Lecture 9:: Off-policy and multi-step learning

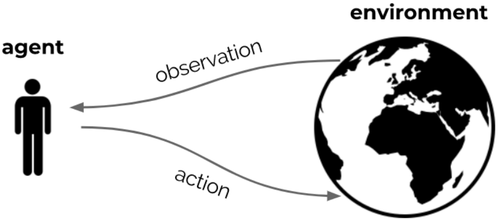
Hado van Hasselt

February 11, 2020, UCL
(Updated post-lecture)

Background

Sutton & Barto 2018, Chapter 5, 7, 11

Recap



- Reinforcement learning is the science of learning to make decisions
- Agents can learn a **policy**, **value function** and/or a **model**
- The general problem involves taking into account **time** and **consequences**
- Decisions affect the **reward**, the **agent state**, and **environment state**

High level

- Previous lectures:
  - Model-free prediction & control
  - Multi-step updates (and eligibility traces)
  - Understanding dynamic programming operators
  - Predictions with function approximation
  - Model-based algorithms
- This lecture:
  - **Off-policy learning**, especially when combined with **multi-step** updates and **function approximation**
- Not yet:
  - Policy gradients and actor-critic algorithms

Recap: why learn off-policy?

Why learn off-policy?

- Off-policy learning is important to learn about **hypothetical**, **counterfactual** events (i.e, “what if” question)
  For instance:
  - To learn about the greedy policy
  - To learn about other policies (e.g., greedy) from observed data (e.g., stored logs / other agents)
  - To learn about from past policies
  - To learn about many things at the same time

Recap: one-step off-policy

With action values, **one-step** off-policy learning seems relatively straightforward:

$$q(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha_t (R_{t+1} + \sum_a \pi(a|S_{t+1}) q(S_{t+1}, a) - q(S_t, A_t))$$

For instance

- Q-learning: let π be greedy $\implies \quad \sum_a \pi_{sa} q_{sa} = \max_a q_{sa}$
- Expected Sarsa: let π be the current behaviour policy
- Sarsa: let π put all probability mass on the action the behaviour picked

Recap: multi-step off-policy

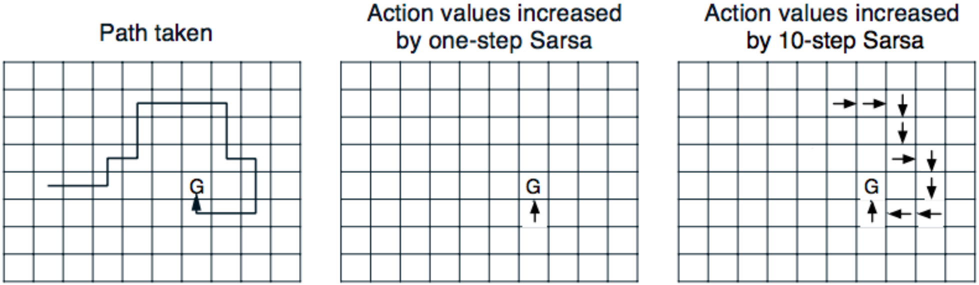
For **multi-step updates**, we can use **importance-sampling corrections**
E.g., for a Monte Carlo return on a trajectory $\tau_t = \{S_t, A_t, R_{t+1}, \dots, S_T\}$

$$\hat{G}_t \quad \equiv \quad \frac{p(\tau_t|\pi)}{p(\tau_t|\mu)} G_t \quad = \quad \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t \,,$$

then $\mathbb{E}[\hat{G}_t \mid \mu] = \mathbb{E}[G_t \mid \pi]$

Multi-step off-policy

- We know multi-step updates often more efficiency propagate information
- But full Monte Carlo is typically not the best trade-off



Issues with off-policy learning

The following issues (especially) arise when learning off-policy

- High variance (especially when using multi-step updates)
- Divergent and inefficient learning (especially when using one-step updates)

We will discuss both in this lecture

Variance of importance sampling corrections

- The big issue in importance-sampling corrections is **high variance**
- First, consider a one-step reward
- Verify the expectation, for a given state s:

$$\begin{aligned}\mathbb{E}\left[\frac{\pi(A_t|s)}{\mu(A_t|s)}R_{t+1} \mid A_t \sim \mu\right] &= \sum_a \mu(a|s)\frac{\pi(a|s)}{\mu(a|s)}r(s,a) \\ &= \sum_a \pi(a|s)r(s,a) \\ &= \mathbb{E}[R_{t+1} \mid A_t \sim \pi]\end{aligned}$$

- But typically the variance will be larger, sometimes greatly so

Variance of importance sampling corrections

Lets consider a concrete example:

action	reward	$\pi(a \vert s)$	$\mu(a \vert s)$
$\rightarrow$	$+10$	0.9	0.9
$\leftarrow$	$+20$	0.1	0.1

$$\mathbb{E}[R_{t+1} \mid \pi] = 11$$

Variance of importance sampling corrections

action	reward	$\pi(a\|s)$	$\mu(a\|s)$
$\rightarrow$	$+10$	0.9	0.9
$\leftarrow$	$+20$	0.1	0.1

$\mathbb{E}[R_{t+1} \mid \pi] = 11$

- Second moment, **on-policy** (when $\mu = \pi$):

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{\pi(A_t|s)}{\mu(A_t|s)} R_{t+1}\right)^2 \mid A_t \sim \mu\right] &= \mathbb{E}\left[(R_{t+1})^2 \mid A_t \sim \pi\right] \qquad \text{(using } \mu = \pi\text{)} \\
&= 0.9 \times 10^2 + 0.1 \times 20^2 \\
&= 90 + 40 \qquad = 130
\end{aligned}$$

- $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = 130 - 11^2 = 9$ (Std dev $= 3$)
  (where $X = \frac{\pi(A_t|s)}{\mu(A_t|s)} R_{t+1}$)

Variance of importance sampling corrections

action	reward	$\pi(a\|s)$	$\mu(a\|s)$
$\rightarrow$	$+10$	0.9	0.5
$\leftarrow$	$+20$	0.1	0.5

$$\mathbb{E}[R_{t+1} \mid \pi] = 11$$

- Second moment, **off-policy**:

$$\mathbb{E}\left[\left(\frac{\pi(A_t|s)}{\mu(A_t|s)}R_{t+1}\right)^2 \mid A_t \sim \mu\right] = \sum_a \mu(a|s)\left(\frac{\pi(a|s)}{\mu(a|s)}r(s,a)\right)^2$$

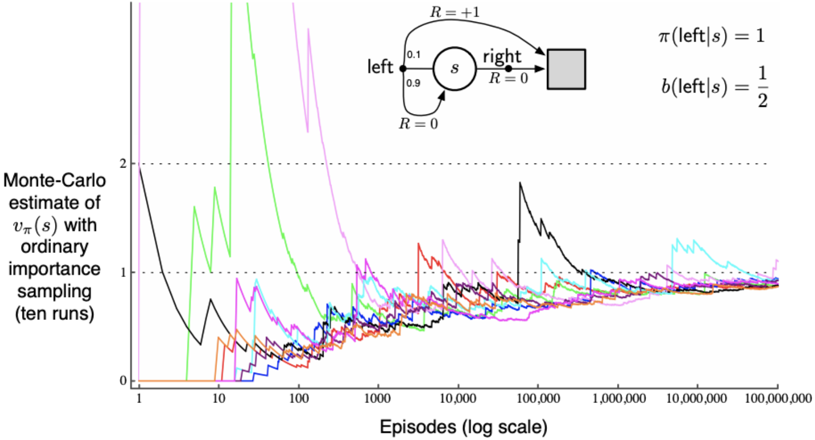
$$= 0.5 \times \frac{0.9^2}{0.5^2} \times 10^2 + 0.5 \times \frac{0.1^2}{0.5^2} \times 20^2$$

$$= 162 + 8 \qquad = 170$$

- Variance: $170 - 11^2 = 49$ (Std dev $= 7$)

Variance of importance sampling corrections

- In some cases the variance of an importance-weighting return can even be **infinite** (see: Sutton & Barto, Example 5.5)



Mitigating variance: with per-decision importance weighting

- There are multiple ways to reduce variance
- We will discuss three:
  - Per-decision importance weighting
  - Control variates
  - (Adaptive) bootstrapping

Reducing variance:
Per-decision importance weighting

Mitigating variance: with per-decision importance weighting

Consider some state s. For any random X that does not correlate with (random) action A we have

$$\mathbb{E}[X \mid \pi] = \mathbb{E}\left[\frac{\pi(A|s)}{\mu(A|s)} X \mid \mu\right] = \mathbb{E}[X \mid \mu]$$

Intuition: the expectation does not depend on the policy, so we don't need to correct

Mitigating variance: with per-decision importance weighting

Proof:

$$
\begin{aligned}
&\mathbb{E}\left[\frac{\pi(A|s)}{\mu(A|s)}X \mid \mu\right] \\
&= \mathbb{E}[X \mid \mu]\mathbb{E}\left[\frac{\pi(A|s)}{\mu(A|s)} \mid \mu\right] && \text{(Because } X \text{ and } \tfrac{\pi}{\mu} \text{ are uncorrelated)} \\
&= \mathbb{E}[X \mid \mu]\sum_a \cancel{\mu(a|S_k)}\frac{\pi(a|s)}{\cancel{\mu(a|s)}} \\
&= \mathbb{E}[X \mid \mu]\sum_a \pi(a|s) \\
&= \mathbb{E}[X \mid \mu] && \text{(Because } \textstyle\sum_a \pi(a|s) = 1\text{)}
\end{aligned}
$$

Similarly, in general, we have $\mathbb{E}[\frac{\pi(A|s)}{\mu(A|s)} \mid \mu] = 1$

Notation

Shorthand notations:

$$\boldsymbol{\rho_t} \equiv \frac{\pi(A_t \mid S_t)}{\mu(A_t \mid S_t)} \qquad\qquad \boldsymbol{\rho_{t:t+n}} \equiv \prod_{k=t}^{t+n} \rho_k = \prod_{k=t}^{t+n} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

Then the reweighted MC return from state S_t terminating at time T can be written as

$$\overbrace{\left(\prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}\right)}^{=\rho_{t:T-1}} \overbrace{\left(\sum_{k=t}^{T-1} \gamma^{k-t} R_{k+1}\right)}^{=G_t} = \rho_{t:T-1} G_t = \sum_{k=t}^{T-1} \rho_{t:T-1} \gamma^{k-t} R_{k+1}$$

We can interpret the importance-weight $\rho_{t:T-1}$ as applying to each reward

Mitigating variance: with per-decision importance weighting

$$\rho_{t:T-1} G_t = \sum_{k=t}^{T-1} \rho_{t:T-1} \gamma^{k-t} R_{k+1}$$

Earlier rewards cannot depend on **later** actions. This means:

$$\begin{aligned}\mathbb{E}[\rho_{t:T-1} G_t \mid \mu] &= \mathbb{E}[\sum_{k=t}^{T-1} \rho_{t:T-1} \gamma^{k-t} R_{k+1} \mid \mu] \\ &= \mathbb{E}[\sum_{k=t}^{T-1} \rho_{t:\boldsymbol{k}} \gamma^{k-t} R_{k+1} \mid \mu]\end{aligned}$$

Recursive definition of the latter:

$$G_t^{\rho} = \rho_t (R_{t+1} + \gamma G_{t+1}^{\rho})$$

Mitigating variance: with per-decision importance weighting

- Per-decision importance-weighted return

$$G_t^\rho = \rho_t(R_{t+1} + \gamma G_{t+1}^\rho)$$

  We can use this to learn v_π from data generated under $\mu \neq \pi$

- To learn **action values** q_π, we can use

$$G_t^\rho = R_{t+1} + \gamma \rho_{t+1} G_{t+1}^\rho$$

- How and why are these different?

Reducing variance:
Control variates

Mitigating variance: control variates

Example:

action	reward	$\pi(a|s)$	$\mu(a|s)$
$\leftarrow$	$+1$	1	0.5
$\rightarrow$	-1	0	0.5

We are trying to estimate $\mathbb{E}[R|\pi]$, using update

$$\Delta v(S_t) = \alpha(\rho_t R_{t+1} - v(S_t)) = \begin{cases} \alpha(2 \times (+1) - v(S_t)) & \text{if } A_t = \leftarrow \\ \alpha(0 \times (-1) - v(S_t)) & \text{if } A_t = \rightarrow \end{cases}$$

$$= \begin{cases} \alpha(2 - v(S_t)) & \text{if } A_t = \leftarrow \\ \alpha(0 - v(S_t)) & \text{if } A_t = \rightarrow \end{cases}$$

So, we either update towards $+2$, or towards 0

These average out to the correct target of $+1$

Mitigating variance: control variates

Example:

action	reward	$\pi(a\|s)$	$\mu(a\|s)$
$\leftarrow$	$+1$	1	0.5
$\rightarrow$	-1	0	0.5

We are trying to estimate the expected immediate reward, using update

$$\Delta v(S_t) = \alpha(\rho_t R_{t+1} - v(S_t)) = \begin{cases} \alpha(2 \times (+1) - v(S_t)) & \text{if } A_t = \leftarrow \\ \alpha(0 \times (-1) - v(S_t)) & \text{if } A_t = \rightarrow \end{cases}$$

$$= \begin{cases} \alpha(2 - v(S_t)) & \text{if } A_t = \leftarrow \\ \alpha(0 - v(S_t)) & \text{if } A_t = \rightarrow \end{cases}$$

But why update towards the arbitrary value of 0?

Could we, instead, just not update at all when we pick $\rightarrow$?

Mitigating variance: control variates

We propose to use, instead

$$\Delta v(S_t) = \boldsymbol{\rho_t \alpha (R_{t+1} - v(S_t))}$$

This is the same as

$$\underbrace{\alpha(\rho_t R_{t+1} - v(S_t))}_{\text{previous update}} + \alpha \underbrace{(1 - \rho_t) v(S_t)}_{\text{control variate}}$$

Note, ρ_t does not correlate with $v(S_t)$, so

$$\mathbb{E}[(1 - \rho_t) v(S_t) \mid \mu] = 0$$

The **control variate** has mean zero, but can (anti-)correlate with the target
$\implies$ mean stays the same, but variance can differ

Mitigating variance: control variates

Example:

action	reward	$\pi(a\|s)$	$\mu(a\|s)$
$\leftarrow$	$+1$	1	0.5
$\rightarrow$	-1	0	0.5

We are trying to estimate $\mathbb{E}[R|\pi]$, using update

$$\Delta v(S_t) = \boldsymbol{\rho_t \alpha (R_{t+1} - v(S_t))} = \begin{cases} 2 \times \alpha(+1 - v(S_t)) & \text{if } A_t = \leftarrow \\ 0 \times \alpha(-1 - v(S_t)) & \text{if } A_t = \rightarrow \end{cases}$$

$$= \begin{cases} 2\alpha(1 - v(S_t)) & \text{if } A_t = \leftarrow \\ 0 & \text{if } A_t = \rightarrow \end{cases}$$

Note: we either update to $+1$ (correctly) with twice the step size
or we do not update at all

Mitigating variance: control variates

Did we lower the variance? Let's check. Suppose $v(S_t) = 1$.

- The previous update

$$\Delta v(S_t) = \alpha(\rho_t R_{t+1} - v(S_t)) = \begin{cases} \alpha(2 - v(S_t)) & \text{if } A_t = \leftarrow \\ \alpha(0 - v(S_t)) & \text{if } A_t = \rightarrow \end{cases}$$

$\implies$ $\mathbf{Var(\Delta v(S_t)) = \alpha^2}$

- The new update

$$\Delta v(S_t) = \rho_t \alpha(R_{t+1} - v(S_t)) = \begin{cases} 2\alpha(1 - v(S_t)) & \text{if } A_t = \leftarrow \\ 0 & \text{if } A_t = \rightarrow \end{cases}$$

$\implies$ $\mathbf{Var(\Delta v(S_t)) = 0}$

(Obviously, this is an extreme example, where our estimate is already correct.)

Control variates for multi-step returns

The idea of control variates can be extended to multi-step returns

- First, recall

$$\begin{aligned}
\delta_t^\lambda &\equiv G_t^\lambda - v(S_t) \\
&= R_{t+1} + \gamma((1-\lambda)v(S_{t+1}) + \gamma\lambda G_{t+1}^\lambda) - v(S_t) \\
&= \underbrace{R_{t+1} + \gamma v(S_{t+1}) - v(S_t)}_{=\delta_t} + \gamma\lambda \underbrace{(G_{t+1}^\lambda - v(S_{t+1}))}_{=\delta_{t+1}^\lambda} \\
&= \delta_t + \gamma\lambda\delta_{t+1}^\lambda
\end{aligned}$$

Control variates for multi-step returns

The idea of control variates can be extended to multi-step returns

- Now, lets add per-decision importance weights

$$\delta_t^{\lambda} = \delta_t + \gamma\lambda\delta_{t+1}^{\lambda}$$
$$\delta_t^{\rho\lambda} = \rho_t(\delta_t + \gamma\lambda\delta_{t+1}^{\rho\lambda})$$

- By design this includes the $(1-\rho_t)v(S_t)$ control variate terms
- Sometimes called 'error weighting' (to contrast to 'reward weighting')

Control variates for multi-step returns

$$\delta_t^{\rho\lambda} = \rho_t(\delta_t + \gamma\lambda\delta_{t+1}^{\rho\lambda})$$

- One can show that
$$\mathbb{E}[\delta_t^{\rho\lambda} \mid \mu] = \mathbb{E}[G_t^{\rho\lambda} - v(S_t) \mid \mu]$$
where
$$G_t^{\rho\lambda} = \rho_t\left(R_{t+1} + \gamma\left((1-\lambda)v(S_{t+1}) + \lambda G_{t+1}^{\rho\lambda}\right)\right)$$
is the per-decision importance-weighted λ-return.
- But $\delta_t^{\rho\lambda}$ can have lower variance than $G_t^{\rho\lambda} - v(S_t)$

Reducing variance: Bootstrapping

Reducing variance: bootstrapping

- For our last technique, we consider bootstrapping
- This amounts to picking $\lambda < 1$ when using either $\delta_t^{\rho\lambda}$ or $G_t^{\rho\lambda}$
- Note that to learn **action values**, we can use

$$G_t^{\rho\lambda} = R_{t+1} + \gamma\left((1-\lambda)\sum_a \pi(a \mid S_{t+1})q(S_{t+1}, a) + \rho_{t+1}\lambda G_{t+1}^{\rho\lambda}\right)$$

  Then, if $\lambda = 0$, we get

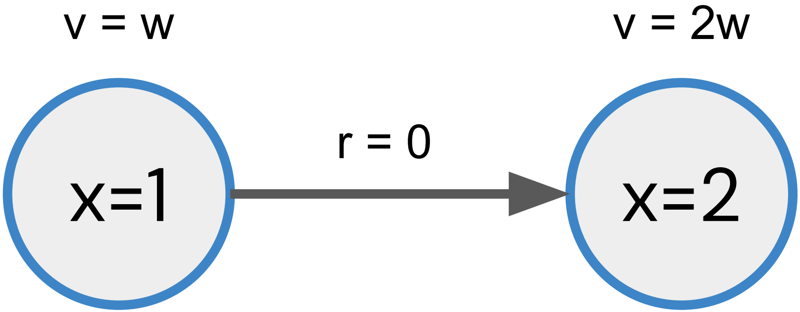
$$G_t = R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1})q(S_{t+1}, a)$$

  $\implies$ no more importance weighted $\implies$ low variance
- However, bootstrapping too much may open us to the **deadly triad**!

Recap: Deadly triad

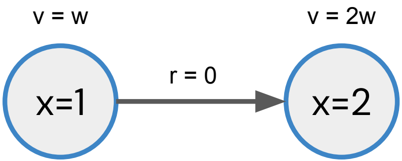
- Recall, the deadly triad refers to the possibility of divergence when we combine
  - **Bootstrapping**
  - **Function approximation**
  - **Off-policy learning**

Recap: Deadly triad



What if we use TD only on this transition?

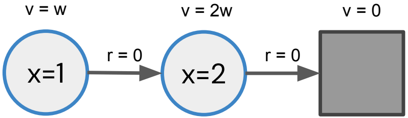
Recap: Deadly triad



$$
\begin{aligned}
w_{t+1} &= w_t + \alpha_t(r + \gamma v(s') - v(s))\nabla v(s) \\
&= w_t + \alpha_t(2\gamma - 1)w_t
\end{aligned}
$$

Suppose $\gamma > \frac{1}{2}$. Then,
When $w_t > 0$ and , then $w_{t+1} > w_t$
When $w_t < 0$ and , then $w_{t+1} < w_t \implies w_t$ diverges to $+\infty$ of $-\infty$

Recap: Deadly triad



- What if we use multi-step returns?
- Still consider only updating the left-most state

$$\begin{aligned}\Delta w &= \alpha(r + \gamma(G_t^{\lambda} - v(s)) \\ &= \alpha(2\gamma(1-\lambda) - 1)w\end{aligned}$$

- The multiplier is negative when $2\gamma(1-\lambda) < 1 \implies \lambda > 1 - \frac{1}{2\gamma}$
- E.g., when $\gamma = 0.9$, then we need $\lambda > 4/9 \approx 0.45$
- Conclusion: if we do not bootstrap too much, we can learn better

Reducing variance: **adaptive** bootstrapping

- We don't want to bootstrap too much $\implies$ **deadly triad**
- We don't want to bootstrap too little $\implies$ **high variance**
- Can we adaptively bootstrap 'just enough'?
- Idea: bootstrap **adaptively** only in as much as you go off-policy

Reducing variance: adaptive bootstrapping

- Recall $\delta_t^{\rho\lambda} = \rho_t(\delta_t + \gamma\lambda\delta_{t+1}^{\rho\lambda})$
- Let's add an initial bootstrap parameter, and make these time-dependent

$$\delta_t^{\rho\lambda} = \lambda_t\rho_t(\delta_t + \gamma\delta_{t+1}^{\rho\lambda})$$

  (If $\lambda_t = 1$, we obtain the previous version)
- We can pick λ_t **separately on each time step**
- Idea: pick it such that, for all t, $\lambda_t\rho_t \leq 1$:

$$\boldsymbol{\lambda_t = \min(1, 1/\rho_t)}$$

- Intuition: when we are too off-policy (ρ is far from one) truncate the sum of errors
- This is the same as bootstrapping there

Reducing variance: adaptive bootstrapping

$$\lambda_t = \min(1, 1/\rho_t)$$

- This is known as **ABTD** (Mahmood et al. 2017) or **v-trace** (Espeholt et al. 2018)
- We are free to choose different ways to bootstrap: in the tabular case all these methods will be updating towards some mixture of multi-step returns, and therefore converge
- In deep RL this really helps, especially for policy gradients
(Policy gradients do not like biased return estimates – we will get back to that)
- This is used a lot these days

Reducing variance: tree backup

- Picking $\lambda_t = \min(1, 1/\rho_t)$ is not the only way to adaptively bootstrap
- One more option, consider the Bellman operator for action values

$$q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) q_\pi(S_{t+1}, a) \mid A_t = a, S_t = s]$$

- Note: the expectation does not depend on π, because we condition on the action a
- Idea: sample this, then replace only the action you selected:

$$G_t = R_{t+1} + \gamma \sum_{\boldsymbol{a \neq A_{t+1}}} \pi(a|S_{t+1}) q(S_{t+1}, a) + \boldsymbol{\gamma \pi(A_{t+1}|S_{t+1}) G_{t+1}}$$

- We remove only the expectation $q(S_{t+1}, A_{t+1})$ of the action actually selected, and replace it with the return
- This is **unbiased**, and **low variance**! ($\pi(A_{t+1}|S_{t+1})$ plays a role similar to λ)
- It might bootstrap too early though — beware of deadly triads!

Questions?