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 versus Prediction

What we have done up to now is *prediction*

Prediction

We are given by a policy and intend to evaluate it by sampling

But, in most applications we deal with a *control problem*

Control

We are looking to move towards optimal policy while we are sampling

+ *But, we already talked about GPI with sampling! Didn't we?*
– Well! What we did makes sense if we learn offline!

GPI in Control Loop: *Offline vs Online Approach*

In *offline* RL, *we sample environment first and find the optimal policy later*

1. *Sample environment with sufficient number of episodes*
2. *Evaluate policies and improve them from the available dataset*
   ↳ *Go over the dataset over and over if needed*

We are however looking for an *online* RL *approach, we sample environment and learn optimal policy gradually as we sample*

1. *Take a single sample from environment, e.g., a single reward-state pair or a terminating trajectory*
2. *Estimate values and improve policy based on this single sample*
3. *Take a new sample* $\cdots$

+ *I am sure that we always thought about the second case! Isn't that right?!*
– Yes! But, if we want to do complete evaluation in each GPI iteration, *we will be extremely slow! Imagine* 1000 *episodes for each iteration!*

Direct GPI with *Prediction*

Recall a generic form of GPI with a prediction approach X

```
X_PolicyItr():
```
1: *Initiate two random policies* π *and* $\bar{\pi}$
2: **while** $\pi \neq \bar{\pi}$ **do**
3: $\hat{q}_\pi = \texttt{X\_QEval}(\pi)$ *and* $\pi \leftarrow \bar{\pi}$
4: $\bar{\pi} = \texttt{Greedy}(\hat{q}_\pi)$
5: **end while**

and $\texttt{Greedy}(\hat{q}_\pi)$ *is the basic improvement strategy*

$\texttt{Greedy}(\hat{q}_\pi)$:
1: **for** $n = 1 : N$ **do**
2: *Improve the by taking deterministically the best action*

$$\bar{\pi}\left(a^m | s^n\right) = \begin{cases} 1 & m = \underset{m}{\operatorname{argmax}}\, \hat{q}_\pi\left(s^n, a^m\right) \\ 0 & m \neq \underset{m}{\operatorname{argmax}}\, \hat{q}_\pi\left(s^n, a^m\right) \end{cases}$$

3: **end for**

Direct GPI with *Prediction*

Recall a generic form of GPI with a prediction approach `X`

```
X_PolicyItr():
```
1: *Initiate two random policies* π *and* $\bar{\pi}$
2: **while** $\pi \neq \bar{\pi}$ **do**
3: $\quad \hat{q}_\pi = \texttt{X\_QEval}(\pi)$ *and* $\pi \leftarrow \bar{\pi}$
4: $\quad \bar{\pi} = \texttt{Greedy}(\hat{q}_\pi)$
5: ***end while***

If we want our evaluation to be accurate enough: we need to keep playing each policy for a large number of episodes

- *This is extremely sample inefficient*
  - ↳ *Imagine how many times we should lose the game to update our strategy!*
  - ↳ *We as human do not really need so many losses*
- *In many practical settings is really cost-inefficient*
  - ↳ *Amount of losses we should pay in each iteration to evaluate the policy is not worth it!*

Online Control Loop via GPI

+ *But, how can we do anything about this?*
– Maybe we could *improve the policy after each update of action-values*

```
X_Control():
1: Initiate two random policies π and π̄
2: while π ≠ π̄ do
3:     q̂_π = X_QUpdate(π) and π ← π̄
4:     π̄ = Greedy(q̂_π)
5: end while
```

Here, `X_QUpdate`(π) *refers to one single update which is typically of the form*

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

for some G

Online Control Loop via GPI

```
X_Control():
1: Initiate two random policies π and π̄
2: while π ≠ π̄ do
3:     q̂_π = X_QUpdate(π) and π ← π̄
4:     π̄ = Greedy(q̂_π)
5: end while
```

+ *It sounds like a loose approach! Why should that work?!*
– We see accurate illustrations about that, *but for the moment*
  ↳ *we could think of a single update as a low-accuracy estimate*
  ↳ *we have already said the GPI is very robust against estimation error*

Let's start by a Monte-Carlo control loop

First Try: *Monte-Carlo Control Loop*

We can build a Monte-Carlo control loop for episodic environments

```
MC_Control(π):
```

1: *Initiate estimator as* $\hat{q}_\pi(s, a) = 0$ *for all states and actions*
2: **for** *episode* $= 1 : K$ *or until* π *stops changing* **do**
3: *Initiate with a random state-action pair* (S_0, A_0)
4: *Act via* $\pi = \texttt{Greedy}(\hat{q}_\pi)$
5: *Sample a trajectory*

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T\text{: terminal}$$

6: *Initiate with* $G = 0$
7: **for** $t = T - 1 : 0$ **do**
8: *Update current return* $G \leftarrow R_{t+1} + \gamma G$
9: *Update* $\hat{q}_\pi(S_t, A_t) \leftarrow \hat{q}_\pi(S_t, A_t) + \alpha(G - \hat{q}_\pi(S_t, A_t))$
10: **end for**
11: **end for**

Monte-Carlo Control: *Updating Action-Values*

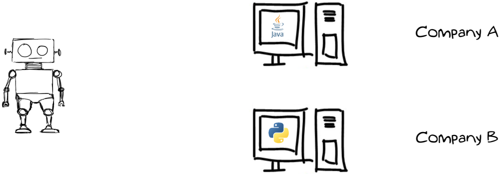
Comparing with `MC_QEval(`π`)`, there is only one difference, i.e.,

in-loop greedy improvement of the policy: line 4

1. *Estimate action-values over a sample trajectory*
2. *Improve the policy using this estimate by greedy approach*
3. *Sample the next trajectory using the improved policy*

+ *Can we guarantee the convergence of this control loop?*
– In the current state, not really! *Let's see an example!*

Example: *Our Multi-armed Bandit*

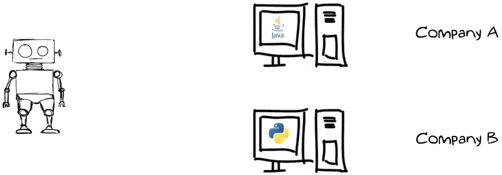


Let's get back to our very first RL problem in which the robot is to decide for a company: say the robot follows Monte-Carlo control loop

1. *it starts with a random decision*
   - ↳ *Say it decides for Company B and receives $150 income*
   - ↳ *Now, we have* $\hat{q}_\pi(\_, A) = 0$ *and* $\hat{q}_\pi(\_, B) = 150$
2. *in the next episode, it would definitely chooses to work at Company B*
   - ↳ *Say it receives $250 income this time*
   - ↳ *Now, we have* $\hat{q}_\pi(\_, A) = 0$ *and* $\hat{q}_\pi(\_, B) = 200$

   ⋮

Example: *Our Multi-armed Bandit*



The robot keeps working at Company B!

- + *But, can we guarantee that company A is not paying better?!*
- – *Well! Not really! In fact, even if we had worked there for a single day or so, we could still not guarantee!*

Greedy Improvement: *Lack of Exploration*

+ *Why is this happening? Why it doesn't happen when we apply direct GPI via Monte-Carlo?*
– In the latter, we do *exploration*; but now, we are only *exploiting*!

This is a general behavior of greedy improvement

Downside of Greedy Improvement

In greedy improvement, we only exploit our knowledge, i.e.,

we always act optimal based on what we know up to now

We thus lack exploration, i.e.,

we remain unaware about states and actions that we have not explored

We may never get the chance to explore them!

Improving via ϵ-Greedy Improvement

A classical approach to handle this issue is to improve by ϵ-greedy approach

ϵ-Greedy Improvement

Choose a small $0 < \epsilon < 1$, and improve after each update of action-values by greedy approach: at begining of each episode

- *with probability $1 - \epsilon$ act by the improved policy*
- *with probability ϵ act randomly*

We implemented this approach for multi-armed bandit in Assignment 1: in this approach we render a trade-off between exploitation and exploration

- *with probability $1 - \epsilon$ we exploit our improved policy*
- *with probability ϵ we explore the environment*

It's hard to find any improvement approach that can beat ϵ-greedy!

ϵ-Greedy Algorithm

We can algorithmically specify ϵ-greedy as

ϵ-`Greedy`$(\hat{q}_\pi)$:

1: **for** $n = 1 : N$ **do**

2: *Take next step randomly as*

$$\bar{\pi}\left(a^m | s^n\right) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{M} & m = \operatorname*{argmax}_m \hat{q}_\pi\left(s^n, a^m\right) \\ \dfrac{\epsilon}{M} & m \neq \operatorname*{argmax}_m \hat{q}_\pi\left(s^n, a^m\right) \end{cases}$$

3: **end for**

+ *That seems to solve exploration problem! But, is there any guarantee that $\bar{\pi}$ is going to be a better policy? For greedy approach, we could prove that we get always better!*
– Yes! We can actually prove it!

ϵ-Greedy Algorithm

Let's assume we have policy π given after ϵ-greedy improvement, and we improved it again via the ϵ-greedy approach from its action-values: *we can then write the value of new policy $\bar{\pi}$ as*

$$
\begin{aligned}
v_{\bar{\pi}}(s) &= \sum_{m=1}^{M} \bar{\pi}(a^m|s)\, q_{\bar{\pi}}(s, a^m) \\
&= \underbrace{\frac{\epsilon}{M} \sum_{m=1}^{M} q_{\pi}(s, a^m)}_{\text{exploration}} + \underbrace{(1-\epsilon) q_{\pi}(s, a^\star)}_{\text{exploitation}}
\end{aligned}
$$

We know that for any non-negative $w_1, \ldots, w_M$ that add up to one, we have

$$
\sum_{m=1}^{M} w_m q_{\pi}(s, a^m) \leqslant q_{\pi}(s, a^\star)
$$

ϵ-Greedy Algorithm

We have the improved value in terms of the initial action-values as

$$v_{\bar{\pi}}(s) = \frac{\epsilon}{M} \sum_{m=1}^{M} q_\pi (s, a^m) + (1-\epsilon) q_\pi (s, a^\star)$$

Let's now define

$$w_m = \frac{\pi(a^m|s) - \epsilon/M}{1-\epsilon}$$

We note that since π is an ϵ-greedy policy, we have $w_m \geqslant 0$ and

$$\sum_{m=1}^{M} w_m = \sum_{m=1}^{M} \frac{\pi(a^m|s) - \epsilon/M}{1-\epsilon} = 1$$

ϵ-Greedy Algorithm

Now, let us replace this bound in the previous equation

$$
\begin{aligned}
v_{\bar{\pi}}(s) &= \frac{\epsilon}{M}\sum_{m=1}^{M} q_\pi(s, a^m) + (1-\epsilon) q_\pi(s, a^\star) \\
&\geqslant \frac{\epsilon}{M}\sum_{m=1}^{M} q_\pi(s, a^m) + (1-\epsilon)\sum_{m=1}^{M} \frac{\pi(a^m|s) - \epsilon/M}{1-\epsilon} q_\pi(s, a^m) \\
&= \sum_{m=1}^{M} \pi(a^m|s) q_\pi(s, a^m) = v_\pi(s)
\end{aligned}
$$

ϵ-Greedy Improvement Theorem

Let π and be an ϵ-greedy policies, i.e., computed from some action-value function using ϵ-greedy algorithm. Assume $\bar{\pi}$ is derived by ϵ-greedy improvement from $q_\pi(s, a)$; then, $\bar{\pi} \geqslant \pi$

Online Control Loop via GPI and ϵ-Greedy Improvement

We can now build our control loop via ϵ-greedy algorithm

```
X_Control():
```

1: *Initiate two random policies* π *and* $\bar{\pi}$
2: ***while*** $\pi \neq \bar{\pi}$ ***do***
3: $\quad \hat{q}_\pi = \texttt{X\_QUpdate}(\pi)$ *and* $\pi \leftarrow \bar{\pi}$
4: $\quad \bar{\pi} = \epsilon\text{-}\texttt{Greedy}(\hat{q}_\pi)$
5: ***end while***

Attention

We are still using single update of action-values for policy improvement: this means that we may have bad estimates of action-values at initial iterations!

First Try: *Monte-Carlo Control Loop*

Monte-Carlo control loop for episodic environments is modified as

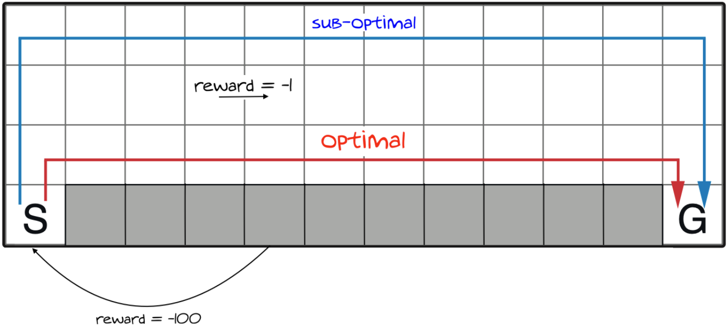
MC_Control(π):

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)
  Act via $\pi = \epsilon\text{-}\texttt{Greedy}(\hat{q}_\pi)$
  Sample a trajectory

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T\text{: terminal}$$

  Initiate with $G = 0$
  for $t = T - 1 : 0$ **do**
    Update current return $G \leftarrow R_{t+1} + \gamma G$
    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

Example: *Cliff Walking*

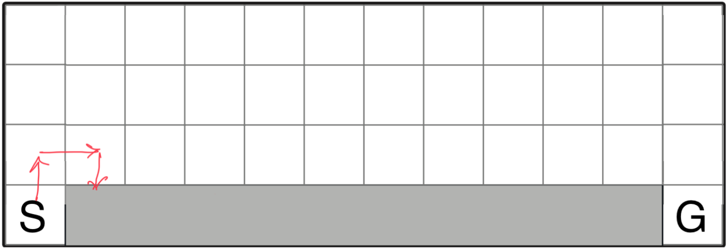


We have seen the cliff walking example in Assignment 1: we want to

- *get from S to G with shortest possible path*
- *avoid hitting the cliff $\equiv$ gray squares*
  - ↳ *each time we hit the cliff, we get back to S with a big negative reward*

Example: *Cliff Walking*

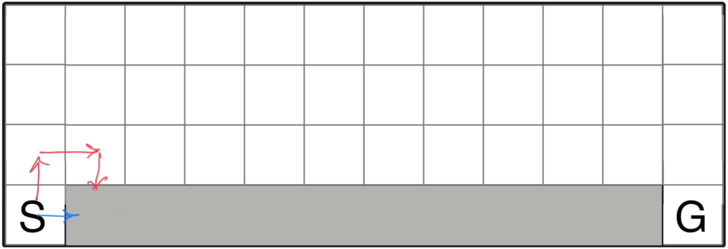
Say we use naive greedy policy: we start sampling trajectory and hit the cliff



We realize that our first action gave bad reward

Example: *Cliff Walking*

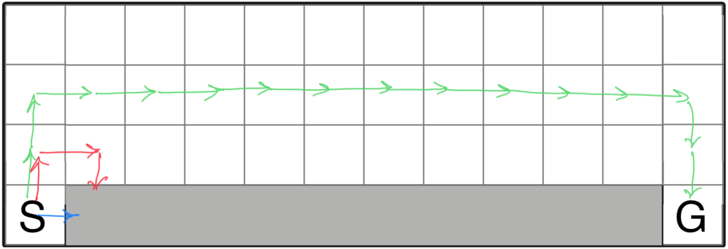
We now follow a better action, but we hit the cliff again



We realize that this action was even worse

Example: *Cliff Walking*

We get back to our first action, but now modify next actions



Say we are lucky and arrive at G

We will never go back to find the optimal path!

But with ϵ-greedy improvement, we get the chance to explore again: we may find the optimal path!

Greedy in Limit with Infinite Exploration Algorithms

+ *Sounds working! But can we guarantee that this approach will converge to optimal path?*
- Under some circumstances: Yes!

Recall that we said the following when we started Monte-Carlo

Asymptotic Convergence of Monte-Carlo

Let $\mathcal{C}_K(s, a)$ denote number of visits at state s followed by action a during K Monte-Carlo episodes. Assume random initialization is distributed such that

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

for any state s and action a; then, we can guarantee $\hat{q}_\pi(s, a) \xrightarrow{K \uparrow \infty} q_\pi(s, a)$

Greedy in Limit with Infinite Exploration Algorithms

The main idea in this result was that

> *As long as we do enough sampling, so that we see all states and actions enough number of times, Monte-Carlo will converge*

We can claim the same thing here

> *If we keep playing enough, we explore all states and actions; then, eventually we get very sure about optimal values and actions*

But there is a small point here: *if we keep on using ϵ-greedy policy even after we got sure, we can still perform sub-optimal*

> *We should stop exploring once we have visited all states and actions*

This is what we call

Greedy in Limit with Infinite Exploration $\equiv$ GLIE

GLIE Algorithms

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

GLIE Algorithms

+ *It seems that they contradict! First one needs us to explore and the second to exploit!*
– We could simply get rid of it by scaling ϵ

Say we choose ϵ to scale reversely by the number of episodes, e.g.,

$$\epsilon_k = \frac{1}{k}$$

Then, we have both the constraints satisfied

1. *We keep exploring a lot in initial episodes*
2. *We focus more on exploiting in later episodes*

This is what we do in practice!

ϵ-Greedy Monte-Carlo is GLIE

It is easy to show that Monte-Carlo with shrinking ϵ-greedy improvement is GLIE

```
MC_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)
  Set $\epsilon = 1/k$ *and act via* $\pi = \epsilon\text{-}\mathtt{Greedy}(\hat{q}_\pi)$
  Sample a trajectory

$$S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T\text{: terminal}$$

  Initiate with $G = 0$
  for $t = T - 1 : 0$ **do**
    Update current return $G \leftarrow R_{t+1} + \gamma G$
    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