))}$$

- if $\mathcal{S}$ is finite, one may use $f_\theta(s,a) = \theta_{s,a}$ $\qquad \Theta = \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
- otherwise, $f_\theta(s,a)$ is a function a some parametric space (e.g. a neural network)

$$\nabla_\theta \log \pi_\theta(a|s) = \kappa \nabla_\theta f_\theta(s,a) - \kappa \sum_{a' \in \mathcal{A}} \pi_\theta(a'|s) \nabla_\theta f_\theta(s,a')$$

Choosing the policy class

Policy gradient algorithms permit to handle continuous action spaces as well. For example, we may use a Gaussian policy with density

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma_{\theta_2}^2(s)}} \exp\left(-\frac{(a-\mu_{\theta_1}(s))^2}{2\sigma_{\theta_2}^2(s)}\right)$$

$$\begin{aligned}
\nabla_{\theta_1} \log \pi(a|s) &= \frac{(a-\mu_{\theta_1}(s))}{\sigma_{\theta_2}^2(s)} \nabla_{\theta_1}\mu_{\theta_1}(s) \\
\nabla_{\theta_2} \log \pi(a|s) &= \frac{(a-\mu_{\theta_1}(s))^2 - \sigma_{\theta_2}^2(s)}{\sigma_{\theta_2}^3(s)} \nabla_{\theta_2}\sigma_{\theta_2}(s)
\end{aligned}$$

Limitation

The gradient estimated by REINFORCE can have a large variance

Two ideas to overcome this problem :

- use better baselines
- use a different estimate of $Q^{\pi_\theta}(s, a)$
  (which will create biais)

Outline

Baseline trick, reloaded

One can further substract the baseline $b(s_1, a_1, \dots, s_t) = V^{\pi_\theta}(s_t)$:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(s_t|a_t)\right] \\
&= \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \left(Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)\right) \nabla_\theta \log \pi_\theta(s_t|a_t)\right] \\
&= \mathbb{E}^{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^{t-1} A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(s_t|a_t)\right]
\end{aligned}
$$

introducing the advantage function

$$
\begin{aligned}
A^\pi(s, a) &= Q^\pi(s, a) - V^\pi(s) \\
&= Q^\pi(s, a) - Q^\pi(s, \pi(s))
\end{aligned}
$$

(how good it is to replace the first action by a when following π ?)

Estimating the advantage

- Assume we have access to $\hat{V}$, an estimate of V^{π_θ}
- The advantage function in (s_t, a_t) can be estimated using the next transition by

$$\hat{A}(s_t, a_t) = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

  or more transitions

$$\hat{A}(s_t, a_t) = \sum_{k=t}^{t+p} \gamma^{k-t} r_k + \gamma^{p+1} \hat{V}(s_{t+p+1}) - \hat{V}(s_t)$$

- This leads to a gradient estimator from (multiple) trajectories

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^{t-1} \hat{A}\left(s_t^{(i)}, a_t^{(i)}\right) \nabla_\theta \log \pi_\theta \left(a_t^{(i)} | s_t^{(i)}\right)$$

Estimating the advantage

- Assume we have access to $\hat{V}$, an estimate of V^{π_θ}
- The advantage function in (s_t, a_t) can be estimated using the next transition by

$$\hat{A}(s_t, a_t) = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

or more transitions

$$\hat{A}(s_t, a_t) = \sum_{k=t}^{t+p} \gamma^{k-t} r_k + \gamma^{p+1} \hat{V}(s_{t+p+1}) - \hat{V}(s_t)$$

- This leads to a gradient estimator from (multiple) trajectories

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma^{t-1} \hat{A}\left(s_t^{(i)}, a_t^{(i)}\right) \nabla_\theta \log \pi_\theta \left(a_t^{(i)} | s_t^{(i)}\right)$$

➜ How do we produce the estimates $\hat{V}$? Use a critic

Actor critic algorithms

- Actor : maintains a policy and performs trajectory under it
- Critic : maintain a value, which estimates the value of the policy followed by the critic

Rationale :

- the critic's policy *improves* the value given by the critic
- the critic uses the trajectories generated by the critic to update its *evaluation* of the value

➜ Generalized Policy Iteration

Both the actor and the critic can use **parametric representation** :

- π_θ : the actor's policy, $\theta \in \Theta$
- V_ω : the critic's value, $\omega \in \Omega$

How to update the critic ?

- **Idea 1** : use TD(0)

after each observed transition under π_θ,

$$\begin{array}{rcl} \delta_t & = & r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t) \\ \omega & \leftarrow & \omega + \alpha \delta_t \nabla_\omega V_\omega(s_t) \end{array}$$

- **Idea 2** : use batches and bootstrapping

$$\hat{V}(s_t^{(i)}) = \sum_{k=t}^{t+p} \gamma^{k+t} r_t + \gamma^{p+1} V_\omega(s_{t+p+1}^{(i)})$$

and minimize the loss with respect to ω :

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\hat{V}(s_t^{(i)}) - V_\omega(s_t^{(i)}) \right)^2$$

The A2C algorithm

[Mnih et al., 2016]

In each iteration :

- collect M transitions under the policy π_θ (with reset of initial states if a terminal state is reached) $\{(s_k, a_k, r_k, s_{k+1})\}_{k\in[M]}$
- compute the (bootstrap) Monte-Carlo estimate

$$\hat{V}(s_k) = \hat{Q}(s_k, a_k) = \sum_{t=K}^{\tau_k \vee M} \gamma^{t-k} r_t + \gamma^{M-k+1} V_\omega(s_{M+1}) \mathbb{1}(\tau_k > M)$$

and advantage estimates $\hat{A}_\omega(s_k, a_k) = \hat{Q}(s_k, a_k) - V_\omega(s_k)$.

- one gradient step to minimize the policy loss : $\theta \leftarrow \theta + \alpha \nabla_\theta L_\pi(\theta)$

$$L_\pi(\omega) = -\frac{1}{M}\sum_{k=1}^{M} A_\omega(s_k, a_k) \log \pi_\theta(a_k|s_k) - \frac{\gamma}{M}\sum_{k=1}^{M}\sum_{a} \pi_\theta(a|s_k) \log \frac{1}{\pi_\theta(a|s_k)}$$

- one gradient step to minimize the value loss : $\omega \leftarrow \omega + \alpha \nabla_\omega L_V(\omega)$

$$L_V(\omega) = \frac{1}{M}\sum_{k=1}^{M}\left(\hat{V}(s_k) - V_\omega(s_k)\right)^2$$

Policy Gradient Algorithms : Pros and Cons

+ allows conservative policy updates (not just taking argmax), which make learning more stable
+ easy to implement and can handle continuous state and action spaces
+ the use of randomized policies allows for some **exploration**...

- ... but not always enough
- requires a lot of samples
- controlling the variance of the grandient can be hard
  (many tricks for variance reduction)
- the loss fonction $J(\theta)$ is *not* concave, how to avoid local maxima ?

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016).
Asynchronous methods for deep reinforcement learning.
In *Proceedings of the 33nd International Conference on Machine Learning, (ICML)*.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (1999).
Policy gradient methods for reinforcement learning with function approximation.
In *Advances in Neural Information Processing Systems (NIPS)*.