香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

COMP7015 Artificial Intelligence

Lecture 11: Markov Decision Process & Reinforcement Learning

Instructor: Dr. Kejing Yin

November 24, 2022

Logistics

- Programming Assignment 2 *Due: 23:59 pm on Nov. 27*
- Written Assignment 2 *Due: 23:59 pm on Nov. 30*
  (submission cut-off at 1 pm on Dec. 1)
- Remaining Office Hours: Nov. 28, and Dec. 5
- Group Project: Final deadline is approaching (Dec. 3). Submit early!

Supervised Learning

- Given data: (x, y), x is data, y is label.
- Goal: Learn a function to map $x \rightarrow y$, namely posterior probability

$$P(Y|X = x)$$

→ cat

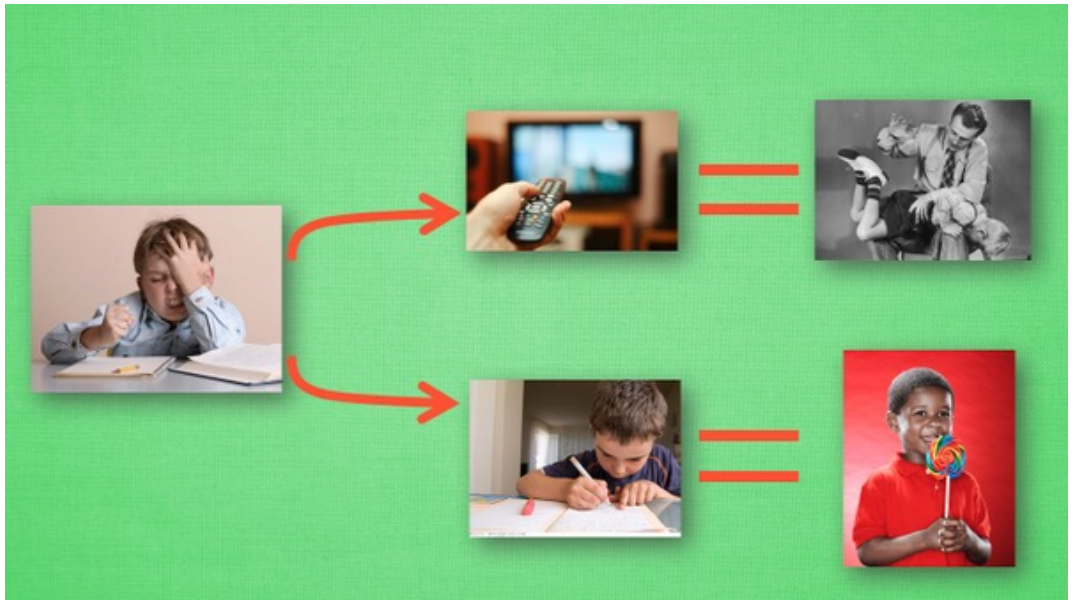
- Examples: Classification, regression, object detection, face recognition, sentiment classification, etc.

→ car

Image source: https://www.autoexpress.co.uk/tesla/model-3
https://timesofindia.indiatimes.com/life-style/relationships/pets/5-things-that-scare-and-stress-your-cat/articleshow/67586673.cms

Reinforcement Learning

- When we are kids, nobody gives us training data for good behaviors and bad behaviors.
- We learn by trial and error with feedbacks from parents and teachers.



Negative feedback

Positive feedback

Image source: https://mofanpy.com/tutorials/machine-learning/reinforcement-learning/intro-q-learning/

Reinforcement Learning

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal.
- Feedback is delayed, not instantaneous.
- Time really matters (sequential, non i.i.d data).
- Agent's actions affect the subsequent data it receives.

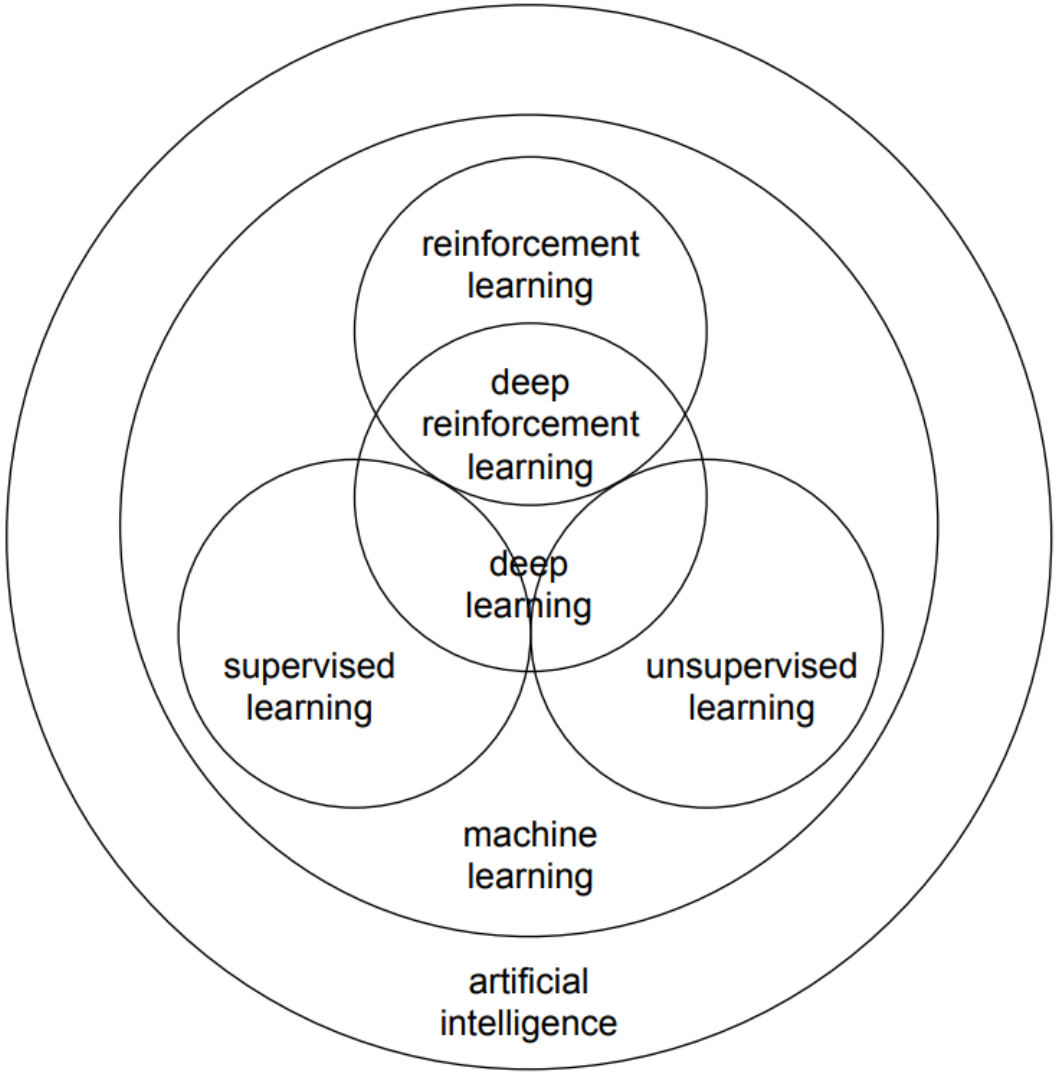
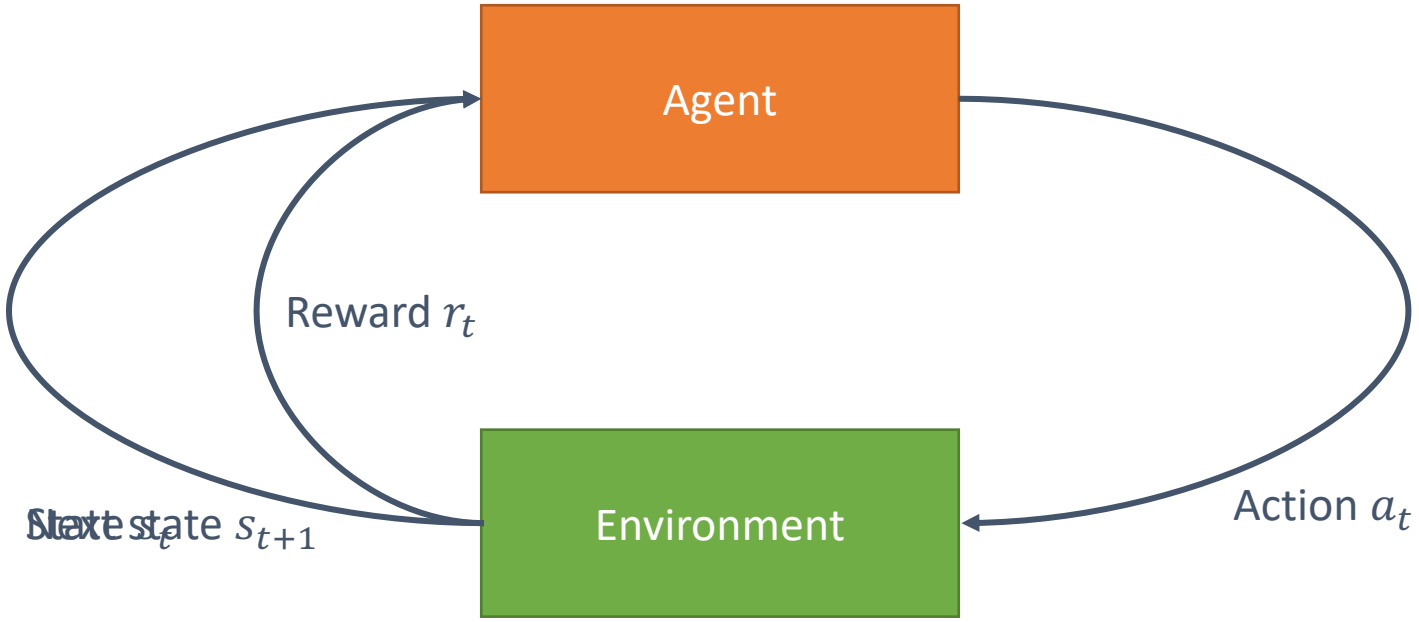


Image source: Li, Yuxi. "Deep reinforcement learning." arXiv preprint arXiv:1810.06339 (2018).

Reinforcement Learning



Reinforcement Learning

Atari Games

- **Objective**: Complete the game with the highest score.
- **State:** Raw pixel inputs of the game state.
- **Action:** Game controls e.g. Left, Right, Up, Down.
- **Reward:** Score increase/decrease at each time step.

Image source: https://syncedreview.com/2020/01/08/slm-lab-new-rl-research-benchmark-software-framework/

Reinforcement Learning

Go

- **Objective**: Win the game.
- **State**: Position of all pieces.
- **Action**: Where to put the next piece down.
- **Reward**: 1 if win at the end of the game, 0 otherwise.

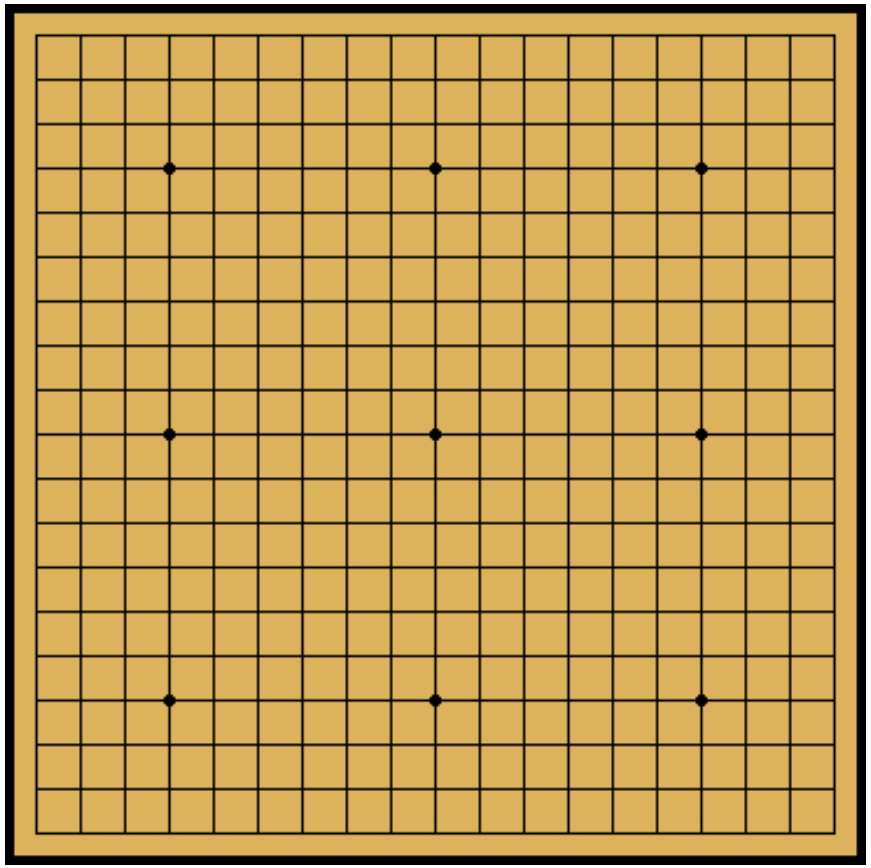
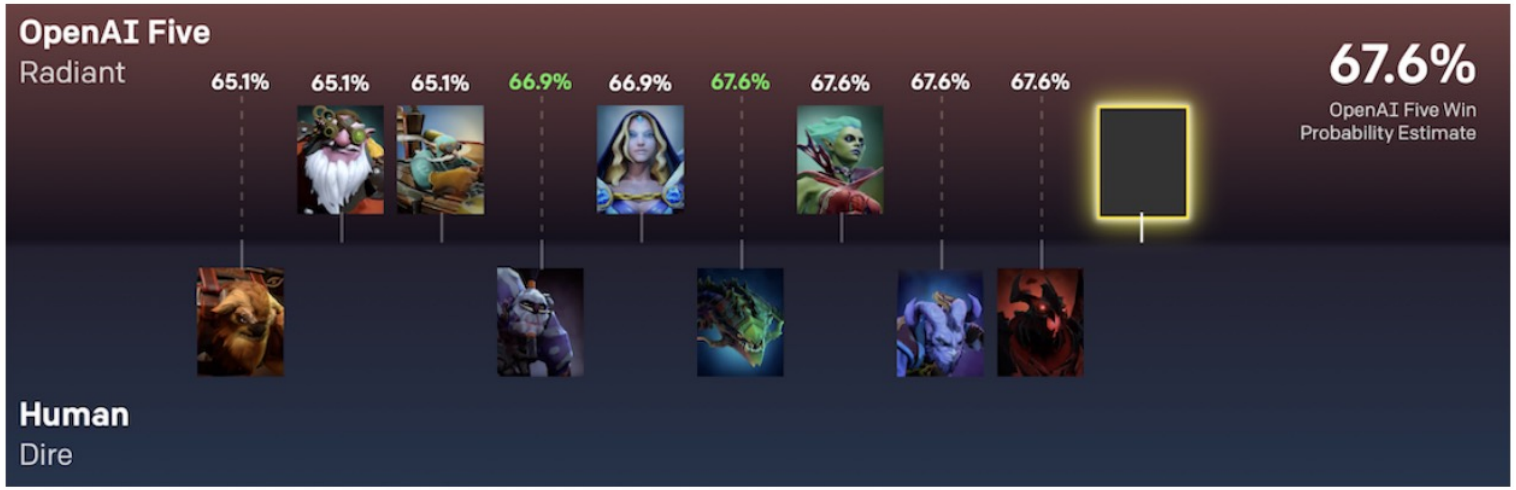


Image source: https://levelup.gitconnected.com/how-deepminds-alphago-became-the-world-s-top-go-player-5b275e553d6a?gi=766cb4dda780

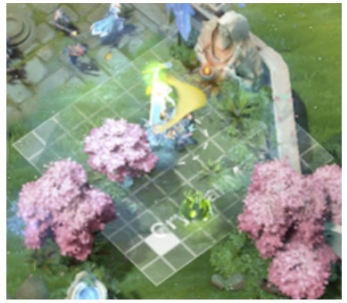
Applications of Reinforcement Learning

- Play game



(a)

(b)



Source: Berner, Christopher, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi et al. "Dota 2 with large scale deep reinforcement learning." arXiv preprint arXiv:1912.06680 (2019).

Applications of Reinforcement Learning

- Play game

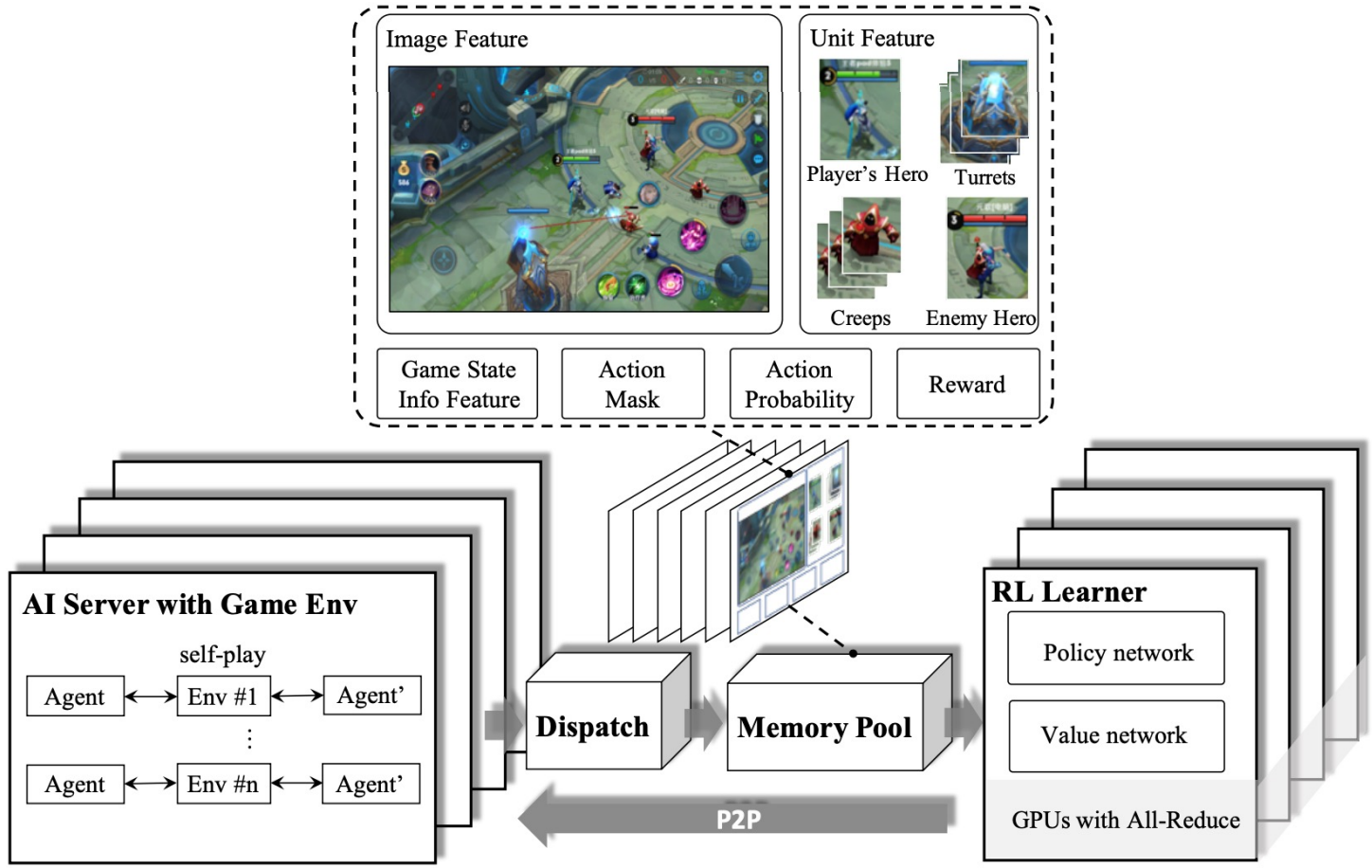


Image source: Ye, Deheng, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu et al. "Mastering Complex Control in MOBA Games with Deep Reinforcement Learning." In AAAI, pp. 6672-6679. 2020.

Applications of Reinforcement Learning

- AlphaGo [DeepMind, Nature 2016]:
  - Required many engineering tricks
  - Bootstrapped from human play
  - Beat 18-time world champion Lee Sedol
- AlphaGo Zero [Nature 2017]:
  - Simplified and elegant version of AlphaGo
  - No longer bootstrapped from human play
  - Beat (at the time) #1 world ranked Ke Jie
- Alpha Zero (Dec. 2017)
  - Generalized to beat world champion programs on chess and shogi as well
- MuZero (Nov. 2019)
  - Plans through a learned model of the game

Image source: https://becominghuman.ai/summary-of-the-alphago-paper-b55ce24d8a7c

Applications of Reinforcement Learning

- Recommander system

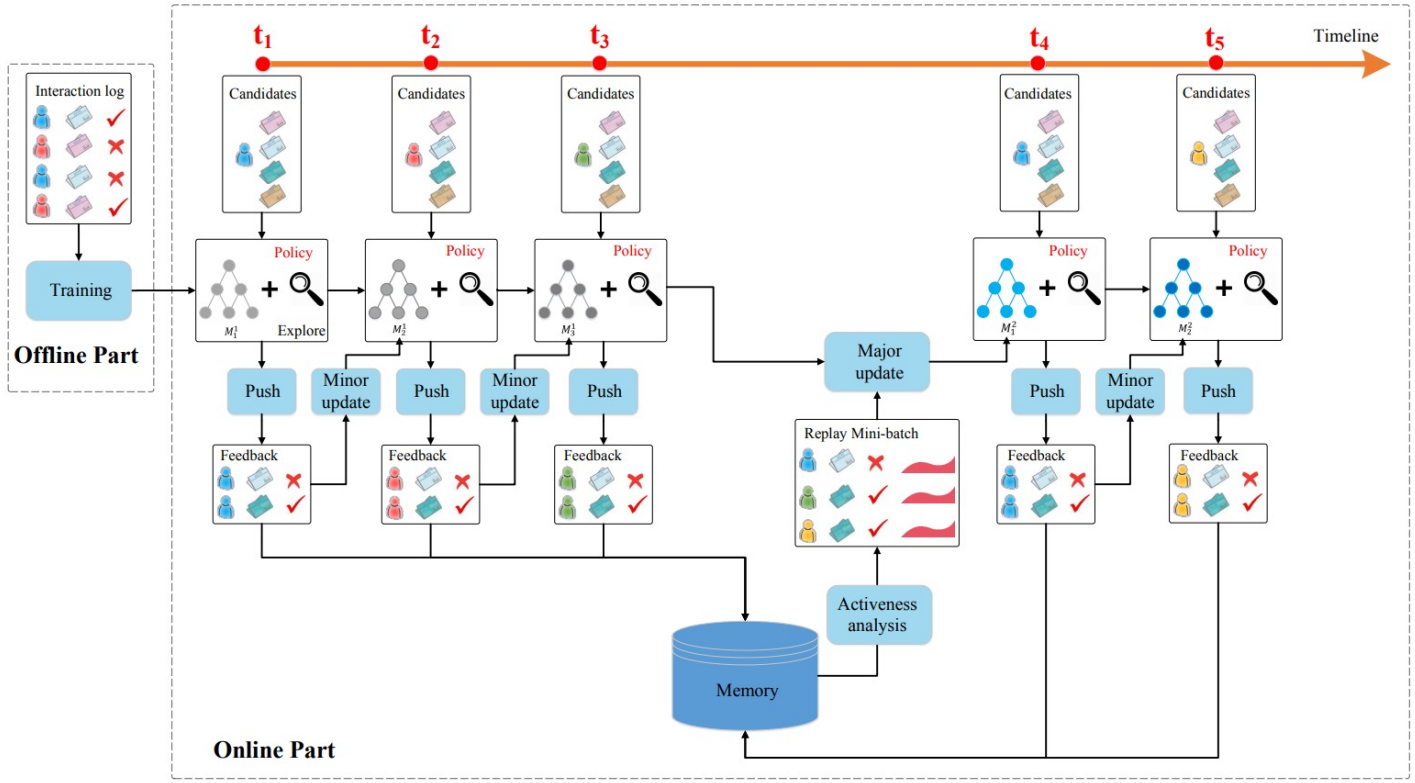


Image source: Zheng, Guanjie, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. "DRN: A deep reinforcement learning framework for news recommendation." In Proceedings of the 2018 World Wide Web Conference, pp. 167-176. 2018.

Applications of Reinforcement Learning

- Financial trading

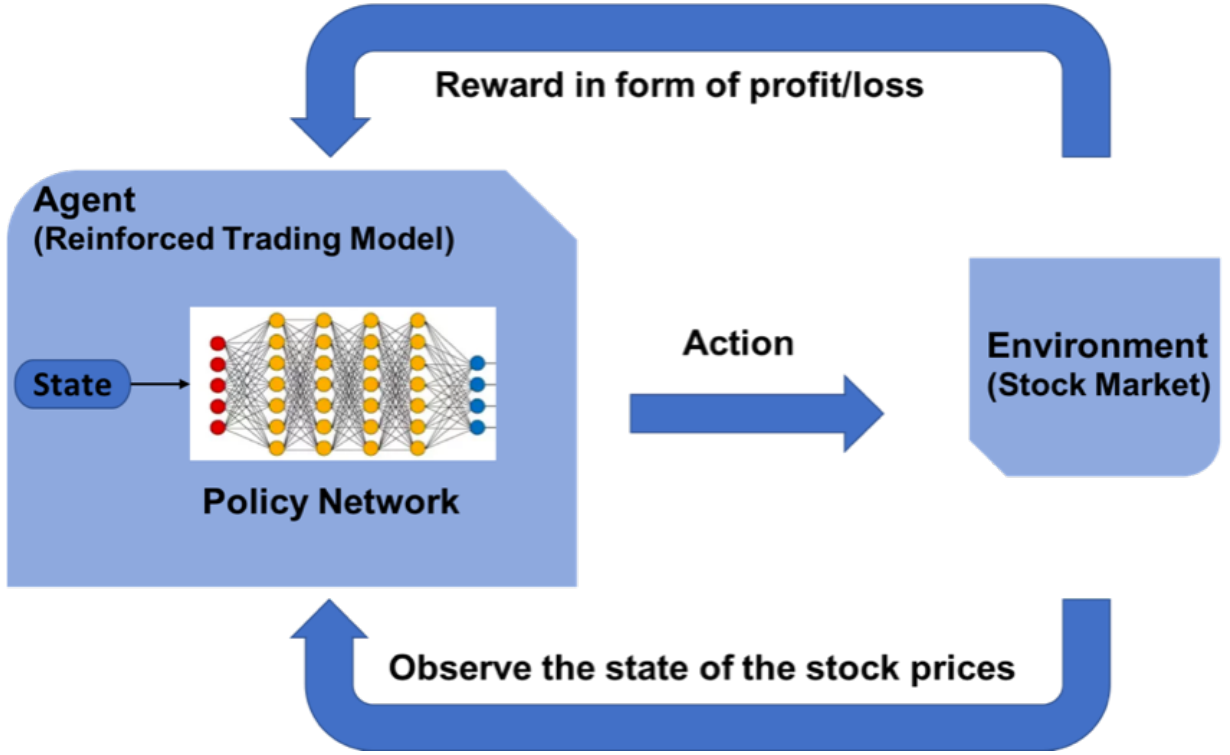
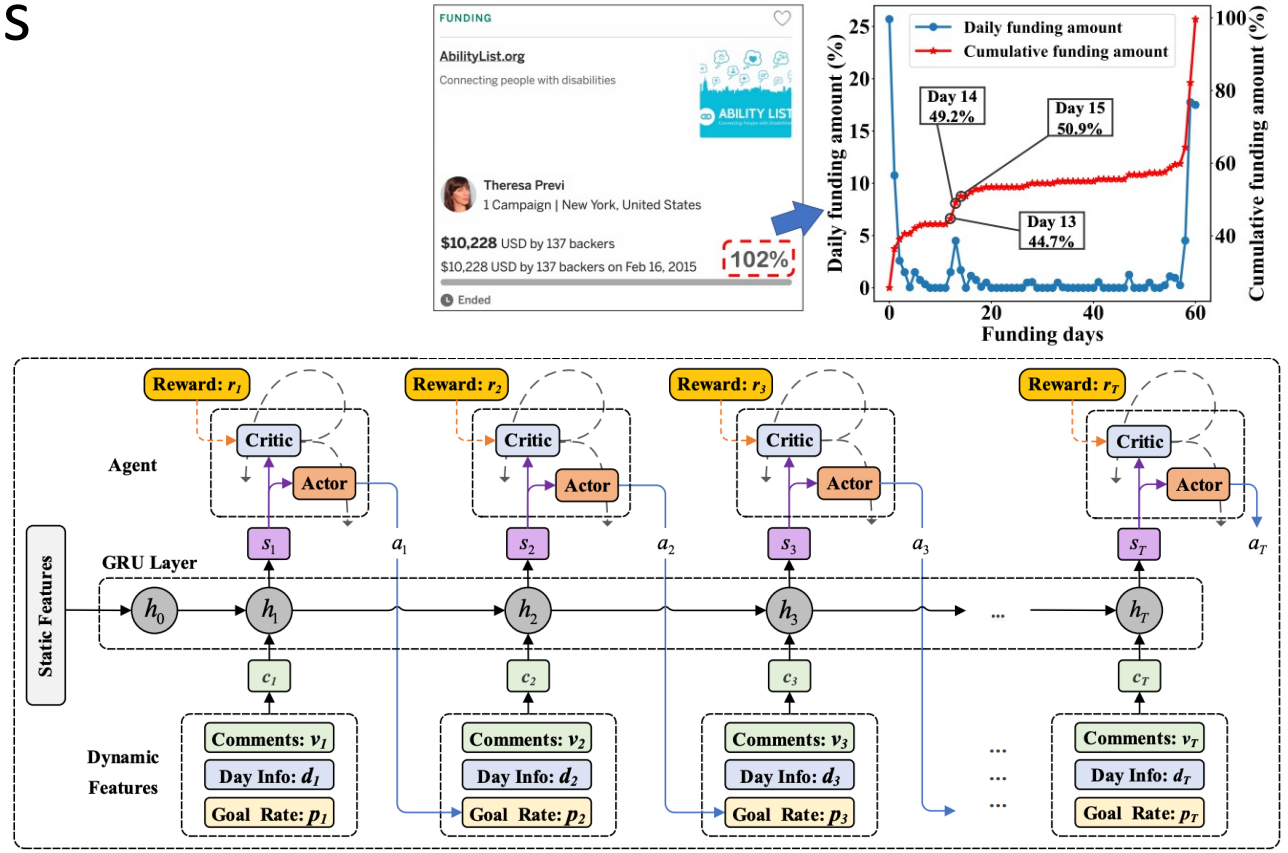


Image source: https://medium.com/inside-machine-learning/reinforcement-learning-the-business-use-case-part-2-c175740999

Applications of Reinforcement Learning

- Business analysis

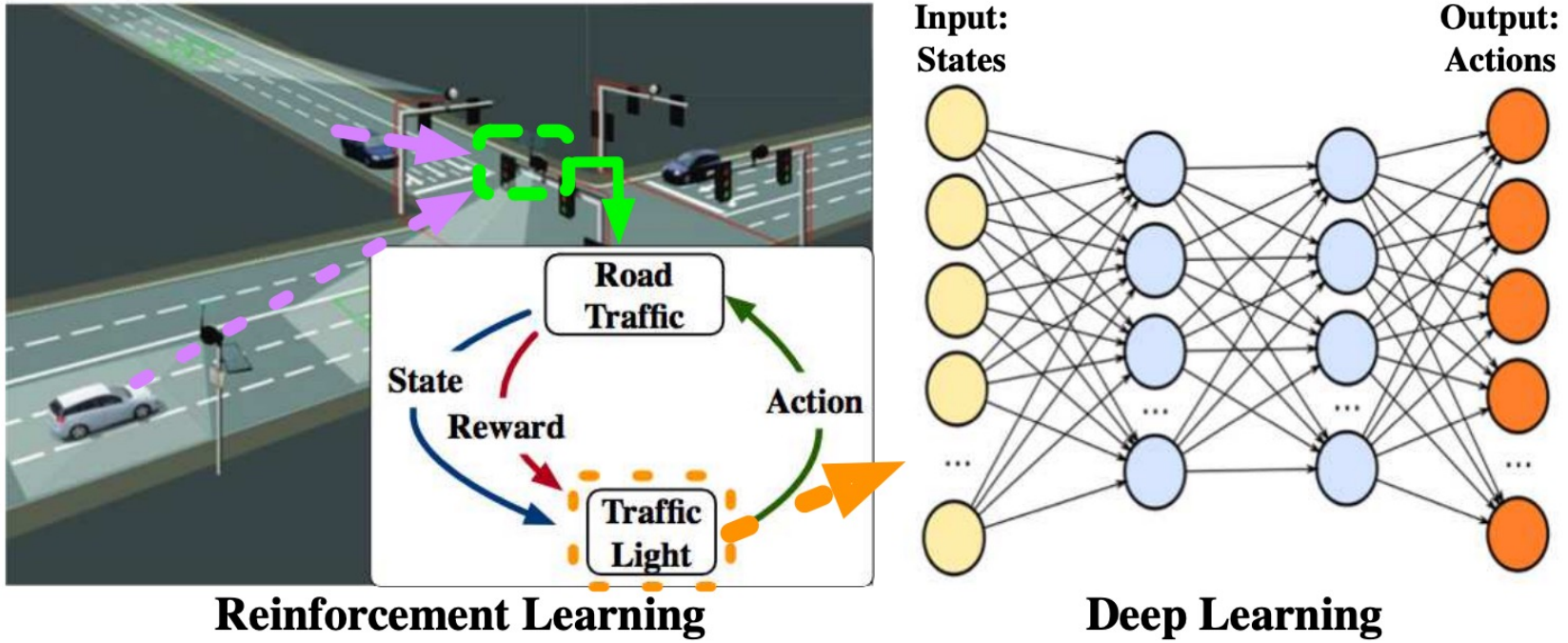


Crowdfunding dynamics tracking

Image source: Wang, Jun, Hefu Zhang, Qi Liu, Zhen Pan, and Hanqing Tao. "Crowdfunding Dynamics Tracking: A Reinforcement Learning Approach." In AAAI, pp. 6210-6218. 2020.

Applications of Reinforcement Learning

- Smart transportation



Traffic light control model

Image source: Liang, Xiaoyuan, Xunsheng Du, Guiling Wang, and Zhu Han. "Deep reinforcement learning for traffic light control in vehicular networks." arXiv preprint arXiv:1803.11115 (2018).

Applications of Reinforcement Learning

- Healthcare

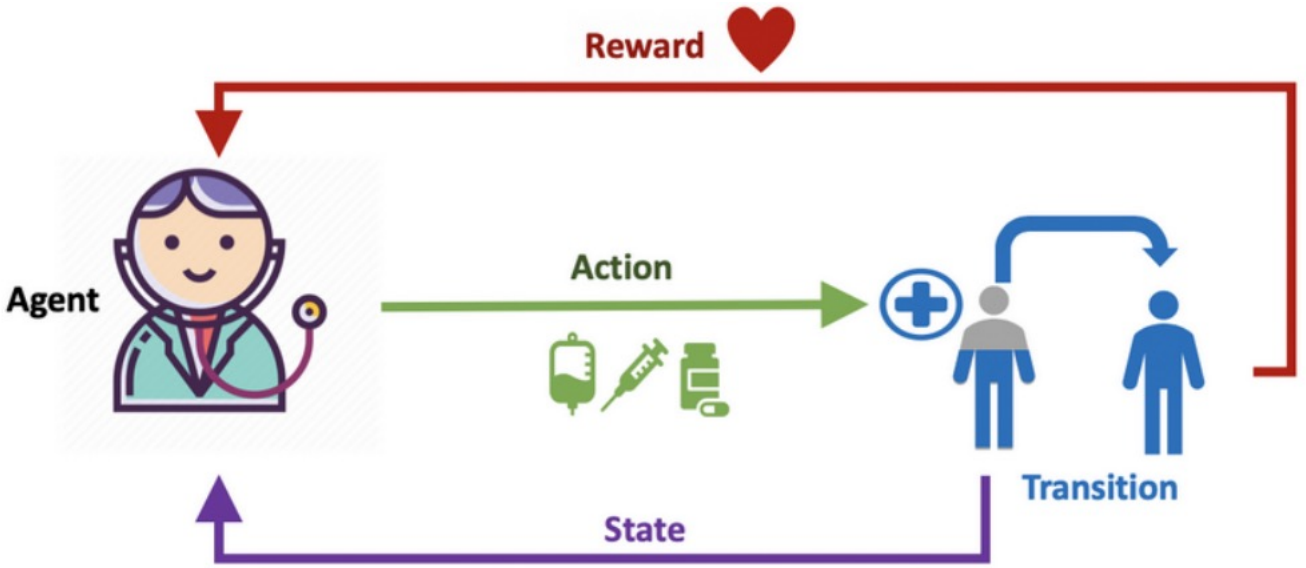


Image source: Liu, Siqi, Kay Choong See, Kee Yuan Ngiam, Leo Anthony Celi, Xingzhi Sun, and Mengling Feng. "Reinforcement learning for clinical decision support in critical care: comprehensive review." Journal of medical Internet research 22, no. 7 (2020): e18477.

Applications of Reinforcement Learning

- Robotics

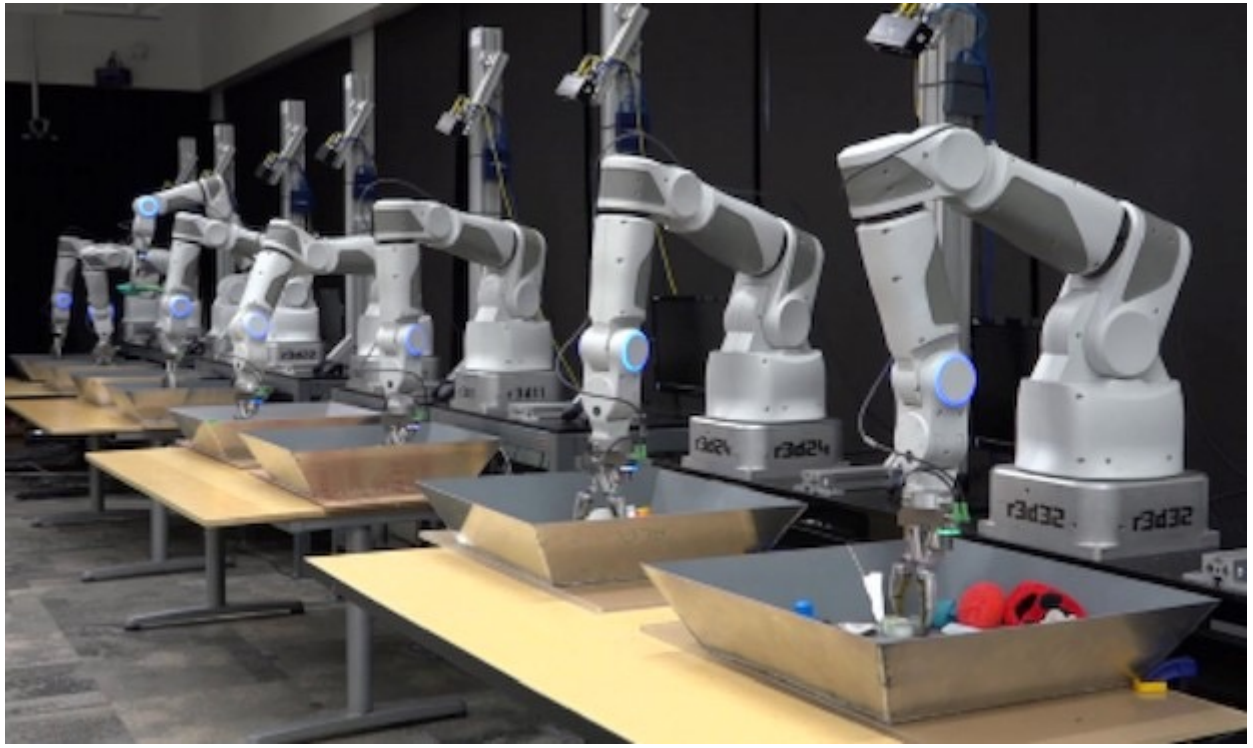


Image source: https://blogs.sw.siemens.com/tecnomatix/part-2-how-digital-twins-help-scale-up-industrial-robotics-ai/

Applications of Reinforcement Learning

- Network architecture search

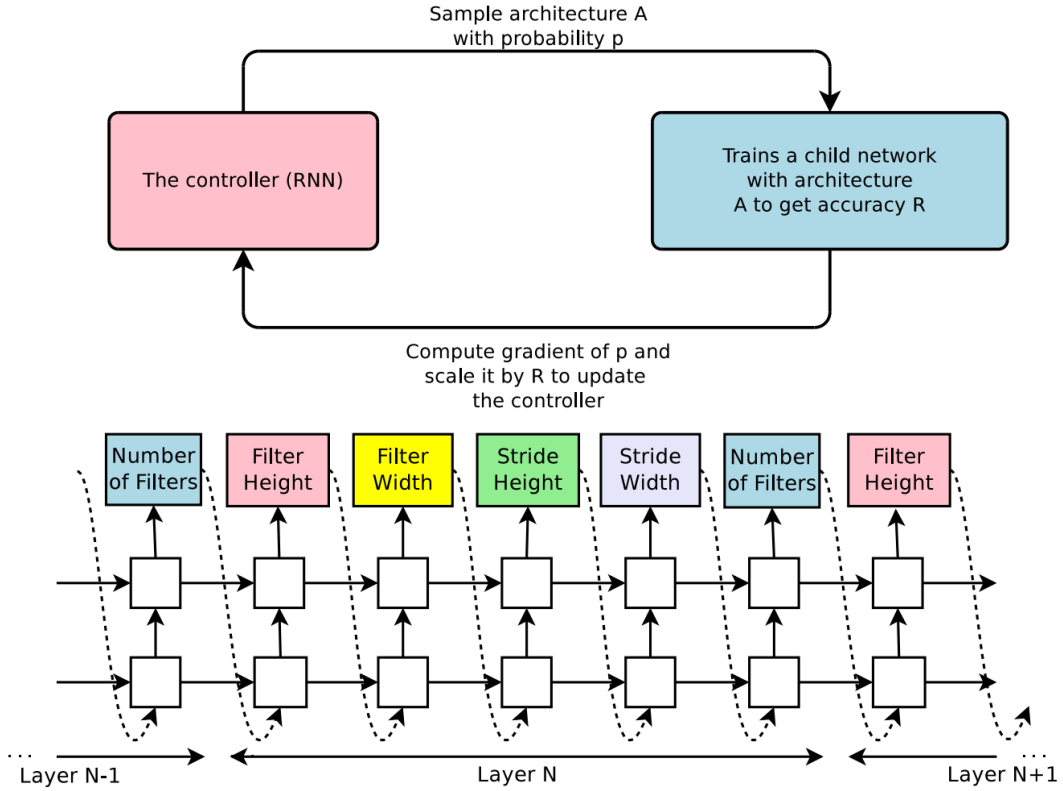
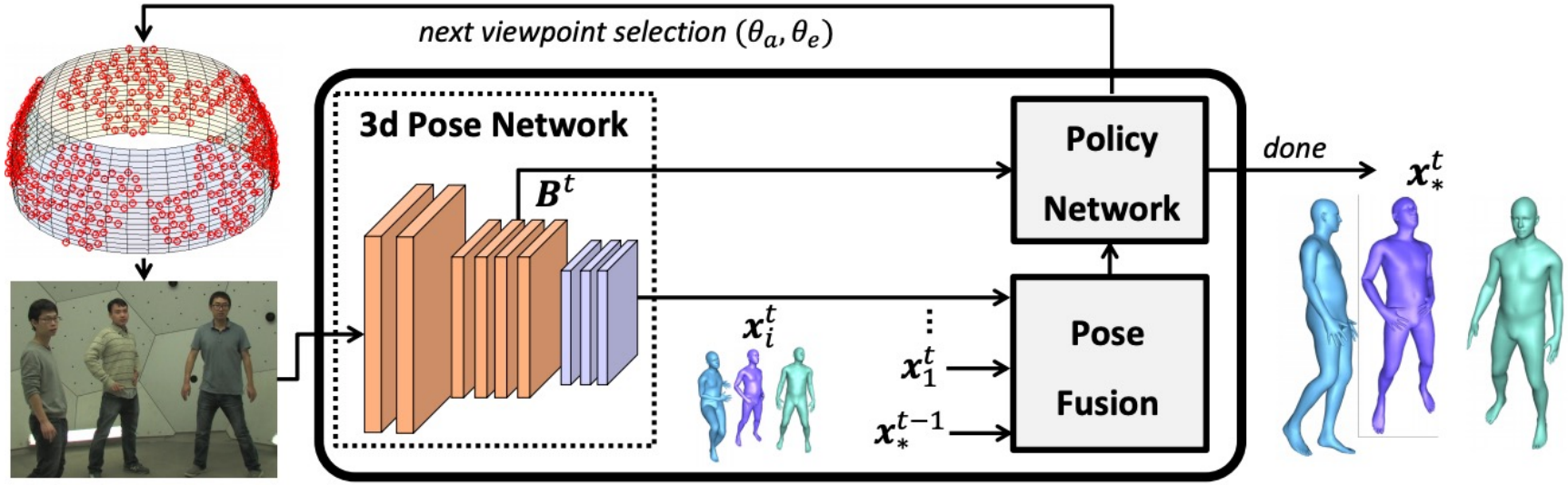


Image source: Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).

Applications of Reinforcement Learning

- CV



Active human pose estimation

Image source: Gärtner, Erik, Aleksis Pirinen, and Cristian Sminchisescu. "Deep Reinforcement Learning for Active Human Pose Estimation." In AAAI, pp. 10835-10844. 2020.

Applications of Reinforcement Learning

- Energy management

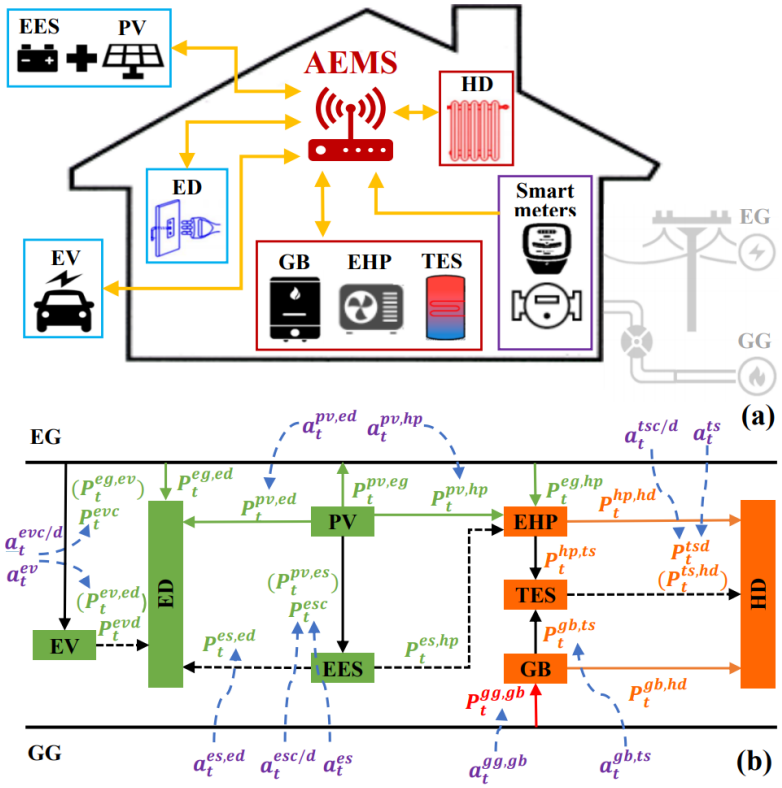


Image source: Ye, Yujian, Dawei Qiu, Jonathan Ward, and Marcin Abram. “Model-Free Real-Time Autonomous Energy Management for a Residential Multi-Carrier Energy System: A Deep Reinforcement Learning Approach.” IJCAI, 2020.

Outline

- Markov Decision Processes
- Policy Evaluation
- Value Iteration
- Reinforcement Learning
- Monte Carlo Methods
- Q-Learning & Deep Q-Learning
- ϵ-greedy Algorithm

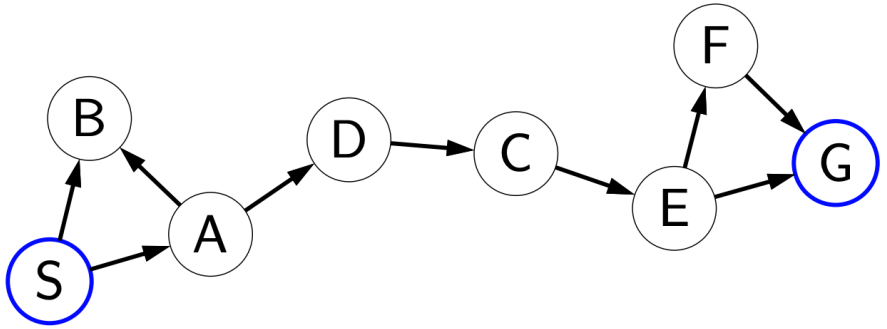
Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Outline

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation
- Value Iteration
- Reinforcement Learning
- Monte Carlo Methods
- Q-Learning & Deep Q-Learning
- ϵ-greedy Algorithm

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Recall the search problems …

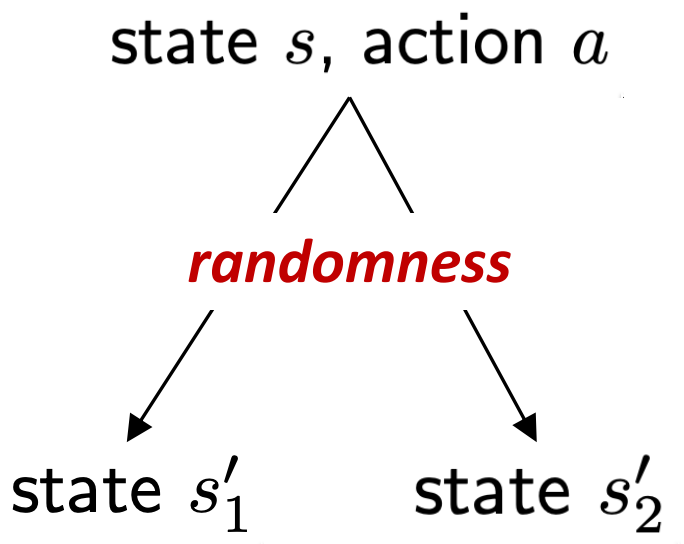


deterministic

state s, action a ⟶ state $\text{Succ}(s, a)$

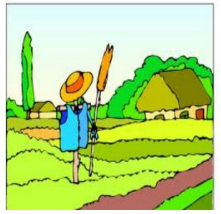
Taking an action from a state results deterministically in a unique successor state

In real world, uncertainty is unavoidable



Robotics: decide where to move, but actuators can fail, hit unseen obstacles, etc.

Resource allocation: decide what to produce, don't know the customer demand for various products

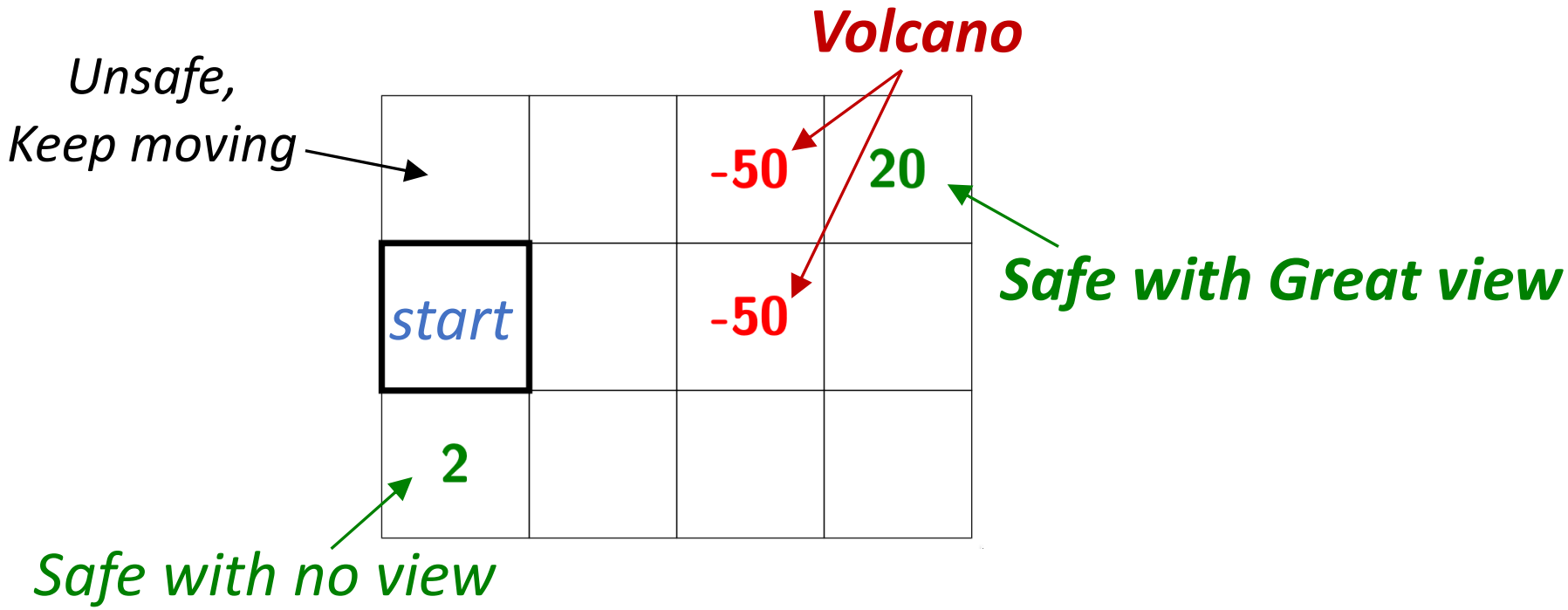


Agriculture: decide what to plant, but don't know weather and thus crop yield

Taking an action might lead to any one of many possible states

Example: Crossing the volcano

You are exploring a South Pacific island, which is modeled as a 3x4 grid of states.



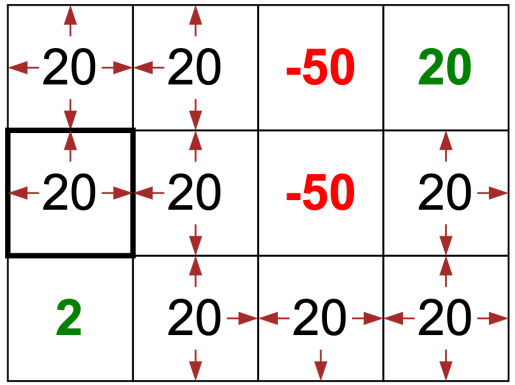
Due to terrains, there is on each action a probability of slipping, p(slip), which results in moving in a random direction.

Example: Crossing the volcano

Arrows: best actions
Value: the reward (we will define later)

p(slip) = 0

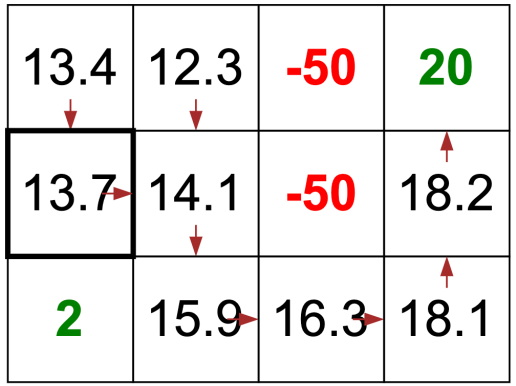
(i.e., a deterministic setting)



Value: 20

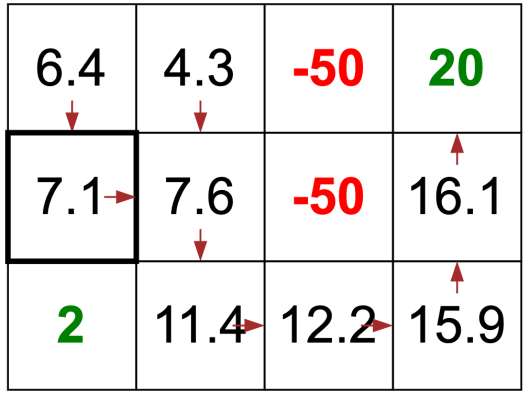
Go for the view!

p(slip) = 0.1



Value: 13.68

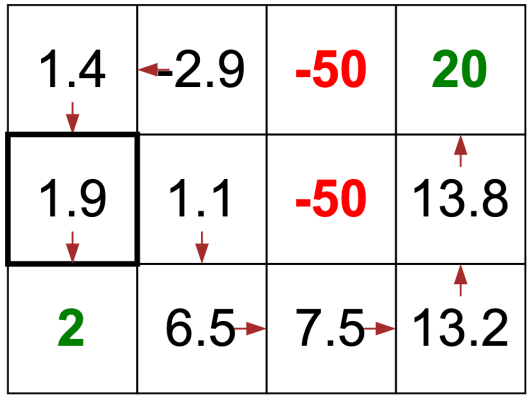
p(slip) = 0.2



Value: 7.07

Still worth it.

p(slip) = 0.3



Value: 1.86

Play safe …

We only specify the dynamics of the world, not directly specifying the best action to take.

Today's focus: How to compute the best actions automatically using algorithms?

Example: Dice game

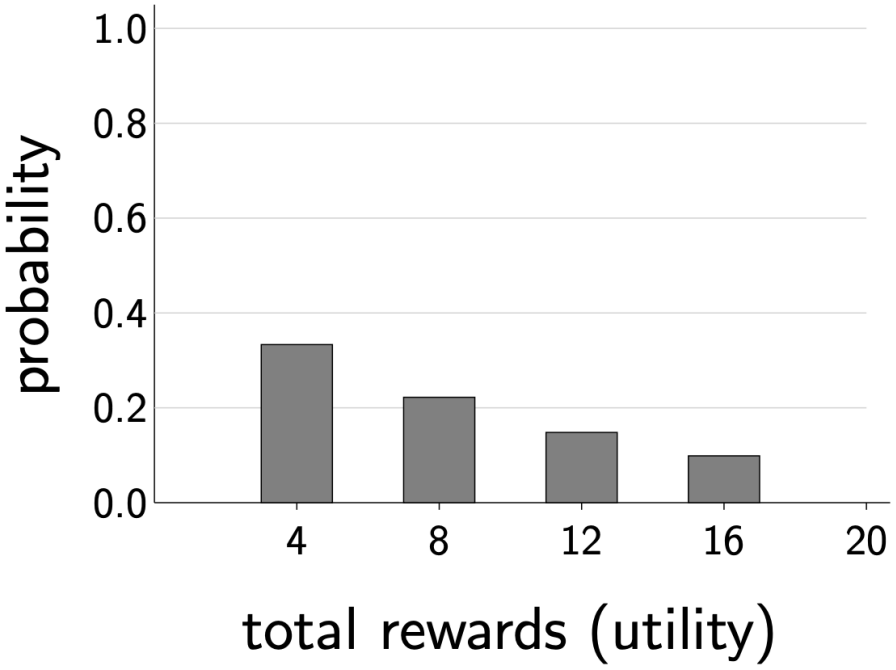
Let's start with a simpler example

For each round $r = 1, 2, \ldots$

- You choose stay or quit.
- If quit, you get $\$10$ and we end the game.
- If stay, you get $\$4$ and then I roll a 6-sided dice.
  - If the dice results in 1 or 2, we end the game.
  - Otherwise, continue to the next round.

Example: Dice game

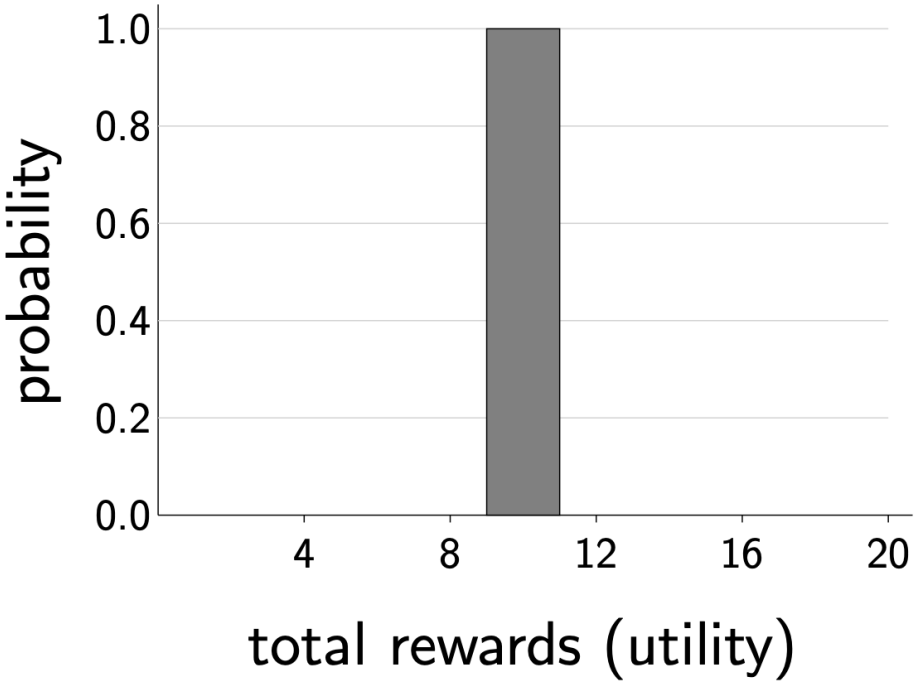
Suppose you always follow the policy “stay”:



Expected utility: $\frac{1}{3}(4) + \frac{2}{3} \cdot \frac{1}{3}(8) + \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}(12) + \cdots = 12$

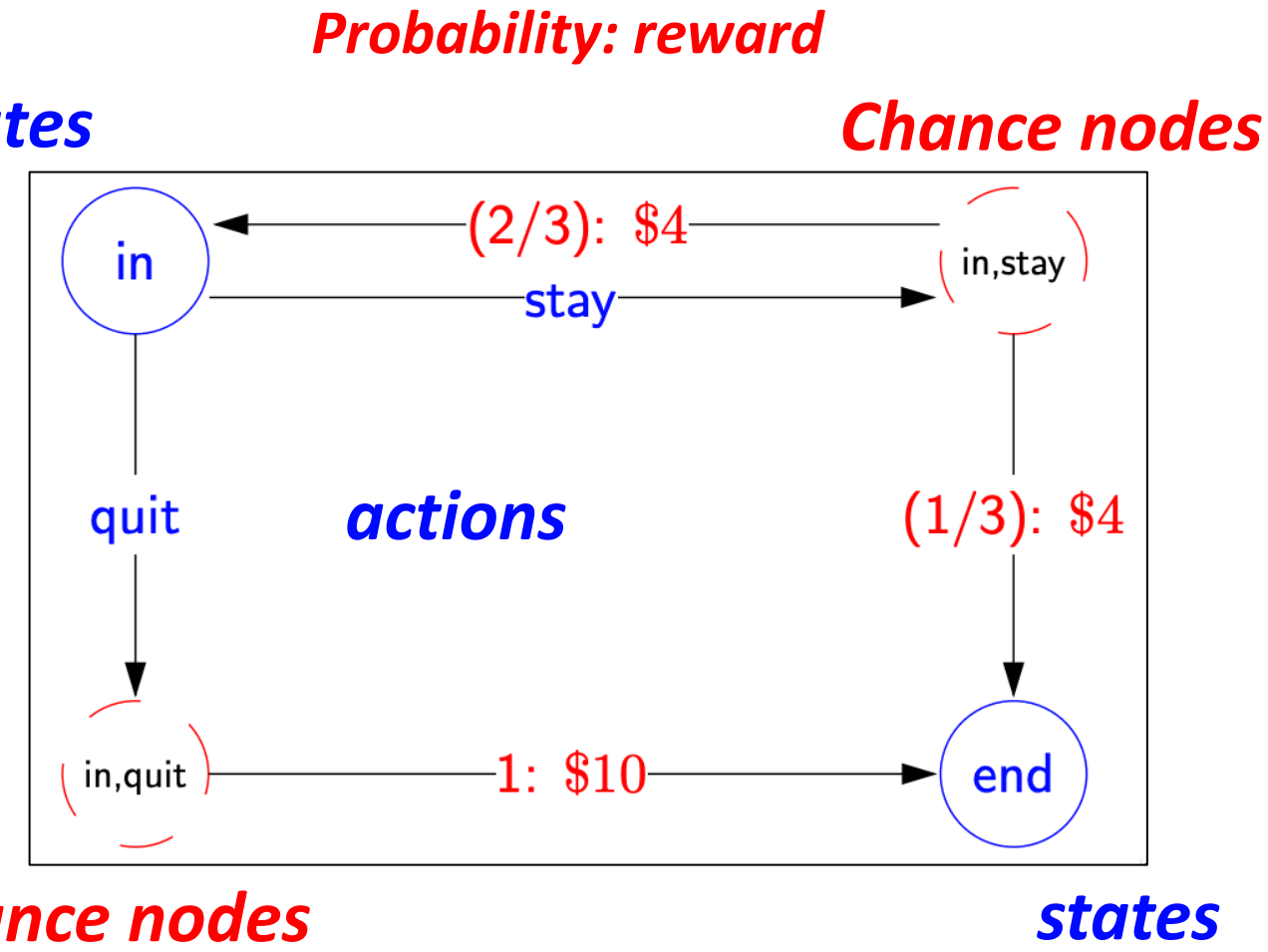
Example: Dice game

Suppose you always follow the policy “quit”:



Expected utility: $1(10) = 10$

A Markov Decision Process



- Edges coming out of states are the possible actions from that state, which lead to chance nodes.
- Edges coming out of a chance nodes are the possible random outcomes of that action, which end up back in states.
- We Label the chance-to-state edges with the transition probability and the associated reward.

A Markov Decision Process

- States: the set of states.
- $s_{\text{start}} \in \text{States}$: starting state.
- Actions(s): Possible actions from state s.
- Transition function $T(s, a, s')$: Probability of reaching s' if take action a in state s.
- Reward(s, a, s'): reward for the transition (s, a, s').
- IsEnd(s): check whether at the end of a game.
- $0 \leq \gamma \leq 1$: discount factor (default: 1).

A Markov Decision Process

Transition function $T(s, a, s')$:

Probability of reaching s' if take action a in state s.

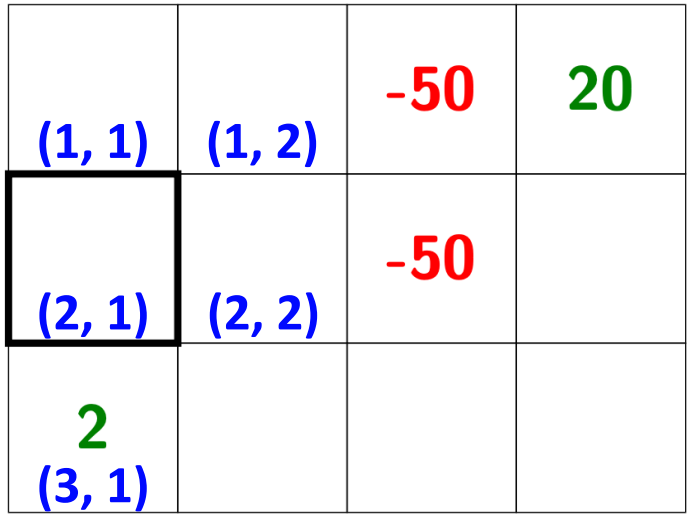
Example of the dice game:

s	a	s'	$T(s, a, s')$
in	quit	end	1
in	stay	in	$2/3$
in	stay	end	$1/3$

A Markov Decision Process

Transition function $T(s, a, s')$:
Probability of reaching s' if take action a in state s.

Example of the volcano crossing with p(slip)=0.2:



s	a	s'	$T(s, a, s')$
(2, 1)	north	(1, 1)	0.8+0.2/4
(2, 1)	north	(1, 2)	0
(2, 1)	north	(2, 1)	0.2/4
(2, 1)	north	(2, 2)	0.2/4
(2, 1)	north	(3, 1)	0.2/4

A Markov Decision Process

$T(s, a, s')$ is a probability, so it sum up to one:

$$\sum_{s' \in \text{States}} T(s, a, s') = 1$$

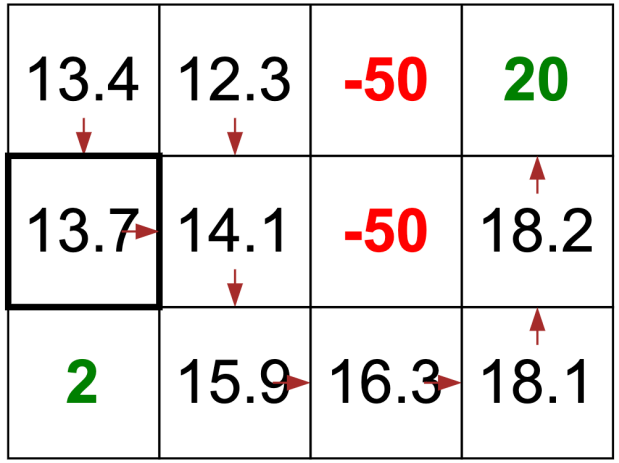
Successors: s' such that $T(s, a, s') > 0$

possible destinations from state s by taking action a

A Markov Decision Process

A policy π is a mapping from each state $s \in$ States to an action $a \in$ Actions(s).

Example:



s	$\pi(s)$
(1,1)	S
(2,1)	E
(3,1)	N

Markov property: the transition functions and rewards only depend on the current state s and the action a.

Outline

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation *Given a policy π, evaluate how good it is. $(MDP, \pi) \rightarrow V_\pi$*
- Value Iteration
- Reinforcement Learning
- Monte Carlo Methods
- Q-Learning & Deep Q-Learning
- ϵ-greedy Algorithm

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Policy Evaluation *(Given the policy π)*

Utility: The utility of a policy is the discounted sum of the rewards on a random path generated by running the policy.

Policy Evaluation *(Given the policy π)*

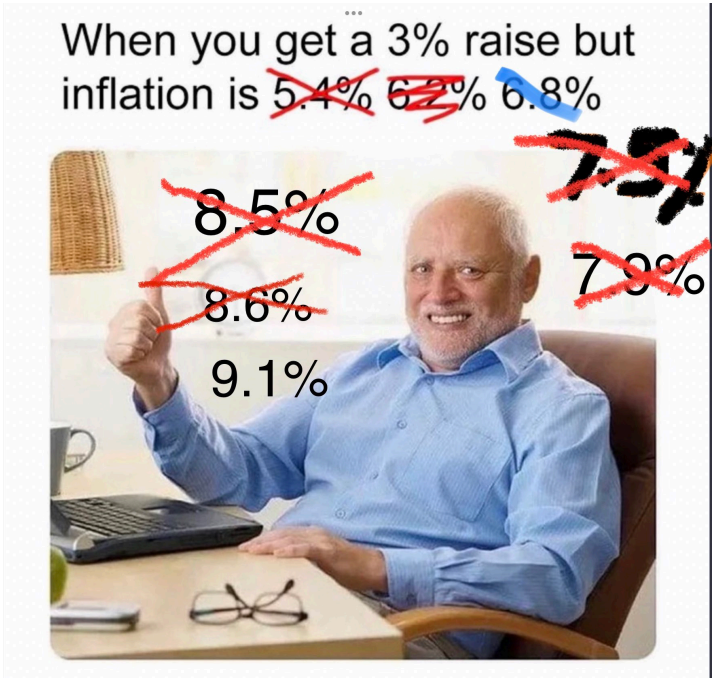
Discounting: a reward today might be worth more than the same reward in the future.

A random path: $s_0, a_1 r_1 s_1, a_2 r_2 s_2, \ldots$ (action, reward, state)

The discounted utility (γ: discount factor) is:

$$u_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots$$

A larger γ: the future is discounted less.



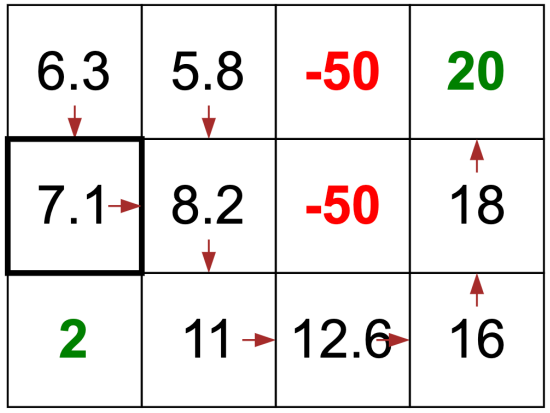
Policy Evaluation *(Given the policy π)*

Utility: The utility of a policy is the discounted sum of the rewards on a random path generated by running the policy.

Example in dice game (γ = 1):

Path	Utility
[in; stay, 4, end]	4
[in; stay, 4, in; stay, 4, in; stay, 4, end]	12
[in; stay, 4, in; stay, 4, end]	8
[in; stay, 4, in; stay, 4, in; stay, 4, in; stay, 4, end]	16
…	…

Example in volcano crossing (γ = 0.9):



a	r	s
		(2,1)
E	0	(2,2)
S	0	(1,2)
S	0	(2,2)
S	0	(3,2)
E	0	(3,3)
E	0	(3,4)
N	0	(2,4)
N	20	(1,4)

Utility: 9.57

a	r	s
		(2,1)
E	0	(2,2)
S	0	(3,2)
E	0	(3,3)
E	0	(3,4)
N	0	(2,4)
N	20	(1,4)

Utility: 11.81

a	r	s
		(2,1)
E	0	(2,2)
S	0	(3,2)
E	0	(3,3)
E	-50	(2,3)

Utility: -36.45

Policy Evaluation *(Given the policy π)*

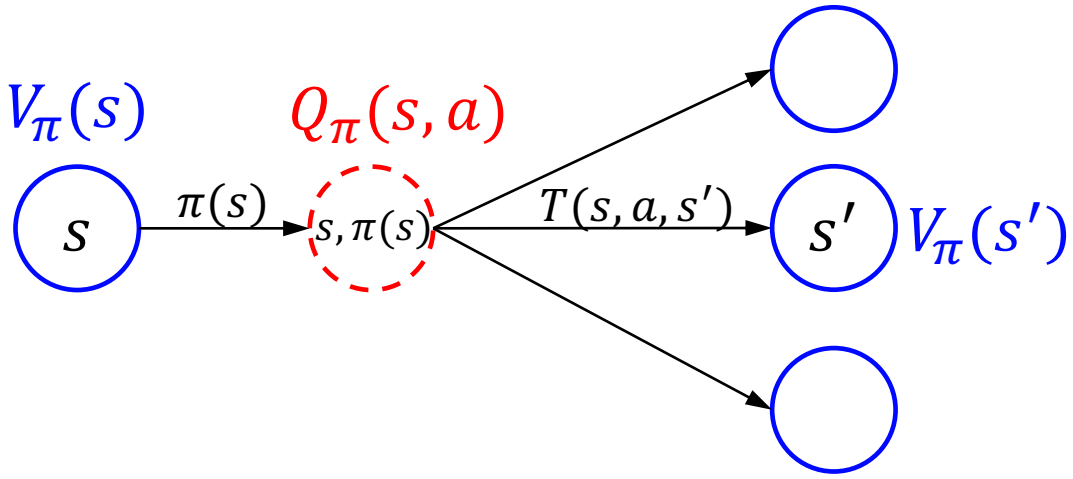
Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards.
- Avoids infinite returns in cyclic Markov processes.
- Uncertainty about the future may not be fully represented.
- If the reward is financial, immediate rewards may earn more interest than delayed rewards.
- Animal/human behavior shows preference for immediate reward.

Policy Evaluation *(Given the policy π)*

Value of a policy $V_\pi(s)$: The expected utility received by following policy π from state s.

Q-value of a policy $Q_\pi(s,a)$: The expected utility of taking action a at s, and then following the policy π.



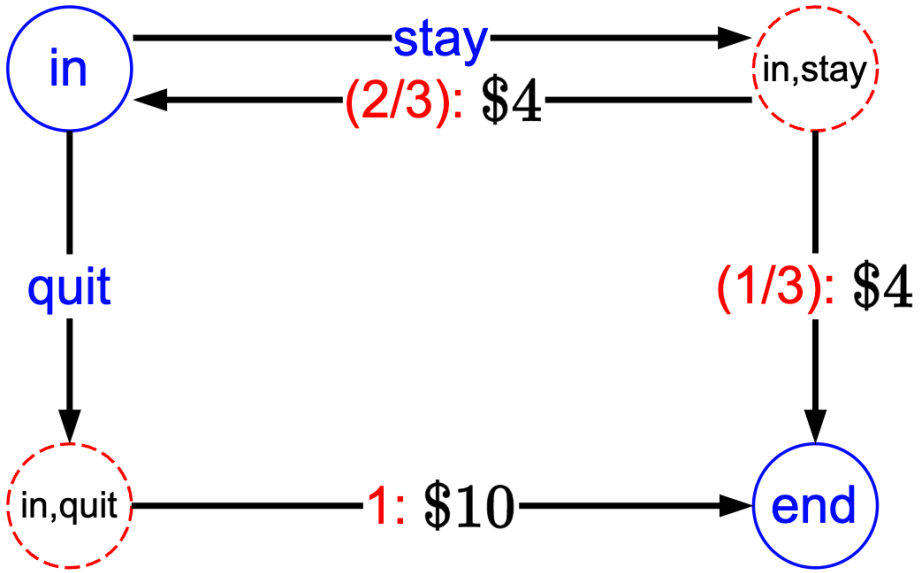
Recurrences relating value and Q-value

$$V_\pi(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ Q_\pi(s,\pi(s)) & \text{otherwise.} \end{cases}$$

$$Q_\pi(s,a) = \sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_\pi(s')]$$

Policy Evaluation *(Given the policy π)*

Example: Dice game



Let π be "always stay", *i.e.*, π(in)=stay

$$V_\pi(\text{end}) = 0$$

$$V_\pi(\text{in}) = Q_\pi(\text{in}, \text{stay})$$

$$V_\pi(\text{in}) = \tfrac{1}{3}\left(4 + V_\pi(\text{end})\right) + \tfrac{2}{3}\left(4 + V_\pi(\text{in})\right)$$

$$V_\pi(\text{in}) = \tfrac{1}{3}\,4 + \tfrac{2}{3}\left(4 + V_\pi(\text{in})\right)$$

$$V_\pi(\text{in}) = 4 + \tfrac{2}{3}\,V_\pi(\text{in})$$

$$V_\pi(\text{in}) = 12$$

Exercise: how about γ = 0.9?

Policy Evaluation *(Given the policy π)*

Iterative algorithms

- Large or complex MDP cannot be solved in closed form.
- Key idea: start with arbitrary value and iteratively apply the recurrence to converge to true values.
- Algorithm:

policy evaluation

Initialize $V_\pi^{(0)}(s) \leftarrow 0$ for all states s.

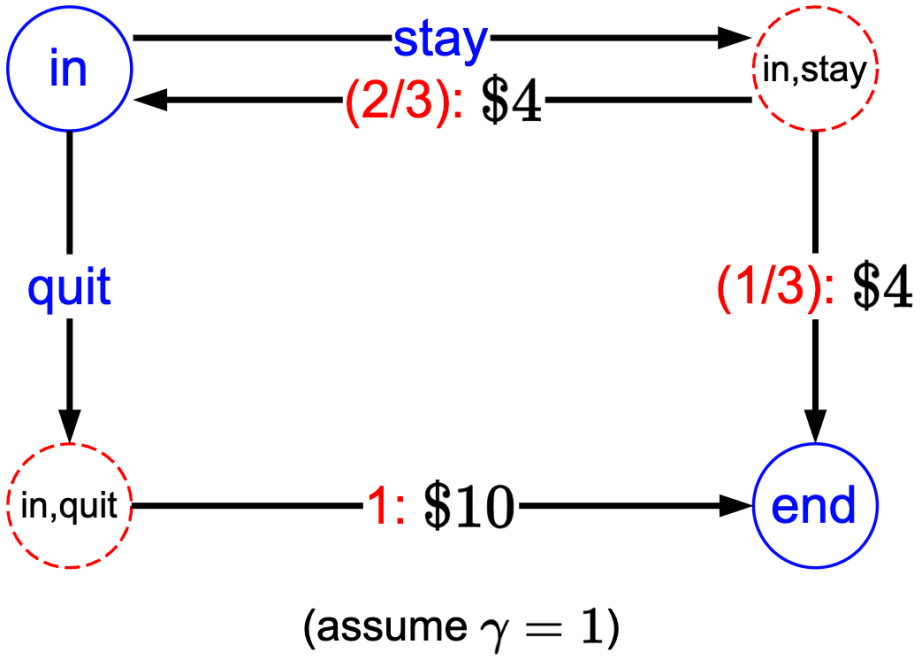
For iteration $t = 1, \ldots, t_{\text{PE}}$:

For each state s:

$$V_\pi^{(t)}(s) \leftarrow \underbrace{\sum_{s'} T(s, \pi(s), s')[\text{Reward}(s, \pi(s), s') + \gamma V_\pi^{(t-1)}(s')]}_{Q^{(t-1)}(s, \pi(s))}$$

Policy Evaluation *(Given the policy π)*

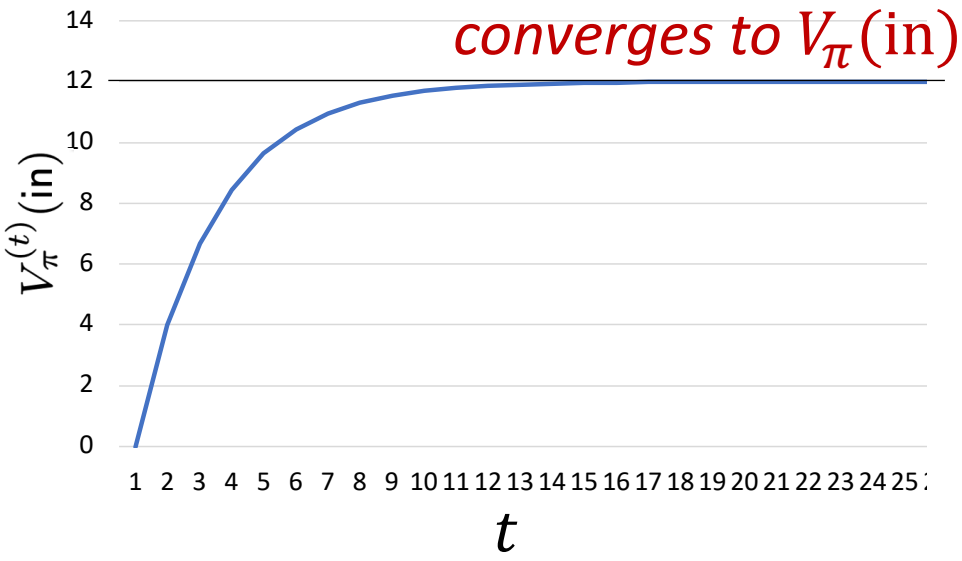
Example: Dice game



Let π be "always stay", *i.e.*, π(in)=stay

$$V_\pi^{(t)}(\text{end}) = 0$$

$$V_\pi^{(t)}(\text{in}) = \tfrac{1}{3}(4 + V_\pi^{(t-1)}(\text{end})) + \tfrac{2}{3}(4 + V_\pi^{(t-1)}(\text{in}))$$



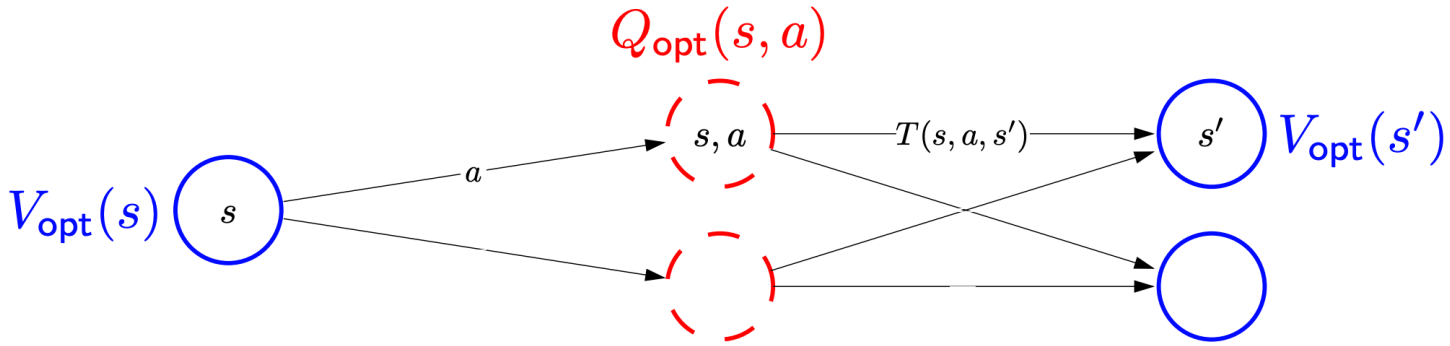
Outline

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation *Given a policy π, evaluate how good it is. $(MDP, \pi) \rightarrow V_\pi$*
- Value Iteration *Compute the optimal value and optimal policy. $MDP \rightarrow (V_{\text{opt}}, \pi_{\text{opt}})$*
- Reinforcement Learning
- Monte Carlo Methods
- Q-Learning & Deep Q-Learning
- ϵ-greedy Algorithm

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Optimal value and policy

Optimal Value $V_{\text{opt}}(s)$: The maximum value attained by any policy.



Optimal value if take action a in state s:

$$Q_{\text{opt}}(s,a) = \sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}(s')].$$

Optimal value from state s:

$$V_{\text{opt}}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ \max_{a \in \text{Actions}(s)} Q_{\text{opt}}(s,a) & \text{otherwise.} \end{cases}$$

Optimal policy:

$$\pi_{\text{opt}}(s) = \arg \max_{a \in \text{Actions}(s)} Q_{\text{opt}}(s,a)$$

Optimal value and policy

Similarly, this can also be solved iteratively:

Value Iteration

Initialize $V_{\text{opt}}^{(0)}(s) \leftarrow 0$ for all states s.

For iteration $t = 1, \ldots, t_{\text{VI}}$:

For each state s:

$$V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in \text{Actions}(s)} \underbrace{\sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}^{(t-1)}(s')]}_{Q_{\text{opt}}^{(t-1)}(s,a)}$$

Summary so far

- Markov Decision Processes
  - Specify the dynamics of the world (cope with the uncertainty).
  - Solutions are policies rather than paths.
- Policy Evaluation
  - Given a policy π, evaluate how good it is (compute the policy value).

  $(\text{MDP}, \pi) \rightarrow V_\pi$
- Value Iteration
  - Compute the optimal value and optimal policy.

  $\text{MDP} \rightarrow (V_{\text{opt}}, \pi_{\text{opt}})$

Outline

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation *Given a policy π, evaluate how good it is. $(MDP, \pi) \rightarrow V_\pi$*
- Value Iteration *Compute the optimal value and optimal policy. $MDP \rightarrow (V_{\text{opt}}, \pi_{\text{opt}})$*
- Reinforcement Learning *Unknown transitions and rewards*
- Monte Carlo Methods
- Q-Learning & Deep Q-Learning
- ϵ-greedy Algorithm

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Recall the Markov Decision Process

- States: the set of states.
- $s_{\text{start}} \in \text{States}$: starting state.
- Actions(s): Possible actions from state s.
- ~~Transition function $T(s, a, s')$: Probability of reaching s' if take action a in state s.~~
- ~~Reward(s, a, s'): reward for the transition (s, a, s').~~
- IsEnd(s): check whether at the end of a game.
- $0 \leq \gamma \leq 1$: discount factor (default: 1).

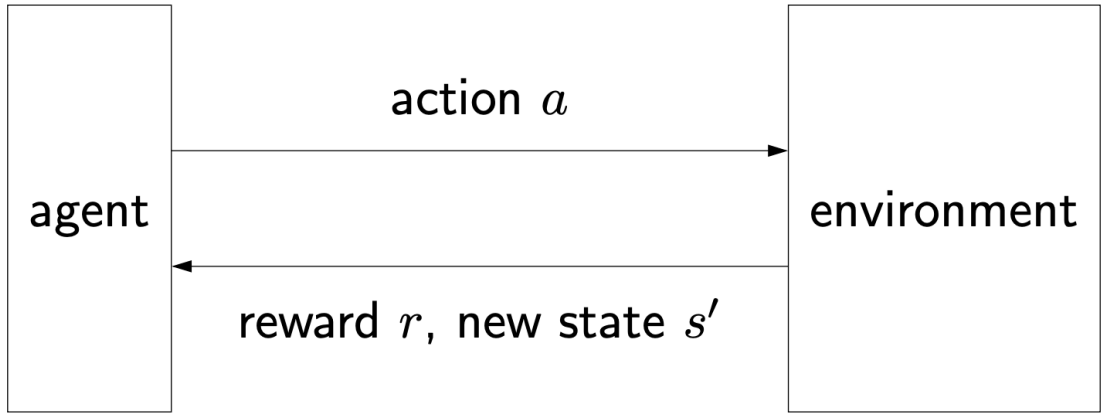
Reinforcement Learning

Maximize the value without knowing the transition function and reward.

- States: the set of states.
- $s_{\text{start}} \in \text{States}$: starting state.
- Actions(s): Possible actions from state s.

- IsEnd(s): check whether at the end of a game.
- $0 \leq \gamma \leq 1$: discount factor (default: 1).

Reinforcement Learning Framework



Algorithm: reinforcement learning template

For $t = 1, 2, 3, \ldots$

Choose action $a_t = \pi_{\text{act}}(s_{t-1})$ (**how?**)

Receive reward r_t and observe new state s_t

Update parameters (**how?**)

Supervised machine learning: passively receive data

Reinforcement learning: actively explore the states

Two questions:

- How to choose actions (what is π_{act})?
- How to update parameters?

We look at the second question first.

Outline

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation *Given a policy π, evaluate how good it is. $(MDP, \pi) \rightarrow V_\pi$*
- Value Iteration *Compute the optimal value and optimal policy. $MDP \rightarrow (V_{\text{opt}}, \pi_{\text{opt}})$*
- Reinforcement Learning *Unknown transitions and rewards*
- Monte Carlo Methods *Estimate the MDP or the optimal value from data*
- Q-Learning & Deep Q-Learning
- ϵ-greedy Algorithm

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Monte Carlo Methods

- Monte Carlo is a standard way to estimate the expectation of a random variable by computing the empirical mean.
- It takes an average over samples of that random variable.

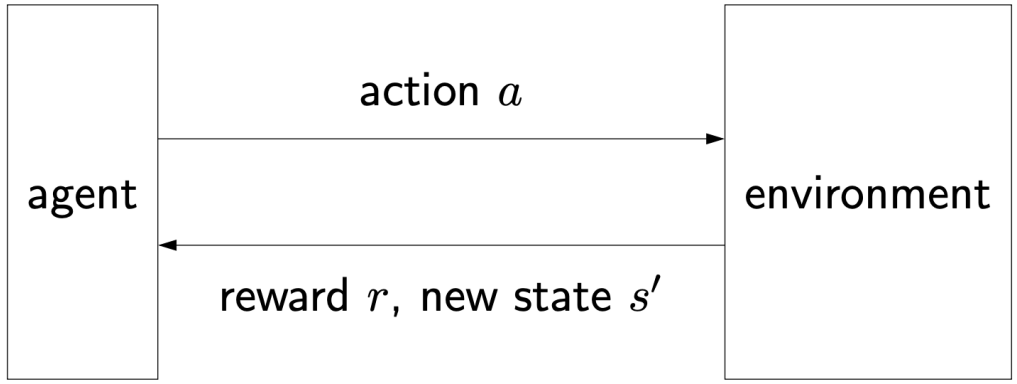
- Example:
  - Coin flip: $p(\text{head}) = \frac{\#\ \text{heads}}{\#\ \text{total\_flips}}$

Image: https://medium.com/swlh/how-many-flips-of-a-coin-does-it-take-to-get-nine-heads-or-tails-in-a-row-a324298e2d00

Model-based Monte Carlo

Idea: Estimate the MDP, ($T(s, a, s')$ and Rewards (s, a, s')), from data

An episode: $s_0; a_1, r_1, s_1; a_2, r_2, s_2; \ldots; a_n, r_n, s_n$



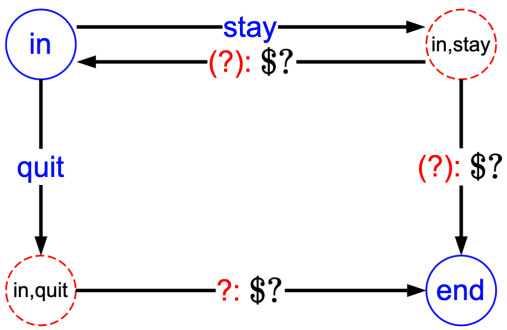
Initially at s_0;
Take action a_1, receive reward r_1, reach r_1;
Take action a_2, receive reward r_2, reach r_2;
…
Take action a_n, receive reward r_n, reach r_n;

$$\hat{T}(s, a, s') = \frac{\#\text{ times } (s, a, s') \text{ occurs}}{\#\text{ times } (s, a) \text{ occurs}}$$

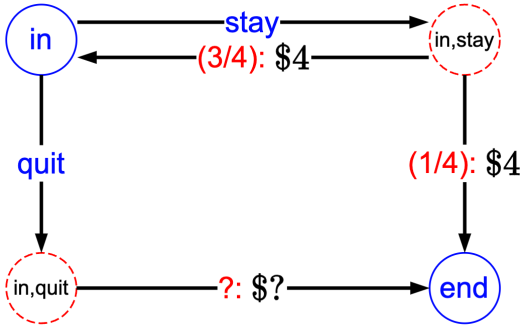
$$\widehat{\text{Reward}}(s, a, s') = r \text{ in } (s, a, r, s')$$

Model-based Monte Carlo

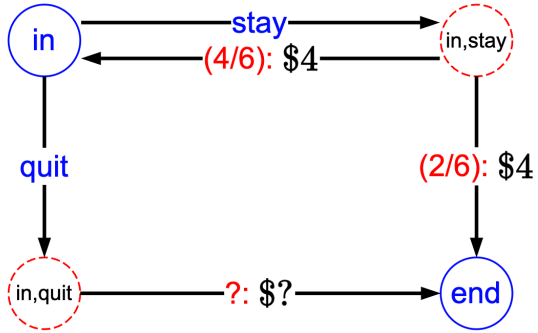
Example: Dice game



update



update

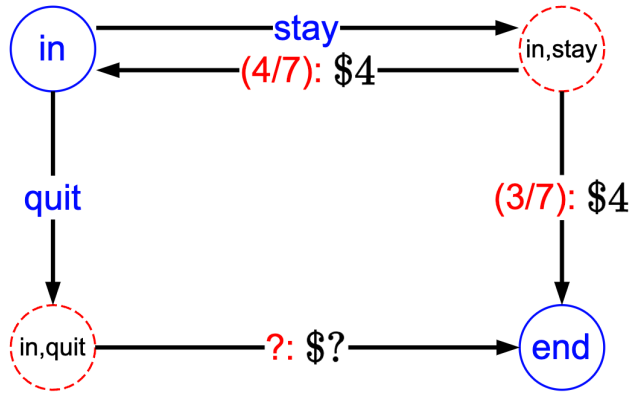


$\pi(s)$ = "stay"

$$\hat{T}(s,a,s') = \frac{\#\text{ times } (s,a,s') \text{ occurs}}{\#\text{ times } (s,a) \text{ occurs}}$$

$$\widehat{\text{Reward}}(s,a,s') = r \text{ in } (s,a,r,s')$$

update

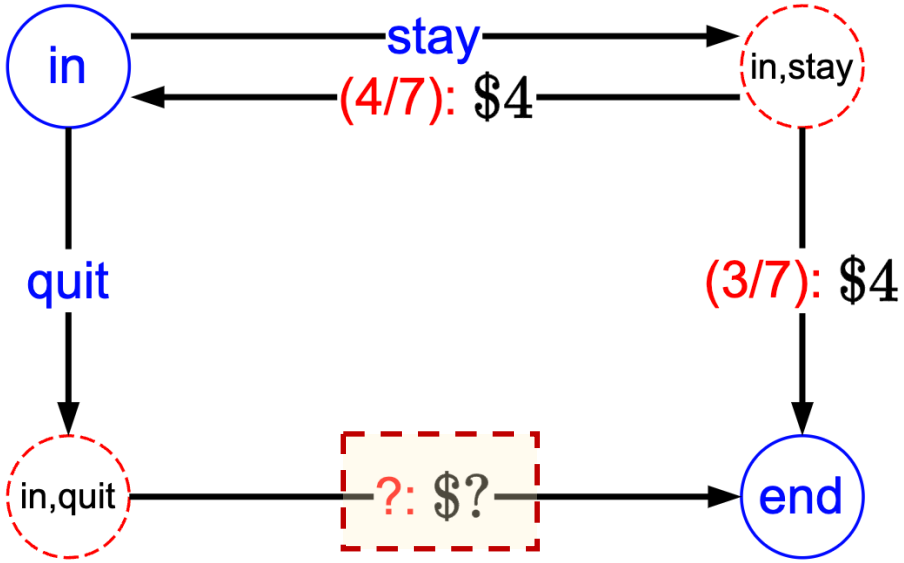


Data (e.g., three episodes):

- [in; stay, 4, in; stay, 4, in; stay, 4, in; stay, 4, end]
- [in; stay, 4, in; stay, 4, end]
- [in; stay, 4, end]

Model-based Monte Carlo

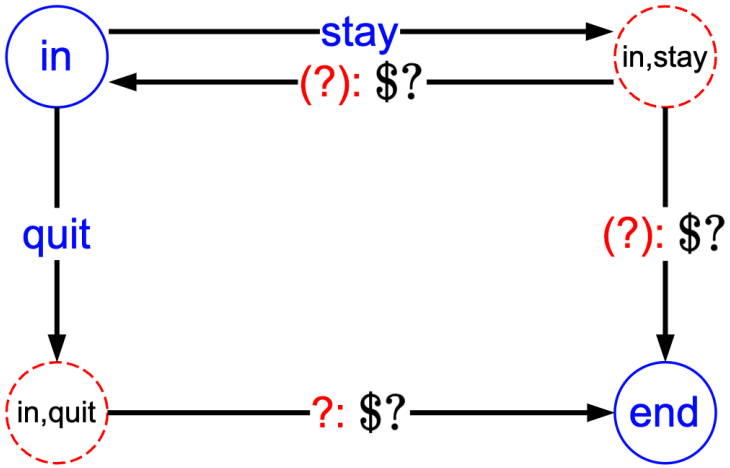
Problem: Cannot see (s, a) if $a \neq \pi(s)$



Solution: Exploration (will see more later)

Model-based Monte Carlo & Value Iteration

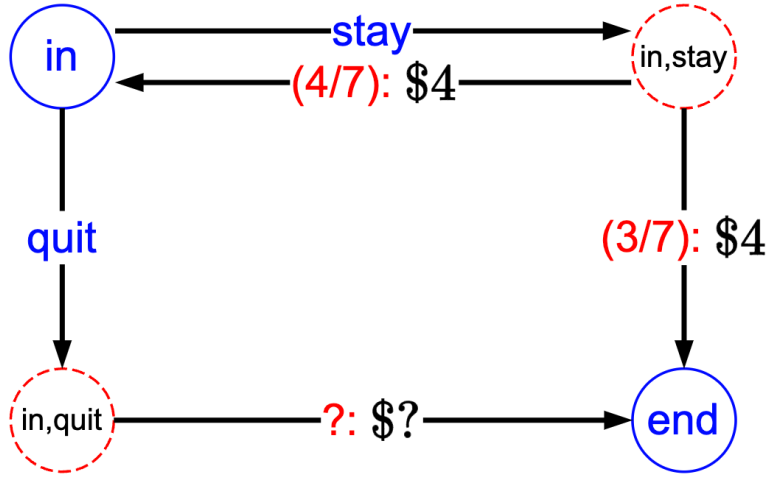
"Model-based": explicitly estimating the MDP



Unknown transitions & rewards

Model-based Monte Carlo

Generate episodes following some policy π, estimate the transitions and rewards.



Estimated MDP

With the estimated MDP ($\hat{T}$, $\widehat{\text{Reward}}$), compute policy using value iteration:

Model-based Value Iteration: $\hat{Q}_{\text{opt}}(s,a) = \sum_{s'} \hat{T}(s,a,s')[\widehat{\text{Reward}}(s,a,s') + \gamma \hat{V}_{\text{opt}}(s')]$

Model-free Monte Carlo

Idea: Estimate $Q_\pi(s,a)$ directly (without explicitly estimating MDP)

Data (following policy π): $s_0; a_1, r_1, s_1; a_2, r_2, s_2; \ldots; a_n, r_n, s_n$

Q-value of a policy $Q_\pi(s,a)$: The expected utility of taking action a at s, and then following the policy π.

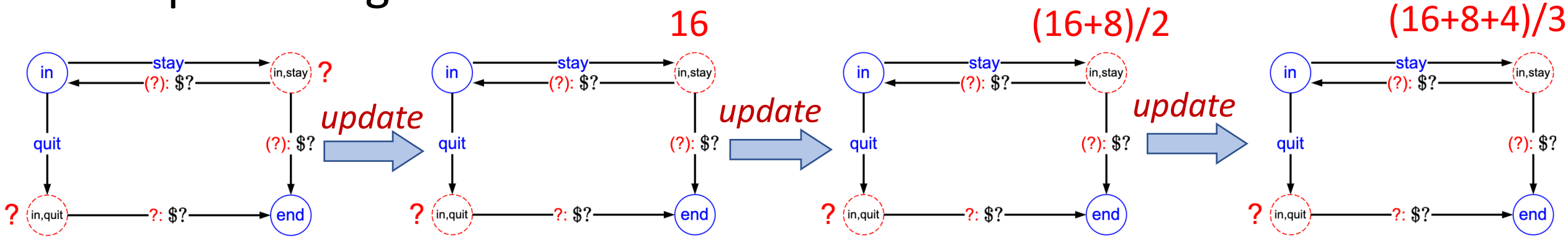
Utility: $u_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$

Monte Carlo Estimation of Q-value:

$\hat{Q}_\pi(s,a)$ = average of u_t where $s_{t-1} = s, a_t = a$

Model-free Monte Carlo

Example: Dice game

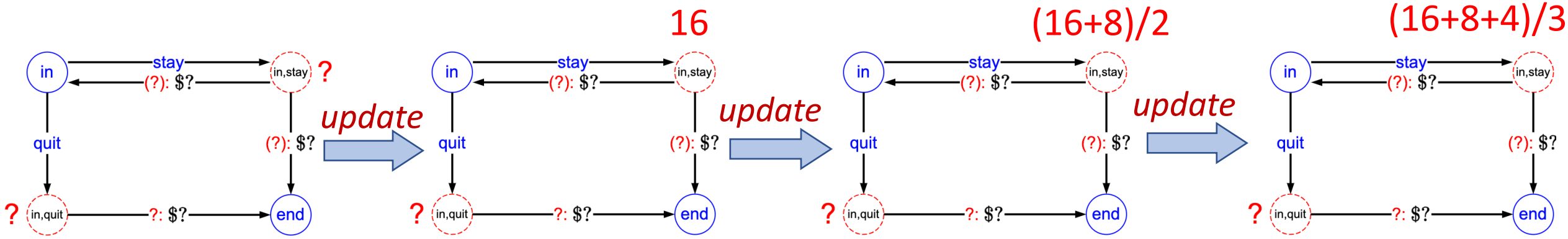


$\pi(s)$ = "stay" *Data (e.g., three episodes):*

- [in; stay, 4, in; stay, 4, in; stay, 4, end] $u = 16$
- [in; stay, 4, in; stay, 4, end] $u = 8$
- [in; stay, 4, end] $u = 4$

Note: We are estimating Q_π, not Q_{opt}!

Model-free Monte Carlo



Incremental Update:

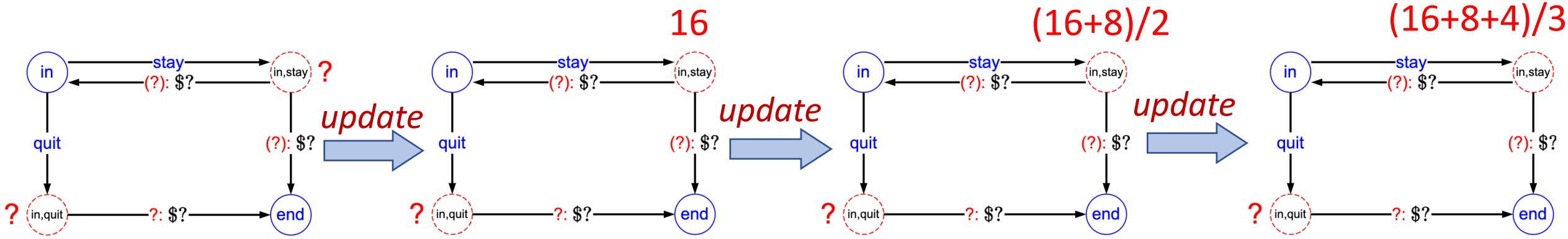
$\hat{Q}_\pi(s,a)$ = average of u_t Let n be the number of updates to (s,a)

new *old* *data*

$$\boxed{\hat{Q}_\pi(s,a)} \leftarrow \frac{n\boxed{\hat{Q}_\pi(s,a)} + \boxed{u}}{n+1} = \frac{n}{n+1}\hat{Q}_\pi(s,a) + \boxed{\frac{1}{n+1}}u = (1-\eta)\hat{Q}_\pi(s,a) + \eta u$$

η

Model-free Monte Carlo



Incremental Update: $\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta u$

Equivalently, $\hat{Q}_\pi(s,a) \leftarrow \hat{Q}_\pi(s,a) - \eta(\hat{Q}_\pi(s,a) - u)$

predictions *data*

Think this as the stochastic gradient update of a loss function of

$$\left(\hat{Q}_\pi(s,a) - u\right)^2$$

Note: We are estimating Q_π, not Q_{opt}!

Monte Carlo Methods

- Model-based Monte Carlo
  - Fully specify the MDP → can estimate anything, including $Q_{\text{opt}}(s, a)$
- Model-free Monte Carlo
  - Estimate $\hat{Q}_{\pi}$; cannot estimate $Q_{\text{opt}}(s, a)$.
- On-policy versus off-policy
  - On-policy: estimate the value of data-generating policy π (estimation depends on π), e.g., model-free Monte Carlo, Q-learning.
  - Off-policy: estimate the value of another policy π (estimation does not depend on π). E.g., in model-based MC, *the exact policy that generates the data does not matter*.

Outline

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation *Given a policy π, evaluate how good it is. $(MDP, \pi) \rightarrow V_{\pi}$*
- Value Iteration *Compute the optimal value and optimal policy. $MDP \rightarrow (V_{\text{opt}}, \pi_{\text{opt}})$*
- Reinforcement Learning *Unknown transitions and rewards*
- Monte Carlo Methods *Estimate the MDP or the optimal value from data*
- Q-Learning & Deep Q-Learning *Estimate Q_{opt} in a model-free manner*
- ϵ-greedy Algorithm

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Q-Learning

- Recall the MDP recurrence:

$$Q_{\text{opt}}(s,a) = \sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}(s')]$$

- Q-learning algorithm:

On each (s,a,r,s'):

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{\text{opt}}(s,a)}_{\text{prediction}} + \eta\underbrace{(r + \gamma \hat{V}_{\text{opt}}(s'))}_{\text{target}}$$

where $\hat{V}_{\text{opt}}(s') = \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s',a')$

(Tabular) Q-Learning

- The Q-values $Q_\pi(s, a)$ can be stored in a table called Q-table.
- It stores the estimated utilities (not reward!) obtained by action a at state s.

s_1

a_1

a_2

Q-table:

	a_1	a_2
s_1	-2	1

a_2

s_2

a_1

a_2

Q-table:

	a_1	a_2
s_2	-4	2

a_2

Reference: https://mofanpy.com/tutorials/machine-learning/reinforcement-learning/intro-q-learning/

(Tabular) Q-Learning

- Store the expected return for every state and every action.
- When we are in a state, just select the action with maximum Q-value.
- How can we learn this Q-table from data?
  - Initialize the Q-table with all zeros
  - Update entries

Q-table:

	a_1	a_2
s_1	-2	1
s_2	-4	2
…	…	…

Initialization:

	a_1	a_2
s_1	0	0
s_2	0	0
…	…	…

Q-Learning

- Update entries: Just like supervised learning, we correct our model by the difference between prediction and ground truth:

$$Q(s,a) \leftarrow Q(s,a) + \eta \left(r + \left[\gamma \max_{a'} Q(s',a') \right] - Q(s,a) \right)$$

Q-Learning

- Update entries: Just like supervised learning, we correct our model by the difference between prediction and ground truth:

$$Q(s,a) \leftarrow \underbrace{Q(s,a)}_{\text{estimation}} + \overbrace{\eta}^{lr}\Big(\underbrace{r}_{\text{reward}} + \Big[\overbrace{\gamma}^{\text{discount}} \underbrace{\max_{a'} Q(s',a')}_{\text{optimal future}}\Big] - \underbrace{Q(s,a)}_{\text{estimation}}\Big)$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{“real value”}}$$

Example of Tabular Q-Learning

The agent lives in a 3x2 grid. The agent cannot take actions that bring it outside the grid. Landing on state 2 and 6 have rewards of 5 and 20, respectively. Reward of landing on other states is 0. Use learning rate of $\eta = 0.8$ and discount rate $\gamma = 1$.

The agent starts from state 1 and takes actions: East, South, West, South, East. After each action, the Q-table is updated. What is the Q-table at the end of this phase?

1	2 reward: 5
3	4
5	6 reward: 20

Example of Tabular Q-Learning

The agent lives in a 3x2 grid. The agent cannot take actions that bring it outside the grid. Landing on state 2 and 6 have rewards of 5 and 20, respectively. Reward of landing on other states is 0. Use learning rate of $\eta = 0.8$ and discount rate $\gamma = 1$.

The agent starts from state 1 and takes actions: East, South, South, West. After each action, the Q-table is updated. What is the Q-table at the end of this phase?

1	2 reward: 5
3	4
5	6 reward: 20

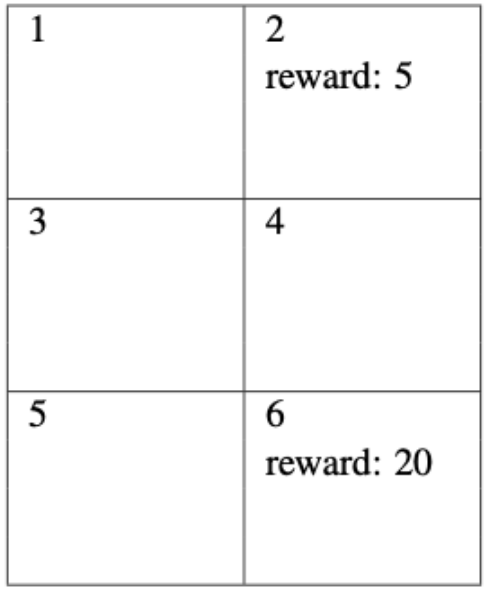
	East	West	North	South
1	0			0
2		0		0
3	0		0	0
4		0	0	0
5	0		0	
6		0	0	

Step 1: Initialize the Q table

Example of Tabular Q-Learning

The agent lives in a 3x2 grid. The agent cannot take actions that bring it outside the grid. Landing on state 2 and 6 have rewards of 5 and 20, respectively. Reward of landing on other states is 0. Use learning rate of $\eta = 0.8$ and discount rate $\gamma = 1$.

The agent starts from state 1 and takes actions: East, South, South, West. After each action, the Q-table is updated. What is the Q-table at the end of this phase?



	East	West	North	South
1	0			0
2		0		0
3	0		0	0
4		0	0	0
5	0		0	
6		0	0	

Step 2: Perform updates

$$Q(s,a) \leftarrow Q(s,a) + \eta\left(r + \left[\gamma \max_{a'} Q(s',a')\right] - Q(s,a)\right)$$

Action 1 (East): $s = 1$, $a =$ East, $s' = 2$, $r = 5$

$$Q(1,\text{East}) \leftarrow Q(1,\text{East}) + 0.8\left(r + \left[0.9 \max_{a'} Q(2,a')\right] - Q(1,\text{East})\right)$$

$$Q(1,\text{East}) \leftarrow 0 + 0.8\left(5 + \left[1 \cdot \max_{a'} Q(2,a')\right] - 0\right)$$

$$Q(1,\text{East}) \leftarrow 0 + 0.8(5 + [1 \cdot 0] - 0) = 4$$

Example of Tabular Q-Learning

The agent lives in a 3x2 grid. The agent cannot take actions that bring it outside the grid. Landing on state 2 and 6 have rewards of 5 and 20, respectively. Reward of landing on other states is 0. Use learning rate of $\eta = 0.8$ and discount rate $\gamma = 1$.

The agent starts from state 1 and takes actions: East, South, South, West. After each action, the Q-table is updated. What is the Q-table at the end of this phase?

1	2 reward: 5
3	4
5	6 reward: 20

	East	West	North	South
1	4			0
2		0		0
3	0		0	0
4		0	0	0
5	0		0	
6		0	0	

Step 2: Perform updates

$$Q(s,a) \leftarrow Q(s,a) + \eta\left(r + \left[\gamma \max_{a'} Q(s',a')\right] - Q(s,a)\right)$$

Action 1 (East): $s = 1$, $a =$ East, $s' = 2$, $r = 5$

$$Q(1,\text{East}) \leftarrow Q(1,\text{East}) + 0.8\left(r + \left[0.9 \max_{a'} Q(2,a')\right] - Q(1,\text{East})\right)$$

$$Q(1,\text{East}) \leftarrow 0 + 0.8\left(5 + \left[1 \cdot \max_{a'} Q(2,a')\right] - 0\right)$$

$$Q(1,\text{East}) \leftarrow 0 + 0.8(5 + [1 \cdot 0] - 0) = 4$$

Example of Tabular Q-Learning

The agent lives in a 3x2 grid. The agent cannot take actions that bring it outside the grid. Landing on state 2 and 6 have rewards of 5 and 20, respectively. Reward of landing on other states is 0. Use learning rate of $\eta = 0.8$ and discount rate $\gamma = 1$.

The agent starts from state 1 and takes actions: East, South, South, West. After each action, the Q-table is updated. What is the Q-table at the end of this phase?

1	2 reward: 5
3	4
5	6 reward: 20

	East	West	North	South
1	4			0
2		0		0
3	0		0	0
4		0	0	0
5	0		0	
6		0	0	

Step 2: Perform updates

$$Q(s,a) \leftarrow Q(s,a) + \eta\left(r + \left[\gamma \max_{a'} Q(s',a')\right] - Q(s,a)\right)$$

Action 2 (South): $s = 2$, $a =$ South, $s' = 4$, $r = 0$

$$Q(2,\text{South}) \leftarrow Q(2,\text{South}) + 0.8\left(r + \left[0.9 \max_{a'} Q(4,a')\right] - Q(2,\text{South})\right)$$

$$Q(2,\text{South}) \leftarrow 0 + 0.8\left(0 + \left[1 \cdot \max_{a'} Q(2,a')\right] - 0\right)$$

$$Q(2,\text{South}) \leftarrow 0 + 0.8(0 + [1 \cdot 0] - 0) = 0$$

Example of Tabular Q-Learning

The agent lives in a 3x2 grid. The agent cannot take actions that bring it outside the grid. Landing on state 2 and 6 have rewards of 5 and 20, respectively. Reward of landing on other states is 0. Use learning rate of $\eta = 0.8$ and discount rate $\gamma = 1$.

The agent starts from state 1 and takes actions: East, South, South, West. After each action, the Q-table is updated. What is the Q-table at the end of this phase?

1	2 reward: 5
3	4
5	6 reward: 20

	East	West	North	South
1	4			0
2		0		0
3	0		0	0
4		0	0	0
5	0		0	
6		0	0	

Exercise 1: perform update for the remining actions.

$$Q(s,a) \leftarrow Q(s,a) + \eta \left(r + \left[\gamma \max_{a'} Q(s',a') \right] - Q(s,a) \right)$$

Example of Tabular Q-Learning

The agent lives in a 3x2 grid. The agent cannot take actions that bring it outside the grid. Landing on state 2 and 6 have rewards of 5 and 20, respectively. Reward of landing on other states is 0. Use learning rate of $\eta = 0.8$ and discount rate $\gamma = 1$.

The agent starts from state 1 and takes actions: East, South, South, West. After each action, the Q-table is updated. What is the Q-table at the end of this phase?

1	2 reward: 5
3	4
5	6 reward: 20

	East	West	North	South
1	4			0
2		0		0
3	0		0	0
4		0	0	0
5	0		0	
6		0	0	

Exercise 2: Continue update for the following actions:

North, North, East, South, South, West, North, North

$$Q(s,a) \leftarrow Q(s,a) + \eta \left(r + \left[\gamma \max_{a'} Q(s',a') \right] - Q(s,a) \right)$$

Deep Q Network

Playing atari with deep reinforcement learning
V Mnih, K Kavukcuoglu, D Silver, A Graves… - arXiv preprint arXiv …, 2013 - arxiv.org
We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the …
☆ 99 Cited by 5071 Related articles All 34 versions »

- Deep Q Network (DQN) uses deep neural network to estimate the action-value function.

$$Q_\pi(s, a; \theta) \approx Q_\pi^*(s, a)$$

  - θ is network parameters.
- Now, the estimated and real Q-value becomes:
  - Estimated value: $Q_\pi(s_t, a_t; \theta)$.
  - Real value: $R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta)$.
- How to update? Gradient descent:

$$\nabla_\theta L_t = \mathbb{E}[R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta) - Q_\pi(s_t, a_t; \theta)]$$

Deep Q Network

- Now new problems appear.

- Training samples in a time interval are highly correlated.
  - They are not i.i.d!
  - Inefficient learning.

- The current parameters determine the next data sample that the parameters are trained on.
  - Get stuck in a poor local minimum, or diverge catastrophically.

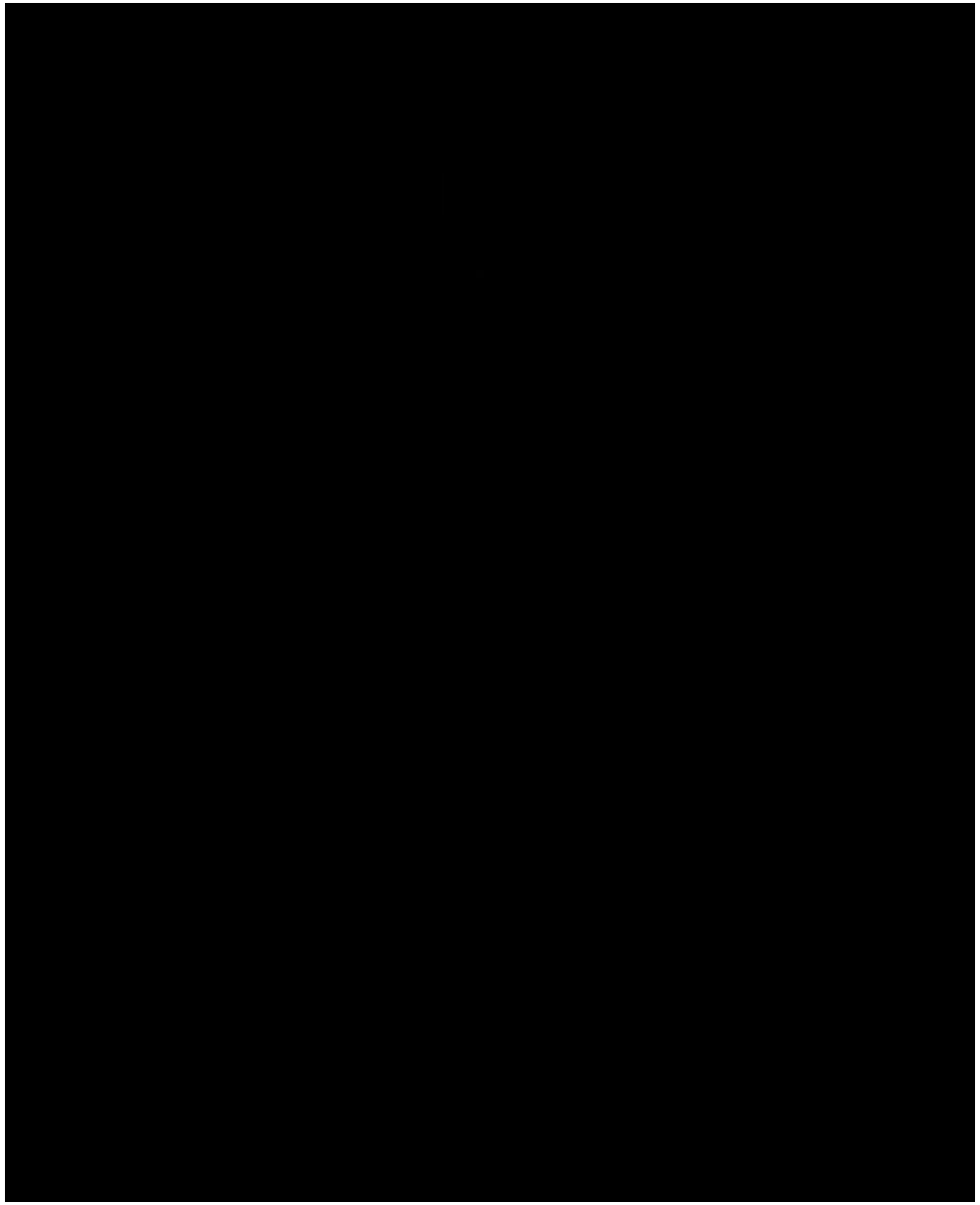
Experience Replay

- Idea: store transitions in memory and random sample some for training at each step.

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Image source: Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).



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

Outline

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation *Given a policy π, evaluate how good it is. $(MDP, \pi) \rightarrow V_{\pi}$*
- Value Iteration *Compute the optimal value and optimal policy. $MDP \rightarrow (V_{\text{opt}}, \pi_{\text{opt}})$*
- Reinforcement Learning *Unknown transitions and rewards*
- Monte Carlo Methods *Estimate the MDP or the optimal value from data*
- Q-Learning & Deep Q-Learning *Estimate Q_{opt} in a model-free manner*
- ϵ-greedy Algorithm *Balance between exploration and exploitation*

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC

Exploration and Exploitation

- Recall the reinforcement learning template:

Algorithm: reinforcement learning template

For $t = 1, 2, 3, \ldots$

Choose action $a_t = \pi_{\text{act}}(s_{t-1})$ (**how?**)

Receive reward r_t and observe new state s_t

Update parameters (**how?**)

- Remining question: what exploration policy π_{act} to use?

Exploration and Exploitation

- No exploration, all exploitation

Set $\pi_{\text{act}}(s) = \arg \max_{a \in \text{Actions}(s)} \hat{Q}_{\text{opt}}(s, a)$

1	2 reward: 5
3	4
5	6 reward: 20

	East	West	North	South
1	4			0
2		0		0
3	0		0	0
4		0	0	0
5	0		0	
6		0	0	

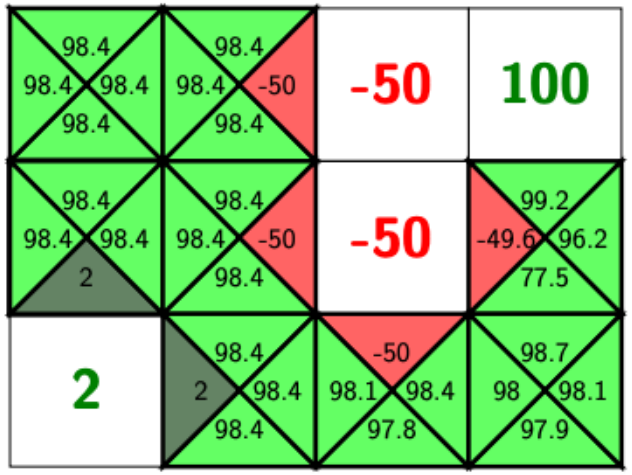
Suppose when state 2 is reached, the agent returns to the start for a new episode.

After updating the Q-table: only action East will be taken. The rest of the world will not explored, even if higher rewards exist. (Too greedy! Nothing new can be learned.)

Exploration and Exploitation

- No exploitation, all exploration

Set $\pi_{\text{act}}(s) = \text{random from Actions}(s)$



Average utility: -19.15

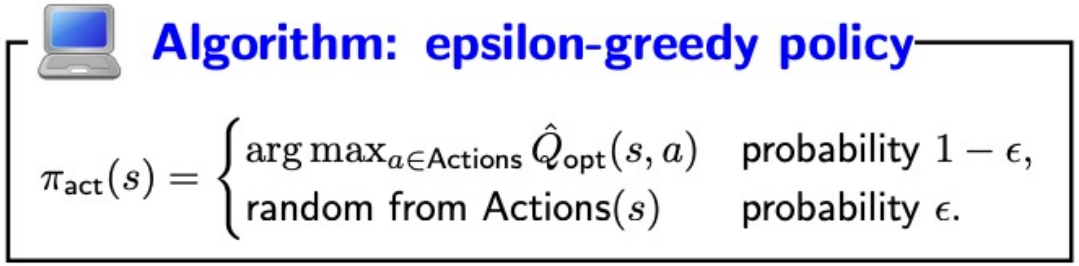
Average utility is low (example: volcano crossing)

Exploration is not guided.

Balance between Exploration and Exploitation

- To maximize reward, an agent must prefer actions that it has tried in the past and found to be effective in producing reward.
- However, to discover such actions, it has to try actions that it has not selected before.
- The agent has to *exploit* what it has already experienced in order to obtain reward
- The agent also has to *explore* in order to make better action selections in the future.
- Therefore, the balance between exploration and exploitation is very important in reinforcement learning.

ϵ-greedy Algorithm



- Idea: with a certain probability, the agent select a random action.

- All exploration: $\epsilon = 1$
- All exploitation: $\epsilon = 0$

Summary

- Markov Decision Processes *Specify the dynamics of the world.*
- Policy Evaluation *Given a policy π, evaluate how good it is. $(MDP, \pi) \rightarrow V_\pi$*
- Value Iteration *Compute the optimal value and optimal policy. $MDP \rightarrow (V_{\text{opt}}, \pi_{\text{opt}})$*
- Reinforcement Learning *Unknown transitions and rewards*
- Monte Carlo Methods *Estimate the MDP or the optimal value from data*
- Q-Learning & Deep Q-Learning *Estimate Q_{opt} in a model-free manner*
- ϵ-greedy Algorithm *Balance between exploration and exploitation*

Credits: The slides are partially adapted from the following sources:
- Lecture 7 & 8, CS221@Stanford
- Lecture 10, Deep Learning@XMU
- Lecture 22-24, CS440@UIUC