Machine Learning

A. G. Schwing & M. Telgarsky

University of Illinois at Urbana-Champaign, 2018

L24: Q-learning

Goals of this lecture

Goals of this lecture

- Extending MDPs

Goals of this lecture

- Extending MDPs
- Getting to know Q-learning

Recap so far: Known MDP

- To compute V^*, Q^*, π^*: use **policy/value iteration or exhaustive**
- To evaluate fixed policy π: use policy evaluation

But what if:

- Transition probabilities and rewards are not known
- No model available (model free RL)

But what if:

- Transition probabilities and rewards are not known
- No model available (model free RL)

Bellman optimality principle:

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \left[R(s,a,s') + \max_{a' \in \mathcal{A}_{s'}} Q^*(s',a')) \right]$$

But what if:

- Transition probabilities and rewards are not known
- No model available (model free RL)

Bellman optimality principle:

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \left[R(s,a,s') + \max_{a' \in \mathcal{A}_{s'}} Q^*(s',a')) \right]$$

Idea:

But what if:

- Transition probabilities and rewards are not known
- No model available (model free RL)

Bellman optimality principle:

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \left[R(s,a,s') + \max_{a' \in \mathcal{A}_{s'}} Q^*(s',a')) \right]$$

Idea:

- Run a simulator to collect experience tuples/samples (s, a, r, s')

But what if:

- Transition probabilities and rewards are not known
- No model available (model free RL)

Bellman optimality principle:

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \left[R(s,a,s') + \max_{a' \in \mathcal{A}_{s'}} Q^*(s',a')) \right]$$

Idea:

- Run a simulator to collect experience tuples/samples (s, a, r, s')
- Approximate transition probability using samples

Algorithm sketch for Q-learning:

Algorithm sketch for Q-learning:

- Obtain a sample transition (s, a, r, s')

Algorithm sketch for Q-learning:

- Obtain a sample transition (s, a, r, s')
- Obtained sample suggests:

$$Q(s,a) \approx y_{(s,a,r,s')} = r + \max_{a' \in \mathcal{A}_{s'}} Q(s', a')$$

Algorithm sketch for Q-learning:

- Obtain a sample transition (s, a, r, s')
- Obtained sample suggests:

$$Q(s,a) \approx y_{(s,a,r,s')} = r + \max_{a' \in \mathcal{A}_{s'}} Q(s', a')$$

- To account for missing transition probability we keep running average

Algorithm sketch for Q-learning:

- Obtain a sample transition (s, a, r, s')
- Obtained sample suggests:

$$Q(s, a) \approx y_{(s,a,r,s')} = r + \max_{a' \in \mathcal{A}_{s'}} Q(s', a')$$

- To account for missing transition probability we keep running average

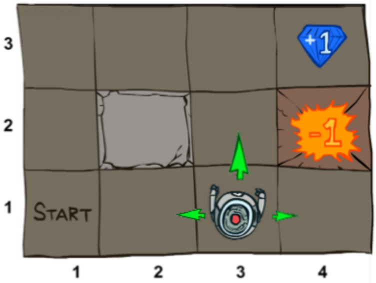
$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha y_{(s,a,r,s')}$$

Summary:

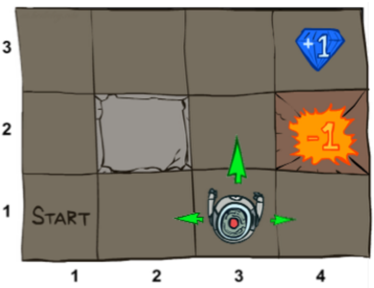
- Known MDP
  - To compute V^*, Q^*, π^*: use value/policy iteration
  - To evaluate fixed policy π: use policy evaluation

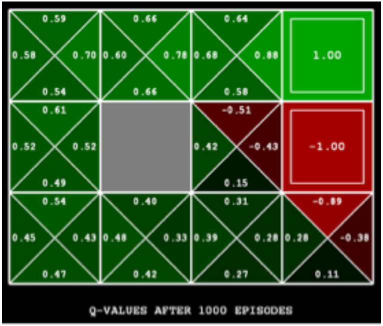
- Unknown MDP: Model free
  - To compute V^*, Q^*, π^*: use Q-learning
  - To evaluate fixed policy π: use value learning

Our current Q-learning works fine in case we have a tabular environment:



Our current Q-learning works fine in case we have a tabular environment:





How to use this idea for playing Atari games?

How to use this idea for playing Atari games?

- What are the actions?

How to use this idea for playing Atari games?

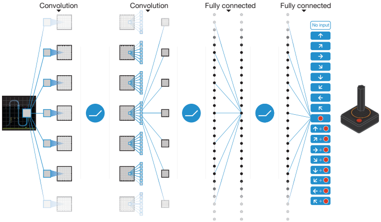
- What are the actions?
- What are the rewards?

How to use this idea for playing Atari games?

- What are the actions?
- What are the rewards?
- What are the states?

How about $Q_\theta(s, a)$ as a neural net which takes images as input and produces a number for each of the possible actions?

How about $Q_\theta(s, a)$ as a neural net which takes images as input and produces a number for each of the possible actions?



Deep Q-learning algorithm (Deep Q-networks (DQN)):

Given dataset $\mathcal{D} = \{(s_j, a_j, r_j, s_{j+1})\}$:

Deep Q-learning algorithm (Deep Q-networks (DQN)):

Given dataset $\mathcal{D} = \{(s_j, a_j, r_j, s_{j+1})\}$:

- Sample minibatch $\mathcal{B} \subseteq \mathcal{D}$

Deep Q-learning algorithm (Deep Q-networks (DQN)):

Given dataset $\mathcal{D} = \{(s_j, a_j, r_j, s_{j+1})\}$:

- Sample minibatch $\mathcal{B} \subseteq \mathcal{D}$
- Compute target $y_j = r_j + \gamma \max_a Q_{\theta^-}(s_{j+1}, a) \quad \forall j \in \mathcal{B}$

Deep Q-learning algorithm (Deep Q-networks (DQN)):

Given dataset $\mathcal{D} = \{(s_j, a_j, r_j, s_{j+1})\}$:

- Sample minibatch $\mathcal{B} \subseteq \mathcal{D}$
- Compute target $y_j = r_j + \gamma \max_a Q_{\theta^-}(s_{j+1}, a) \quad \forall j \in \mathcal{B}$
- Use stochastic (semi-)gradient descent to optimize w.r.t. parameters θ

$$\min_\theta \sum_{(s_j, a_j, r_j, s_{j+1}) \in \mathcal{B}} \left(Q_\theta(s_j, a_j) - y_j\right)^2$$

Deep Q-learning algorithm (Deep Q-networks (DQN)):

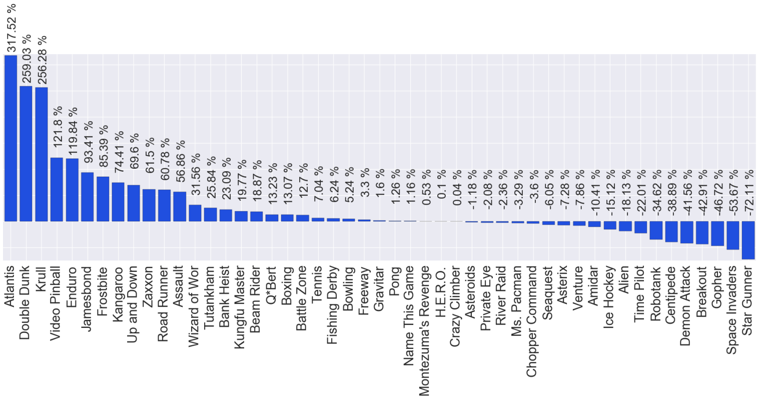
Given dataset $\mathcal{D} = \{(s_j, a_j, r_j, s_{j+1})\}$:

- Sample minibatch $\mathcal{B} \subseteq \mathcal{D}$
- Compute target $y_j = r_j + \gamma \max_a Q_{\theta^-}(s_{j+1}, a) \quad \forall j \in \mathcal{B}$
- Use stochastic (semi-)gradient descent to optimize w.r.t. parameters θ

$$\min_\theta \sum_{(s_j, a_j, r_j, s_{j+1}) \in \mathcal{B}} \left(Q_\theta(s_j, a_j) - y_j\right)^2$$

- Perform ϵ-greedy action and augment $\mathcal{D}$

Results:



317.52 %
259.03 %
256.28 %
121.8 %
119.84 %
93.41 %
85.39 %
74.41 %
69.6 %
61.5 %
60.78 %
56.86 %
31.56 %
25.84 %
23.09 %
19.77 %
18.87 %
13.23 %
13.07 %
12.7 %
7.04 %
6.24 %
5.24 %
3.3 %
1.6 %
1.26 %
1.16 %
0.53 %
0.1 %
0.04 %
-1.18 %
-2.08 %
-2.36 %
-3.29 %
-3.6 %
-6.05 %
-7.28 %
-7.86 %
-10.41 %
-15.12 %
-18.13 %
-22.01 %
-34.62 %
-38.89 %
-41.56 %
-42.91 %
-46.72 %
-53.67 %
-72.11 %
Atlantis
Double Dunk
Krull
Video Pinball
Enduro
Jamesbond
Frostbite
Kangaroo
Up and Down
Zaxxon
Road Runner
Assault
Wizard of Wor
Tutankham
Bank Heist
Kungfu Master
Beam Rider
Q*Bert
Boxing
Battle Zone
Tennis
Fishing Derby
Bowling
Freeway
Gravitar
Pong
Name This Game
Montezuma's Revenge
H.E.R.O.
Crazy Climber
Asteroids
Private Eye
River Raid
Ms. Pacman
Chopper Command
Seaquest
Asterix
Venture
Amidar
Ice Hockey
Alien
Time Pilot
Robotank
Centipede
Demon Attack
Breakout
Gopher
Space Invaders
Star Gunner

Quiz:

Quiz:

- What differentiates RL from supervised learning?

Quiz:

- What differentiates RL from supervised learning?
- What is a MDP?

Quiz:

- What differentiates RL from supervised learning?
- What is a MDP?
- What to do if no transition probabilities are available?

Important topics of this lecture

Important topics of this lecture

- Getting a feeling for reinforcement learning

Important topics of this lecture

- Getting a feeling for reinforcement learning
- Understanding how to use reinforcement learning

Important topics of this lecture

- Getting a feeling for reinforcement learning
- Understanding how to use reinforcement learning

What's next:

Policy Gradient Methods