Introduction to Machine Learning (67577)
Reinforcement Learning

Shai Shalev-Shwartz

School of CS and Engineering,
The Hebrew University of Jerusalem

Reinforcement Learning

Outline

1. Reinforcement Learning

2. Multi-Armed Bandit
   - ϵ-greedy exploration
   - EXP3
   - UCB

3. Markov Decision Process (MDP)
   - Value Iteration
   - Q-Learning
   - Deep-Q-Learning
   - Temporal Abstraction

Reinforcement Learning

Goal: Learn a policy, mapping from state space, S, to action space, A

Learning Process:

For $t = 1, 2, \ldots$

- Agent observes state $s_t \in S$
- Agent decides on action $a_t \in A$ based on the current policy
- Environment provides reward $r_t \in \mathbb{R}$
- Environment moves the agent to next state $s_{t+1} \in S$

Reinforcement Learning

Goal: Learn a policy, mapping from state space, S, to action space, A

Learning Process:
For $t = 1, 2, \ldots$

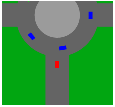
- Agent observes state $s_t \in S$
- Agent decides on action $a_t \in A$ based on the current policy
- Environment provides reward $r_t \in \mathbb{R}$
- Environment moves the agent to next state $s_{t+1} \in S$

Many applications, e.g.: Robotics, Playing games, Finance, Inventory management, ...

Examples

Merge into traffic:

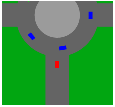
- Goal: Adjust the speed of the car according to traffic
- State is positions and velocities of the car and the preceding car
- Action is acceleration/braking command
- Reward is composed of avoiding accidents, smooth driving, and making progress



Examples

Merge into traffic:

- Goal: Adjust the speed of the car according to traffic
- State is positions and velocities of the car and the preceding car
- Action is acceleration/braking command
- Reward is composed of avoiding accidents, smooth driving, and making progress



Playing Atari Game:

- https://www.youtube.com/watch?v=V1eYniJ0Rnk

Average Reward and Discounted Reward

Average Reward: Given time horizon T, the average reward of following a policy π is

$$R_T(\pi) \;=\; \mathbb{E}\,\frac{1}{T}\sum_{t=1}^{T} r_t$$

Average Reward and Discounted Reward

Average Reward: Given time horizon T, the average reward of following a policy π is

$$R_T(\pi) \; = \; \mathbb{E} \, \frac{1}{T} \sum_{t=1}^{T} r_t$$

Discounted Reward: Given $\gamma \in (0,1)$, the discounted reward of following a policy π is

$$R_\gamma(\pi) \; = \; \mathbb{E} \sum_{t=1}^{\infty} \gamma^t \, r_t$$

Reinforcement Learning vs. Supervised Learning

SL is a special case of RL in which s_t is the "instance", a_t is the predicted label, $-r_t$ is the loss measuring the discrepancy between a_t and the "true" label, y_t, and s_{t+1} is chosen independent of s_t and a_t.

Reinforcement Learning vs. Supervised Learning

SL is a special case of RL in which s_t is the "instance", a_t is the predicted label, $-r_t$ is the loss measuring the discrepancy between a_t and the "true" label, y_t, and s_{t+1} is chosen independent of s_t and a_t.

Differences:

- In SL, actions do not effect the environment, therefore we can collect training examples in advance, and only then search for a policy
- In SL, the effect of actions is local, while in RL, actions have long-term effect
- In SL we are given the correct answer, while in RL we only observe a reward

Outline

The Multi-Armed Bandit Problem (Robbins 1952)

- States: The state is constant (has no effect)

The Multi-Armed Bandit Problem (Robbins 1952)

- States: The state is constant (has no effect)
- Actions: n slot machines ("arms").

The Multi-Armed Bandit Problem (Robbins 1952)

- States: The state is constant (has no effect)
- Actions: n slot machines ("arms").
- Reward: There exists a deterministic function ρ from $A = [n]$ to all distributions over $[0, 1]$ s.t. for every t, $r_t \sim \rho(a_t)$

The Multi-Armed Bandit Problem (Robbins 1952)

- States: The state is constant (has no effect)
- Actions: n slot machines ("arms").
- Reward: There exists a deterministic function ρ from $A = [n]$ to all distributions over $[0, 1]$ s.t. for every t, $r_t \sim \rho(a_t)$
- Denote: $\mu_i = \mathbb{E}[r_t | a_t = i]$, $i^* = \operatorname{argmax}_i \mu_i$, $\mu^* = \mu_{i^*}$, $\Delta_i = \mu^* - \mu_i$

The Multi-Armed Bandit Problem (Robbins 1952)

- States: The state is constant (has no effect)
- Actions: n slot machines ("arms").
- Reward: There exists a deterministic function ρ from $A = [n]$ to all distributions over $[0, 1]$ s.t. for every t, $r_t \sim \rho(a_t)$
- Denote: $\mu_i = \mathbb{E}[r_t | a_t = i]$, $i^* = \operatorname{argmax}_i \mu_i$, $\mu^* = \mu_{i^*}$, $\Delta_i = \mu^* - \mu_i$
- Regret:

$$\mu^* - \mathbb{E}\, R_T(\pi)$$

The Exploration-Exploitation Tradeoff

How to pick the next action?

- Exploitation: Choose the most promising action based on your current understanding

The Exploration-Exploitation Tradeoff

How to pick the next action?

- Exploitation: Choose the most promising action based on your current understanding
- Exploration: Maybe there is a better arm ?

Naive approach: first explore then exploit

- Procedure:

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)
  - Let $\hat{i} = \mathrm{argmax}_i \, \hat{\mu}_i$, where $\hat{\mu}_i = \mathrm{avg}(r_t : a_t = i)$

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)
  - Let $\hat{i} = \mathrm{argmax}_i\, \hat{\mu}_i$, where $\hat{\mu}_i = \mathrm{avg}(r_t : a_t = i)$
  - Pure exploitation for the rest of the $T - m$ iterations (always pick $\hat{i}$)

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)
  - Let $\hat{i} = \text{argmax}_i \, \hat{\mu}_i$, where $\hat{\mu}_i = \text{avg}(r_t : a_t = i)$
  - Pure exploitation for the rest of the $T - m$ iterations (always pick $\hat{i}$)
- Analysis:

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)
  - Let $\hat{i} = \mathrm{argmax}_i \, \hat{\mu}_i$, where $\hat{\mu}_i = \mathrm{avg}(r_t : a_t = i)$
  - Pure exploitation for the rest of the $T - m$ iterations (always pick $\hat{i}$)
- Analysis:
  - **Claim:** If m is order of $n \log(n)/\epsilon^2$ then for all i, $|\mu_i - \hat{\mu}_i| \le \epsilon$

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)
  - Let $\hat{i} = \mathrm{argmax}_i \, \hat{\mu}_i$, where $\hat{\mu}_i = \mathrm{avg}(r_t : a_t = i)$
  - Pure exploitation for the rest of the $T - m$ iterations (always pick $\hat{i}$)
- Analysis:
  - **Claim:** If m is order of $n \log(n)/\epsilon^2$ then for all i, $|\mu_i - \hat{\mu}_i| \le \epsilon$
  - **Proof:** Hoeffding + union bound

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)
  - Let $\hat{i} = \mathrm{argmax}_i \, \hat{\mu}_i$, where $\hat{\mu}_i = \mathrm{avg}(r_t : a_t = i)$
  - Pure exploitation for the rest of the $T - m$ iterations (always pick $\hat{i}$)
- Analysis:
  - **Claim:** If m is order of $n \log(n)/\epsilon^2$ then for all i, $|\mu_i - \hat{\mu}_i| \le \epsilon$
  - **Proof:** Hoeffding + union bound
  - Regret:

$$\mu^* - \frac{m\bar{\mu} + (T-m)\mu_{\hat{i}}}{T} = (\mu^* - \mu_{\hat{i}}) + \tfrac{m}{T}(\mu_{\hat{i}} - \bar{\mu})$$

$$\le (\mu^* - \hat{\mu}_{i^*} + \hat{\mu}_{i^*} - \hat{\mu}_{\hat{i}} + \hat{\mu}_{\hat{i}} - \mu_{\hat{i}}) + \tfrac{m}{T} \le 2\epsilon + \frac{n \log(n)}{T\,\epsilon^2}$$

Naive approach: first explore then exploit

- Procedure:
  - Pure exploration for the first m iterations (pick actions at random)
  - Let $\hat{i} = \text{argmax}_i \, \hat{\mu}_i$, where $\hat{\mu}_i = \text{avg}(r_t : a_t = i)$
  - Pure exploitation for the rest of the $T - m$ iterations (always pick $\hat{i}$)
- Analysis:
  - **Claim:** If m is order of $n \log(n)/\epsilon^2$ then for all i, $|\mu_i - \hat{\mu}_i| \le \epsilon$
  - **Proof:** Hoeffding + union bound
  - Regret:

$$\mu^* - \frac{m\bar{\mu} + (T-m)\mu_{\hat{i}}}{T} = (\mu^* - \mu_{\hat{i}}) + \tfrac{m}{T}(\mu_{\hat{i}} - \bar{\mu})$$

$$\le (\mu^* - \hat{\mu}_{i^*} + \hat{\mu}_{i^*} - \hat{\mu}_{\hat{i}} + \hat{\mu}_{\hat{i}} - \mu_{\hat{i}}) + \tfrac{m}{T} \le 2\epsilon + \frac{n \log(n)}{T\,\epsilon^2}$$

  - For the best ϵ, the regret is order of $\left(\frac{n \log(n)}{T}\right)^{1/3}$

SGD with ϵ-greedy exploration

- Want to minimize $L(w) = -w^\top \mu$ over $\{w \in [0,1]^n : \sum_i w_i = 1\}$

SGD with ϵ-greedy exploration

- Want to minimize $L(w) = -w^\top \mu$ over $\{w \in [0,1]^n : \sum_i w_i = 1\}$
- A convex objective with convex constraint — can we use Stochastic Gradient Descent ?

SGD with ϵ-greedy exploration

- Want to minimize $L(w) = -w^\top \mu$ over $\{w \in [0,1]^n : \sum_i w_i = 1\}$
- A convex objective with convex constraint — can we use Stochastic Gradient Descent ?
- For every probability vector p, if we choose $i_t \sim p$ and set $\hat{\nabla} L(w^{(t)}) = -r_t \frac{1}{p_{i_t}} e_{i_t}$, then

$$\mathbb{E}[\hat{\nabla} L(w^{(t)})] = \sum_{i=1}^{n} p_i \cdot \left(-\mathbb{E}[r_t] \frac{1}{p_i} e_i \right) = -\mu = \nabla L(w^{(t)})$$

SGD with ϵ-greedy exploration

- Want to minimize $L(w) = -w^\top \mu$ over $\{w \in [0,1]^n : \sum_i w_i = 1\}$
- A convex objective with convex constraint — can we use Stochastic Gradient Descent ?
- For every probability vector p, if we choose $i_t \sim p$ and set $\hat{\nabla} L(w^{(t)}) = -r_t \frac{1}{p_{i_t}} e_{i_t}$, then

$$\mathbb{E}[\hat{\nabla} L(w^{(t)})] = \sum_{i=1}^{n} p_i \cdot \left(-\mathbb{E}[r_t] \frac{1}{p_i} e_i \right) = -\mu = \nabla L(w^{(t)})$$

- Problem: we need that $\mathbb{E}[\|\hat{\nabla} L(w^{(t)})\|^2]$ will be bounded

SGD with ϵ-greedy exploration

- Want to minimize $L(w) = -w^\top \mu$ over $\{w \in [0,1]^n : \sum_i w_i = 1\}$
- A convex objective with convex constraint — can we use Stochastic Gradient Descent ?
- For every probability vector p, if we choose $i_t \sim p$ and set $\hat{\nabla} L(w^{(t)}) = -r_t \frac{1}{p_{i_t}} e_{i_t}$, then

$$\mathbb{E}[\hat{\nabla} L(w^{(t)})] = \sum_{i=1}^{n} p_i \cdot \left(-\mathbb{E}[r_t] \frac{1}{p_i} e_i \right) = -\mu = \nabla L(w^{(t)})$$

- Problem: we need that $\mathbb{E}[\|\hat{\nabla} L(w^{(t)})\|^2]$ will be bounded
- ϵ-greedy exploration: set $p = (1-\epsilon) w^{(t)} + \epsilon \mathbf{1}/n$
  That is, we explore w.p. ϵ and exploit w.p. $(1-\epsilon)$

SGD with ϵ-greedy exploration

- Want to minimize $L(w) = -w^\top \mu$ over $\{w \in [0,1]^n : \sum_i w_i = 1\}$
- A convex objective with convex constraint — can we use Stochastic Gradient Descent ?
- For every probability vector p, if we choose $i_t \sim p$ and set $\hat{\nabla} L(w^{(t)}) = -r_t \frac{1}{p_{i_t}} e_{i_t}$, then

$$\mathbb{E}[\hat{\nabla} L(w^{(t)})] = \sum_{i=1}^{n} p_i \cdot \left(- \mathbb{E}[r_t] \frac{1}{p_i} e_i \right) = -\mu = \nabla L(w^{(t)})$$

- Problem: we need that $\mathbb{E}[\|\hat{\nabla} L(w^{(t)})\|^2]$ will be bounded
- ϵ-greedy exploration: set $p = (1-\epsilon) w^{(t)} + \epsilon \mathbf{1}/n$
  That is, we explore w.p. ϵ and exploit w.p. $(1-\epsilon)$
- Regret analysis: it can be show that the regret is order of $\left(\frac{n}{T}\right)^{1/3}$

EXP3 (Auer, Cesa-Bianchi, Freund, Schapire)

- Same as SGD, but we pick $p = w^{(t)}$ and update using Stochastic Gradient in the Exponent

EXP3 (Auer, Cesa-Bianchi, Freund, Schapire)

- Same as SGD, but we pick $p = w^{(t)}$ and update using Stochastic Gradient in the Exponent
- Initialize: $w^{(1)} = (1/n, \ldots, 1/n)$

EXP3 (Auer, Cesa-Bianchi, Freund, Schapire)

- Same as SGD, but we pick $p = w^{(t)}$ and update using Stochastic Gradient in the Exponent
- Initialize: $w^{(1)} = (1/n, \ldots, 1/n)$
- Update: $w_i^{(t+1)} = \frac{1}{Z_t}\, w_i^{(t)} \exp(-\eta \hat{\nabla} L(w^{(t)})[i])$

EXP3 (Auer, Cesa-Bianchi, Freund, Schapire)

- Same as SGD, but we pick $p = w^{(t)}$ and update using Stochastic Gradient in the Exponent
- Initialize: $w^{(1)} = (1/n, \ldots, 1/n)$
- Update: $w_i^{(t+1)} = \frac{1}{Z_t} \, w_i^{(t)} \, \exp(-\eta \hat{\nabla} L(w^{(t)})[i])$
- The update makes sure that we have some exploration (we never completely zero components of w)

EXP3 (Auer, Cesa-Bianchi, Freund, Schapire)

- Same as SGD, but we pick $p = w^{(t)}$ and update using Stochastic Gradient in the Exponent
- Initialize: $w^{(1)} = (1/n, \ldots, 1/n)$
- Update: $w_i^{(t+1)} = \frac{1}{Z_t} \, w_i^{(t)} \, \exp(-\eta \hat{\nabla} L(w^{(t)})[i])$
- The update makes sure that we have some exploration (we never completely zero components of w)
- Regret analysis: it can be show to be order of $\left(\frac{n \log(n)}{T}\right)^{1/2}$

EXP3 (Auer, Cesa-Bianchi, Freund, Schapire)

- Same as SGD, but we pick $p = w^{(t)}$ and update using Stochastic Gradient in the Exponent
- Initialize: $w^{(1)} = (1/n, \ldots, 1/n)$
- Update: $w_i^{(t+1)} = \frac{1}{Z_t}\, w_i^{(t)}\, \exp(-\eta \hat{\nabla} L(w^{(t)})[i])$
- The update makes sure that we have some exploration (we never completely zero components of w)
- Regret analysis: it can be show to be order of $\left(\frac{n\log(n)}{T}\right)^{1/2}$
- EXP3 stands for "**Exp**loration-**Exp**loitation using **Exp**onentiated Gradient"

EXP3 (Auer, Cesa-Bianchi, Freund, Schapire)

- Same as SGD, but we pick $p = w^{(t)}$ and update using Stochastic Gradient in the Exponent
- Initialize: $w^{(1)} = (1/n, \ldots, 1/n)$
- Update: $w_i^{(t+1)} = \frac{1}{Z_t}\, w_i^{(t)}\, \exp(-\eta \hat{\nabla} L(w^{(t)})[i])$
- The update makes sure that we have some exploration (we never completely zero components of w)
- Regret analysis: it can be show to be order of $\left(\frac{n \log(n)}{T}\right)^{1/2}$
- EXP3 stands for "**Exp**loration-**Exp**loitation using **Exp**onentiated Gradient"
- Remark: EXP3 works also in the adversarial setting

Upper Confidence Bound (UCB)

- Optimism in the face of uncertainty (Lai and Robbins' 1985)

Upper Confidence Bound (UCB)

- Optimism in the face of uncertainty (Lai and Robbins' 1985)
- Using Hoeffding's inequality, if we pulled arm i for $N_i(t)$ times then:

$$\mu_i \;\leq\; \hat{\mu}_i + \sqrt{\frac{2\log(T)}{N_i(t)}} := \mathrm{UCB}_i(t)$$

Upper Confidence Bound (UCB)

- Optimism in the face of uncertainty (Lai and Robbins' 1985)
- Using Hoeffding's inequality, if we pulled arm i for $N_i(t)$ times then:

$$\mu_i \;\leq\; \hat{\mu}_i + \sqrt{\frac{2\log(T)}{N_i(t)}} := \mathrm{UCB}_i(t)$$

- The UCB rule is to pull the arm that maximizes $\mathrm{UCB}_i(t)$

Upper Confidence Bound (UCB)

- Optimism in the face of uncertainty (Lai and Robbins' 1985)
- Using Hoeffding's inequality, if we pulled arm i for $N_i(t)$ times then:

$$\mu_i \ \leq \ \hat{\mu}_i + \sqrt{\frac{2\log(T)}{N_i(t)}} := \mathrm{UCB}_i(t)$$

- The UCB rule is to pull the arm that maximizes $\mathrm{UCB}_i(t)$
- Regret can be shown to be bounded by $\frac{\log(T)}{T} \sum_{i:\Delta_i>0} \frac{1}{\Delta_i}$

Outline

1. Reinforcement Learning

2. Multi-Armed Bandit
   - ϵ-greedy exploration
   - EXP3
   - UCB

3. Markov Decision Process (MDP)
   - Value Iteration
   - Q-Learning
   - Deep-Q-Learning
   - Temporal Abstraction

Markov Decision Process (MDP)

The Markovian Assumption:

- For every t, $s_{t+1} \sim \tau(s_t, a_t)$ where τ is a deterministic function over $S \times A$
- For every t, r_t is a random variable over $[0, 1]$ whose distribution depends deterministically only on (s_t, a_t) and we denote its expected value by $\rho(s_t, a_t)$,
- It follows that (s_{t+1}, r_t) is conditionally independent of $(s_{t-1}, a_{t-1}), (s_{t-2}, a_{t-2}), \ldots, (s_1, a_1)$ given (s_t, a_t)

MDP — algorithms

- Value Iteration: Find the optimal policy when τ and ρ are known
- Q-Learning: Find the optimal policy when τ and ρ are not known

The Value Function and the Q-Function

- The optimal value function is $V^* : S \to \mathbb{R}$ s.t. $V^*(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_1 = s\right]$

The Value Function and the Q-Function

- The optimal value function is $V^* : S \to \mathbb{R}$ s.t. $V^*(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_1 = s\right]$
- Observe (this is known as Bellman's Equation:)

$$V^*(s) = \max_{a \in A}\left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V^*(s')\right]$$

The Value Function and the Q-Function

- The optimal value function is $V^* : S \to \mathbb{R}$ s.t. $V^*(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_1 = s\right]$
- Observe (this is known as Bellman's Equation:)

$$V^*(s) = \max_{a \in A}\left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V^*(s')\right]$$

- The objective function in the above maximization problem is called the *optimal action-value function*, and is denoted by

$$Q^*(s,a) = \rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V^*(s') \ .$$

The Value Function and the Q-Function

- The optimal value function is $V^* : S \to \mathbb{R}$ s.t. $V^*(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_1 = s\right]$
- Observe (this is known as Bellman's Equation:)

$$V^*(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V^*(s')\right]$$

- The objective function in the above maximization problem is called the *optimal action-value function*, and is denoted by

$$Q^*(s,a) = \rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V^*(s') \; .$$

- corollary: The optimal policy is the greedy policy w.r.t. Q^*, namely, $\pi^*(s) = \mathrm{argmax}_a \, Q^*(s,a)$

The Value Function and the Q-Function

- The optimal value function is $V^* : S \to \mathbb{R}$ s.t. $V^*(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_1 = s\right]$
- Observe (this is known as Bellman's Equation:)

$$V^*(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V^*(s')\right]$$

- The objective function in the above maximization problem is called the *optimal action-value function*, and is denoted by

$$Q^*(s,a) = \rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V^*(s') \ .$$

- corollary: The optimal policy is the greedy policy w.r.t. Q^*, namely, $\pi^*(s) = \mathrm{argmax}_a Q^*(s,a)$
- In particular, the optimal a_t is a deterministic function of s_t

Value Iteration

- Iterative algorithm for finding V^*:
  Start with some arbitrary V_0 and update

$$V_{t+1}(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V_t(s') \right]$$

Value Iteration

- Iterative algorithm for finding V^*:
  Start with some arbitrary V_0 and update

$$V_{t+1}(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V_t(s') \right]$$

- Theorem: $\|V_t - V^*\|_\infty \;\leq\; \gamma^t \, \|V_0 - V^*\|_\infty$

Value Iteration

- Iterative algorithm for finding V^*:
  Start with some arbitrary V_0 and update

$$V_{t+1}(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V_t(s') \right]$$

- Theorem: $\|V_t - V^*\|_\infty \;\leq\; \gamma^t \, \|V_0 - V^*\|_\infty$
- Proof idea:

Value Iteration

- Iterative algorithm for finding V^*:
  Start with some arbitrary V_0 and update

$$V_{t+1}(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V_t(s') \right]$$

- Theorem: $\|V_t - V^*\|_\infty \;\leq\; \gamma^t \, \|V_0 - V^*\|_\infty$
- Proof idea:
  - Define $T^* : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ to be the operator s.t. $V_{t+1} = T^*(V_t)$

Value Iteration

- Iterative algorithm for finding V^*:
  Start with some arbitrary V_0 and update

$$V_{t+1}(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V_t(s') \right]$$

- Theorem: $\|V_t - V^*\|_\infty \;\le\; \gamma^t \, \|V_0 - V^*\|_\infty$
- Proof idea:
  - Define $T^* : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ to be the operator s.t. $V_{t+1} = T^*(V_t)$
  - Show that T^* is a contraction mapping: for any two vector in $\mathbb{R}^{|S|}$ we have $\|T^*(u) - T^*(v)\|_\infty \le \gamma \|u - v\|_\infty$

Value Iteration

- Iterative algorithm for finding V^*:
  Start with some arbitrary V_0 and update

$$V_{t+1}(s) = \max_{a \in A} \left[\rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} V_t(s') \right]$$

- Theorem: $\|V_t - V^*\|_\infty \;\leq\; \gamma^t \, \|V_0 - V^*\|_\infty$
- Proof idea:
  - Define $T^* : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ to be the operator s.t. $V_{t+1} = T^*(V_t)$
  - Show that T^* is a contraction mapping: for any two vector in $\mathbb{R}^{|S|}$ we have $\|T^*(u) - T^*(v)\|_\infty \leq \gamma \|u - v\|_\infty$
  - The proof follows from Banach's fixed point theorem

Naive Learner

- Step 1: Estimate τ and ρ by applying purely random policy
- Step 2: Apply Value Iteration to learn the optimal policy

Q-Learning

- Bellman's equation for the Q function:

$$Q^*(s,a) = \rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} \max_{a'} Q^*(s',a')$$

Q-Learning

- Bellman's equation for the Q function:

$$Q^*(s,a) = \rho(s,a) + \gamma \underset{s' \sim \tau(s,a)}{\mathbb{E}} \max_{a'} Q^*(s',a')$$

- Given (s_t, a_t, s_{t+1}, r_t), define

$$\delta_{s_t,a_t}(Q) = Q(s_t,a_t) - \left(r_t + \gamma \max_{a'} Q(s_{t+1},a')\right)$$

Q-Learning

- Bellman's equation for the Q function:

$$Q^*(s,a) = \rho(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \tau(s,a)} \max_{a'} Q^*(s',a')$$

- Given (s_t, a_t, s_{t+1}, r_t), define

$$\delta_{s_t,a_t}(Q) = Q(s_t,a_t) - \left(r_t + \gamma \max_{a'} Q(s_{t+1},a') \right)$$

- Initialize Q_1 and update

$$Q_{t+1}(s,a) = Q_t(s,a) - \eta_t \delta_{s_t,a_t}(Q_t) 1[s = s_t, a = a_t]$$

Q-Learning

- Bellman's equation for the Q function:

$$Q^*(s,a) = \rho(s,a) + \gamma \underset{s' \sim \tau(s,a)}{\mathbb{E}} \max_{a'} Q^*(s',a')$$

- Given (s_t, a_t, s_{t+1}, r_t), define

$$\delta_{s_t,a_t}(Q) = Q(s_t,a_t) - \left(r_t + \gamma \max_{a'} Q(s_{t+1},a') \right)$$

- Initialize Q_1 and update

$$Q_{t+1}(s,a) = Q_t(s,a) - \eta_t \delta_{s_t,a_t}(Q_t) 1[s = s_t, a = a_t]$$

- The above update aims at converging to Bellman's equation

Exploration for Q-Learning

- Q-Learning can be applied for any choice of a_t (it is an “off policy” learner)

Exploration for Q-Learning

- Q-Learning can be applied for any choice of a_t (it is an "off policy" learner)
- Speed of convergence can be improved if we balance the exploration-exploitation tradeoff (by one of the methods described previously)

The Curse of Dimensionality

- The Q function is a table of size $|S| \times |A|$
- This size grows exponentially with the dimensions of S and A
- The convergence of the "tabular" Q-learning (namely, maintaing Q is a table of size $|S| \times |A|$) becomes very slow
- We describe two approaches to overcome this problem:
  - Function Approximation
  - Temporal Abstractions

Function Approximation for Q-Learning

- Maintain a parametric hypothesis class of Q functions

Function Approximation for Q-Learning

- Maintain a parametric hypothesis class of Q functions
- Rewrite δ as a function of the parameter θ:

$$\delta_{s_t,a_t}(\theta) = Q_\theta(s_t, a_t) - \left(r_t + \gamma \max_{a'} Q_{\theta_t}(s_{t+1}, a')\right)$$

Function Approximation for Q-Learning

- Maintain a parametric hypothesis class of Q functions
- Rewrite δ as a function of the parameter θ:

$$\delta_{s_t,a_t}(\theta) = Q_\theta(s_t, a_t) - \left(r_t + \gamma \max_{a'} Q_{\theta_t}(s_{t+1}, a')\right)$$

- Since we want to minimize $\frac{1}{2}\delta_{s_t,a_t}(\theta)^2$ we take a gradient step:

$$\theta_{t+1} = \theta_t - \eta_t \delta_{s_t,a_t}(\theta_t)\, \nabla Q_\theta(s_t, a_t)$$

Deep-Q-Learning

- Used by DeepMind to learn to play Atari games

Deep-Q-Learning

- Used by DeepMind to learn to play Atari games
- Let $Q_\theta : S \rightarrow \mathbb{R}^{|A|}$ be a deep network, where we take $S \subset \mathbb{R}^d$ and assume that $|A|$ is not too larger

Deep-Q-Learning

- Used by DeepMind to learn to play Atari games
- Let $Q_\theta : S \to \mathbb{R}^{|A|}$ be a deep network, where we take $S \subset \mathbb{R}^d$ and assume that $|A|$ is not too larger
- Exploration: ϵ-greedy

Deep-Q-Learning

- Used by DeepMind to learn to play Atari games
- Let $Q_\theta : S \to \mathbb{R}^{|A|}$ be a deep network, where we take $S \subset \mathbb{R}^d$ and assume that $|A|$ is not too larger
- Exploration: ϵ-greedy
- Memory replay: After executing a_t and observing r_t, s_{t+1} we store the example (s_t, a_t, r_t, s_{t+1}) in a database. Instead of updating just based on the last example, update based on a mini-batch of random examples from the database

Deep-Q-Learning

- Used by DeepMind to learn to play Atari games
- Let $Q_\theta : S \to \mathbb{R}^{|A|}$ be a deep network, where we take $S \subset \mathbb{R}^d$ and assume that $|A|$ is not too larger
- Exploration: ϵ-greedy
- Memory replay: After executing a_t and observing r_t, s_{t+1} we store the example (s_t, a_t, r_t, s_{t+1}) in a database. Instead of updating just based on the last example, update based on a mini-batch of random examples from the database
- Freezing Q: Every C step, freeze the value of Q_θ and denote it by $\hat{Q}$. Then, redefine δ to be

$$\delta_{s_t,a_t}(\theta) = Q_\theta(s_t, a_t) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a')\right)$$

  This has some stabilization effect on the algorithm

Intuition: Structuring a State Space

- Consider some state space $S \subset \mathbb{R}^d$
- Suppose we partition it to $S = S_1 \mathbin{\dot\cup} S_2 \mathbin{\dot\cup} \ldots \mathbin{\dot\cup} S_k$
- Assuming homogenous actions within each S_i, we can apply Q learning while using $[k]$ as a new state space
- One can think of Deep-Q-Learning as automatically finding the partition (the first layers of the network)

Temporal Abstraction

- Decisions are often structured into sub-tasks with a broad range of time scale. E.g.:
  - Task: Call a taxi
  - Step 1: finding my phone
  - Step 2: finding the number
  - Step 3: dialing the first digit
  - ...
  - Step 20: commanding my finger muscle to move into the right place ...

Temporal Abstraction

- Decisions are often structured into sub-tasks with a broad range of time scale. E.g.:
  - Task: Call a taxi
  - Step 1: finding my phone
  - Step 2: finding the number
  - Step 3: dialing the first digit
  - ...
  - Step 20: commanding my finger muscle to move into the right place ...
- Options: (Sutton, Precup, Singh)
  - An option is a pair (π, β) where
    - $\pi : S \to A$ is the policy to apply while within the "option"
    - $\beta : S \to [0, 1]$ is a stochastic termination function

Temporal Abstraction

- Decisions are often structured into sub-tasks with a broad range of time scale. E.g.:
  - Task: Call a taxi
  - Step 1: finding my phone
  - Step 2: finding the number
  - Step 3: dialing the first digit
  - ...
  - Step 20: commanding my finger muscle to move into the right place ...
- Options: (Sutton, Precup, Singh)
  - An option is a pair (π, β) where
    - $\pi : S \to A$ is the policy to apply while within the "option"
    - $\beta : S \to [0, 1]$ is a stochastic termination function
  - Instead of directly choosing actions, the agent picks an option $o_t \in O$, and this option is applied until it terminates
  - That is, we should learn a policy over options, $\mu : S \to O$

Temporal Abstraction

- Decisions are often structured into sub-tasks with a broad range of time scale. E.g.:
  - Task: Call a taxi
  - Step 1: finding my phone
  - Step 2: finding the number
  - Step 3: dialing the first digit
  - ...
  - Step 20: commanding my finger muscle to move into the right place ...
- Options: (Sutton, Precup, Singh)
  - An option is a pair (π, β) where
    - $\pi : S \to A$ is the policy to apply while within the "option"
    - $\beta : S \to [0, 1]$ is a stochastic termination function
  - Instead of directly choosing actions, the agent picks an option $o_t \in O$, and this option is applied until it terminates
  - That is, we should learn a policy over options, $\mu : S \to O$
  - We can learn μ similarly to how we learn a vanilla policy, and the advantage is that mt may be easier to pick O than picking A

Limitations of MDPs

- The Markovian assumption is mathematically convenient but rarely holds in practice
- POMDP = Partially Observed MDP: There is a hidden Markovian state, but we only observe a view that depends on it
- Another approach is "direct policy search", that do not necessarily rely on the Markovian assumption.

Summary

- Reinforcement Learning is a powerful and useful learning setting, but is much harder than Supervised Learning
- The Exploration-Exploitation Tradeoff
- MDP: Connecting the future rewards to current actions using a Markovian assumption

Appendix

Stationary Distribution of an MDP

- A MDP and a deterministic policy function π induces a Markov chain over S, because $\mathbb{P}[s_{t+1}|s_t, a_t, \ldots, s_1, a_1] = \mathbb{P}[s_{t+1}|s_t]$
- The stationary distribution over S is the probability vector q such that $q_s = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} 1[s_t = s]$
- We have that $q_s = \sum_{s'} q_{s'}\, \mathbb{P}[s|s']$
- We have $R_T(\pi) \to \sum_s q_s \rho_s$ where $\rho_s = (s, \pi(s))$
- Using P to denote the matrix s.t. $P_{s,s'} = \mathbb{P}[s|s']$, we obtain that the average reward is the solution of the following Linear Program (LP):

$$\min_q \langle q, -\rho\rangle \text{ s.t. } q \geq 0, \langle q, 1\rangle = 1, (P - I)q = 0$$

The Dual Problem and the Value Function

- Primal

$$\min_{q \in \mathbb{R}^{|S|}} \langle q, -\rho \rangle \text{ s.t. } q \geq 0, \langle q, 1 \rangle = 1, (P - I)q = 0$$

- Dual: define $A = [(P^\top - I), \mathbf{1}]$

$$\max_{v \in \mathbb{R}^{|S|+1}} \langle v, [0, \ldots, 0, 1] \rangle \text{ s.t. } Av \leq -\rho$$

- Equivalently:

$$\max_{v \in \mathbb{R}^{|S|}, \beta \in \mathbb{R}} \beta \text{ s.t. } \beta \leq -\rho + (I - P^\top)v = v - [\rho + P^\top v]$$

- Equivalently (since at the optimum, $\beta = \min_s [v_s - (\rho_s + (P^\top v)_s)]$)

$$\max_{v \in \mathbb{R}^{|S|}} \min_s [v_s - (\rho_s + (P^\top v)_s)]$$

Solution

- Assumption: rewards are ≥ 0
- Claim: If there's a solution to $(I - P^\top)v = \rho$, then it is an optimal solution for which $\beta = 0$
- Proof: For any v, choose s s.t. v_s is minimal, then $(P^\top v)_s \geq v_s$, because the rows of $P^\top$ are probabilities vector. Since $\rho_s \geq 0$, we have that for this s, $v_s - (\rho_s + (P^\top v)_s) \leq 0$, so $\beta \leq 0$, which concludes our proof.