Introduction To Reinforcement Learning

Felipe Glicério Gomes Marcelino

18 April 2020

Slide 3

Today's Lecture

1. Definition of a Markov decision process
2. Definition of reinforcement learning problem
3. Anatomy of a RL algorithm
4. Brief overview of RL algorithm types

- Goals:
  - Understand definitions & notation
  - Understand the underlying reinforcement learning objective
  - Get summary of possible algorithms

Slide 4

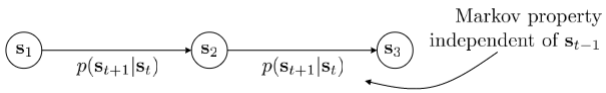
Some observations:

- Behavioral Cloning works with partial observations (o_t), but
- Dagger assumption about error ϵ is only possible with fully observations states (s_t)

Slide 5

Reward functions

- Reward functions tell us which states and actions are better.
- Finds out a good reward function is one of the toughest problems in reinforcement learning.
- Objective: Taking actions that will maximize long-term rewards rather than rewards at immediate next step
- Markov decision process definition: $s, a, r(s, a), p(s'|s, a)$



.png

Figure 1: Markov Chain

Slide 8

Markov Chain

Definitions:

$$M = \{S, T\}$$

$$S - \text{State Space} \quad \text{states } s \in S \text{ (discrete or continuous)}$$

$$\tau - \text{Transition Operation} \quad p(s_{t+1}|s_t) \tag{1}$$

$$\mu_{t,i} = p(s_t = i) \quad \text{Probability of being in state I at time step T} \tag{2}$$

$$T_{i,j} = p(s_{t+1} = i|s_t = j) \quad (3) \tag{4}$$

- (1) - Probability of go to state s_{t+1} from s_t
- (2) - $\vec{\mu_t}$ is a vector of probabilities
- (3) - Matrix - Probability of going at a state i given that currently state J
- (4) - then $\vec{\mu_{t+1}} = \tau\vec{\mu_t}$

Figure [1] shows how is a markov chain graphic model.

Slide 9 - 10

Markov Decision Process

Defitions:

S - state space — states $s \in S$ (discrete or continuous)

A - action space — actions $a \in A$ (discrete or continuous)

τ - transition operator (Tensor: 3D)

let $\mu_{t,j} = p(s_t = j)$ — $\mu_{t,i} = \sum_{j,k} \tau_{i,j,k}\mu_{t,j}\xi_{t,k}$

let $\xi_{i,k} = p(a_t = k)$

let $\tau_{i,j,k} = p(s_{t+1} = i|s_t = j, a_j = k)$ r - reward function — $r : S \times A \to \mathbb{R}$

$r(s_t, a_t)$ - reward

0.1 Slide 12-14

Figure [2] and Figure [3] show how is a markov process graphic model and markov chain with augmentation state space.

The goal of reinforcement learning. Markov decision giving us:

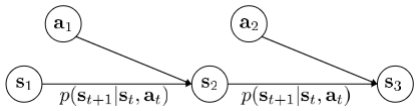


Figure 2: Markov Process

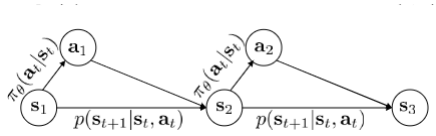


Figure 3: Markov chain with augmentation state space

$$\underbrace{p_\theta(s_1, a_1, \cdots, s_T, a_T)}_{p_\theta(\tau)} = \underbrace{p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)}_{\text{Markov chain on}(s,a)}$$

τ is a trajectory. It is easy to see that adding the policy inside $\prod$, the MDP models transform to a markov chain. Using above, we want find out the best parameters to maximize the expectation of reward:

$$\theta^* = \underset{\theta}{argmax} E_{\tau\sim p_\theta(\tau)} = \left[\sum_t r(s_t, a_t)\right]$$

Using MDP definition above, all actions are possible on all states. But, it is possible to put negative rewards when some actions is illegal given a specific state. Or maybe remap illegal actions to do something else.

Tip:

$$E_{x\sim p(x)}[f(x)] = \int p(x)f(x)dx$$

Redefining the state space to group the S and A, it is possible using the graphic model in Figure [4](Markov Chain). So the probability of next state and the next action given the current stete and current action is just obtained by multiplying together transition dynamics of the original MDP and the policy.

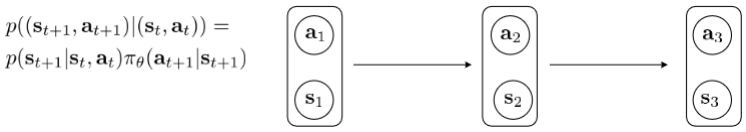


Figure 4: Markov chain notation using A and S space together.

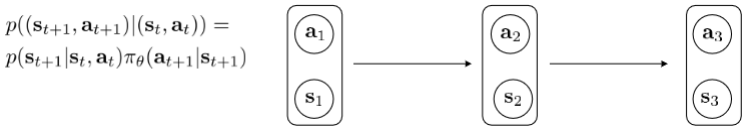


Figure 5: Transition Operation

Slide 15

Finite Horizon Case: State-Action Marginal

$$\theta^* = \underset{\theta}{argmax} E_{\tau \sim p_\theta(\tau)} = \left[\sum_t r(s_t, a_t) \right] \tag{1}$$

$$\theta^* = \underset{\theta}{\text{argmax}} \sum_{t=1}^{T} E_{(s_t, a_t) \sim p_\theta(s_t, a_t)[}[r(s_t, a_t)] \tag{2}$$

E can be pushing inside the $\sum$, by linearity of expectations. And because the term inside the sum only depends on s_t and a_t, I can rewrite it instead of having it be an expectation over all trajectoreis, I can have it be expectation over just s_t and a_t, that calls state action marginal $\rightarrow p_\theta(s_t, a_t)$

Slide 16

Infinite Horizon case: Stationary Distribution

Now, if $T = \infty$? Considering:

$$\theta^* = \underset{\theta}{\text{argmax}} \sum_{t=1}^{T} E_{(s_t, a_t) \sim p_\theta(s_t, a_t)[}[r(s_t, a_t)] \tag{3}$$

$$\theta^\star = \arg\max_\theta E_{(\mathbf{s},\mathbf{a})\sim p_\theta(\mathbf{s},\mathbf{a})}[r(\mathbf{s},\mathbf{a})] \qquad \theta^\star = \arg\max_\theta \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim p_\theta(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)]$$

infinite horizon case — finite horizon case

Figure 6: Finite Horizon

$$\mu = \begin{bmatrix} p(s_1,a_1) \\ p(s_1,a_2) \\ \vdots \\ p(s_1,a_m) \\ p(s_2,a_1) \\ \vdots \end{bmatrix} \tag{4}$$

$p(s_t, a_t)$ converges to a stationary distribution. Therefore, $\mu = \tau\mu$. Solving $(\tau - \mathbb{I})\mu = 0$ result in Stationary Distribution. **OBS:** μ is eigenvector of τ. It means that the distribution over μ doesn't change after a transition. $\mu = p_\theta(s,a) \leftarrow$ Stationary distribution.

As T go to ∞ in (3), the sum becomes dominate by terms of stationary distribution. However, the result of $\sum$ over ∞ is ∞. So, is necessary to adding a new term to have well-defined answer and sum up to something finite. The term is $\frac{1}{T}$. It is called **Undiscounted Average Return Formulation of RL**. Then the optimal policy can be represent as follows:

$$\theta^* = \underset{\theta}{\text{argmax}} \frac{1}{T} \sum_{t=1}^{T} E_{(s_t,a_t)\sim p_\theta(s_t,a_t)[}[r(s_t,a_t)] \rightarrow E_{(s,a)\sim p_\theta(s,a)}[r(s,a)] \tag{5}$$

Slide 18

Figure [6] shows as we don't care about the reward of a particular trajectory, we care about the expectation reward average over all the trajectories that could happen. Dealing with expectation is better because expectation functions are smooth and they are better to derivate.

Slides 26-29

Again, maximize the expectation of reward is what we want to. So, how do we deal with all these expectation?

Maximize:

$$E_{\tau\sim p_\theta(\tau)} \left[\sum_{t=1}^{T} r(s_t,a_t) \right] \tag{6}$$

We can express the equation (6) as a recursive way:

$$E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right] = E_{s_1\sim p(s_1)}\big[E_{a_1\sim\pi(a_1|b_1)} \underbrace{\big[r(s1,a_1) + E_{s_2\sim p(s_2|s_1,a_1)}\big[E_{a_2\sim\pi(a_2|s_2)}\big[r(s_2,a_2)+\cdots|s_2\big]|s_1,a_1\big]}_{Q(s_1,a_1)}|s_1\big]\big] \tag{7}$$

Simplified equation (6) is:

$$E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right] = E_{s_1\sim p(s_1)}\big[E_{a_1\sim\pi(a_1|s_1)}\big[Q(s_1,a_1)|s_1\big]\big] \tag{8}$$

If we know $Q(s_1,a_1)$, the policy can be modified to improve the reward expectation. For instance: Maximize the probability of the action a_1 so that the probability of $\pi_{a_x|s_1} = 1$ if $a_1 = argmax_{a_1} Q(s_1,a_1)$

Definition: Q-function

$Q^\pi(s_t,a_t = \sum_{t'=t}^{T} E_{\pi_\theta}\big[r(s_{t'},a_{t'})|s_t,a_t\big]$: total reward from taking a_t in s_t for the rest of trajectory. Evaluate it exactly is intractable, but is is possible to use some algorithms to approximate Q-values.

Definition: Value-Function

It is similar to Q-Function, but not taking into account the action as input for calculations.

$V^\pi(s_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\big[r(s_{t'},a_{t'})|s_t\big]$: total reward from s_t. Using Q-function to express the Value-function: sum of all actions $Q(s_t,a_t)$(Expectation) $V^\pi(s_t) = E_{a_t\sim\pi(a_t|s_t)}\big[Q^\pi(s_t,a_t)\big]$. As a result, maximize the value in expectation of at the first state, consequently maximizes the total expected reward of the entire policy.

$$E_{s_1\sim p(s_1)}\big[V^\pi(s_1)\big] \text{ is the RL objective!} \tag{9}$$

How to use Q-Functions and Value-Functions to improve

Idea 2 in Figure [7] have been using in **actor-critic** models.

Slide 30-35

Types of RL Algorithms

$$\theta_* = \underset{\theta}{argmax} E_{\tau\sim p_\theta(\tau)}\Big[\sum_t r_(s_t,a_t)\Big] \tag{10}$$

Policy Gradients:

Directly differentiate the equation (10). Diagram in Figure [8]

Using Q-functions and value functions

Idea 1: if we have policy π, and we know $Q^\pi(\mathbf{s}, \mathbf{a})$, then we can *improve* π:

set $\pi'(\mathbf{a}|\mathbf{s}) = 1$ if $\mathbf{a} = \arg\max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$

this policy is at least as good as π (and probably better)!

and it doesn't matter what π is

Idea 2: compute gradient to increase probability of good actions $\mathbf{a}$:

if $Q^\pi(\mathbf{s}, \mathbf{a}) > V^\pi(\mathbf{s})$, then $\mathbf{a}$ is *better than average* (recall that $V^\pi(\mathbf{s}) = E[Q^\pi(\mathbf{s}, \mathbf{a})]$ under $\pi(\mathbf{a}|\mathbf{s})$)

modify $\pi(\mathbf{a}|\mathbf{s})$ to increase probability of $\mathbf{a}$ if $Q^\pi(\mathbf{s}, \mathbf{a}) > V^\pi(\mathbf{s})$

These ideas are *very* important in RL; we'll revisit them again and again!

Figure 7: Ideas to improve policy

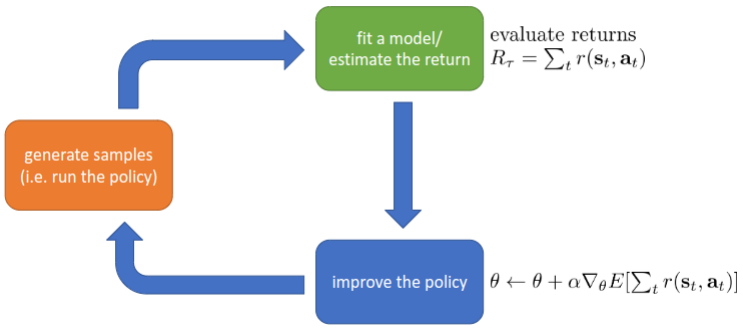


Figure 8: Policy-Gradients Diagram

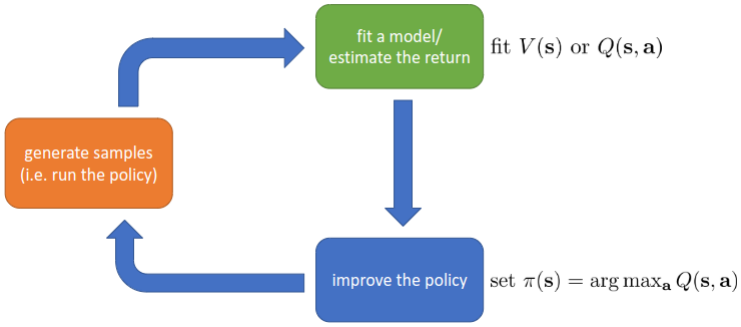


Figure 9: Value-Function Diagram

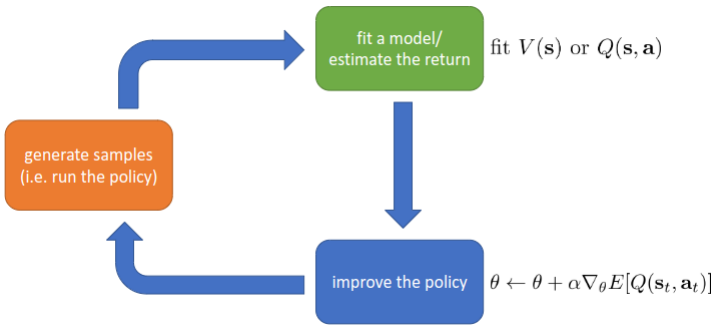


Figure 10: Actor-Critic Diagram

Value-Based

Estimate value function or Q-function of the optimal policy without representing the policy explicitly. Only keep tracking Q or V and improve these two functions through an iterative procedure. Diagram in Figure [9]

Actor-Critic

Combination of Policy Gradients and Value-Based. They estimate a value function or a Q-function and then they improve the policy using something very similar to policy gradients. They calculate a gradient using value functions and Q-functions. Diagram in Figure [10]

Model-Based RL

Estimate the transition model between states using s_t and a_t.

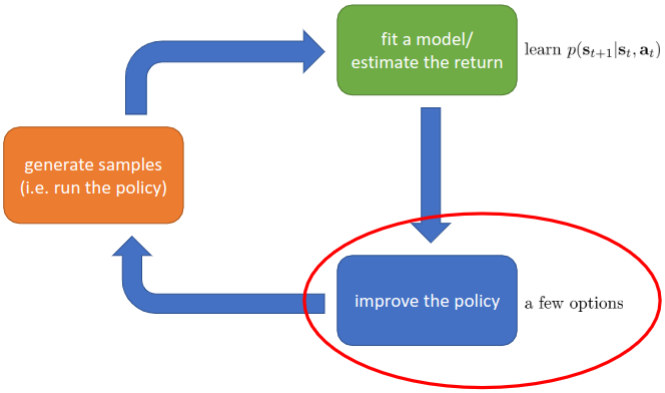


Figure 11: Model-Based Diagram

- Use the model for planning(no explicit policy) - Optimal Control or Planning techniques
- Use the model to improve policy by pushing gradients or using the model to generate some synthetic experience and plugging it into more standard model-free RL algorithms, which maybe can be any of this tree types of model above.

Diagram model-based RL in Figure [11]

Slide 36-43

Tradeoffs

- Different tradeoffs
  - Sample Efficiency: How many times are necessary to run the policy to correct collect samples.
  - Stability and ease of use.
- Different assumptions
  - Stochastic or deterministic?
  - Continuous or discrete?
  - Episodic or infinite horizon?
- Different things are easy or had in different settings
  - Easier to represent the policy?
  - Easier to represent the model?

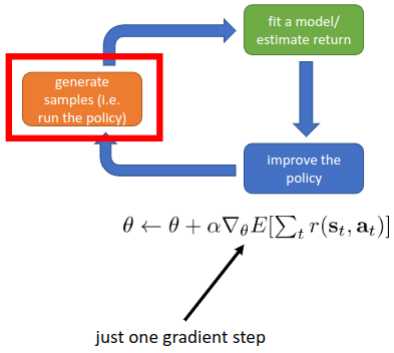


Figure 12: Sample Efficiency

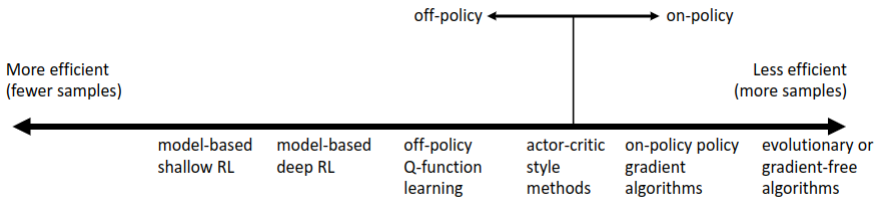


Figure 13: Sample efficiency for different algorithms

Sample Efficiency

- Sample efficiency: How many samples do we need to get a good policy? A sample means: Taking policy and run it and see what it does.
- Is the algorithm *off policy*?
  - *Off policy*: Able to improve the policy without generating new samples from that policy.
  - *On policy*: Each time the policy is changed, even a little bit, we need to generate new samples.

Figure [13] shows a comparison between different algorithms according to sample efficiency.

Stability and ease of use

- Does it converge?
- And if it converges, to what?

- And does it converge every time?
- Supervised learning: Almost always gradient descent
- Reinforcement Learning: often not gradient descent
  - Q-learning: Fixed Point Iteration
  - Model-based RL: Model is not optimized for expected reward
  - Policy Gradient: is gradient descent, but also often the least efficient!
- Value function fitting
  - At best, minimizes error of fit("Bellman error")
    - Not the same as expected reward
  - At worst, doesn't optimize anything
    - Many popular deep RL value fitting algorithms are not guaranteed to converge to anything in the nonlinear case.
- Model-based RL
  - Model minimizes error fit
    - This will converge
  - Not guarantee that better model = better policy
- Policy gradient
  - The only one that actually performs gradient descent on the true objective.

Assumptions

- Common assumption #1: Full Observability - Seeing states and not observations
  - Generally assumed by value function fitting methods
  - Can be mitigate by adding recurrence
- Common assumed #2: Episodic Learning - Delimit of trials.
  - Often assumed by pure policy methods
  - Assumed by some model-base RL methods
- Common assumption #3: Continuity or Smoothness
  - Assumed by some continuous value function learning methods
  - Often assumed by some model-based RL methods

Examples of specific algorithms

- Value function fitting methods
  - Q-learning, DQN
  - Temporal difference learning
  - Fitted value iteration
- Policy gradient methods
  - Reinforce
  - Natural policy gradient
  - Trust region policy optimization
- Actor-critic algorithms
  - Asynchronous advantage actor-critic (A3C)
  - Soft actor-critic(SAC)
- Model-based RL algorithms
  - Dyna
  - Guided policy search