

Model-free Policy Learning

Policy learning without knowing how the world works

Cathy Wu

6.7920 Reinforcement Learning: Foundations and Methods

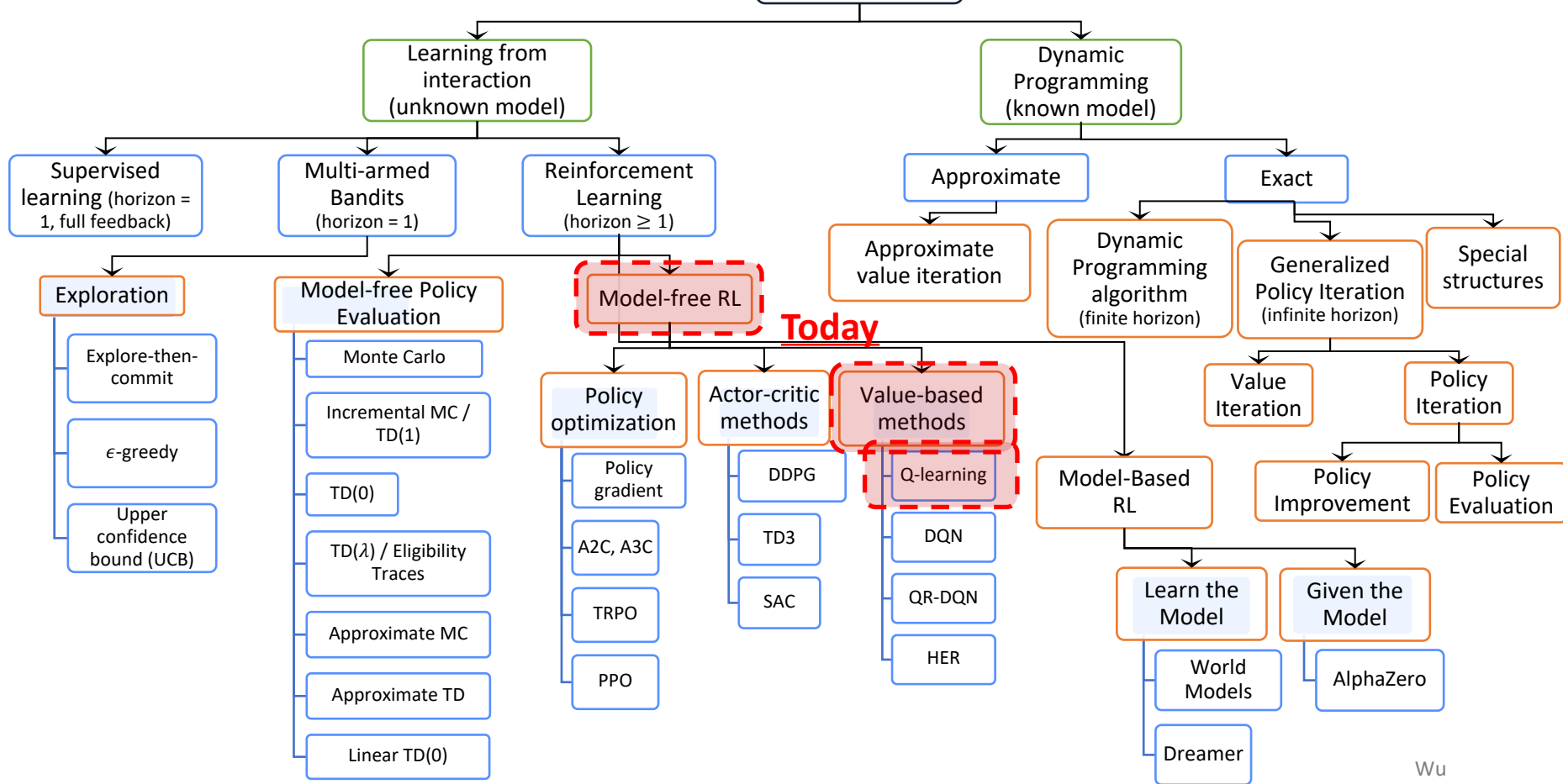
Readings

1. Neuro-dynamic Programming (NDP). §5.6, §4.1-4.3. Skim Ch 3-5 as needed.

Taxonomy of RL Methods

Methods for
sequential
decision making

Reinforcement learning (RL) is a framework for
sequential decision making under uncertainty.



Outline

1. State-action value function
2. Q-iteration
3. Q-learning
4. Exploration vs exploitation
5. On-policy vs off-policy learning

Policy Learning

Learn optimal policy π^*

For $i = 1, \dots, n$ [each of n episodes]

1. Set $t = 0$

2. Set initial state s_0

3. **While** ($s_{t,i}$ not terminal) [execute one trajectory]

1. Take action $a_{t,i}$ [Compare Policy Evaluation: Take action $a_{t,i} = \pi(s_{t,i})$]

2. Observe next state $s_{t+1,i}$ and reward $r_{t+1,i} = r(s_{t,i}, a_{t,i})$

3. Set $t = t + 1$

EndWhile

Endfor

Return: Estimate of the value function $\hat{\pi}^*$

Recall: Policy Evaluation

Fixed policy π

For $i = 1, \dots, n$ [each of n episodes]

1. Set $t = 0$
2. Set initial state s_0
3. **While** ($s_{t,i}$ not terminal) [execute one trajectory]
 1. Take action $a_{t,i} = \pi(s_{t,i})$
 2. Observe next state $s_{t+1,i}$ and reward $r_{t+1,i} = r(s_{t,i}, a_{t,i})$
 3. Set $t = t + 1$

EndWhile

Endfor

Return: Estimate of the value function $\hat{V}^\pi(\cdot)$

Recall: Value iteration algorithm

1. Let $V_0(s)$ be any function $V_0: S \rightarrow \mathbb{R}$.
2. At each iteration $k = 1, 2, \dots, K$
 - Compute $V_{k+1} = \mathcal{T}V_k$
 - Equivalently: $V_{k+1}(s) = \max_{a \in A} r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot | s, a)} V_k(s') \quad \forall s$
3. Terminate when V_k stops improving
 - e.g. when $\max_s |V_{k+1}(s) - V_k(s)|$ is small.
4. Return the greedy policy
$$\pi_K(s) \in \arg \max_{a \in A} r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot | s, a)} V_K(s')$$

State-Action Value Function (“Q”)

Definition

In discounted infinite horizon problems, for any policy π , the state-action value function (or Q-function) $Q^\pi : S \times A \mapsto \mathbb{R}$ is

$$Q^\pi(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a, a_t = \pi(s_t), \forall t \geq 1 \right]$$

The optimal Q-function is

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a)$$

and the optimal policy can be obtained as

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

- Recall: definition of value function, $V^\pi(s) = \mathbb{E} \left[\sum_{t=0}^T \gamma^t r(s_t, \pi(s_t)) \mid s_0 = s; \pi \right]$

State-Action Value Function Operators*

- $\mathcal{T}^\pi Q(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} [Q(s', \pi(s'))]$
 - Compare: $\mathcal{T}^\pi V(s) := r(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, \pi(s))} [V(s')]$
- $\mathcal{T}Q(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[\max_{a'} Q(s', a') \right]$
 - Compare: $\mathcal{T}V(s) := \max_{a \in A} r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} V(s')$

Still true:

- $Q^* = \mathcal{T}Q^*$
- $Q^\pi = \mathcal{T}^\pi Q^\pi$

Note: Abuse of notation for the operators

State-Action and State Value Function

- $Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)}[V^\pi(s')]$
- $V^\pi(s) = Q^\pi(s, \pi(s))$

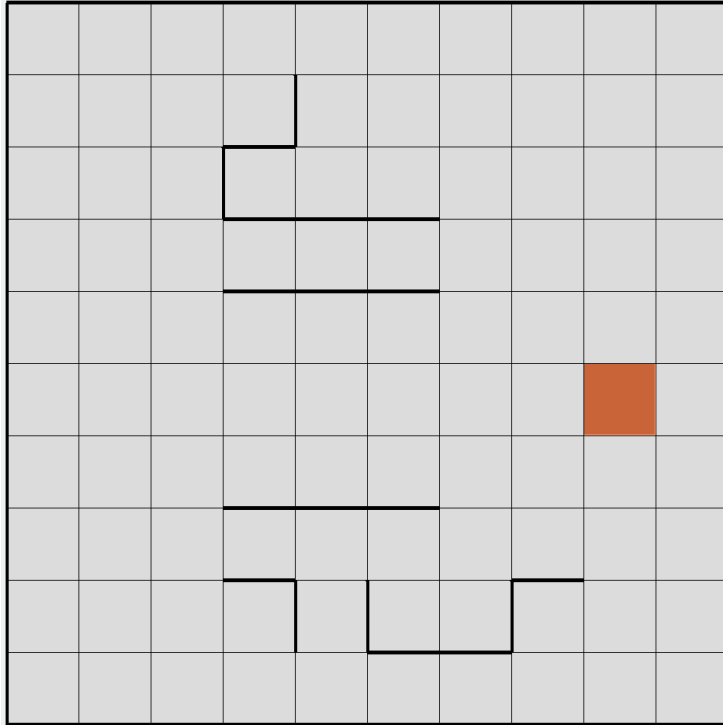
- $Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)}[V^*(s')]$
- $V^*(s) = Q^*(s, \pi^*(s)) = \max_{a \in A} Q^*(s, a)$

Q-value Iteration

1. Let $Q_0(s, a)$ be any Q-function $Q_0: S \times A \rightarrow \mathbb{R}$
2. At each iteration $k = 1, 2, \dots, K$
 - Compute $Q_{k+1} = \mathcal{T}Q_k$
 - Equivalently: $Q_{k+1}(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} [\max_{a' \in A} Q_k(s', a')] \quad \forall s, a$
3. Terminate when Q_k stops improving
 - e.g. when $\max_{s, a} |Q_{k+1}(s, a) - Q_k(s, a)|$ is small.
4. Return the greedy policy

$$\pi_K(s) \in \arg \max_{a \in A} Q_K(s, a)$$

The Grid-World Problem

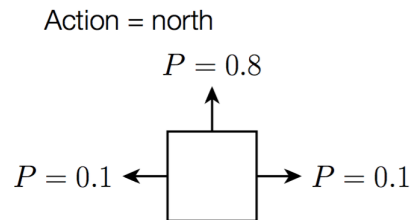


State: agent location

Example: Winter parking (with ice and potholes)

- Simple grid world with a *goal state* (green, desired parking spot) with reward (+1), a *“bad state”* (red, pothole) with reward (-100), and all other states neutral (+0).
- *Omnidirectional vehicle (agent)* can head in any direction. Actions move in the desired direction with probability 0.8, in one of the perpendicular directions with.
- Taking an action that would bump into a wall leaves agent where it is.

0	0	0	1
0		0	-100
0	0	0	0



[Source: adapted from Kolter, 2016]

Example: value iteration

Running value iteration with $\gamma = 0.9$

0	0	0	1
0		0	-100
0	0	0	0

Original reward function

(a)

Running value iteration with $\gamma = 0.9$

0	0	0.72	1.81
0		0	-99.91
0	0	0	0

$\hat{V}$ at one iteration

(b)

Running value iteration with $\gamma = 0.9$

0.809	1.598	2.475	3.745
0.268		0.302	-99.59
0	0.034	0.122	0.004

$\hat{V}$ at five iterations

(c)

Running value iteration with $\gamma = 0.9$

2.686	3.527	4.402	5.812
2.021		1.095	-98.82
1.390	0.903	0.738	0.123

$\hat{V}$ at 10 iterations

(d)

Running value iteration with $\gamma = 0.9$

5.470	6.313	7.190	8.669
4.802		3.347	-96.67
4.161	3.654	3.222	1.526

$\hat{V}$ at 1000 iterations

(e)

Running value iteration with $\gamma = 0.9$

→	→	→	↑
↑		←	←
↑	←	←	↓

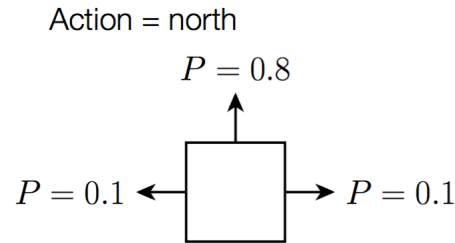
Resulting policy after 1000 iterations

(f)

State-Action Value Function (“Q table”)

- Example: Winter parking (with ice and potholes)

0	0	0	1
0		0	-100
0	0	0	0



Running value iteration with $\gamma = 0.9$

$V(s)$

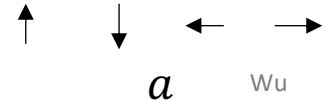
5.470	6.313	7.190	8.669
4.802		3.347	-96.67
4.161	3.654	3.222	1.526

$\hat{V}$ at 1000 iterations

$Q(s, a)$

S

2.5	1.4	3.2	5.4
1.0	3.2	5.1	6.3
5.2	4.2	5.5	7.2
8.7	3.4	2.0	8.0
4.8	2.5	3.5	4.2
1.0	3.0	3.3	1.2
-180	-172	-99.7	-150
4.2	2.1	3.2	3.7
2.1	2.0	3.7	3.1
3.0	1.2	3.2	2.7
0.1	1.5	0.1	1.0



It is convenient to keep track of not only the long term value of a state, but also the state, jointly with the next action.

Convenient for selecting next action!

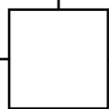
- Winter parking (with ice and potholes)

0	0	0	1
0		0	-100
0	0	0	0

Action = north

$P = 0.8$

$P = 0.1$



$P = 0.1$

$Q(s, a)$

S

2.5	1.4	3.2	5.4
1.0	3.2	5.1	6.3
5.2	4.2	5.5	7.2
8.7	3.4	2.0	8.0
4.8	2.5	3.5	4.2
1.0	3.0	3.3	1.2
-180	-172	-99.7	-150
4.2	2.1	3.2	3.7
2.1	2.0	3.7	3.1
3.0	1.2	3.2	2.7
0.1	1.5	0.1	1.0



a

Wu

Before

Running value iteration with $\gamma = 0.9$

5.470	6.313	7.190	8.669
4.802		3.347	-96.67
4.161	3.654	3.222	1.526

$\hat{V}$ at 1000 iterations

Running value iteration with $\gamma = 0.9$

→	→	→	↑
↑		←	←
↑	←	←	↓

Resulting policy after 1000 iterations

$$\pi_K(s) = \arg \max_{a \in A} r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot | s, a)} V_K(s')$$

Policy Iteration (w/ Q-value function)

1. Let π_0 be **any** stationary policy
2. At each iteration $k = 1, 2, \dots, K$
 - **Policy evaluation**: given π_k , compute Q^{π_k}
 - **Policy improvement**: compute the greedy policy
$$\pi_{k+1}(s) \in \arg \max_{a \in A} Q_k^{\pi}(s, a)$$
3. Stop if $Q^{\pi_k} = Q^{\pi_{k-1}}$
4. Return the last policy π_K

Compare: Policy Iteration

1. Let π_0 be any stationary policy
2. At each iteration $k = 1, 2, \dots, K$
 - Policy evaluation: given π_k , compute V^{π_k}
 - Policy improvement: compute the greedy policy

$$\pi_{k+1}(s) \in \arg \max_{a \in A} \left[r(s, a) + \gamma \sum_{s'} p(s'|s, a) V^{\pi_k}(s') \right]$$

1. Stop if $V^{\pi_k} = V^{\pi_{k-1}}$
2. Return the last policy π_K

Q-learning update

- Key idea: **Update the value of one state at a time** (a little bit)

- Dynamic programming update

$$Q_k(s, a) = r_k(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot | s, a)} \left[\max_{a' \in \mathcal{A}} Q_{k+1}(s', a') \right] \quad \forall s, a$$

- Q-learning update

$$Q_{t+1}(s, a) = (1 - \eta_t) Q_t(s, a) + \eta_t \left(r + \gamma \max_{a' \in \mathcal{A}} Q_t(s', a') \right)$$

- with learning rates $\eta_t \ll 1$

- Recall: **Optimal** Bellman equation

$$Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot | s, a)} \left[\max_{a'} Q^*(s', a') \right]$$

Recall: Temporal-Difference $TD(0)$ Estimation

- Approach #3: Conduct **incremental updates** within trajectories, leveraging the **Bellman equation**
- Recall: The Bellman equation

$$V^\pi(s) = r(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim P(\cdot | s, \pi(s))} [V^\pi(s')]$$

- Incremental update: At each step t , observe s_t, r_{t+1}, s_{t+1} and update estimate $\hat{V}^\pi$ as

$$\hat{V}^\pi(s_t) = (1 - \eta) \hat{V}^\pi(s_t) + \eta \left(r_{t+1} + \gamma \hat{V}^\pi(s_{t+1}) \right)$$

Q Learning Algorithm

1. Let Q_0 be any Q-function, s be an initial state,

2. At each iteration $t = 0, 1, 2, \dots, T$

- Use Q_t to select an action a

learning rates $\eta_t \in [0, 1]$.

- Observe next state s' and reward r

- Update Q function: $Q_{t+1}(s, a) = (1 - \eta_t)Q_t(s, a) + \eta_t \left(r + \gamma \max_{a'} Q_t(s', a') \right)$

- $s \leftarrow s'$

$$= Q_t(s, a) + \eta_t \left(\underbrace{r + \gamma \max_{a'} Q_t(s', a')}_{\text{TD / bootstrap target}} - \underbrace{Q_t(s, a)}_{\text{Current guess of value}} \right)$$

3. Return the greedy policy

$$\pi_T(s) = \arg \max_{a \in A} Q_T(s, a)$$

Temporal difference (TD) error δ_t

Additional assumption needed for convergence:

- **Coverage:** All the state-action pairs are visited infinitely often
- **Learning rates** set appropriately (Robbins-Monro conditions)

Exploration vs exploitation

The agent's dilemma:

- The learner should **maximize return** by repeatedly selecting the best action $\Rightarrow$ **exploitation**

$$\arg \max_{a \in A} Q_t(s, a)$$

- The learner should **gain information** by repeatedly selecting all actions $\Rightarrow$ **exploration**

$$\pi(a|s) = \frac{1}{|\mathcal{A}|}$$

Compromise:

- **ϵ -greedy**: with probability ϵ , explore. Else, exploit

Learning the Optimal Policy

1. Let Q_0 be any Q-function, set $t \leftarrow 0$, observe initial state s_t
2. **While** (s_t not terminal)
 1. Take action $a_t \sim \pi_t(\cdot | s_t)$ according to a suitable exploration policy

$$\pi_t(a|s_t) = \begin{cases} \operatorname{argmax}_a Q_t(s_t, a) & \text{w.p. } 1 - \epsilon \\ \operatorname{Unif}(A) & \text{w.p. } \epsilon \end{cases} \quad (\epsilon\text{-greedy policy})$$

$$\pi_t(a|s_t) = \frac{\exp\left(\frac{Q_t(s,a)}{\tau_t}\right)}{\sum_{a'} \exp\left(\frac{Q_t(s,a')}{\tau_t}\right)} \quad \tau_t \rightarrow 0 \quad (\text{soft-max policy})$$

1. Observe reward r_{t+1} and next state s_{t+1}
 - And (for SARSA) take action a_{t+1}
2. Compute the temporal difference δ_t

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \quad (\text{Q-learning})$$

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \quad (\text{SARSA})$$
1. Update the Q-function

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \eta_t(s_t, a_t) \delta_t$$
2. Set $t = t + 1$; set $s_t \leftarrow s_{t+1}$
 - And $a_t \leftarrow a_{t+1}$ for SARSA

EndWhile

Terminology: on-policy vs off-policy learning

- Two uses of policies
 - Behavior policy: Policy used for **interacting** (collecting data)
 - Target policy: Policy used for learning (updating the representation)
- Q-learning
 - Interacting policy: **ϵ -greedy**
 - Learning policy: **greedy**
 - Different $\rightarrow$ off-policy
- SARSA
 - Interacting policy: **softmax**
 - Learning policy: **softmax**
 - Same $\rightarrow$ on-policy
- Off-policy = “learning from others”
- On-policy = “learning from oneself”

Q-Learning: Properties

Understanding this Proposition is the main subject of today + next time.

Proposition

If the learning rate satisfies the Robbins-Monro conditions in all states $s, a \in S \times A$

$$\sum_{i=0}^{\infty} \eta_t(s, a) = \infty \quad \sum_{i=0}^{\infty} \eta_t^2(s, a) < \infty$$

And all state-action pairs are tried **infinitely often**, then for all $s, a \in S \times A$

$$\hat{Q}(s, a) \xrightarrow{a.s.} Q^*(s, a)$$

- **Remark:** “infinitely often” requires a steady exploration policy.

Summary

- **State-action value function** (Q) vs state value function (V)
 - State-action value function permits model-free extraction of the policy
 - DP methods: Q-value iteration, Policy iteration with Q-value function
- Policy learning: **Q-learning** and SARSA (definition, guarantees)
- Concepts:
 - **Tabular** setting
 - **Exploration vs exploitation**, ϵ -greedy exploration strategy
 - **Off-policy vs on-policy learning**