Reinforcement Learning

Chapter 3: Model-free RL

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Fall 2025

Control Loop via Temporal Difference

+ *But still we are not fully online! We need to wait till end of each episode!*
– Well! That's right! But, we could use TD!

Using TD in the control loop will make our algorithm fully online

- *We update values after each state-action pair*
- *We then improve the policy*

We should yet use ϵ-greedy improvement to keep exploration

SARSA: *State-Action-Reward State-Action*

SARSA $\equiv$ *State-Action Reward State-Action*

SARSA algorithms use TD along with ϵ-greedy update for the control loop

In general, we can develop various forms of SARSA

- *We may use TD-0 for updating action-values*
  - ↳ *This is the basic SARSA*
- *We may use TD-n for updating action-values*
  - ↳ *This is n-SARSA*
- *We may use TD$_\lambda$ for updating action-values*
  - ↳ *This is SARSA(λ)*

SARSA: *First Try*

Let's try to make a simple TD-based control loop

`TD_Control():`

Initiate estimator as $\hat{q}_\pi(s, a) = 0$ *for all states and actions*
for *episode* $= 1 : K$ *or until* π *stops changing* **do**
  Initiate with a random state-action pair (S_0, A_0)
  for $t = 0 : T - 1$ *that is either terminal or terminated* **do**
    Act A_t *and observe*

$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}$$

    Update policy to $\pi \leftarrow \epsilon\text{-}\texttt{Greedy}(\hat{q}_\pi)$
    Draw the new action A_{t+1} *from* $\pi(\cdot|S_{t+1})$
    Compute $\hat{v}_\pi(S_{t+1})$ *from* $\hat{q}_\pi(S_{t+1}, a)$ *and* $\pi(\cdot|S_{t+1})$
    Set $G \leftarrow R_{t+1} + \gamma \hat{v}_\pi(S_{t+1})$
    Update $\hat{q}_\pi(S_t, A_t) \leftarrow \hat{q}_\pi(S_t, A_t) + \alpha(G - \hat{q}_\pi(S_t, A_t))$
  end for
end for

SARSA: *Going On-Policy*

In *line 8* of our control algorithm: *we compute* $\hat{v}_\pi(S_{t+1})$ *as*

$$\hat{v}_\pi(S_{t+1}) = \sum_{m=1}^{M} \pi(a^m | S_{t+1}) \hat{q}_\pi(S_{t+1}, a^m)$$

But, we do know that

1. *our estimates* $\hat{q}_\pi(S_{t+1}, a^m)$ *are* ***not*** *that good, and also*
2. *our policy has led use to next action* A_{t+1}

So, we could move on our policy and write

$$\pi(a|S_{t+1}) = \begin{cases} 1 & a = A_{t+1} \\ 0 & a \neq A_{t+1} \end{cases} \rightsquigarrow \hat{v}_\pi(S_{t+1}) = \hat{q}_\pi(S_{t+1}, A_{t+1})$$

We call this approach on-policy, since move on our policy

SARSA: *Basic Algorithm*

`SARSA():`

Initiate estimator as $\hat{q}_\pi(s, a) = 0$ *for all states and actions*
for *episode* $= 1 : K$ *or until* π *stops changing* **do**
  Initiate with a random state-action pair (S_0, A_0)
  for $t = 0 : T - 1$ *that is either terminal or terminated* **do**
    Act A_t *and observe*

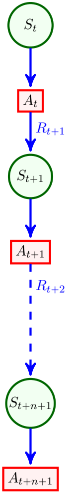
$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}$$

    Update policy to $\pi \leftarrow \epsilon\text{-}\mathtt{Greedy}(\hat{q}_\pi)$
    Draw the new action A_{t+1} *from* $\pi(\cdot|S_{t+1})$ *and move on policy*

$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}, A_{t+1}$$

    Set $G \leftarrow R_{t+1} + \gamma \hat{q}_\pi(S_{t+1}, A_{t+1})$
    Update $\hat{q}_\pi(S_t, A_t) \leftarrow \hat{q}_\pi(S_t, A_t) + \alpha(G - \hat{q}_\pi(S_t, A_t))$
  end for
end for

SARSA: *Deeper Return Samples*



We can use a longer trajectory while we learn on-policy, i.e.,

$$G^n = \sum_{i=0}^{n} R_{t+i+1} + \gamma \hat{q}_\pi \left(S_{t+n+1}, A_{t+n+1} \right)$$

This will however add extra delay!

As a practice, you could

re-write the basic SARSA with n-return ☺

SARSA(λ): *Tracing Eligibility of State-Action Pairs*

We can extend SARSA to the case with λ-return: *we have two options*

- *the case with forward-view*
  - ↳ *We know this is not practical! So, let's skip the details*
- *the case with backward-view and eligibility tracing*
  - ↳ *Let's look into this one*

We first extend eligibility tracing to the case with state-action pairs

```
ElgTrace(S_t, A_t, E(·) |λ):
```

1: Eligibility tracing function has NM components, i.e., $E(s, a)$ for all state-action pairs
2: **for** all state-action pairs (s, a) **do**
3: Update $E(s, a) \leftarrow \gamma\lambda E(s, a)$
4: **end for**
5: Update $E(S_t, A_t) \leftarrow E(S_t, A_t) + 1$

SARSA: *Alternative via TD-λ*

`SARSA`(λ):

```
Initiate $\hat{q}_\pi(s,a) = 0$ and $E(s,a) = 0$ for all states and actions
for episode $= 1 : K$ or until $\pi$ stops changing do
    Initiate with a random state-action pair $(S_0, A_0)$
    for $t = 0 : T-1$ that is either terminal or terminated do
        $E(\cdot) \leftarrow$ ElgTrace($S_t, A_t, E(\cdot) \ |\lambda$)
        Act $A_t$ and observe $R_{t+1}$ and $S_{t+1}$
        Update policy to $\pi \leftarrow \epsilon$-Greedy($\hat{q}_\pi$)
        Draw the new action $A_{t+1}$ from $\pi(\cdot|S_{t+1})$ and move on policy

                $S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}, A_{t+1}$

        Set $\Delta \leftarrow R_{t+1} + \gamma \hat{q}_\pi(S_{t+1}, A_{t+1}) - \hat{q}_\pi(S_t, A_t)$
        for all state-action pairs $(s,a)$ do
            Update $\hat{q}_\pi(s,a) \leftarrow \hat{q}_\pi(s,a) + \alpha \Delta E(s,a)$
        end for
    end for
end for
```

Going Off-Policy

Let's think about a fundamental question: *while sampling the environment with a specific policy π, can we estimate the values of another policy $\bar{\pi}$?*

- + *Why should this be a fundamental question?*
- – Well! There are several reasons
  - ↳ *Maybe we sampled environment with our bad policy: can't we use our sample again?*
  - ↳ *Maybe we are looking at other players: can't we learn something about the environment from their samples?*
    - ↳ *Maybe they are good players: can't we use this fact to improve our policy?*
    - ↳ *Maybe they are bad players: can't we use this fact to avoid doing mistakes?*

This is the idea of off-policy control

Let's start with some baiscs

Importance Sampling

Consider following problem: *we have random variable X drawn as $X \sim p(x)$ whose mean is*

$$\mu_p = \mathbb{E}_p \{X\} = \sum_x p(x) x$$

We want to know how would be the expectation if we had $X \sim q(x)$: we write

$$\begin{aligned}\mu_q = \mathbb{E}_q \{X\} &= \sum_x q(x) x \\ &= \sum_x p(x) \frac{q(x)}{p(x)} x = \mathbb{E}_p \left\{ \frac{q(X)}{p(X)} X \right\}\end{aligned}$$

This gives us possibility to

estimate $\mathbb{E}_q \{X\}$ using samples drawn from $p(x)$

Importance Sampling

Say we have drawn K samples from $p(x)$, i.e., we have

$$X_1, X_2, \dots, X_K$$

We can use Monte-Carlo to estimate μ_p as

$$\hat{\mu}_p = \frac{1}{K} \sum_{k=1}^{K} X_k$$

We can also use Monte-Carlo to estimate μ_q as

$$\hat{\mu}_q = \frac{1}{K} \sum_{k=1}^{K} \frac{q(X_k)}{p(X_k)} X_k$$

We call this method importance sampling

Off-Policy Control via Importance Sampling

Now, let's get back to our problem: *assume we have played with policy π and collected K sample trajectories of length T all started at state $S_0 = s$, i.e.,*

$$s = S_0[k], A_0[k] \xrightarrow{R_1[k]} S_1[k], A_1[k] \xrightarrow{R_2[k]} \cdots \xrightarrow{R_T[k]} S_T[k]$$

for $k = 1 : K$; then, we could write

$$\hat{v}_\pi(s) = \frac{1}{K} \sum_{k=1}^{K} G[k]$$

This is the basic Monte-Carlo

Off-Policy Control via Importance Sampling

But now, we want to use samples to evaluate another policy $\bar{\pi}$

$$s = S_0[k], A_0[k] \xrightarrow{R_1[k]} S_1[k], A_1[k] \xrightarrow{R_2[k]} \cdots \xrightarrow{R_T[k]} S_T[k]$$

We could also use importance sampling to write

$$\begin{aligned}\hat{v}_{\bar{\pi}}(s) &= \frac{1}{K}\sum_{k=1}^{K} \frac{\Pr\{\text{same action sequence with } \bar{\pi}\}}{\Pr\{\text{same action sequence with } \pi\}} G[k] \\ &= \frac{1}{K}\sum_{k=1}^{K} \frac{\bar{\pi}(A_0[k]|S_0[k]) \cdots \bar{\pi}(A_{T-1}[k]|S_{T-1}[k])}{\pi(A_0[k]|S_0[k]) \cdots \pi(A_{T-1}[k]|S_{T-1}[k])} G[k] \\ &= \frac{1}{K}\sum_{k=1}^{K} \prod_{\ell=0}^{T-1} \frac{\bar{\pi}(A_\ell[k]|S_\ell[k])}{\pi(A_\ell[k]|S_\ell[k])} G[k]\end{aligned}$$

Off-Policy Control via Importance Sampling

We can further update the estimate in an online fashion from

$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}, A_{t+1} \xrightarrow{R_{t+2}} \cdots \xrightarrow{R_T} S_T$$

by online averaging as

$$\hat{v}_{\bar{\pi}}(S_t) \leftarrow \hat{v}_{\bar{\pi}}(S_t) + \alpha \left(\prod_{\ell=t}^{T-1} \frac{\bar{\pi}(A_\ell | S_\ell)}{\pi(A_\ell | S_\ell)} G_t - \hat{v}_{\bar{\pi}}(S_t) \right)$$

So, we are evaluating $\bar{\pi}$ via Monte-Carlo

off our policy π

This is off-policy control

Off-Policy Control via Importance Sampling

We can further apply off-policy control via TD

$$\hat{v}_{\bar{\pi}}(S_t) \leftarrow \hat{v}_{\bar{\pi}}(S_t) + \alpha \left(\frac{\bar{\pi}(A_t|S_t)}{\pi(A_t|S_t)} (R_{t+1} + \gamma \hat{v}_{\bar{\pi}}(S_{t+1})) - \hat{v}_{\bar{\pi}}(S_t) \right)$$

Note that for action-values estimate

R_{t+1} *does not depend any more on policy as we know action* A_t

Therefore, we have for action-value update

$$\hat{q}_{\bar{\pi}}(S_t, A_t) \leftarrow \hat{q}_{\bar{\pi}}(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \frac{\bar{\pi}(A_t|S_t)}{\pi(A_t|S_t)} \hat{v}_{\bar{\pi}}(S_{t+1}) - \hat{q}_{\bar{\pi}}(S_t, A_t) \right)$$

Q-Learning

Q-Learning

Q-learning is an off-policy TD control algorithm, where we sample with ϵ-greedy policy but update the action-values to evaluate greedy policy

This means in Q-learning π is ϵ-greedy policy and $\bar{\pi}$ is greedy. Let's consider basic TD evaluation: so, we can write

$$\hat{q}_{\bar{\pi}}\left(S_t, A_t\right) \leftarrow \hat{q}_{\bar{\pi}}\left(S_t, A_t\right) + \alpha\left(G - \hat{q}_{\bar{\pi}}\left(S_t, A_t\right)\right)$$

where G should be

$$G = R_{t+1} + \gamma \hat{v}_{\bar{\pi}}\left(\bar{S}_{t+1}\right)$$

Since we sample by ϵ-greedy policy π, we use importance sampling and write

$$G = R_{t+1} + \gamma \frac{\bar{\pi}\left(A_t | S_t\right)}{\pi\left(A_t | S_t\right)} \hat{v}_{\bar{\pi}}\left(S_{t+1}\right)$$

Q-Learning

But, we really don't need importance sampling: we can simply observe that

$$\hat{v}_{\bar{\pi}} \left(S_{t+1} \right) = \sum_{m=1}^{M} \hat{q}_{\bar{\pi}} \left(S_{t+1}, a^m \right) \bar{\pi} \left(a^m | S_{t+1} \right) = \max_m \hat{q}_{\bar{\pi}} \left(S_{t+1}, a^m \right)$$

and we do know that

$$\frac{\bar{\pi} \left(A_t | S_t \right)}{\pi \left(A_t | S_t \right)} = \mathbf{1} \left\{ A_t = \operatorname*{argmax}_a \hat{q}_{\bar{\pi}} \left(S_t, a \right) \right\}$$

So, we could directly update as

$$\hat{q}_{\bar{\pi}} \left(S_t, A_t \right) \leftarrow \hat{q}_{\bar{\pi}} \left(S_t, A_t \right) + \alpha \left(R_{t+1} + \gamma \max_m \hat{q}_{\bar{\pi}} \left(S_{t+1}, a^m \right) - \hat{q}_{\bar{\pi}} \left(S_t, A_t \right) \right)$$

This concludes Q-learning algorithm

Q-Learning: *Basic Algorithm*

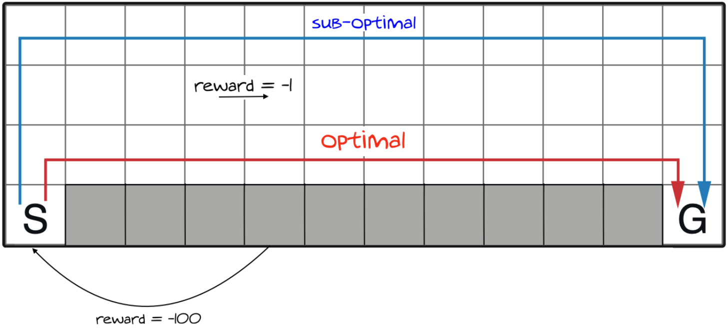
`Q-Learning():`

1: *Initiate estimator as* $\hat{q}_\star(s, a) = 0$ *for all states and actions*
2: ***for*** *episode* $= 1 : K$ *or until* π *stops changing* ***do***
3: *Initiate with a random state* S_0
4: ***for*** $t = 0 : T - 1$ *that is either terminal or terminated* ***do***
5: *Update policy to* $\pi \leftarrow \epsilon\text{-}\mathtt{Greedy}(\hat{q}_\star)$
6: *Draw action* A_t *from* $\pi(\cdot|S_t)$ *and observe*

$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}$$

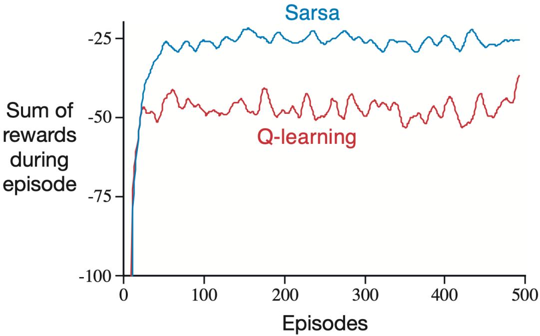
7: *Set* $G \leftarrow R_{t+1} + \gamma \max_m \hat{q}_\star(S_{t+1}, a^m)$
8: *Update* $\hat{q}_\star(S_t, A_t) \leftarrow \hat{q}_\star(S_t, A_t) + \alpha(G - \hat{q}_\star(S_t, A_t))$
9: ***end for***
10: ***end for***

Example: *Cliff Walking*



Let's compare SARSA to Q-Learning algorithm!

Example: *Cliff Walking*



Don't Mistake!

Q-learning collects less reward since it goes off-policy; however, it estimates optimal action-values: at some point it can start playing optimally

Convergence of SARSA

Recall: *GLIE Algorithms*

A GPI-type control loop is GLIE, if for any state-action pair (s, a), *we have the following asymptotic properties*

1. *The number of visits to all state-action pair grows large*

$$\lim_{K \to \infty} \mathcal{C}_K (s, a) = \infty$$

2. *The improved policy in last episode converges to greedy policy*

$$\lim_{K \to \infty} \pi_K (a^m | s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_{\pi_K} (s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_{\pi_K} (s, a^m) \end{cases}$$

GLIE control algorithms converge to optimal policy

Convergence of SARSA

- + *But do we really have large number of episodes with SARSA?*
- – Not necessarily! We may have only one infinitely long trajectory
- + *What should we do then?*
- – We can simply treat it as a large number of episodes of length 1

In (basic) SARSA, we only need one step in the trajectory

$$S_t, A_t \xrightarrow{R_{t+1}} S_{t+1}$$

We could hence think of it as one episode

1. *each time step t we update the action-values*
2. *each time step we improve the policy*

Convergence of SARSA

Modification: *GLIE Algorithms*

An online control loop is GLIE, if we have asymptotically in time t

1. *The number of visits to all state-action pair grows large*

$$\lim_{t\to\infty} \mathcal{C}_t(s, a) = \infty$$

2. *The improved policy converges to greedy policy*

$$\lim_{t\to\infty} \pi_t(a^m|s) = \begin{cases} 1 & m = \operatorname*{argmax}_m q_{\pi_t}(s, a^m) \\ 0 & m \neq \operatorname*{argmax}_m q_{\pi_t}(s, a^m) \end{cases}$$

Convergence of SARSA: *Make it GLIE*

+ *Can we guarantee that both conditions hold with SARSA?*
– The second one is easy: *we need to scale ϵ down with t, e.g., $\epsilon_t = 1/t$*
+ *What about the first condition?*
– We should scale *the step-size α according to Robbins-Monro*

Robbins-Monro Sequence

Sequence α_t is Robbins-Monro if we have

$$\sum_{t=0}^{\infty} \alpha_t = \infty \qquad \textit{and} \qquad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

For instance, $\alpha_t = 1/t$ is a Robbins-Monro sequence

Convergence of SARSA

Convergence of SARSA

SARSA online control loop converges to the optimal action-values if

1. *Step-size is scheduled by a Robbins-Monro sequence*
2. *Exploration factor ϵ decays in time*

In practice however

- ϵ *is a hyperparameter*
  - ↳ *We know that we should schedule it*
  - ↳ *How we should do the scheduling? This is hyperparameter tuning*
- α *is a hyperparameter: some people call it learning rate*
  - ↳ *Its scheduling is again hyperparameter tuning*

Convergence of Q-Learning

Convergence of Q-Learning

Q-learning online control loop with exploration (non-zero ϵ) converges to the optimal action-values as $t \to \infty$

+ *That's it?*
- Yes!

Since we are evaluating off-policy, we don't care about behaving policy

Q-Learning vs SARSA

+ *So! Does it mean that Q-learning is always better?*
– Not always!

In general Q-learning has several benefits

- *Minimal convergence requirements*
- *It converges faster to the optimal policy*
  - ↳ *If we want to make SARSA that fast, we may get to a sub-optimal policy*
- *It has more flexibility and sample-efficiency*

But, SARSA also has some benefits

- *It is better suited for online control*
  - ↳ *Our behaving policy is the one going towards optimal one*
  - ↳ *In Q-learning, the behaving policy is not the optimal one*
- *It has lower complexity*
  - ↳ *We just deal with one policy*

End of Story!

Model-Based RL
- Bellman Equation
  - value iteration
  - policy iteration

Model-free RL
- on-policy methods
  - temporal difference
    - Monte Carlo
    - SARSA
- off-policy methods
  - Q-learning

I would strongly suggest to start with programming part of Assignment 2!

There you solve Froozen Lake with SARSA and Q-Learning