ECEN 743: Reinforcement Learning

TD Learning and Q-learning

Dileep Kalathil
Assistant Professor
Department of Electrical and Computer Engineering
Texas A&M University

References

- [SB, Chapter 5-6]
- "Neuro-Dynamic Programming", D. Bertsekas and J. Tsitsiklis, Chapter 5

TD Learning: Model-Free Policy Evaluation

MDP Questions

- How do we find the value of a policy π?
  - Policy evaluation iteration
- How do we find the optimal value function V^*?
  - Value iteration
- How do we find the optimal policy?
  - Value iteration, policy iteration

Reinforcement Learning Questions

- How do we find the value of a policy π?
- How do we find the optimal value function V^*?
- How do we find the optimal policy?

Reinforcement Learning Questions

- How do we find the value of a policy π?
- How do we find the optimal value function V^*?
- How do we find the optimal policy?

... without the knowledge of the model P

Reinforcement Learning Questions

- How do we find the value of a policy π?
- How do we find the optimal value function V^*?
- How do we find the optimal policy?

... without the knowledge of the model P

- Need to learn from the observed sequence of states, actions, and rewards

Policy Evaluation Problem

- How do we learn the value of a policy π?

Policy Evaluation Problem

- How do we learn the value of a policy π?
- Objective: Learn V_π using the the observed sequence of states, actions, and rewards, generated according to the policy π

Policy Evaluation Problem

- How do we learn the value of a policy π?
- Objective: Learn V_π using the the observed sequence of states, actions, and rewards, generated according to the policy π
- Assumption: The agent can interact with the environment (real-world environment or a simulator) to generate data

Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π

Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π with $s_0^k = s$

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \;\; a_t^k \sim \pi(s_t^k, \cdot), \;\; s_{t+1}^k \sim P(\cdot|s_t^k, a_t^k)$$

Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π with $s_0^k = s$

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \;\; a_t^k \sim \pi(s_t^k, \cdot), \;\; s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k as

$$G^k(s) = r_0^k(s_0, a_0) + \gamma r_1^k(s_1, a_1) + \gamma^2 r_2^k(s_2, a_2) + \ldots$$

Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π with $s_0^k = s$

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \; a_t^k \sim \pi(s_t^k, \cdot), \; s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k as

$$G^k(s) = r_0^k(s_0, a_0) + \gamma r_1^k(s_1, a_1) + \gamma^2 r_2^k(s_2, a_2) + \ldots$$

- Estimate the value

$$\hat{V}_\pi(s) = \frac{1}{K} \sum_{k=1}^{K} G^k(s)$$

Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π with $s_0^k = s$

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \ a_t^k \sim \pi(s_t^k, \cdot), \ s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k as

$$G^k(s) = r_0^k(s_0, a_0) + \gamma r_1^k(s_1, a_1) + \gamma^2 r_2^k(s_2, a_2) + \ldots$$

- Estimate the value

$$\hat{V}_\pi(s) = \frac{1}{K} \sum_{k=1}^{K} G^k(s)$$

- Monte Carlo Methods: Use empirical average instead of expectation

Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π with $s_0^k = s$

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \ a_t^k \sim \pi(s_t^k, \cdot), \ s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k as

$$G^k(s) = r_0^k(s_0, a_0) + \gamma r_1^k(s_1, a_1) + \gamma^2 r_2^k(s_2, a_2) + \ldots$$

- Estimate the value

$$\hat{V}_\pi(s) = \frac{1}{K} \sum_{k=1}^{K} G^k(s)$$

- Monte Carlo Methods: Use empirical average instead of expectation
- Problems with naive MC policy evaluation:

Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π with $s_0^k = s$

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \ a_t^k \sim \pi(s_t^k, \cdot), \ s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k as

$$G^k(s) = r_0^k(s_0, a_0) + \gamma r_1^k(s_1, a_1) + \gamma^2 r_2^k(s_2, a_2) + \ldots$$

- Estimate the value

$$\hat{V}_\pi(s) = \frac{1}{K} \sum_{k=1}^{K} G^k(s)$$

- Monte Carlo Methods: Use empirical average instead of expectation
- Problems with naive MC policy evaluation:
  - We need to generate K trajectories for each and every state!

Every-Visit Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π

Every-Visit Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \;\; a_t^k \sim \pi(s_t^k, \cdot), \;\; s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

Every-Visit Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \; a_t^k \sim \pi(s_t^k, \cdot), \; s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k at timestep t as as

$$G_t^k = r_t^k(s_t, a_t) + \gamma r_{t+1}^k(s_{t+1}, a_{t+1}) + \gamma^2 r_{t+2}^k(s_{t+2}, a_{t+2}) + \ldots$$

Every-Visit Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \;\; a_t^k \sim \pi(s_t^k, \cdot), \;\; s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k at timestep t as as

$$G_t^k = r_t^k(s_t, a_t) + \gamma r_{t+1}^k(s_{t+1}, a_{t+1}) + \gamma^2 r_{t+2}^k(s_{t+2}, a_{t+2}) + \ldots$$

- Estimate the value

$$\hat{V}_\pi(s) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k(s)} \sum_t G_t^k \mathbb{1}\{s_t = s\}$$

where $n_k(s)$ is the number of times state s is observed in trajectory τ^k

Every-Visit Monte Carlo (MC) Policy Evaluation

- Objective: Learn V_π
- Generate trajectories $\tau^k, 1 \leq k \leq K$, according to policy π

$$\tau^k = (s_0^k, a_0^k, r_0^k(s_0^k, a_0^k), s_1^k, a_1^k, r_1^k(s_1^k, a_1^k), \ldots), \;\; a_t^k \sim \pi(s_t^k, \cdot), \;\; s_{t+1}^k \sim P(\cdot | s_t^k, a_t^k)$$

- Compute the return of the trajectory τ^k at timestep t as as

$$G_t^k = r_t^k(s_t, a_t) + \gamma r_{t+1}^k(s_{t+1}, a_{t+1}) + \gamma^2 r_{t+2}^k(s_{t+2}, a_{t+2}) + \ldots$$

- Estimate the value

$$\hat{V}_\pi(s) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k(s)} \sum_t G_t^k \mathbb{1}\{s_t = s\}$$

  where $n_k(s)$ is the number of times state s is observed in trajectory τ^k
- We only need a (very long) *single trajectory* actually!

Incremental MC Updates

- Incremental averaging: suppose we get m samples of a random variable Y, $y_1, y_2, \ldots, y_m$. How do we estimate the mean of Y?

Incremental MC Updates

- Incremental averaging: suppose we get m samples of a random variable Y, $y_1, y_2, \ldots, y_m$. How do we estimate the mean of Y?

$$\mu_m = \frac{1}{m}\sum_{i=1}^{m} y_i$$

Incremental MC Updates

- Incremental averaging: suppose we get m samples of a random variable Y, $y_1, y_2, \ldots, y_m$. How do we estimate the mean of Y?

$$\mu_m = \frac{1}{m} \sum_{i=1}^{m} y_i$$

- Now we get one more sample, y_{m+1}. How do we update the estimate of the mean?

Incremental MC Updates

- Incremental averaging: suppose we get m samples of a random variable Y, $y_1, y_2, \ldots, y_m$. How do we estimate the mean of Y?

$$\mu_m = \frac{1}{m}\sum_{i=1}^{m} y_i$$

- Now we get one more sample, y_{m+1}. How do we update the estimate of the mean?

$$\mu_{m+1} = \frac{1}{m+1}\sum_{i=1}^{m+1} y_i = \frac{1}{m+1} y_{m+1} + \frac{1}{m+1}\sum_{i=1}^{m} y_i$$
$$= \frac{1}{m+1} y_{m+1} + \frac{m}{m+1}\mu_m = \mu_m + \frac{1}{m+1}(y_{m+1} - \mu_m)$$

Incremental MC Updates

- Incremental averaging: suppose we get m samples of a random variable Y, $y_1, y_2, \ldots, y_m$. How do we estimate the mean of Y?

$$\mu_m = \frac{1}{m} \sum_{i=1}^{m} y_i$$

- Now we get one more sample, y_{m+1}. How do we update the estimate of the mean?

$$\begin{aligned}\mu_{m+1} &= \frac{1}{m+1} \sum_{i=1}^{m+1} y_i = \frac{1}{m+1} y_{m+1} + \frac{1}{m+1} \sum_{i=1}^{m} y_i \\ &= \frac{1}{m+1} y_{m+1} + \frac{m}{m+1} \mu_m = \mu_m + \frac{1}{m+1}(y_{m+1} - \mu_m)\end{aligned}$$

- Incremental MC: $V_{t+1}(s_t) = V_t(s_t) + \frac{1}{n_t(s_t)}(G_t - V_t(s_t))$ and $V_{t+1}(s) = V_t(s)$ for all $s \neq s_t$

Incremental MC Updates

- Incremental averaging: suppose we get m samples of a random variable Y, $y_1, y_2, \ldots, y_m$. How do we estimate the mean of Y?

$$\mu_m = \frac{1}{m}\sum_{i=1}^{m} y_i$$

- Now we get one more sample, y_{m+1}. How do we update the estimate of the mean?

$$\mu_{m+1} = \frac{1}{m+1}\sum_{i=1}^{m+1} y_i = \frac{1}{m+1} y_{m+1} + \frac{1}{m+1}\sum_{i=1}^{m} y_i$$
$$= \frac{1}{m+1} y_{m+1} + \frac{m}{m+1}\mu_m = \mu_m + \frac{1}{m+1}(y_{m+1} - \mu_m)$$

- Incremental MC: $V_{t+1}(s_t) = V_t(s_t) + \frac{1}{n_t(s_t)}(G_t - V_t(s_t))$ and $V_{t+1}(s) = V_t(s)$ for all $s \neq s_t$
- In most practical problems, we use a constant step size

$$V_{t+1}(s_t) = V_t(s_t) + \alpha(G_t - V_t(s_t))$$

Monte-Carlo Policy Evaluation

- MC methods learn directly from episodes of experience
- MC is model-free

- MC learns from complete episodes: no bootstrapping
- MC does not really exploit the properties of the underlying MDP
- MC estimate has generally high variance

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data
  $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots),\ a_t \sim \pi(s_t, \cdot),\ s_{t+1} \sim P(\cdot | s_t, a_t)$

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data
  $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots), \; a_t \sim \pi(s_t, \cdot), \; s_{t+1} \sim P(\cdot | s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, $a_t \sim \pi(s_t, \cdot)$, $s_{t+1} \sim P(\cdot | s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots),\ a_t \sim \pi(s_t, \cdot),\ s_{t+1} \sim P(\cdot | s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return
  - Needs to wait until the episode termination

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots),\ a_t \sim \pi(s_t, \cdot),\ s_{t+1} \sim P(\cdot|s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return
  - Needs to wait until the episode termination
  - Does not make use of the properties of the MDP

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots), \; a_t \sim \pi(s_t, \cdot), \; s_{t+1} \sim P(\cdot | s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return
  - Needs to wait until the episode termination
  - Does not make use of the properties of the MDP
- Temporal-Difference (TD) Learning

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t(r_t + \gamma V_t(s_{t+1}) - V_t(s_t)), \text{ and, } V_{t+1}(s) = V_t(s) \text{ for all } s \neq s_t$$

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data
  $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots),\ a_t \sim \pi(s_t, \cdot),\ s_{t+1} \sim P(\cdot|s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return
  - Needs to wait until the episode termination
  - Does not make use of the properties of the MDP
- Temporal-Difference (TD) Learning

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t(r_t + \gamma V_t(s_{t+1}) - V_t(s_t)), \text{ and, } V_{t+1}(s) = V_t(s) \text{ for all } s \neq s_t$$

  - Intuition: If we have an estimate of the value, use that to estimate the return, and use that in turn to estimate the value

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, $a_t \sim \pi(s_t, \cdot)$, $s_{t+1} \sim P(\cdot|s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return
  - Needs to wait until the episode termination
  - Does not make use of the properties of the MDP
- Temporal-Difference (TD) Learning

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t(r_t + \gamma V_t(s_{t+1}) - V_t(s_t)), \text{ and, } V_{t+1}(s) = V_t(s) \text{ for all } s \neq s_t$$

  - Intuition: If we have an estimate of the value, use that to estimate the return, and use that in turn to estimate the value
  - Can immediately update value estimate after each new (s, a, r, s') sample

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, $a_t \sim \pi(s_t, \cdot)$, $s_{t+1} \sim P(\cdot | s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return
  - Needs to wait until the episode termination
  - Does not make use of the properties of the MDP
- Temporal-Difference (TD) Learning

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t(r_t + \gamma V_t(s_{t+1}) - V_t(s_t)), \text{ and, } V_{t+1}(s) = V_t(s) \text{ for all } s \neq s_t$$

  - Intuition: If we have an estimate of the value, use that to estimate the return, and use that in turn to estimate the value
  - Can immediately update value estimate after each new (s, a, r, s') sample
- TD Target: $r_t + \gamma V(s_{t+1})$

Temporal-Difference (TD) Learning

- TD learning is a combination of Monte Carlo ideas and dynamic programming ideas
- Objective: estimate V_π from the trajectory data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, $a_t \sim \pi(s_t, \cdot)$, $s_{t+1} \sim P(\cdot|s_t, a_t)$
- Recall every-visit MC: $V_{t+1}(s_t) = V_t(s_t) + \alpha_t(G_t - V_t(s_t))$
  - Updates value estimate towards actual return
  - Needs to wait until the episode termination
  - Does not make use of the properties of the MDP
- Temporal-Difference (TD) Learning

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t(r_t + \gamma V_t(s_{t+1}) - V_t(s_t)), \text{ and, } V_{t+1}(s) = V_t(s) \text{ for all } s \neq s_t$$

  - Intuition: If we have an estimate of the value, use that to estimate the return, and use that in turn to estimate the value
  - Can immediately update value estimate after each new (s, a, r, s') sample
- TD Target: $r_t + \gamma V(s_{t+1})$
- TD Error: $r_t + \gamma V(s_{t+1}) - V(s_t)$

Convergence of TD Learning

- TD learns a guess from a guess. Will it converge to V_π?

Convergence of TD Learning

- TD learns a guess from a guess. Will it converge to V_π?
- TD will (provably) converge to V_π under suitable assumptions

Convergence of TD Learning

- TD learns a guess from a guess. Will it converge to V_π?
- TD will (provably) converge to V_π under suitable assumptions
- Proof requires stochastic approximation theory. We will discuss this later in the course

Convergence of TD Learning

- One intuitive explanation (to be proved using stochastic approximation later)

Convergence of TD Learning

- One intuitive explanation (to be proved using stochastic approximation later)

Lemma

Let $H : \mathbb{R}^n \to \mathbb{R}^n$ be a contraction w.r.t. $\|\cdot\|$ with contraction factor γ. Then, for any $\alpha \in (0, 1)$, the function $F : \mathbb{R}^n \to \mathbb{R}^n$, defined as $F = (1 - \alpha I) + \alpha H$ is also a contraction.

Convergence of TD Learning

- One intuitive explanation (to be proved using stochastic approximation later)

Lemma

Let $H : \mathbb{R}^n \to \mathbb{R}^n$ be a contraction w.r.t. $\|\cdot\|$ with contraction factor γ. Then, for any $\alpha \in (0, 1)$, the function $F : \mathbb{R}^n \to \mathbb{R}^n$, defined as $F = (1 - \alpha I) + \alpha H$ is also a contraction.

- Consider the operator $F_\pi = (1 - \alpha)I + \alpha T_\pi$, which is a contraction by the above result

Convergence of TD Learning

- One intuitive explanation (to be proved using stochastic approximation later)

Lemma

Let $H : \mathbb{R}^n \to \mathbb{R}^n$ be a contraction w.r.t. $\|\cdot\|$ with contraction factor γ. Then, for any $\alpha \in (0, 1)$, the function $F : \mathbb{R}^n \to \mathbb{R}^n$, defined as $F = (1 - \alpha I) + \alpha H$ is also a contraction.

- Consider the operator $F_\pi = (1 - \alpha)I + \alpha T_\pi$, which is a contraction by the above result
- Then, the iteration $V_{k+1} = (1 - \alpha)V_k + \alpha T_\pi V_k$ will converge to V_π

Convergence of TD Learning

- One intuitive explanation (to be proved using stochastic approximation later)

Lemma

Let $H : \mathbb{R}^n \to \mathbb{R}^n$ be a contraction w.r.t. $\|\cdot\|$ with contraction factor γ. Then, for any $\alpha \in (0,1)$, the function $F : \mathbb{R}^n \to \mathbb{R}^n$, defined as $F = (1 - \alpha I) + \alpha H$ is also a contraction.

- Consider the operator $F_\pi = (1-\alpha)I + \alpha T_\pi$, which is a contraction by the above result
- Then, the iteration $V_{k+1} = (1-\alpha)V_k + \alpha T_\pi V_k$ will converge to V_π
- Rewrite the iteration as

$$V_{k+1}(s) = V_k(s) + \alpha(T_\pi V_k(s) - V_k(s)) = V_k(s) + \alpha(\mathbb{E}_{a\sim\pi(s,\cdot),s'\sim P(\cdot|s,a)}[r(s,a) + \gamma V_k(s')] - V_k(s))$$

Convergence of TD Learning

- One intuitive explanation (to be proved using stochastic approximation later)

Lemma

Let $H : \mathbb{R}^n \to \mathbb{R}^n$ be a contraction w.r.t. $\|\cdot\|$ with contraction factor γ. Then, for any $\alpha \in (0, 1)$, the function $F : \mathbb{R}^n \to \mathbb{R}^n$, defined as $F = (1 - \alpha I) + \alpha H$ is also a contraction.

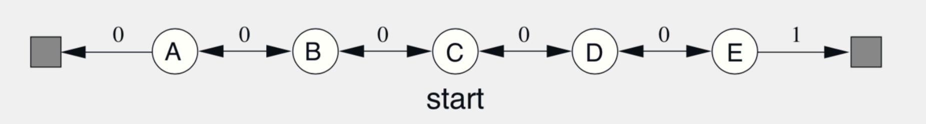
- Consider the operator $F_\pi = (1 - \alpha)I + \alpha T_\pi$, which is a contraction by the above result
- Then, the iteration $V_{k+1} = (1 - \alpha)V_k + \alpha T_\pi V_k$ will converge to V_π
- Rewrite the iteration as

$$V_{k+1}(s) = V_k(s) + \alpha(T_\pi V_k(s) - V_k(s)) = V_k(s) + \alpha(\mathbb{E}_{a\sim\pi(s,\cdot),s'\sim P(\cdot|s,a)}[r(s,a) + \gamma V_k(s')] - V_k(s))$$

- We will replace the exact expectation by one sample approximation

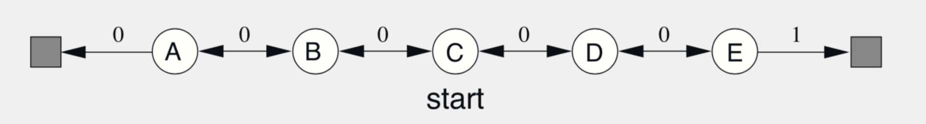
$$V_{t+1}(s_t) = V_t(s_t) + \alpha([r(s_t, a_t) + \gamma V_t(s_{t+1})] - V_t(s_t))$$

MC vs TD: Random Walk Example (from [SB])



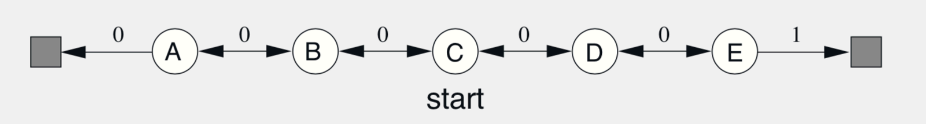
- Policy is *random*: move left or right with probability 0.5

MC vs TD: Random Walk Example (from [SB])



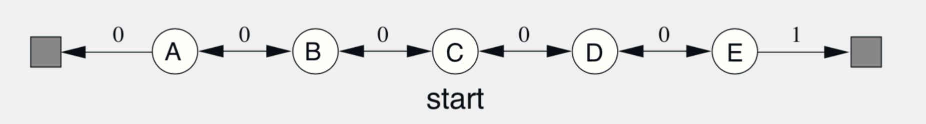
- Policy is *random*: move left or right with probability 0.5
- Each episode starts at state C and ends at L or R

MC vs TD: Random Walk Example (from [SB])



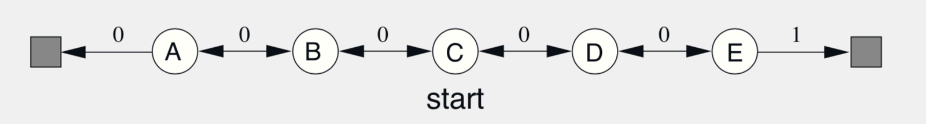
- Policy is *random*: move left or right with probability 0.5
- Each episode starts at state C and ends at L or R
- Assume $\gamma = 1$. What is V_π?

MC vs TD: Random Walk Example (from [SB])



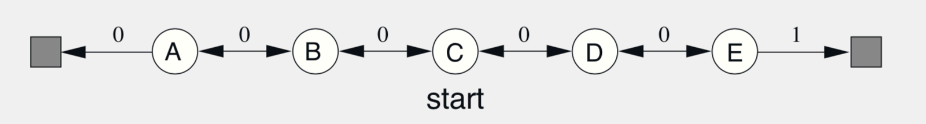
- Policy is *random*: move left or right with probability 0.5
- Each episode starts at state C and ends at L or R
- Assume $\gamma = 1$. What is V_π?

MC vs TD: Random Walk Example (from [SB])



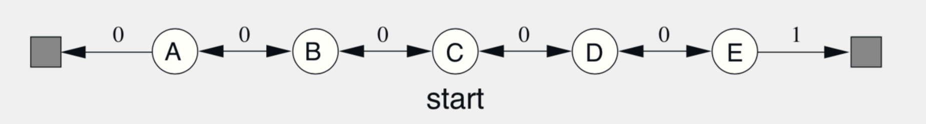
- Policy is *random*: move left or right with probability 0.5
- Each episode starts at state C and ends at L or R
- Assume $\gamma = 1$. What is V_π?
  - This is similar to a random walk problem

MC vs TD: Random Walk Example (from [SB])



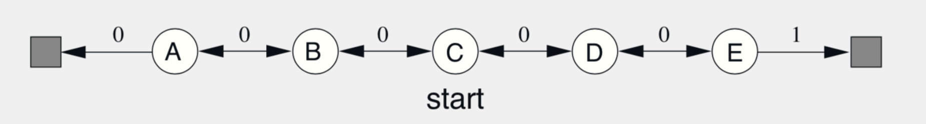
- Policy is *random*: move left or right with probability 0.5
- Each episode starts at state C and ends at L or R
- Assume $\gamma = 1$. What is V_π?
  - This is similar to a random walk problem
  - $V_\pi(s) = \mathbb{P}(\text{reaching R, starting from } s)$

MC vs TD: Random Walk Example (from [SB])



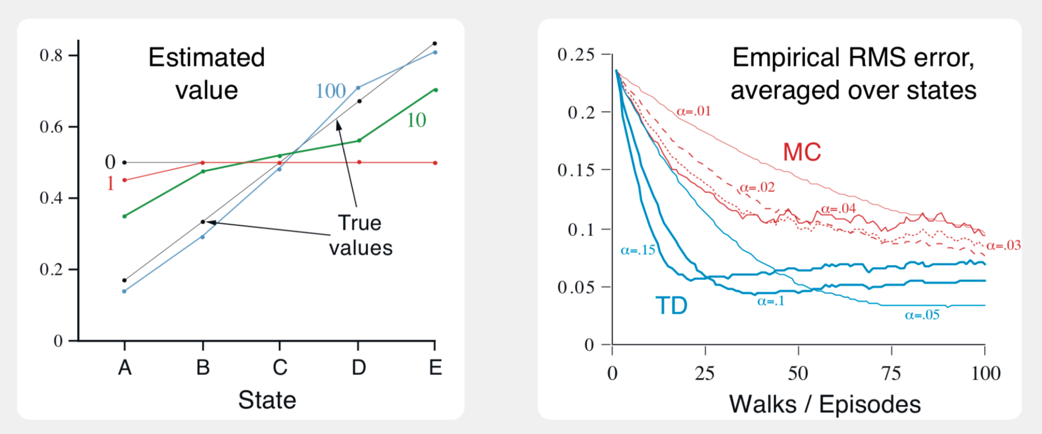
- Policy is *random*: move left or right with probability 0.5
- Each episode starts at state C and ends at L or R
- Assume $\gamma = 1$. What is V_π?
  - This is similar to a random walk problem
  - $V_\pi(s) = \mathbb{P}(\text{reaching R, starting from } s)$
  - This can be show to be $1 - \frac{k}{n}$, where k is the index of the state and n is the total number of states

MC vs TD: Random Walk Example (from [SB])

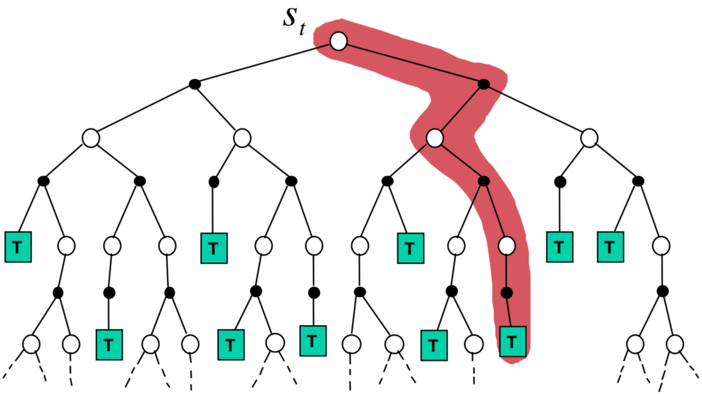


- Policy is *random*: move left or right with probability 0.5
- Each episode starts at state C and ends at L or R
- Assume $\gamma = 1$. What is V_π?
  - This is similar to a random walk problem
  - $V_\pi(s) = \mathbb{P}$(reaching R, starting from s)
  - This can be show to be $1 - \frac{k}{n}$, where k is the index of the state and n is the total number of states
  - $V(A) = 1/6, V(C) = 3/6, V(E) = 5/6$

MC vs TD: Random Walk Example (from [SB])

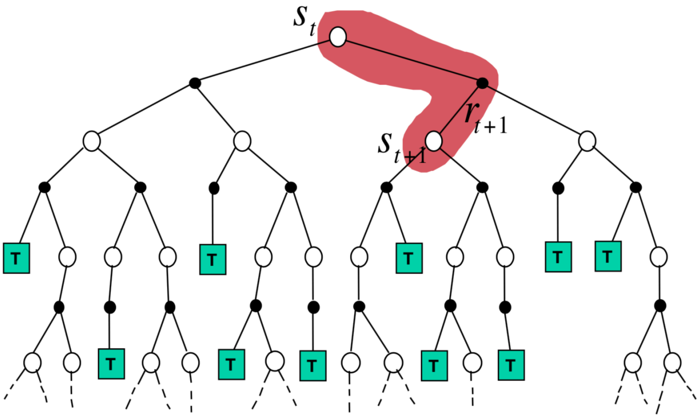


MC Backup



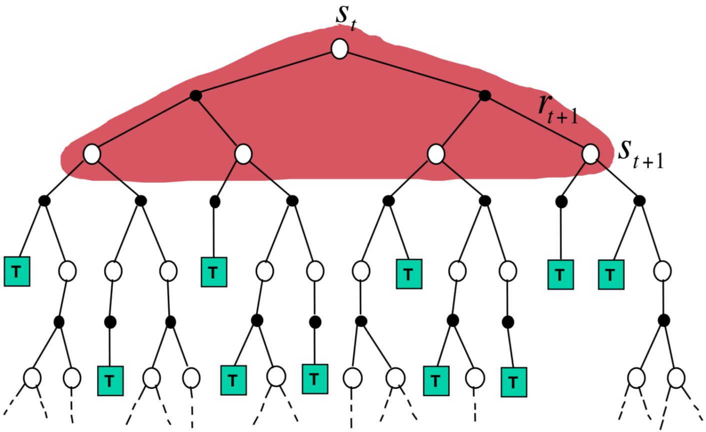
$$V_{t+1}(s_t) = V_t(s_t) + \frac{1}{n(s_t)}\left(G_t - V_t(s_t)\right)$$

TD Backup



$$V_{t+1}(s_t) = V_t(s_t) + \alpha(r_t + \gamma V_t(s_{t+1}) - V_t(s_t))$$

DP Backup



$$V_{t+1}(s_t) = r_t + \gamma \mathbb{E}[V_t(s_{t+1})]$$

Q-Learning: Model-free Learning of the Optimal Q-function

Reinforcement Learning Questions

- How do we learn the value of a policy π?
- How do we learn the optimal action-value function Q^*?
- How do we learn the optimal policy π^*?

... without the knowledge of the model P

- Need to learn from the observed sequence of states, actions, and rewards

Optimal Q-value Function

- Q-value function of a policy π

$$Q_\pi(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$$

Optimal Q-value Function

- Q-value function of a policy π

$$Q_\pi(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$$

- Optimal Q-value function

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

Optimal Q-value Function

- Q-value function of a policy π

$$Q_\pi(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$$

- Optimal Q-value function

$$Q^*(s,a) = \max_\pi \; Q_\pi(s,a)$$

- π^* from Q^*

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

Optimal Q-value Function

- Q-value function of a policy π

$$Q_\pi(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$$

- Optimal Q-value function

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

- π^* from Q^*

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

- Can we learn Q^* only using the trajectory data?

Intuitive Idea

- Recall the Bellman operator for Q-value function

$$TQ(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\max_b Q(s',b)]$$

Intuitive Idea

- Recall the Bellman operator for Q-value function

$$TQ(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\max_b Q(s',b)]$$

- T is a contraction. Then, $F = (1 - \alpha I) + \alpha T$ is also a contraction. So, the iteration

$$Q_{k+1}(s,a) = Q_{k+1}(s,a) + \alpha(TQ_k(s,a) - Q_k(s,a))$$

will converge to Q^*

Intuitive Idea

- Recall the Bellman operator for Q-value function

$$TQ(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\max_b Q(s',b)]$$

- T is a contraction. Then, $F = (1 - \alpha I) + \alpha T$ is also a contraction. So, the iteration

$$Q_{k+1}(s,a) = Q_{k+1}(s,a) + \alpha(TQ_k(s,a) - Q_k(s,a))$$

will converge to Q^*

- We will replace the exact expectation by one sample (s,a,r,s') approximation as

$$Q_{t+1}(s_t,a_t) = Q_{t+1}(s_t,a_t) + \alpha \left(r(s_t,a_t) + \gamma[\max_b Q_t(s_{t+1},b)] - Q_t(s_t,a_t) \right)$$

Intuitive Idea

- Recall the Bellman operator for Q-value function

$$TQ(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\max_b Q(s',b)]$$

- T is a contraction. Then, $F = (1 - \alpha I) + \alpha T$ is also a contraction. So, the iteration

$$Q_{k+1}(s,a) = Q_{k+1}(s,a) + \alpha(TQ_k(s,a) - Q_k(s,a))$$

  will converge to Q^*
- We will replace the exact expectation by one sample (s,a,r,s') approximation as

$$Q_{t+1}(s_t,a_t) = Q_{t+1}(s_t,a_t) + \alpha\left(r(s_t,a_t) + \gamma[\max_b Q_t(s_{t+1},b)] - Q_t(s_t,a_t)\right)$$

- Question: How do we select the next action a_{t+1}?

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a 'clever' trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a 'clever' trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a 'clever' trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions
- The optimal long-term strategy may involve sub-optimal short-term decisions

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions
- The optimal long-term strategy may involve sub-optimal short-term decisions
- How to optimally balance exploration and exploitation?

Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions
- The optimal long-term strategy may involve sub-optimal short-term decisions
- How to optimally balance exploration and exploitation?
  - This is one of the fundamental (research) problems of RL

Exploration vs Exploitation: Examples

Exploration vs Exploitation: Examples

- Restaurant Selection

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location
- Games

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location
- Games
  - Exploitation: Play the move you believe is best

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location
- Games
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs
  - Experience from each state-action pair is necessary for optimal learning

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs
  - Experience from each state-action pair is necessary for optimal learning
- Exploration via randomization

$$a_{t+1} = \epsilon\text{-greedy}(Q_t) = \begin{cases} \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } 1 - \epsilon\,(|\mathcal{A}| - 1) \\ a, a \neq \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } \epsilon \end{cases}$$

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs
  - Experience from each state-action pair is necessary for optimal learning
- Exploration via randomization

$$a_{t+1} = \epsilon\text{-greedy}(Q_t) = \begin{cases} \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } 1 - \epsilon\;(|\mathcal{A}| - 1) \\ a, a \neq \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } \epsilon \end{cases}$$

- Under some conditions on the transition probability matrix, all state-action pairs will be visited infinitely often (in the limit)

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs
  - Experience from each state-action pair is necessary for optimal learning
- Exploration via randomization

$$a_{t+1} = \epsilon\text{-greedy}(Q_t) = \begin{cases} \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } 1 - \epsilon\,(|\mathcal{A}| - 1) \\ a, a \neq \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } \epsilon \end{cases}$$

- Under some conditions on the transition probability matrix, all state-action pairs will be visited infinitely often (in the limit)
- Is ϵ-greedy the "best" exploration strategy?

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs
  - Experience from each state-action pair is necessary for optimal learning
- Exploration via randomization

$$a_{t+1} = \epsilon\text{-greedy}(Q_t) = \begin{cases} \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } 1 - \epsilon\ (|\mathcal{A}| - 1) \\ a, a \neq \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } \epsilon \end{cases}$$

- Under some conditions on the transition probability matrix, all state-action pairs will be visited infinitely often (in the limit)
- Is ϵ-greedy the "best" exploration strategy?
  - In general, ϵ-greedy is not the best exploration strategy.

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs
  - Experience from each state-action pair is necessary for optimal learning
- Exploration via randomization

$$a_{t+1} = \epsilon\text{-greedy}(Q_t) = \begin{cases} \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } 1 - \epsilon \; (|\mathcal{A}| - 1) \\ a, a \neq \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } \epsilon \end{cases}$$

- Under some conditions on the transition probability matrix, all state-action pairs will be visited infinitely often (in the limit)
- Is ϵ-greedy the "best" exploration strategy?
  - In general, ϵ-greedy is not the best exploration strategy.
  - However, we use this in deep RL most of the time.

Exploration by Randomization

- Given Q_t, how do we select the next action a_{t+1}?
- Greedy update: $a_{t+1} = \arg\max_a \; Q_t(s_{t+1}, a)$
  - Policies resulting from the greedy updates may not visit every state-action pairs
  - Experience from each state-action pair is necessary for optimal learning
- Exploration via randomization

$$a_{t+1} = \epsilon\text{-greedy}(Q_t) = \begin{cases} \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } 1 - \epsilon \; (|\mathcal{A}| - 1) \\ a, a \neq \arg\max_a Q_t(s_{t+1}, a) & \text{with prob. } \epsilon \end{cases}$$

- Under some conditions on the transition probability matrix, all state-action pairs will be visited infinitely often (in the limit)
- Is ϵ-greedy the "best" exploration strategy?
  - In general, ϵ-greedy is not the best exploration strategy.
  - However, we use this in deep RL most of the time.
  - There are many clever exploration strategies. This is an active area of research.

Q-Learning Algorithm

Algorithm Q-Learning Algorithm

Initialization: Initial Q-value, Q_0, a behavior policy π_b, $t = 0$, initial state s_0
for each $t = 0, 1, 2, \ldots$ **do**
    Observe the current state s_t
    Take action a_t according to the behavior policy π_b, $a_t \sim \pi_b(s_t, \cdot)$
    Observe the reward r_t and the next state s_{t+1}
    Update Q:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q_t(s_{t+1}, b) - Q_t(s_t, a_t)\right)$$

end for

Q-Learning Algorithm

Algorithm Q-Learning Algorithm

1: Initialization: Initial Q-value, Q_0, a behavior policy π_b, $t = 0$, initial state s_0
2: **for** each $t = 0, 1, 2, \ldots$ **do**
3: Observe the current state s_t
4: Take action a_t according to the behavior policy π_b, $a_t \sim \pi_b(s_t, \cdot)$
5: Observe the reward r_t and the next state s_{t+1}
6: Update Q:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q_t(s_{t+1}, b) - Q_t(s_t, a_t)\right)$$

7: **end for**

Theorem (Convergence of Q-learning)

If: (i) all state-action pairs are visited infinitely often, and
(ii) step size satisfies Robbins-Munro condition, $\sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty$,
then Q-Learning will converge, i.e., $Q_t \to Q^*$ *almost surely.*

Q-Learning Algorithm

- Select action according to a behavior policy $a_t \sim \pi_b(s_t, \cdot)$

Q-Learning Algorithm

- Select action according to a behavior policy $a_t \sim \pi_b(s_t, \cdot)$
- Update the Q-value function

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\right)$$

Q-Learning Algorithm

- Select action according to a behavior policy $a_t \sim \pi_b(s_t, \cdot)$
- Update the Q-value function

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\right)$$

- Action can be taken with respect to any behavior policy π_b as long as it visits all state-action pairs infinitely often

Q-Learning Algorithm

- Select action according to a behavior policy $a_t \sim \pi_b(s_t, \cdot)$
- Update the Q-value function

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\right)$$

- Action can be taken with respect to any behavior policy π_b as long as it visits all state-action pairs infinitely often
- Off-policy learning: Evaluate a policy using experience gathered from following a different policy/polices

Q-Learning Algorithm

- Select action according to a behavior policy $a_t \sim \pi_b(s_t, \cdot)$
- Update the Q-value function

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\right)$$

- Action can be taken with respect to any behavior policy π_b as long as it visits all state-action pairs infinitely often
- Off-policy learning: Evaluate a policy using experience gathered from following a different policy/polices
- Q-learning is an off-policy learning algorithm

Q-Learning Algorithm

- Select action according to a behavior policy $a_t \sim \pi_b(s_t, \cdot)$
- Update the Q-value function

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\right)$$

- Action can be taken with respect to any behavior policy π_b as long as it visits all state-action pairs infinitely often
- Off-policy learning: Evaluate a policy using experience gathered from following a different policy/polices
- Q-learning is an off-policy learning algorithm
  - We are estimating the value of π^* while acting according to another policy π_b

Q-Learning Algorithm

- We know that $Q_t \to Q^*$ according to Q-learning, under suitable assumption

Q-Learning Algorithm

- We know that $Q_t \to Q^*$ according to Q-learning, under suitable assumption
- Will the actual rewards obtained converge to the optimal?

Q-Learning Algorithm

- We know that $Q_t \to Q^*$ according to Q-learning, under suitable assumption
- Will the actual rewards obtained converge to the optimal?
  - The actual rewards observed will depend on the behavioral policy used

Q-Learning Algorithm

- We know that $Q_t \to Q^*$ according to Q-learning, under suitable assumption
- Will the actual rewards obtained converge to the optimal?
  - The actual rewards observed will depend on the behavioral policy used
- It is better to use a policy that will also converge to the optimal policy

Q-Learning Algorithm

- We know that $Q_t \to Q^*$ according to Q-learning, under suitable assumption
- Will the actual rewards obtained converge to the optimal?
  - The actual rewards observed will depend on the behavioral policy used
- It is better to use a policy that will also converge to the optimal policy
  - Often we use ϵ-greedy policy with respect to the current Q-value

Q-Learning Algorithm

- We know that $Q_t \to Q^*$ according to Q-learning, under suitable assumption
- Will the actual rewards obtained converge to the optimal?
  - The actual rewards observed will depend on the behavioral policy used
- It is better to use a policy that will also converge to the optimal policy
  - Often we use ϵ-greedy policy with respect to the current Q-value

Q-Learning Algorithm

- We know that $Q_t \to Q^*$ according to Q-learning, under suitable assumption
- Will the actual rewards obtained converge to the optimal?
  - The actual rewards observed will depend on the behavioral policy used
- It is better to use a policy that will also converge to the optimal policy
  - Often we use ϵ-greedy policy with respect to the current Q-value

Algorithm Q-Learning Algorithm

Initialization: Initial Q-value, Q_0, initial state s_0
for each $t = 0, 1, 2, \ldots$ **do**
    Observe the current state s_t
    Take action $a_t \sim \epsilon$-greedy$(Q_t)(s_t, \cdot)$
    Observe the reward r_t and the next state s_{t+1}
    Update Q:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b Q_t(s_{t+1}, b) - Q_t(s_t, a_t)\right)$$

end for