RL Notes - Module 3 Learning Objectives

- Specifying Policies → Stochastic vs Deterministic policies

→ Policy is a distribution over actions for each possible state

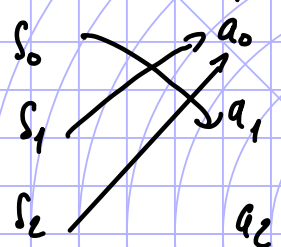
Deterministic policy notation

$\pi(s) = a$

States $\xrightarrow{\text{map}}$ Actions

→ for taking each action

→ only depend on current state

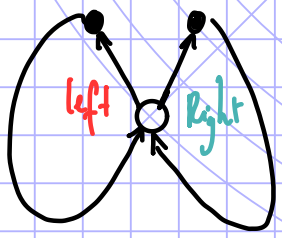


Stochastic policy notation

$\pi(a|s)$

$\sum_{a \in \mathcal{A}(s)} \pi(a|s) = 1 \quad , \quad \pi(a|s) \geq 0$

- Valid and invalid policies



→ Generate valid policies for a given MDP

Value functions

→ In RL, reward captures the notion of short-term gain.

→ the objective, is to learn a policy to achieve the most reward on the long run

⇒ Value funcs formalize what this means.

- Relationship between value funcs and policies
- Brief story of RL (Interesting)
- Bellman Equation Derivation

→ to formalize between value of state and its possible successors

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s], \quad G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_\pi(s')]$$

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

$$= \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a'|s') \cdot q_\pi(s', a')\right]$$

- State-value and action-value funcs under policy

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

↓ not immediate ↳ random

→ recall: $G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

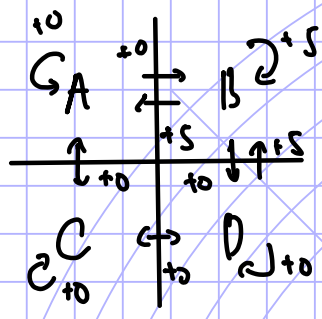
after taking action

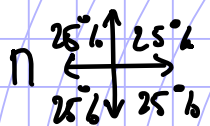
$p(s'|s,a) = V_\pi(s)$ → How to be computed?

$$\mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s'] \uparrow$$

- Valid value functions

E.g. Gridworld





$\gamma = 0,7$

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \cdot [r + \gamma V_\pi(s')]$$

for each a, there's only one possible associated next s' and r

$$= \sum_a \pi(a|s) \cdot [r + \gamma V_\pi(s')]$$

$$\begin{cases} V_\pi(A) = \frac{1}{4}(5 + 0,7\, V_\pi(B)) + \frac{1}{4} \cdot 0,7\, V_\pi(C) + \frac{1}{4} 0,7 \cdot V_\pi(A) \\ \dots \Rightarrow \\ \text{4 linear equations} \end{cases} \Rightarrow \begin{cases} V_\pi(A) = 4,2 \\ V_\pi(B) = 4,1 \\ V_\pi(C) = 2,2 \\ V_\pi(D) = 1,2 \end{cases}$$

⇒ Large MDP states ⇒ won't be practical

E.g. Chess 10^{45} equation?

- Optimal policies

→ Define an optimal policies

An optimal policy π_* is as good as or better than all the other policies

- Optimal value functions

Recall that $\pi_1 \geq \pi_2 \iff v_{\pi_1}(s) \geq v_{\pi_2}(s)$ for all $s \in \mathcal{S}$

v_* $v_{\pi_*}(s) \doteq \mathbb{E}_{\pi_*}[G_t \mid S_t = s] = \max_\pi v_\pi(s)$ for all $s \in \mathcal{S}$

q_* $q_{\pi_*}(s,a) \doteq \max_\pi q_\pi(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$

$\pi, p, \gamma \longrightarrow$ [Linear System Solver] $\longrightarrow v_\pi$

→ Bellman Optimal Equation

$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r \mid s, a) [r + \gamma v_*(s')]$$

$$q_*(s,a) = \sum_{s'} \sum_r p(s', r \mid s, a) [r + \gamma \max_{a'} q_*(s', a')]$$

Using Optimal Value functions to Get Optimal Policies

- The connection between the optimal value functions and optimal policies
  - Verify OVF

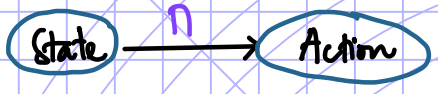
$$\pi_*(s) = \arg\max_a \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_*(s')]$$

$$\pi_*(s) = \arg\max_a q_*(s, a)$$

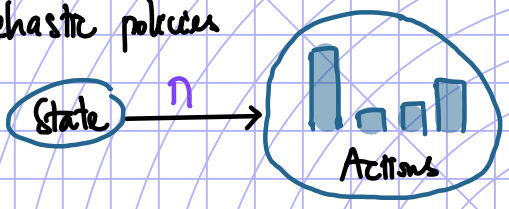
→ Summary

- **Policies** tell an agent how to behave in their environment

- Deterministic policies



- Stochastic policies



A policy depends only on the **current state**

- The goal: to find a policy that obtains as much reward as possible

An **optimal policy** achieves the highest value possible in every state

v_* $\quad v_{\pi_*}(s) = \max_\pi v_\pi(s)$ for all $s \in \mathcal{S}$

$$= \max_a \sum_{s'} \sum_r p(s', r \mid s, a)\,[r + \gamma v_*(s')]$$

q_* $\quad q_{\pi_*}(s, a) = \max_\pi q_\pi(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$

$$= \sum_{s'} \sum_r p(s', r \mid s, a)\,[r + \gamma \max_{a'} q_*(s', a')]$$

- **Value functions** estimate future return under a specific policy

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

- **Bellman equations** define a relationship between the value of a state, or state-action pair, and its possible successors

$$v_\pi(s) \doteq \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\,[r + \gamma v_\pi(s')]$$

$$q_\pi(s, a) \doteq \sum_{s'} \sum_r p(s', r \mid s, a)\,[r + \gamma \sum_{a'} \pi(a' \mid s')\, q_\pi(s', a')]$$

- $v_*(s) = \max_a q_{\pi_*}(s, a)$

Module 4

- Policy Evaluation vs. Control
  (1) (2)

(1) is the task of determining the value function for a specific policy. → necessary first-step

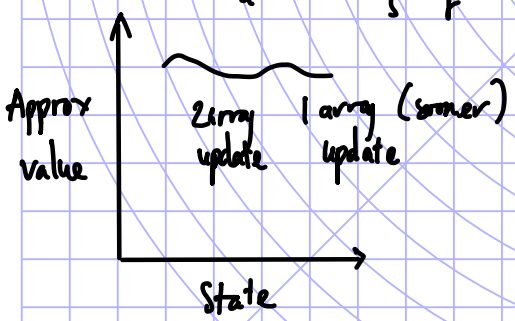
(2) is the task of finding a policy to obtain as much reward as possible.
↳ ultimate goal of RL

- Dynamic programming applied, limitations

- Iterative Policy Evaluation

→ DP are obtained by turning Bellman equations into update rules.

$$v_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_k(s')]$$



Sweep until $v_k = v_{k+1}$

for any v_0, $\lim_{k \to \infty} v_k = v_\pi$

(1) $\pi \rightarrow v_\pi$

Linear System Solver → Dynamic Programming

(2) improve a policy → strictly better

$\pi, p, \gamma \rightarrow DP \rightarrow v_\pi$ Policy Evaluation

$p, \gamma \rightarrow DP \rightarrow \pi_*$ Control

↳ Along with knowledge of p

- Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
$V \leftarrow \vec{0}$, $V' \leftarrow \vec{0}$

Loop:
$\Delta \leftarrow 0$
loop for $s \in S$:
$V'(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
$\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$
$V \leftarrow V'$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

- Policy Improvement

$$\pi_*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} \sum_{r} p(s', r \mid s, a)\,[r + \gamma v_*(s')]$$

- Policy Improvement Theorem

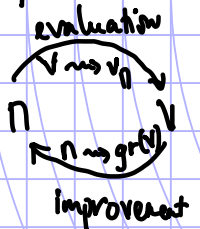
$\pi' \geq \pi$: $q_\pi(s, \pi'(s)) \geq q_\pi(s, \pi(s))$ for all $s \in \mathcal{S}$

$\pi' > \pi$: $q_\pi(s, \pi'(s)) > q_\pi(s, \pi(s))$ for at least one $s \in \mathcal{S}$

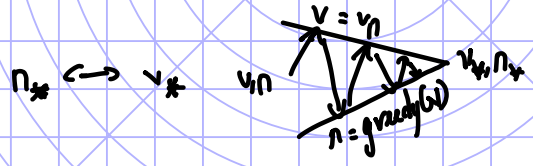
- Policy Iteration
  → the dance of policy and value

Evaluation $\pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_{\pi_*} \xrightarrow{I} \pi_*$

Improvement



Deterministic



- Generalized Policy Iteration unified classifications
  → Flexibility of the Policy Iteration framework

Synchronous vs Asynchronous

→ Value Iteration

Eff: $V_s \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma V(s')]$

and $V(\text{terminal}) = 0$

→ Avoid full sweeps
→ Update the states near those that have recently changed value

- Efficiency of Dynamic Programming

→ learning a value function:
Monte-Carlo Sampling → Average
Recall
$v_\pi(s) \doteq E_\pi[G_t | S_t = s]$

→ Finding an optimal policy: Brute Force Search
$|A|^{|S|}$ policies

$$v_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a).[r + \gamma v_k(s')]$$

- Advantage of DP and bootstrapping over these other strategies
  - much faster

→ RL to solve high-dim problem

- ADP for fleet optimization:

$$\bar{V}^n(a) = (1-\alpha)\bar{V}^{n-1}(a) + \alpha \hat{v}^n(a)$$

attribute of car
$\bar{V}^n(a)$: new est
$\bar{V}^{n-1}(a)$: old est
$\hat{v}^n(a)$: marginal value

merge old with new → $\bar{V}^2(TX) = (1-\alpha)\bar{V}^1(TX) + \alpha \hat{V}^2(TX)$
Let's take $\alpha = 0.1$

$$|States| = \binom{\text{Nb. trucks} + |\text{Locations}| - 1}{|\text{Locations}| - 1}$$

$a_1 = \begin{bmatrix} \text{Time} \\ \text{Location} \\ \text{Domicile} \\ \text{Hours} \end{bmatrix}$

Linear Programming
→ Get marginal value $\bar{v}(a)$

ADP for fleet optimization

Step 1: Start with a pre-decision state S_t^n

Step 2: Solve the deterministic optimization using an approximate value function: (Deterministic optimization)
$\hat{v}_t^n = \min_x \left(C_t(S_t^n, x_t) + \bar{V}_t^{n-1}(S^{M,x}(S_t^n, x_t))\right)$
to obtain x^n.

Step 3: Update the value function approximation (Recursive statistics)
$\bar{V}_{t-1}^n(S_{t-1}^{x,n}) = (1-\alpha_{n-1})\bar{V}_{t-1}^{n-1}(S_{t-1}^{x,n}) + \alpha_{n-1}\hat{v}_t^n$

Step 4: Obtain Monte Carlo sample of W_t and compute the next pre-decision state: (Simulation)
$S_{t+1}^n = S^M(S_t^n, x_t^n, W_{t+1}(\omega^n))$

Step 5: Return to step 1.

→ Multiattribute resources

$$V_t(S_t) = \max\left(C_t(S_t, x_t) + \sum_{a \in A} \bar{v}_{ta} \cdot R_{ta}^x\right)$$

→ Hierarchical learning

aggregate vs. disaggregate vs. weighted

Warren Powell

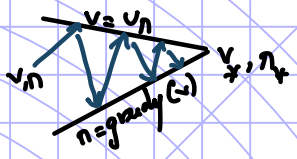
$\begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix} \longrightarrow w_a^0 \bar{v}^0(G_{11}) + w_a^1 \bar{v}^1 \begin{pmatrix} a_{11} \\ a_{12} \end{pmatrix} + \dots$

An Approx DP Algo for

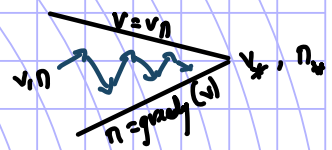
Large-Scale Fleet Management : A case application

DP → Depends on Algo

Policy Iteration



Generalized Policy Iteration



→ Summary

Policy evaluation is the task of determining the state-value func v_π for a particular policy π

Iterative Policy evaluation

$$v_\pi(s) \doteq \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_\pi(s')]$$

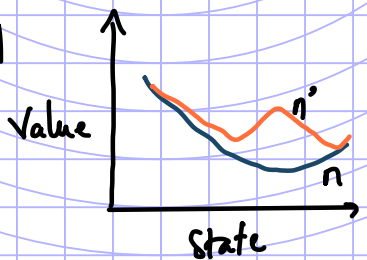
$$v_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_k(s')]$$

Control refers to the task of improving a policy

Policy improvement theorem

$$\pi'(s) \doteq \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_\pi(s')]$$

$\pi' > \pi$ unless π is optimal



→ Asynchronous methods vs. synchronous

1

q

γ

w. $\frac{5}{4}(x-1) + \frac{1}{4}(-21) = x$