Reinforcement Learning Tutorial

Dilip Arumugam

Stanford University

CS330: Deep Multi-Task & Meta Learning

Learning Goals

Walk away with a cursory understanding of the following concepts in RL:

- Markov Decision Processes
- Value Functions
- Planning
- Temporal-Difference Methods
- Q-Learning

Learning Goals

Walk away with a cursory understanding of the following concepts in RL:

- Markov Decision Processes
- Value Functions
- Planning
- Temporal-Difference Methods
- Q-Learning

Much more to cover than we have time for today

Learning Goals

Walk away with a cursory understanding of the following concepts in RL:

- Markov Decision Processes
- Value Functions
- Planning
- Temporal-Difference Methods
- Q-Learning

Much more to cover than we have time for today
Many other Stanford courses that study RL to varying degrees:

- CS229, CS234, CS236, CS238, CS239, CS332
- MS&E338, MS&E346
- EE277

Some details & disclaimers

- Please do ask questions as they come up

Some details & disclaimers

- Please do ask questions as they come up
  - In the interest of time, I may defer some questions to the end

Some details & disclaimers

- Please do ask questions as they come up
  - In the interest of time, I may defer some questions to the end
- Be aware that these slides use one particular notation
  - This should (for the most part) align with the notation used in lectures
  - You will find many equivalent, alternative, or more general notations in other places

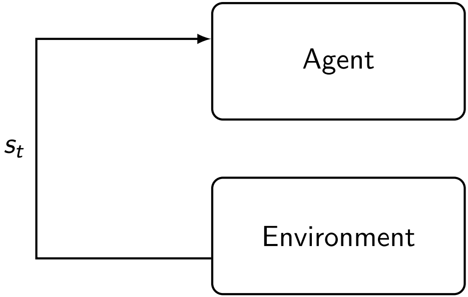
Some details & disclaimers

- Please do ask questions as they come up
  - In the interest of time, I may defer some questions to the end
- Be aware that these slides use one particular notation
  - This should (for the most part) align with the notation used in lectures
  - You will find many equivalent, alternative, or more general notations in other places
- Use office hours to resolve any lingering confusions after today
  - The rest of the course builds heavily upon these foundational concepts

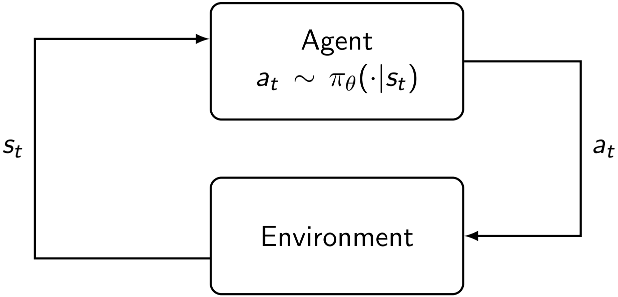
Agent-Environment Interface

Agent-Environment Interface

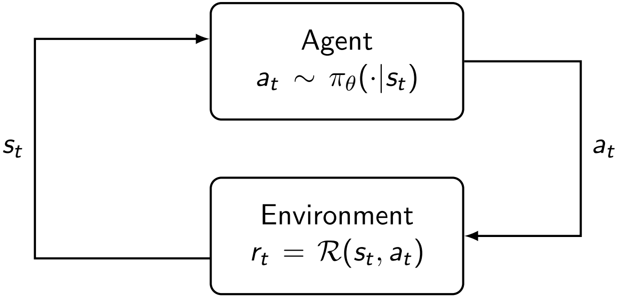
Agent-Environment Interface



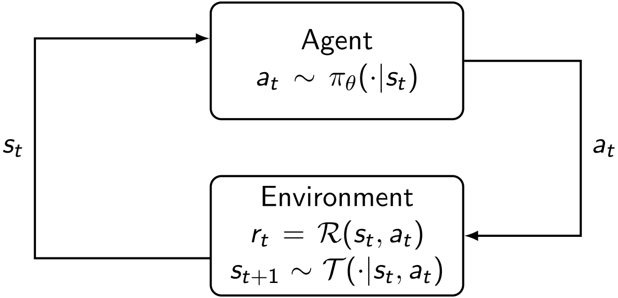
Agent-Environment Interface



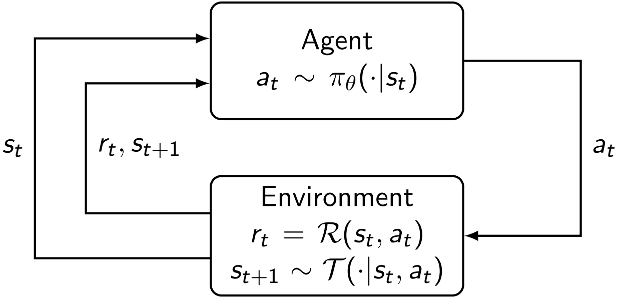
Agent-Environment Interface



Agent-Environment Interface



Agent-Environment Interface

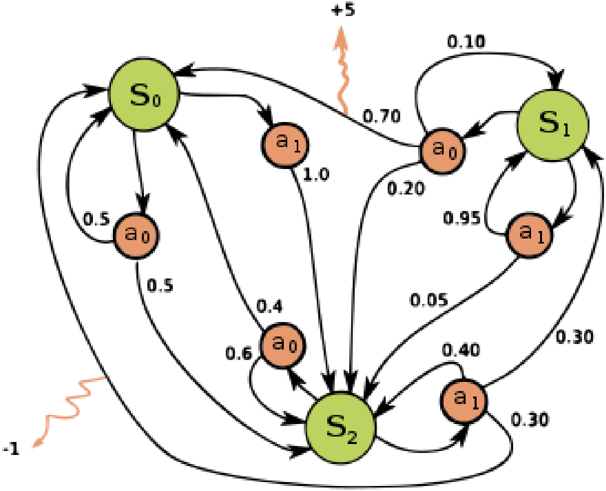


Running the Agent-Environment Interface

Let's watch a reinforcement-learning agent!

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]



Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

$\mathcal{S}$ Set of states

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

- $\mathcal{S}$ Set of states
- $\mathcal{A}$ Set of actions

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

- $\mathcal{S}$ Set of states
- $\mathcal{A}$ Set of actions
- $\mathcal{R}$ Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

- $\mathcal{S}$ Set of states
- $\mathcal{A}$ Set of actions
- $\mathcal{R}$ Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- $\mathcal{T}$ Transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

- $\mathcal{S}$ Set of states
- $\mathcal{A}$ Set of actions
- $\mathcal{R}$ Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- $\mathcal{T}$ Transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$
- γ Discount factor $\gamma \in [0, 1)$

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

- $\mathcal{S}$ Set of states
- $\mathcal{A}$ Set of actions
- $\mathcal{R}$ Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- $\mathcal{T}$ Transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$
- γ Discount factor $\gamma \in [0, 1)$

Behavior is encoded via a stationary, stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$

Markov Decision Processes (MDPs) [Bellman, 1957, Puterman, 1994]

Infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$

- $\mathcal{S}$ Set of states
- $\mathcal{A}$ Set of actions
- $\mathcal{R}$ Reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- $\mathcal{T}$ Transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$
- γ Discount factor $\gamma \in [0, 1)$

Behavior is encoded via a stationary, stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$

How do we assess the performance of a given policy π?

Value Functions

Value Functions

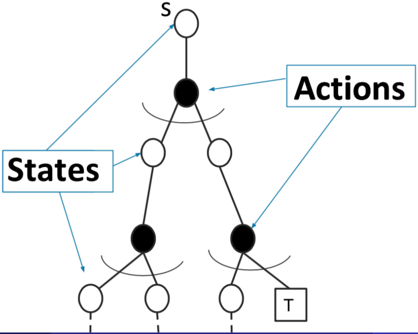
$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \,\middle|\, s_0 = s\right]$$

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right]$$

Value Functions

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s\right]$$

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s, a_0 = a\right]$$



Value Functions & Bellman Equations

- Bellman Equations (Policy evaluation)

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s\right] \qquad Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s, a_0 = a\right]$$

Value Functions & Bellman Equations

- Bellman Equations (Policy evaluation)

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s\right] \qquad Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s, a_0 = a\right]$$

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[Q^\pi(s,a)] \qquad Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s,a)}[V^\pi(s')]$$

Value Functions & Bellman Equations

- Bellman Equations (Policy evaluation)

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s\right] \qquad Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \;\middle|\; s_0 = s, a_0 = a\right]$$

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)] \qquad Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{T}(\cdot|s,a)}[V^\pi(s')]$$

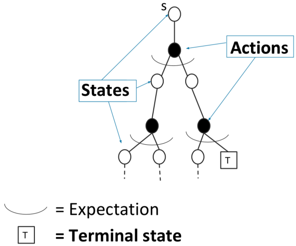


Image from CS234 - Lecture 3

Value Functions & Bellman Equations

- Bellman Equations (Policy evaluation)

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \,\middle|\, s_0 = s\right] \qquad Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right]$$

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)] \qquad Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{T}(\cdot|s,a)}[V^\pi(s')]$$

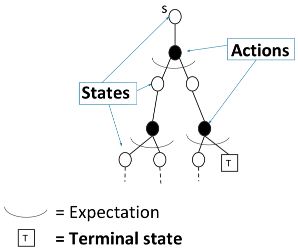


Image from CS234 - Lecture 3

- Bellman Optimality Equations - identify policy $\pi^\star$ achieving maximal value

Value Functions & Bellman Equations

- Bellman Equations (Policy evaluation)

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \,\middle|\, s_0 = s\right] \qquad Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right]$$

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)] \qquad Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{T}(\cdot|s,a)}[V^\pi(s')]$$

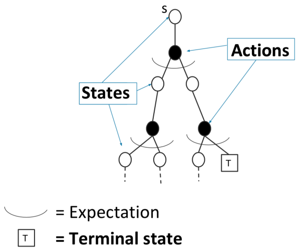


Image from CS234 - Lecture 3

- Bellman Optimality Equations - identify policy $\pi^\star$ achieving maximal value

$$V^\star(s) \triangleq V^{\pi^\star}(s) = \max_{a\in\mathcal{A}} Q^\star(s,a) \qquad Q^\star(s,a) \triangleq Q^{\pi^\star}(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{T}(\cdot|s,a)}[V^\star(s')].$$

Checkpoint #1 – Questions?

✓ Markov Decision Processes

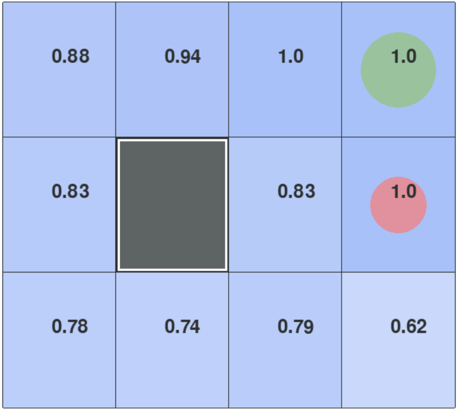
✓ Value Functions

- Planning
- Temporal-Difference Methods
- Q-Learning

Reinforcement Learning vs. Planning [Sutton and Barto, 1998, Kaelbling et al., 1996, Littman, 2015]

Planning Algorithms

- Take a full MDP as input
  - We know the transition function and the reward function!
- Alternative perspective: the agent has a perfect simulator of the environment in its brain
- Sit and think until an optimal policy has been computed



Value Iteration [Bellman, 1957]

Algorithm 1 Value Iteration (VI)

Input: Finite MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, Tolerance $\varepsilon > 0$

Initialize $V^{(0)}(s) = 0, \forall s \in \mathcal{S}$

Value Iteration [Bellman, 1957]

Algorithm 2 Value Iteration (VI)

Input: Finite MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, Tolerance $\varepsilon > 0$
Initialize $V^{(0)}(s) = 0, \forall s \in \mathcal{S}$
$\Delta = \infty$, $k = 0$
while $\Delta > \varepsilon$ **do**

Value Iteration [Bellman, 1957]

Algorithm 3 Value Iteration (VI)

Input: Finite MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, Tolerance $\varepsilon > 0$
Initialize $V^{(0)}(s) = 0, \forall s \in \mathcal{S}$
$\Delta = \infty, \ k = 0$
while $\Delta > \varepsilon$ **do**
  for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q^{(k+1)}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V^{(k)}(s')$

Value Iteration [Bellman, 1957]

Algorithm 4 Value Iteration (VI)

Input: Finite MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, Tolerance $\varepsilon > 0$
Initialize $V^{(0)}(s) = 0, \ \forall s \in \mathcal{S}$
$\Delta = \infty, \ k = 0$
while $\Delta > \varepsilon$ **do**
  for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q^{(k+1)}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V^{(k)}(s')$
    $V^{(k+1)}(s) = \max_{a \in \mathcal{A}} Q^{(k+1)}(s, a)$

Value Iteration [Bellman, 1957]

Algorithm 5 Value Iteration (VI)

Input: Finite MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, Tolerance $\varepsilon > 0$
Initialize $V^{(0)}(s) = 0, \forall s \in \mathcal{S}$
$\Delta = \infty, \; k = 0$
while $\Delta > \varepsilon$ **do**
  for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q^{(k+1)}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V^{(k)}(s')$
    $V^{(k+1)}(s) = \max_{a \in \mathcal{A}} Q^{(k+1)}(s, a)$
  end for
  $\Delta = \max_{s \in \mathcal{S}} |V^{(k+1)}(s) - V^{(k)}(s)|, \; k = k + 1$
end while

Value Iteration [Bellman, 1957]

Algorithm 6 Value Iteration (VI)

Input: Finite MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, Tolerance $\varepsilon > 0$
Initialize $V^{(0)}(s) = 0, \ \forall s \in \mathcal{S}$
$\Delta = \infty, \ k = 0$
while $\Delta > \varepsilon$ **do**
  for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q^{(k+1)}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V^{(k)}(s')$
    $V^{(k+1)}(s) = \max_{a \in \mathcal{A}} Q^{(k+1)}(s, a)$
  end for
  $\Delta = \max_{s \in \mathcal{S}} |V^{(k+1)}(s) - V^{(k)}(s)|, \ k = k + 1$
end while

Output: $\pi^\star(s) = \arg\max_{a \in \mathcal{A}} \overbrace{\left[\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V^\star(s') \right]}^{Q^\star(s,a)}$

Bellman Operators

- Let $\{\mathcal{S} \to \mathbb{R}\}$ denote the space of all real-valued functions on the MDP state space $\mathcal{S}$

Bellman Operators

- Let $\{\mathcal{S} \to \mathbb{R}\}$ denote the space of all real-valued functions on the MDP state space $\mathcal{S}$
- An operator maps from input functions to output functions

Bellman Operators

- Let $\{\mathcal{S} \to \mathbb{R}\}$ denote the space of all real-valued functions on the MDP state space $\mathcal{S}$
- An operator maps from input functions to output functions
- For an arbitrary value function $V : \mathcal{S} \to \mathbb{R}$, we define the **Bellman operator** $\mathcal{B} : \{\mathcal{S} \to \mathbb{R}\} \to \{\mathcal{S} \to \mathbb{R}\}$ as

Bellman Operators

- Let $\{\mathcal{S} \to \mathbb{R}\}$ denote the space of all real-valued functions on the MDP state space $\mathcal{S}$
- An operator maps from input functions to output functions
- For an arbitrary value function $V : \mathcal{S} \to \mathbb{R}$, we define the **Bellman operator** $\mathcal{B} : \{\mathcal{S} \to \mathbb{R}\} \to \{\mathcal{S} \to \mathbb{R}\}$ as

$$\mathcal{B}V(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V(s') \right].$$

Bellman Operators

- Let $\{\mathcal{S} \to \mathbb{R}\}$ denote the space of all real-valued functions on the MDP state space $\mathcal{S}$
- An operator maps from input functions to output functions
- For an arbitrary value function $V : \mathcal{S} \to \mathbb{R}$, we define the **Bellman operator** $\mathcal{B} : \{\mathcal{S} \to \mathbb{R}\} \to \{\mathcal{S} \to \mathbb{R}\}$ as

$$\mathcal{B}V(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V(s') \right].$$

- For a finite MDP ($|\mathcal{S}| < \infty$), $\{\mathcal{S} \to \mathbb{R}\} = \mathbb{R}^{|\mathcal{S}|} \implies \mathcal{B} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$

Bellman Operators

- Let $\{\mathcal{S} \to \mathbb{R}\}$ denote the space of all real-valued functions on the MDP state space $\mathcal{S}$
- An operator maps from input functions to output functions
- For an arbitrary value function $V : \mathcal{S} \to \mathbb{R}$, we define the **Bellman operator** $\mathcal{B} : \{\mathcal{S} \to \mathbb{R}\} \to \{\mathcal{S} \to \mathbb{R}\}$ as

$$\mathcal{B}V(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) V(s') \right].$$

- For a finite MDP ($|\mathcal{S}| < \infty$), $\{\mathcal{S} \to \mathbb{R}\} = \mathbb{R}^{|\mathcal{S}|} \implies \mathcal{B} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$
- From the VI algorithm:

$$V^{(k+1)} = \mathcal{B}V^{(k)}.$$

Convergence of Value Iteration

- Why does VI converge to the optimal value function of the input MDP?

Convergence of Value Iteration

- Why does VI converge to the optimal value function of the input MDP?
- An operator is a contraction mapping if applying it to separate inputs brings the resulting outputs “closer” together

Convergence of Value Iteration

- Why does VI converge to the optimal value function of the input MDP?
- An operator is a contraction mapping if applying it to separate inputs brings the resulting outputs "closer" together

Fact

The Bellman operator $\mathcal{B} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ *is a* γ*-contraction mapping with respect to* $|| \cdot ||_\infty$*. That is, for any two value functions* $V, V' \in \mathbb{R}^{|\mathcal{S}|}$,

$$||\mathcal{B}V - \mathcal{B}V'||_\infty \leq \gamma ||V - V'||_\infty = \gamma \max_{s \in \mathcal{S}} |V(s) - V'(s)|.$$

Convergence of Value Iteration

- Why does VI converge to the optimal value function of the input MDP?
- An operator is a contraction mapping if applying it to separate inputs brings the resulting outputs "closer" together

Fact

The Bellman operator $\mathcal{B} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is a γ-contraction mapping with respect to $||\cdot||_\infty$. That is, for any two value functions $V, V' \in \mathbb{R}^{|\mathcal{S}|}$,

$$||\mathcal{B}V - \mathcal{B}V'||_\infty \leq \gamma||V - V'||_\infty = \gamma \max_{s \in \mathcal{S}} |V(s) - V'(s)|.$$

- VI converges as a consequence of the Banach Fixed-Point Theorem
- Technically, we looked at an approximate version [Tseng, 1990, Littman et al., 1995]

Policy Evaluation

- An illustrative warm-up before we get to full RL

Policy Evaluation

- An illustrative warm-up before we get to full RL
- A step up from planning – no model of the environment

Policy Evaluation

- An illustrative warm-up before we get to full RL
- A step up from planning – no model of the environment
- Suppose we have a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$
- Question: how well does this policy perform?

Policy Evaluation

- An illustrative warm-up before we get to full RL
- A step up from planning – no model of the environment
- Suppose we have a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$
- Question: how well does this policy perform?
- Need to compute V^{π} using trajectories or rollouts sampled from executing π in the environment

Policy Evaluation

- An illustrative warm-up before we get to full RL
- A step up from planning – no model of the environment
- Suppose we have a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$
- Question: how well does this policy perform?
- Need to compute V^π using trajectories or rollouts sampled from executing π in the environment
- Different goal than trying to learn $\pi^\star$

Policy Evaluation

- An illustrative warm-up before we get to full RL
- A step up from planning – no model of the environment
- Suppose we have a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$
- Question: how well does this policy perform?
- Need to compute V^π using trajectories or rollouts sampled from executing π in the environment
- Different goal than trying to learn $\pi^\star$
- First attempt: recall that

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \,\middle|\, s_0 = s\right].$$

Monte-Carlo Policy Evaluation

- Let's help ourselves to an episodic MDP $\implies$ guaranteed termination

Algorithm 7 Monte-Carlo Policy Evaluation

Input: Learning rate $\alpha > 0$, Total episodes K

Initialize $G(s) = 0$ and $N(s) = 0$, $\forall s \in \mathcal{S}$

Monte-Carlo Policy Evaluation

- Let's help ourselves to an episodic MDP $\implies$ guaranteed termination

Algorithm 8 Monte-Carlo Policy Evaluation

Input: Learning rate $\alpha > 0$, Total episodes K
Initialize $G(s) = 0$ and $N(s) = 0$, $\forall s \in \mathcal{S}$
for $k \in [K]$ **do**
  Sample trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$

Monte-Carlo Policy Evaluation

- Let's help ourselves to an episodic MDP $\implies$ guaranteed termination

---

Algorithm 9 Monte-Carlo Policy Evaluation

Input: Learning rate $\alpha > 0$, Total episodes K
Initialize $G(s) = 0$ and $N(s) = 0$, $\forall s \in \mathcal{S}$
for $k \in [K]$ **do**
  Sample trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$
  for $t = 1, 2, 3, \ldots, T$ **do**
    if $t == \texttt{first\_occurrence}(s_t)$ **then**
      $N(s_t) = N(s_t) + 1$

Monte-Carlo Policy Evaluation

- Let's help ourselves to an episodic MDP $\implies$ guaranteed termination

Algorithm 10 Monte-Carlo Policy Evaluation

Input: Learning rate $\alpha > 0$, Total episodes K
Initialize $G(s) = 0$ and $N(s) = 0$, $\forall s \in \mathcal{S}$
for $k \in [K]$ **do**
  Sample trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$
  for $t = 1, 2, 3, \ldots, T$ **do**
    if $t == \texttt{first\_occurrence}(s_t)$ **then**
      $N(s_t) = N(s_t) + 1$
      $G(s_t) = G(s_t) + \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$

Monte-Carlo Policy Evaluation

- Let's help ourselves to an episodic MDP $\implies$ guaranteed termination

Algorithm 11 Monte-Carlo Policy Evaluation

Input: Learning rate $\alpha > 0$, Total episodes K
Initialize $G(s) = 0$ and $N(s) = 0$, $\forall s \in \mathcal{S}$
for $k \in [K]$ **do**
  Sample trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$
  for $t = 1, 2, 3, \dots, T$ **do**
    if $t == \texttt{first\_occurrence}(s_t)$ **then**
      $N(s_t) = N(s_t) + 1$
      $G(s_t) = G(s_t) + \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
      $V^\pi(s_t) = \frac{G(s_t)}{N(s_t)}$

Monte-Carlo Policy Evaluation

- Let's help ourselves to an episodic MDP $\implies$ guaranteed termination

Algorithm 12 Monte-Carlo Policy Evaluation

Input: Learning rate $\alpha > 0$, Total episodes K
Initialize $G(s) = 0$ and $N(s) = 0$, $\forall s \in \mathcal{S}$
for $k \in [K]$ **do**
  Sample trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$
  for $t = 1, 2, 3, \ldots, T$ **do**
    if $t == \texttt{first\_occurrence}(s_t)$ **then**
      $N(s_t) = N(s_t) + 1$
      $G(s_t) = G(s_t) + \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
      $V^\pi(s_t) = \frac{G(s_t)}{N(s_t)}$
    end if
  end for
end for

Temporal-Difference Methods [Sutton, 1988]

- A central idea to reinforcement learning

Algorithm 13 TD(0)

Input: Learning rate $\alpha > 0$
Initialize $V^\pi(s) = 0, \forall s \in \mathcal{S}$

Temporal-Difference Methods [Sutton, 1988]

- A central idea to reinforcement learning

Algorithm 14 TD(0)

Input: Learning rate $\alpha > 0$
Initialize $V^{\pi}(s) = 0, \forall s \in \mathcal{S}$
for $t = 1, 2, 3, \ldots$ **do**
  Observe current state s_t
  Execution action $a_t \sim \pi(\cdot \mid s_t)$
  Observe reward r_t and next state s_{t+1}

Temporal-Difference Methods [Sutton, 1988]

- A central idea to reinforcement learning

Algorithm 15 TD(0)

Input: Learning rate $\alpha > 0$
Initialize $V^{\pi}(s) = 0, \forall s \in \mathcal{S}$
for $t = 1, 2, 3, \ldots$ **do**
  Observe current state s_t
  Execution action $a_t \sim \pi(\cdot \mid s_t)$
  Observe reward r_t and next state s_{t+1}
  Compute TD(0)-error $\delta_t = (r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t))$

Temporal-Difference Methods [Sutton, 1988]

- A central idea to reinforcement learning

Algorithm 16 TD(0)

Input: Learning rate $\alpha > 0$
Initialize $V^\pi(s) = 0, \forall s \in \mathcal{S}$
for $t = 1, 2, 3, \ldots$ **do**
  Observe current state s_t
  Execution action $a_t \sim \pi(\cdot \mid s_t)$
  Observe reward r_t and next state s_{t+1}
  Compute TD(0)-error $\delta_t = (r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$
  $V^\pi(s_t) = V^\pi(s_t) + \alpha\delta_t$
end for

Temporal-Difference Methods [Sutton, 1988]

- A central idea to reinforcement learning

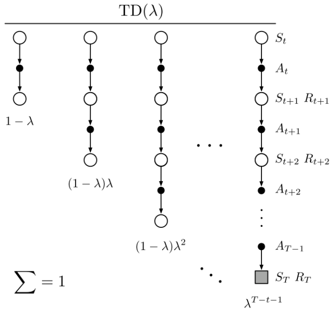
Algorithm 17 TD(0)

Input: Learning rate $\alpha > 0$
Initialize $V^\pi(s) = 0, \forall s \in \mathcal{S}$
for $t = 1, 2, 3, \ldots$ **do**
  Observe current state s_t
  Execution action $a_t \sim \pi(\cdot \mid s_t)$
  Observe reward r_t and next state s_{t+1}
  Compute TD(0)-error $\delta_t = (r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$
  $V^\pi(s_t) = V^\pi(s_t) + \alpha\delta_t$
end for

- Leverage bootstrapping to incrementally align value function estimates

TD(λ) [Sutton, 1984, 1988]

- So what's the difference?
- Consider what happens to Monte-Carlo policy evaluation when run on a highly-stochastic MDP
- A general bias-variance trade-off [Kearns and Singh, 2000]
  - Greater reliance on environment increases variance but incurs no bias
  - Greater reliance on bootstrapping increases bias with reduced variance
- Can we live in between Monte-Carlo and dynamic programming?



Q-Learning [Watkins and Dayan, 1992]

- Value-based reinforcement-learning algorithms exploit the fact that

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

Q-Learning [Watkins and Dayan, 1992]

- Value-based reinforcement-learning algorithms exploit the fact that

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

Algorithm 19 Tabular Q-learning with ε-greedy exploration

Input: Learning rate $\alpha > 0$, Initial Q-value q_{init}, Exploration probability $\varepsilon \geq 0$

Initialize $\widehat{Q}^\star(s, a) = q_{\text{init}}, \forall s, a \in \mathcal{S} \times \mathcal{A}$

Q-Learning [Watkins and Dayan, 1992]

- Value-based reinforcement-learning algorithms exploit the fact that

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

Algorithm 20 Tabular Q-learning with ε-greedy exploration

Input: Learning rate $\alpha > 0$, Initial Q-value q_{init}, Exploration probability $\varepsilon \geq 0$

Initialize $\widehat{Q}^\star(s, a) = q_{\text{init}}, \forall s, a \in \mathcal{S} \times \mathcal{A}$

for $t = 1, 2, 3, \ldots$ **do**

Observe current state s_t

$\pi(a \mid s) = (1 - \varepsilon)\mathbb{1}\left(a = \arg\max_{a^\star \in \mathcal{A}} \widehat{Q}^\star(s, a^\star)\right) + \frac{\varepsilon}{|\mathcal{A}|}$

Execution action $a_t \sim \pi(\cdot \mid s_t)$

Observe reward r_t and next state s_{t+1}

Q-Learning [Watkins and Dayan, 1992]

- Value-based reinforcement-learning algorithms exploit the fact that

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

Algorithm 21 Tabular Q-learning with ε-greedy exploration

Input: Learning rate $\alpha > 0$, Initial Q-value q_{init}, Exploration probability $\varepsilon \geq 0$

Initialize $\widehat{Q}^\star(s, a) = q_{\text{init}}, \forall s, a \in \mathcal{S} \times \mathcal{A}$

for $t = 1, 2, 3, \ldots$ **do**

Observe current state s_t

$\pi(a \mid s) = (1 - \varepsilon)\mathbb{1}\left(a = \arg\max_{a^\star \in \mathcal{A}} \widehat{Q}^\star(s, a^\star)\right) + \frac{\varepsilon}{|\mathcal{A}|}$

Execution action $a_t \sim \pi(\cdot \mid s_t)$

Observe reward r_t and next state s_{t+1}

$\widehat{Q}^\star(s_t, a_t) = \widehat{Q}^\star(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a' \in \mathcal{A}} \widehat{Q}^\star(s_{t+1}, a') - \widehat{Q}^\star(s_t, a_t)\right)$

end for

Tabular Q-Learning in Action!

Sutton & Barto's Cliff Walking Example – Project Malmo

Tabular Q-Learning in Action!

Sutton & Barto's Cliff Walking Example – Project Malmo

Some questions to ponder:

- How does the q_{init} parameter influence learning?
- Why didn't we start executing the optimal policy after collecting the coin for the first time?

Tabular Q-Learning in Action!

Sutton & Barto's Cliff Walking Example – Project Malmo

Some questions to ponder:

- How does the q_{init} parameter influence learning?
- Why didn't we start executing the optimal policy after collecting the coin for the first time?
- Why did the agent fail a small handful of times at the end, even after seeming to have found the optimal policy?

Tabular Q-Learning in Action!

Sutton & Barto's Cliff Walking Example – Project Malmo

Some questions to ponder:

- How does the q_{init} parameter influence learning?
- Why didn't we start executing the optimal policy after collecting the coin for the first time?
- Why did the agent fail a small handful of times at the end, even after seeming to have found the optimal policy?
- What would happen if the Minecraft agent was traversing slippery ice (where going straight might end up moving left) instead of stone?

Tabular Q-Learning in Action!

Sutton & Barto's Cliff Walking Example – Project Malmo

Some questions to ponder:

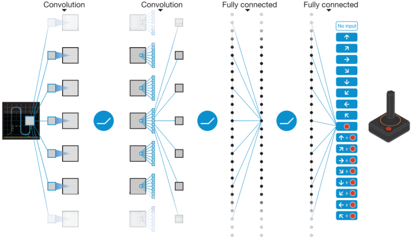
- How does the q_{init} parameter influence learning?
- Why didn't we start executing the optimal policy after collecting the coin for the first time?
- Why did the agent fail a small handful of times at the end, even after seeming to have found the optimal policy?
- What would happen if the Minecraft agent was traversing slippery ice (where going straight might end up moving left) instead of stone?
- Could the environment have given rewards in some other way that could have made learning faster?

Deep Q-Network (DQN) [Mnih et al., 2015]

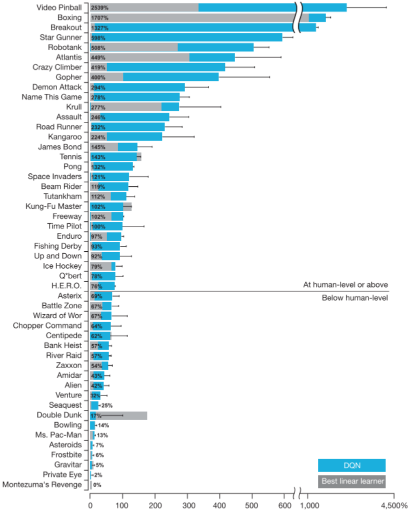
- Augment Q-learning with general function approximation
- $\widehat{Q}^\star_\theta : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$
  - One forward pass yields $Q^\star$-values for all actions
- Experience replay [Lin, 1992]
  - Maintain a FIFO buffer $\mathcal{D}$ of past (s, a, r, s') experiences for training
  - Sample mini-batches uniformly at random for updating θ
- Target networks
  - Maintain old parameters θ^- from C updates ago
  - Compute TD(0)-target as $r + \gamma \max_{a' \in \mathcal{A}} \widehat{Q}^\star_{\theta^-}(s_{t+1}, a')$
  - Bring us closer to supervised learning for stability
- Final loss function

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{a' \in \mathcal{A}} \widehat{Q}^\star_{\theta^-}(s', a') - \widehat{Q}^\star_\theta(s, a)\right)^2\right].$$

DQN Results



Convolution
Convolution
Fully connected
Fully connected
No input



Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*bert 78%
H.E.R.O. 76%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pac-Man 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%

Final Questions, Takeaways, & Parting Thoughts

- ✓ Markov Decision Processes
- ✓ Value Functions
- ✓ Planning
- ✓ Temporal-Difference Methods
- ✓ Q-Learning

Richard Bellman. A Markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.

Michael J Kearns and Satinder P Singh. Bias-variance error bounds for temporal difference updates. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 142–147, 2000.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Michael L Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521(7553):445–451, 2015.

Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh conference on Uncertainty in Artificial Intelligence*, pages 394–402, 1995.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K

Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Richard S Sutton and Andrew G Barto. Introduction to reinforcement learning. 1998.

Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

Paul Tseng. Solving H-horizon, stationary Markov decision problems in time proportional to log(H). *Operations Research Letters*, 9(5):287–297, 1990.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.