Lecture 3: Model-free prediction and control: MC and TD methods

SUMS 707 - Basic Reinforcement Learning

Gabriela Moisescu-Pareja and Viet Nguyen

McGill University, Mila

January 28, 2021

Recap of last lecture

Last time we introduced:

- Bellman equations, Bellman optimality equations
- Solving MDPs
  - Policy iteration
  - Value iteration

Recap of last lecture: Bellman equations I

Recall that the value functions involve an infinite sum, but one key observation allows you to express them in a recursive fashion:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s, \pi\right] \\ &= \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots \middle| s_0 = s, \pi\right] \\ &= \mathbb{E}\left[r_1 + \gamma V^{\pi}(s_1) \middle| s_0 = s, \pi\right] \end{aligned}$$

and similarly,

$$Q^{\pi}(s, a) = \mathbb{E}\left[r_1 + \gamma Q^{\pi}(s_1, a_1) \middle| s_0 = s, a_0 = a, \pi\right]$$

Recap of last lecture: Bellman equations II

We can take this even further to note that there is a relationship between the state value function and the state-action value function! Given a policy π, $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ are related as follows:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^{\pi}(s,a)\right] && (1) \\ &= \sum_{a\in\mathcal{A}} \pi(a|s) Q^{\pi}(s,a) && (2) \end{aligned}$$

and

$$Q^{\pi}(s,a) = r_{(s,a)} + \gamma \sum_{s'\in\mathcal{S}} \mathbb{P}\left(s'|s,a\right) V^{\pi}(s') \qquad (3)$$

Recap of last lecture: Bellman equations III

Assuming a deterministic policy π, we can re-write V^π and Q^π as follows:

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, a) V^\pi(s')$$

and

$$Q^\pi(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, a) Q^\pi(s', \pi(s'))$$

.

Solving MDPs

We introduced Dynamic Programming algorithms that were obtained by turning the Bellman equations into update rules for improving approximations of the desired value functions.

Recap of last lecture: Policy iteration I

Policy iteration is an algorithm for the *control problem*. At a high-level, policy iteration performs the following steps:

1. Policy evaluation: value function is computed for the currently policy
2. Policy improvement: the policy is made greedy w.r.t. the value function
3. Repeat steps 1, 2 until convergence to optimal policy.

Recap of last lecture: Policy iteration II

Policy Evaluation. Consider a sequence of approximations $V_0, V_1, V_2, \ldots$

- Initial approximation V_0 is chosen arbirtrarily (though often set to 0)
- Each successive approximation is obtained by using the Bellman equation for V^π as an update rule:

$$\begin{aligned} V_{k+1}(s) &= \mathbb{E}\left[r_{t+1} + \gamma V_k(s_{t+1}) \middle| s_t = s, \pi\right] \\ &= \sum_a \pi(s,a) \sum_{s'} \mathbb{P}\left(s'|s,a\right)\left[r(s,a) + \gamma V_k(s')\right] \end{aligned}$$

for all states $s \in S$.

We have that $V_k = V^\pi$ is a fixed point for this update rule thanks to the Bellman equation for V^π. The sequence $\{V_k\}$ can be shown, in general, to converge to V^π as $k \to \infty$.

Recap of last lecture: Policy iteration III

Policy evaluation in the Grid World MDP:

- Action set: $\{N, S, E, W\}$
- Undiscounted: $\gamma = 1$
- Two terminal states
- Uniform random policy:

$$\pi(N|s) = \pi(S|s) = \pi(E|s) = \pi(W|s) = 0.25$$

- Reward: -1 until a terminal state is reached.

Question: How many steps in expectation will it take until we hit one of the terminal states? Intuitively, this should tell us how good a state is (its value).

Recap of last lecture: Policy iteration IV

v_k for the Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

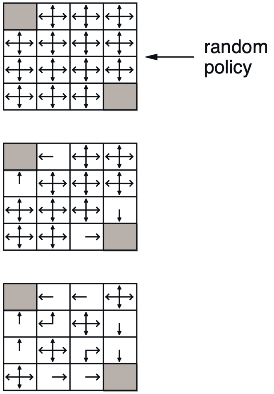
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

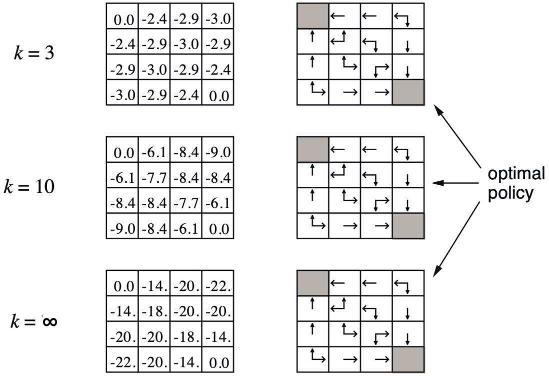
0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

Greedy Policy w.r.t. v_k



Recap of last lecture: Policy iteration V

For any non-terminal state, to compute its current value, we look at the reward we can obtain by taking any action, and then we look at the value according to our previous estimate and sum these.



$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

This is iterating the Bellman equation and feeding the value back into itself.

Recap of last lecture: Value iteration I

Value Iteration. Consider a sequence of approximations $V_0, V_1, V_2, \dots$

- Initial approximation V_0 is chosen arbitrarily.
- Each successive approximation is obtained by

$$\begin{aligned} V_{k+1}(s) &= \max_{a \in A} \mathbb{E}\left[r_{t+1} + \gamma V_k(s_{t+1}) \middle| s_t = s, a_t = a\right] \\ &= \max_{a \in A} \sum_{s'} \mathbb{P}\left(s' | s, a\right) \left[r(s,a) + \gamma V_k(s')\right] \end{aligned}$$

  for all $s \in S$.

And $\{V_k\}$ can be shown to converge to V^* under the same conditions that guarantee the existence of V^*.

Recap of last lecture: Value iteration II

Value iteration in the Grid World MDP, but we don't have a starting policy. We want to find the shortest path from any square to the terminal state.

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

One-step lookahead by maximizing over all possible things you can do from that state.

Plan for today

- Model-free prediction and control:
    - Temporal difference learning methods
    - Monte-Carlo methods
    - SARSA, Q-learning

Unknown environment

- The dynamics of the world is not given to us
- We will have to explore it, gain information, and act accordingly

What we know I

- We know what an MDP looks like
- We know how how to find the value function given an MDP (prediction)
- We know how to optimize a policy by looping between predicting the value function and acting greedily (control)
- Can we do random things to formulate an empirical MDP and solve it using known methods?

Model-based vs. Model-free

Yes! This is called model-based reinforcement learning, a topic for the future.

- Model-based RL:
  - Learn a model from experience
  - Plan a value function (and/or policy) from the model
- Model-free RL:
  - No model
  - Learn a value function (and/or policy) directly from experience

Model-free prediction I

We would like to compute some value functions given some policy π, this time, without knowing transitions and rewards. Recall the value function:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s, \pi\right], \quad \forall s \in \mathcal{S}$$

A potential idea:

- For state s, compute an average over multiple independent realizations started from s
- $\rightarrow$ Monte-Carlo vibes

Model-free prediction II

Are there any problems with that approach? Maybe!

- What if the *variance* of the returns are high? The quality of your estimates are poor
- When you're interacting in a closed-loop fashion (estimation happens when interacting with system) might be even impossible to reset back to some particular state! We won't even be able to sample many times from that state...
- Monte-Carlo methods can be used but only after introducing additional bias

We will look at temporal difference (TD) learning (Sutton, 1984, 1988), which uses *bootstrapping*.

TD learning

- Learn directly from episodes of experience
- Model-free: no knowledge of MDP transitions/rewards
- Learn from incomplete episodes, learn on-the-fly
- Updates a guess towards a guess

Big brain manipulations

Recall

$$\begin{aligned} V^\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s, \pi\right] \\ &= \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \middle| s_0 = s, \pi\right] \\ &= \mathbb{E}\left[r_1 + \gamma V^\pi(s_1) \middle| s_0 = s, \pi\right] \end{aligned}$$

In general, this holds:

$$V^\pi(s_t) = \mathbb{E}\left[r_{t+1} + \gamma V^\pi(s_{t+1}) \middle| \pi\right]$$

How about using the integrand as an update rule?

TD(0)

Given some function $V : \mathcal{S} \to \mathbb{R}$, we update $V(s_t)$ towards an *estimated* return $r_{t+1} + \gamma V(s_{t+1})$

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

where α is the *learning rate*.

- $r_{t+1} + \gamma V(s_{t+1})$ is the *TD target*
- $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ is the *TD error*

TD(0) in the tabular case

Algorithm 1: TD(0) algorithm. This must be called after each transition

```
**Function** TD0($X$, $R$, $Y$, $V$):
    $\delta \rightarrow R + \gamma V[Y] - V[X]$
    $V[X] \rightarrow V[X] + \alpha\delta$
    **return** V
```

- X is the last state
- Y is the next state
- R is the immediate reward
- V is the *array* of size $|\mathcal{S}|$ storing current value estimates

Here, the *TD* target depends on the estimated value function, we say that the algorithm uses *bootstrapping*.

Stochastic approximation

We can show that if $TD(0)$ converges, it converges to a function $\hat{V}$ such that the *expected* temporal difference given $\hat{V}$

$$F\hat{V}(s) \overset{def}{=} \mathbb{E}\left[r_{t+1} + \gamma\hat{V}(s_{t+1}) - \hat{V}(s_t) | s_t = s\right]$$

is *zero* for all states $s \in \mathcal{S}$ that would be sampled infinitely often.

We can also show that $F\hat{V} = T\hat{V} - \hat{V}$ where T is the Bellman operator! As we saw previously, $T\hat{V} - \hat{V} = 0$ has a unique solution, which is exactly the value function V!

- If TD(0) converges and all states are sampled infinitely often, then it must converge to V, the true value function

Monte-Carlo I

What if instead of bootstrapping, we want to consider the actual rewards? Recall the idea of computing an average over multiple realizations... We shall now formalize this idea with Monte-Carlo value prediction.

- Learn directly from *episodes* of experience
- Model free
- Learn from *complete episodes*: no bootstrapping
- Simple idea: value = mean return
- Can only apply MC to episodic MDPs, all episodes must terminate somehow

Monte-Carlo II

We want to learn V^{π} from episodes of experience under policy π. Assume our episodes terminate in T steps. Recall the return:

$$\mathcal{R} = \sum_{t=0}^{T} \gamma^t r_{t+1}$$

We can consider the return at any time $t \in [T]$:

$$\mathcal{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$$

The value function is the expected return:

$$V^{\pi}(s) = \mathbb{E}\left[\mathcal{R}_t | s_t = s, \pi\right]$$

Monte-Carlo III

MC policy evaluation uses *empirical mean return* to approximate expected return.

- TD target:
$$r_{t+1} + \gamma V(s_{t+1})$$
- MC target:
$$\mathcal{R}_t$$
- Instead of updating:
$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$
- We update:
$$V(s_t) \leftarrow V(s_t) + \alpha(\mathcal{R}_t - V(s_t))$$

Implementing Every-Visit MC

One can do the following: To evaluate a state s,

- Every time-step t that state s is visited in an episode
- Increment counter $N(s) \leftarrow N(s)1$
- Increment total return $S(s) \leftarrow S(s) + \mathcal{R}_t$
- $V(s) = S(s)/N(s)$

We can prove that $V(s) \rightarrow V^{\pi}(s)$ as $N(s) \rightarrow \infty$.

Smol incremental mean trick

How does this relate to the update:

$$V(s_t) \leftarrow V(s_t) + \alpha(\mathcal{R}_t - V(s_t))$$

The mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally!

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})
\end{aligned}$$

Incremental MC updates

After sampling an episode $s_0, a_0, r_1, \dots, r_T, s_T$, For each state s_t with return $\mathcal{R}_t$,

$$N(s_t) \leftarrow N(s_t) + 1$$
$$V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(\mathcal{R}_t - V(s_t))$$

Here, α is chosen to be $\frac{1}{N(s_t)}$, but we can tune it as a hyperparameter.

MC is a stochastic approximation method, can prove that our estimates V converges to the true V^π almost surely.

MC vs TD

Both algorithms achieve the same goal, but which one is better?

- TD learns *before* knowing the final outcome
    - TD learns on-the-fly after every step without having to wait an episode to termination
    - MC needs to wait (to be able to calculate episodic returns $\mathcal{R}_t$)
- TD can learn without the final outcome
    - TD can learn from incomplete sequences
    - MC needs complete sequences
    - TD works in continuing environments
    - MC only works for episodic environments

There are interesting pathological environments where one converges faster than the other.

Bias/Variance

- Return $\mathcal{R}_t$ is an *unbiased* estimate of $V^{\pi}(s_t)$
- true TD target $r_{t+1} + \gamma V^{\pi}(s_{t+1})$ unbiased estimate of $V^{\pi}(s_t)$
- TD target $r_{t+1} + \gamma V(s_{t+1})$ is biased (from initialization)! Although bias decays exponentially as we experience more of the environment

But because of this bias, *TD* target has a much lower variance than the return used in MC policy evaluation.

- $\mathcal{R}_t$ depends on many random actions, transitions, rewards...
- TD target only depends on one random action, transition, reward

Looking a bit further into the future

We saw $TD(0)$ for one-step look-ahead into the future. What if we want to look n steps ahead?

$$\begin{aligned}
&r_{t+1} + \gamma V(s_{t+1}) \\
&r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}) \\
&\dots \\
&r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T
\end{aligned}$$

We went from $TD(0)$ to MC!

n-step TD learning

Define the *n*-step return

$$\mathcal{R}_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$$

We have the *n*-step TD-learning update rule

$$V(s_t) \leftarrow V(s_t) + \alpha \left(\mathcal{R}_t^{(n)} - V(s_t) \right)$$

- When $n = 1$, this is TD(0)
- When $n = T$, this is MC

Generalization: λ- return

But which n to choose? We could simply take an average of all estimates of $V(s_t)$ from $n = 1$ to $n = T$ in the episodic setting, but what happens when our horizon is infinite? We could take a *weighted* average:

$$\mathcal{R}_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1}\mathcal{R}_t^{(n)}$$

Here, our weights $(1-\lambda)\lambda^{n-1}$ are exponentially decaying, we weigh small n's exponentially more than large n's. We update:

$$V(s_t) \leftarrow V(s_t) + \alpha\left(\mathcal{R}_t^\lambda - V(s_t)\right)$$

This is the *forward-view* of TD(λ).

TD(λ) and TD(1)

There is a *backward* view of TD(λ) using *eligibility traces*, which allows direct computation instead of requiring complete eepisodes like the forward view.

- TD(1) is roughly equivalent to every-visit Monte-Carlo
- TD(λ) combines both TD(0) and MC in a smart way

Control

We saw two model-free algorithms (TD, MC) that can be used to predict the value function given some policy π.

We also saw that they recycle roughly the same ideas via the TD(λ) generalization.

We know how to predict and control when the environment is given to us. We just learned how to predict when the environment is unknown.

How do we control our agent when the environment is unknown?

Control problems

- Playing computer games
- Beating Magnus at chess
- Protein folding
- Robot walking
- Helicopter control

For these problems, either the MDP is unknown or is way too large. However, experiences can be sampled (sometimes at a high cost, :( ). We can use model-free control to solve these problems.

The policy iteration blueprint

One is now highly familiar with the pattern:

- We estimate V^{π}
- Then we generate $\pi' \geq \pi$
- We now estimate $V^{\pi'}$
- Then we generate $\pi'' \geq \pi'$
- $\ldots$

Eventually, we cross our fingers and hope that we found an optimal policy.

Does greedy policy improvement over V work here?

Greedy policy improvement

Yes! But only if we work with the action-value function!

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$$

If we work with V, this requires the model of the MDP

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} r_{(s,a)} + \mathbb{E}\left[V(s') | s' \sim \mathbb{P}(\cdot | s, a)\right]$$

We will then be working with the action-value function.

Policy Iteration with action-value function

Recall that we are in a model-free setting: the environment is not given to us, and we're not approximating the environment either.

From our blueprint, we fill in the gaps:

- Policy evaluation: MC evaluation, but instead of using V, we use Q to approximate Q^{π}
- Policy improvement: Greedy ?

Is there a problem with this approach?

Food for thought

- You open door 1 and get a reward of 0, you update your action-value function
- You open door 2 and get a reward of $+1$, you update your action-value function
- You are greedy. You open door 2 again and get a reward of $+3$. You update your action-value function
- You suddenly remember rewards ar stochastic
- But you are greedy
- You open door 2 again and get a reward of $+1$
- ...

ϵ-greedy exploration

Smol brain idea but extremely useful

- With probability $1 - \epsilon$ we choose the greedy action
- With probability ϵ we choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon, & a = \arg\max_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/m, & otherwise \end{cases}$$

ϵ-greedy policy improvement

Theorem

For any ϵ-greedy policy π, the ϵ-greedy policy π' w.r.t. Q^π is an improvement, $V^{\pi'}(s) \geq V^\pi(s)$

Recall

$$Q^\pi(s, \pi'(s)) = \sum_{a \in \mathcal{A}} \pi'(a|s) Q^\pi(s, a)$$

But you know π' (given in the previous slide), so you can algebra your way to victory.

MC Control with ϵ-greedy

- Sample the k^{th} episode using π: $s_0, a_0, r_1, \ldots, r_T, s_T$
- For each s_t and a_t in the episode,

$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$$
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(\mathcal{R}_t - Q(s_t, a_t))$$

- Improve the policy according to the updated Q:

$$\epsilon \leftarrow 1/k$$
$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

Convergence guarantees

Theorem

With the assumptions

- *All state-action pairs are explored infinitely many times*
- *The policy converges on a greedy policy MC control converges to the optimal action-value function, that is, $Q(s, a) \rightarrow Q^*(s, a)$.*

TD Control

MC is cool but the potential problems persist:

- MC methods in general have higher variance than TD methods
- TD can be performed online
- TD can learn from incomplete sequences

What if instead of looping MC control $\leftrightarrow$ ϵ-greedy, we loop TD control $\leftrightarrow$ ϵ-greedy? How would this look like?

SARSA updates

After every time-step, recall TD learning updates for policy evaluation:

$$V(s_t) \leftarrow V(s_t) + \alpha \left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

We mirror the same idea, but now with Q. Given that we're in state s we take action a, the environment transitions us to state s' and gives reward r, we take a subsequent action a':

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q(s',a') - Q(s,a)\right)$$

We emphasize that SARSA is an *on-policy* algorithm. This will be made clear in the next lecture

Next lecture

What do we do when we don't know the dynamics of the environment?

- On/Off-policy, Q-learning
- Function approximation to deal with very thicc spaces
- Deeeeeeeep reinforcement learning
- Why the above is a meme

References

- Reinforcement Learning: Theory and Algorithms by Alekh Agarwal, Nan Jiang, Sham M. Kakade
- Algorithms for Reinforcement Learning by Csaba Szepesvári
- Reinforcement Learning: An Introduction by Andrew Barto and Richard S. Sutton
- "Introduction to Reinforcement Learning" Lectures by David Silver
- "CS 598 - Statistical Reinforcement Learning" Notes by Nan Jiang