Reinforcement Learning

CPSC 470 – Artificial Intelligence

Brian Scassellati

From Supervised Learning to Reinforcement Learning

- In supervised learning, when the agent makes a mistake, it is immediately given feedback in the form of the correct response
- In reinforcement learning, when the agent makes a mistake, it will later be given feedback in the form of a punishment or reward

Passive Learning in Known Environments

Actual Utility Values

	1	2	3	4
3	−0.0380	0.0886	0.2152	+ 1
2	−0.1646		−0.4430	− 1
1	−0.2911	−0.0380	−0.5443	−0.7722

Estimated Utility Values

	1	2	3	4
3	-0.1	0.1	0.2	+ 1
2	-0.2		-0.4	− 1
1	-0.3	0	-0.5	-0.7

- Given a set of training sequences that end in a terminal state (with a reward)

  (1,1) → (2,1) → (3,1) → (3,2) → (3,3) → (4,3) → **+1**

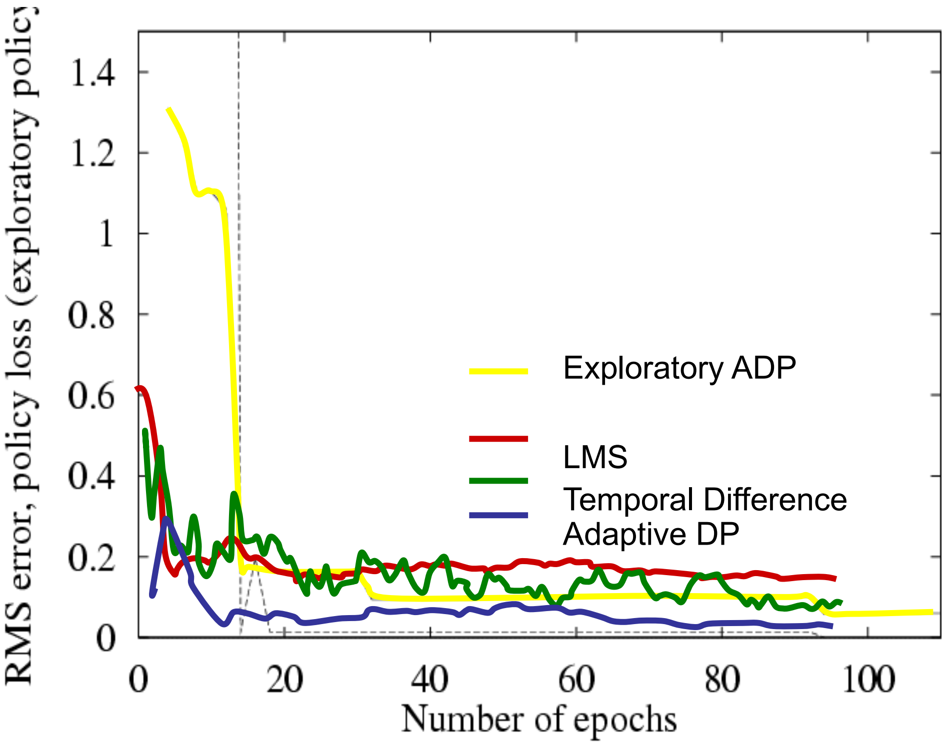
  (1,1) → (2,1) → (3,1) → (3,2) → (4,2) → **-1**

  (1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3) → **+1**

  (1,1) → (2,1) → (3,1) → (4,1) → (3,1) → (3,2) → (4,2) → **-1**

- Determine the expected utility ***U(i)*** associated with each non-terminal state ***i***

Comparison of Reinforcement Learning (Utility) Techniques



What if we have no good model of the environment?

Two Basic Types of Reinforcement Learning

Utility Learning

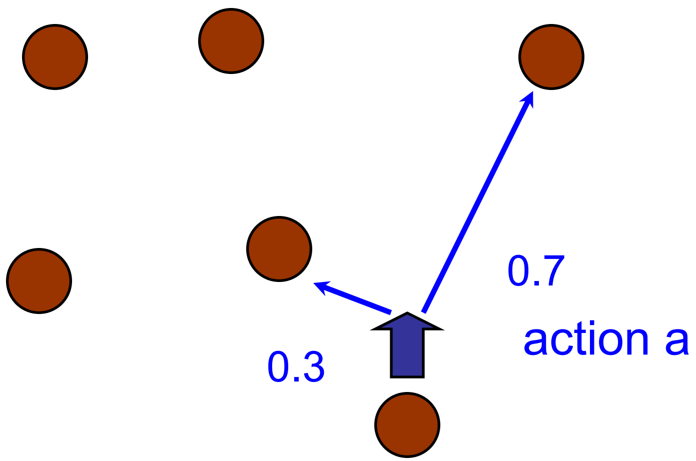
- Learn a utility function that maps states to utilities and select an action by maximizing the expected value
- Needs a model of the environment (needs to know which state an action will result in)
- Predictive

Action-Value Learning

- Learn an action-value function that gives the expected utility of taking a given action in a given state
- No need for an environment model
- Do not know where actions lead, so it cannot look ahead

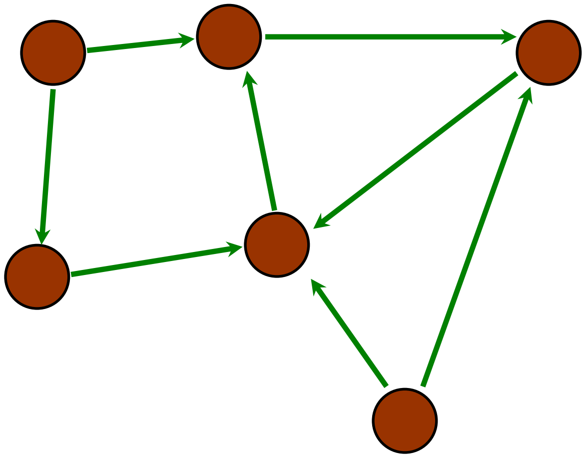
States and Actions

Environment = states



Actions = transitions $\delta(s, a, s')$

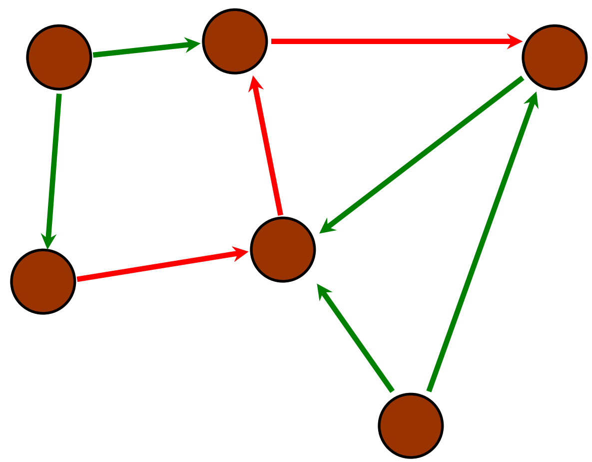
Rewards

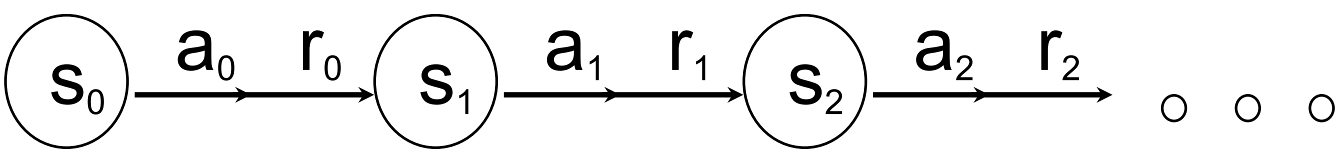


R(s,a) = reward at state ***s***

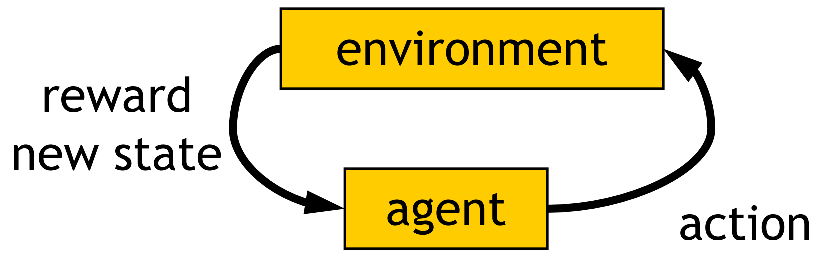
for doing action ***a***

Trajectories





Markov Decision Process (MDP)



- set of states S, set of actions A, initial state S_0
- transition model $P(s'|s,a)$
  - Markov assumption
- Reward function $R(s,a)$
- policy: mapping from S to A
  - $\pi(s)$ or $\pi(s,a)$

Agent Learns a Policy

Policy at step t, π_t :

a mapping from states to action probabilities

$\pi_t(s, a) =$ probability that $a_t = a$ when $s_t = s$

- Reinforcement learning methods specify how the agent changes its policy as a result of experience.
- Roughly, the agent's goal is to get as much reward as it can over the long run.

Goals and Rewards

- Is a scalar reward signal an adequate notion of a goal?
  - Maybe not, but it is surprisingly flexible.
- A goal should specify **what** we want to achieve, not **how** we want to achieve it.
- A goal must be outside the agent's direct control—thus outside the agent.
- The agent must be able to measure success:
  - explicitly;
  - frequently during its lifespan.

Returns

Suppose the sequence of rewards after step t is :

$$r_{t+1}, r_{t+2}, r_{t+3}, \ldots$$

What do we want to maximize?

In general,

we want to maximize the **expected return**, $E\{R_t\}$, for each step t.

Episodic tasks: interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze.

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T,$$

where T is a final time step at which a **terminal state** is reached, ending an episode.

Returns for Continuing Tasks

Continuing tasks: interaction does not have natural episodes.

Discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where γ, $0 \le \gamma \le 1$, is the **discount rate**.

shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

(Learning sequences of actions that generate rewards)

Q-LEARNING

Learning Sequences

- Q-Learning allows an agent to learn *chains* of actions.
- Even though the agent lives only in the present, it acts as if it can see into the future
  - Cannot even predict the next state
- Propagation of credit from the *consummatory* behavior back through the chain of *appetitive* behaviors

Temporal Assignment Problem

- How do you determine which actions successfully produced the goal state?
- Q-learning solves this problem without needing to explicitly remember the sequence of states that lead to a reward.
- The price you pay is that it depends upon repeated visits to each state/action combination.

Q-Values

- Big idea:
  - Compute the quality value (or Q-value) for each possible action ***a*** in state ***x***
  - Choose actions stochastically on the Q-values

Q-Learning Approach

- Update the Q-values according to the following rule:

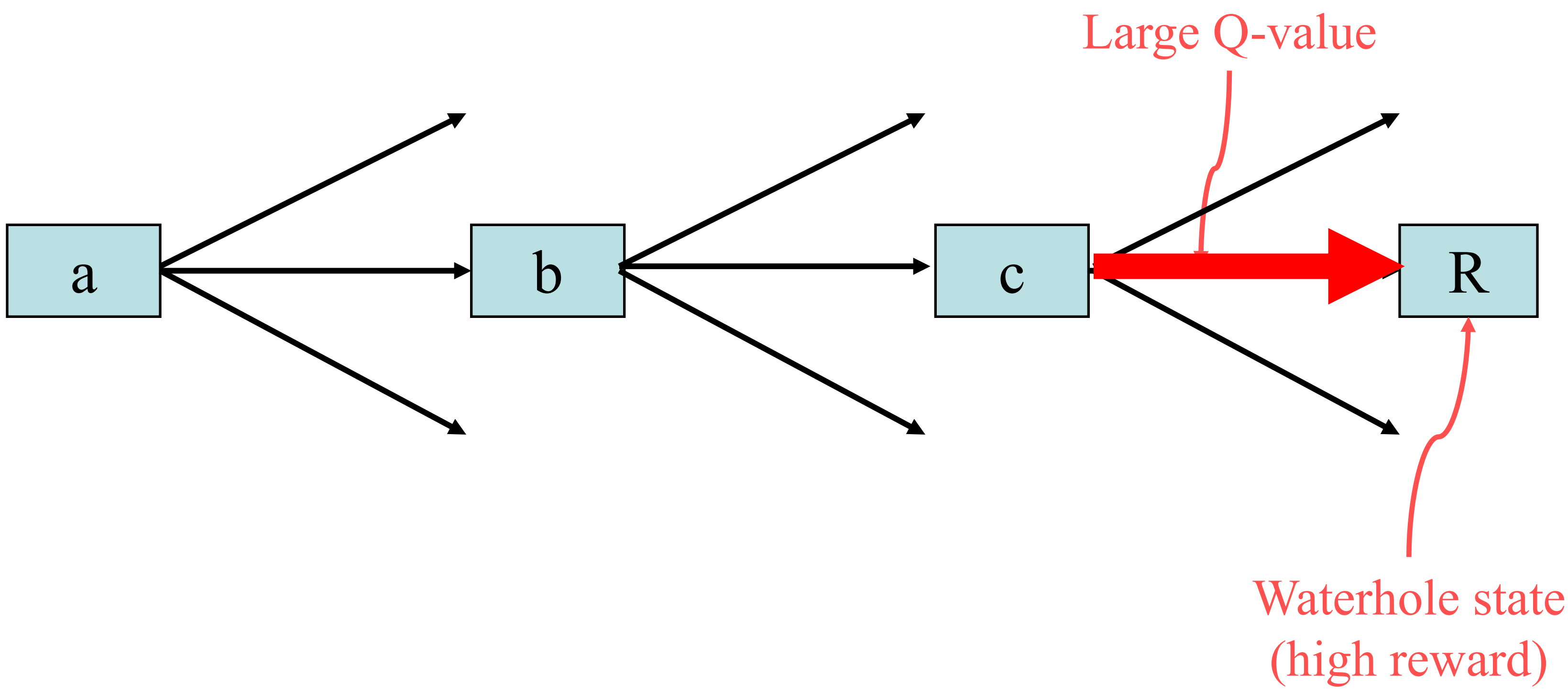
$$Q(x,a) \equiv (1-\alpha)Q(x,a) + \alpha(r + \gamma \max_{b \in A} Q(y,b))$$

Discounted prior Q value

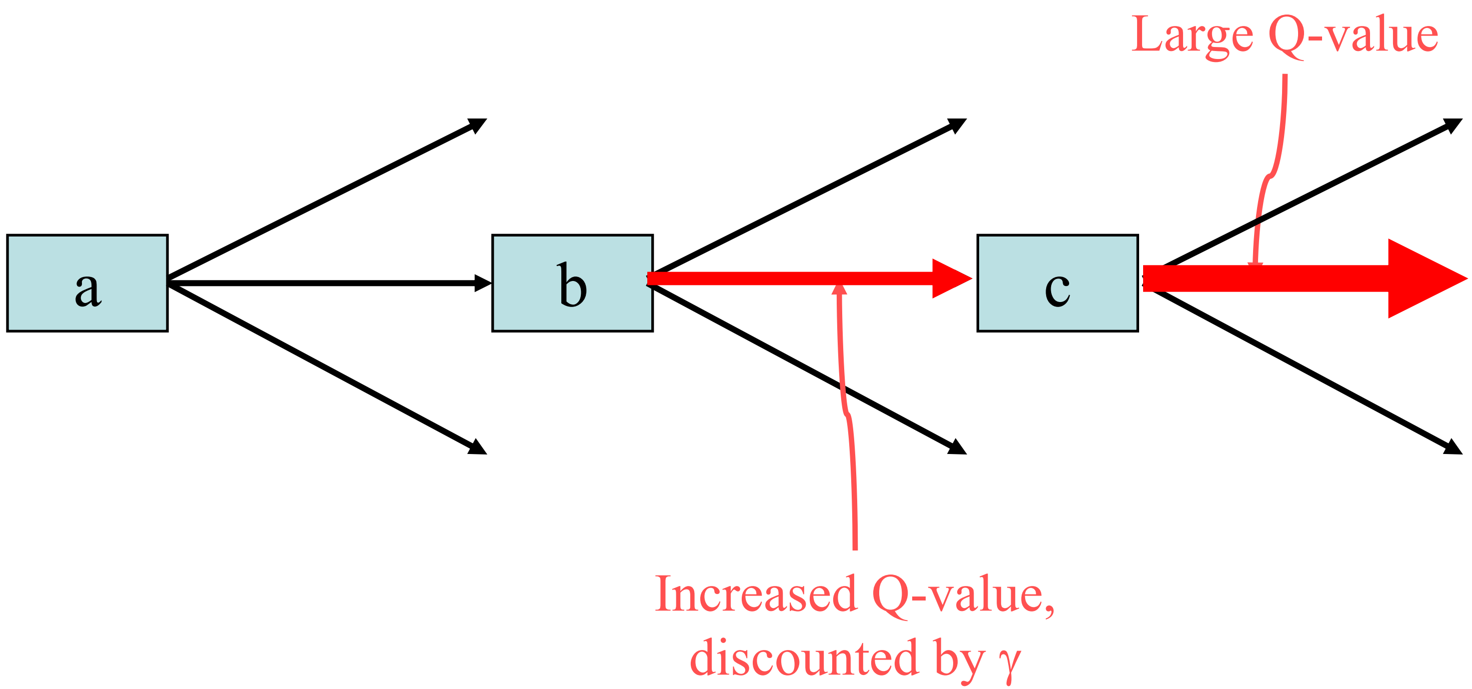
Updated with new Q value

- For a given learning rate α and a discount rate of γ

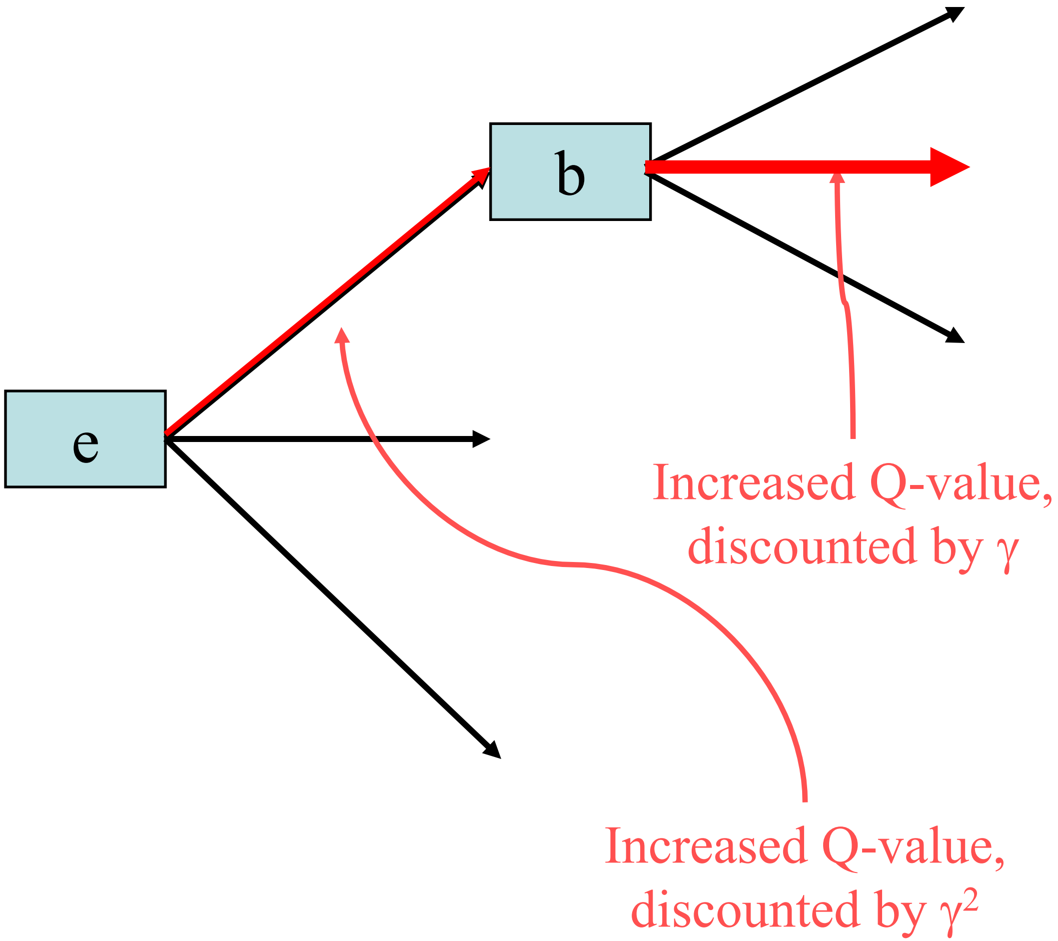
How Q-Learning Works



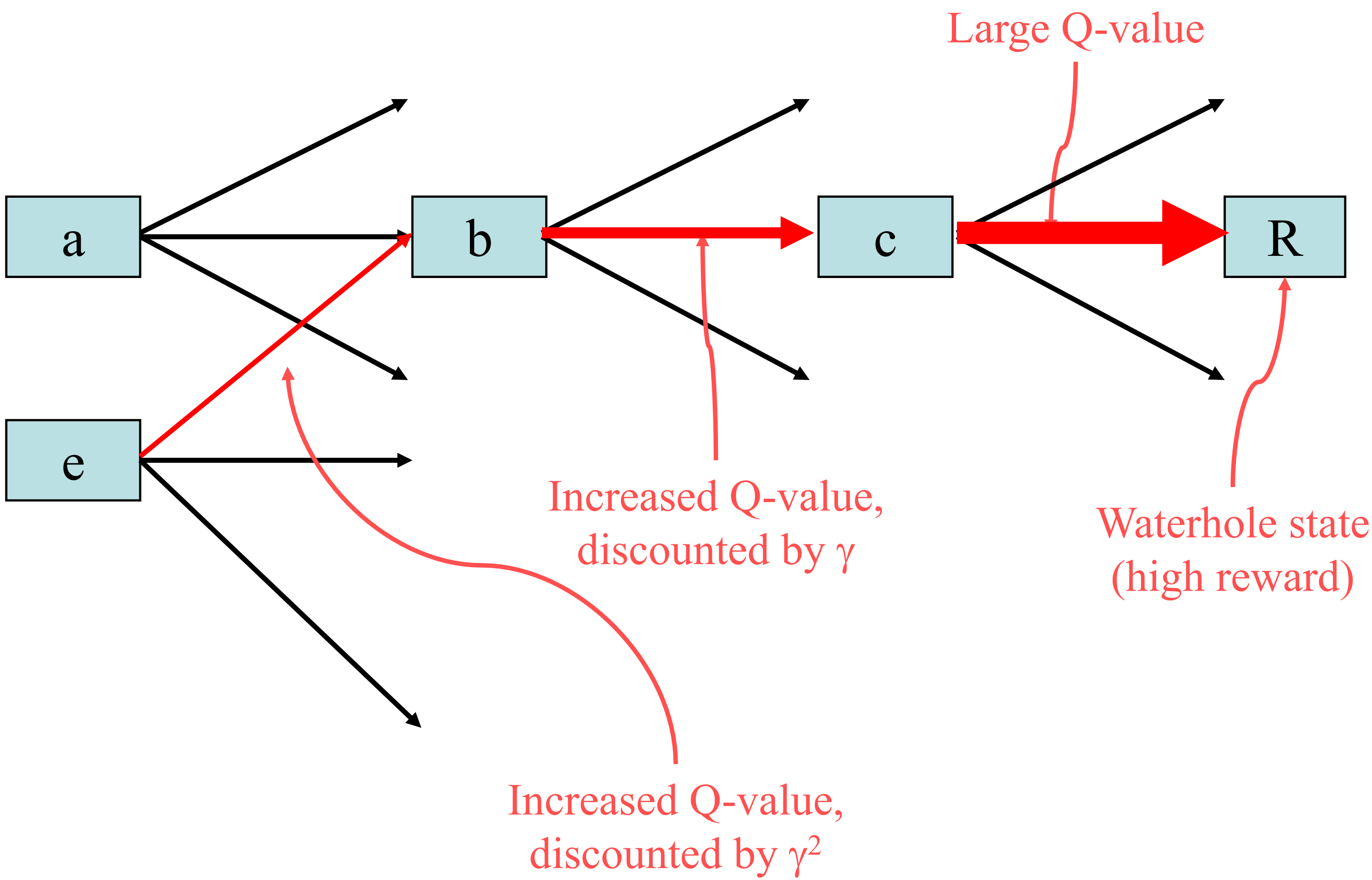
How Q-Learning Works



How Q-Learning Works



How Q-Learning Works



How to Change the Learning Rate

- What happens in the standard update rule if we just gradually drop the learning rate over time?

$$Q(x,a) \equiv (1-\alpha)Q(x,a) + \alpha(r + \gamma \max_{b \in A} Q(y,b))$$

How to Change the Learning Rate

- What happens in the standard update rule if we just gradually drop the learning rate over time?

$$Q(x,a) \equiv (1-\alpha)Q(x,a) + \alpha(r + \gamma \max_{b \in A} Q(y,b))$$

- Solution: Make a different learning rate for each state/action combination based on the number of times visited:

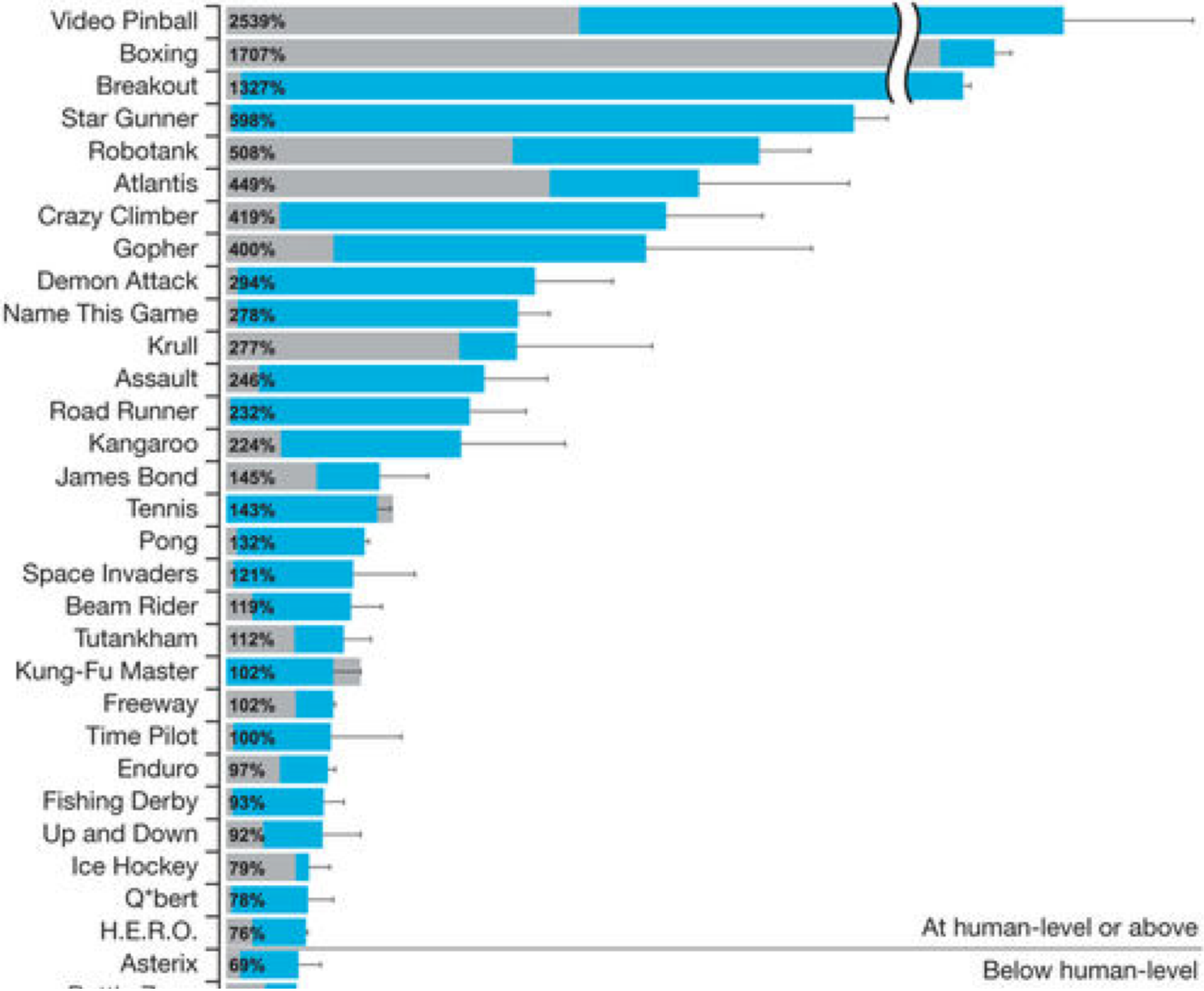
$$\alpha(x,a) = \frac{1}{n(x,a)}$$

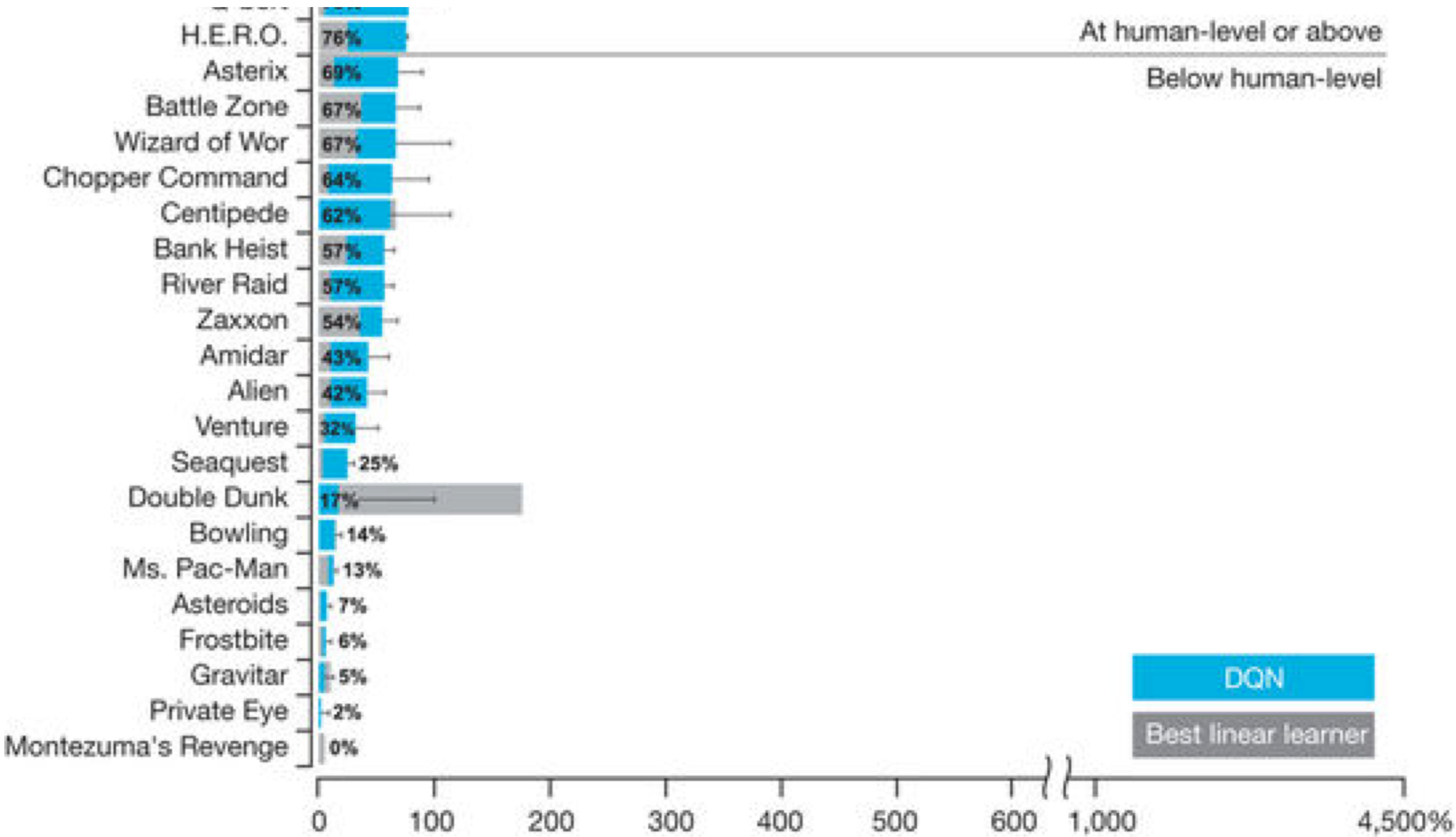
Problems with RL

- Clearly defined states and actions
- Infinite visits to each state
- No look-ahead
  - States with no exit

Reinforcement Learning for Classic Arcade Games

- 2014 Nature paper from Google's Deep Mind project
- Input: Pixels & score (reward)
- Output: joystick controls & button press

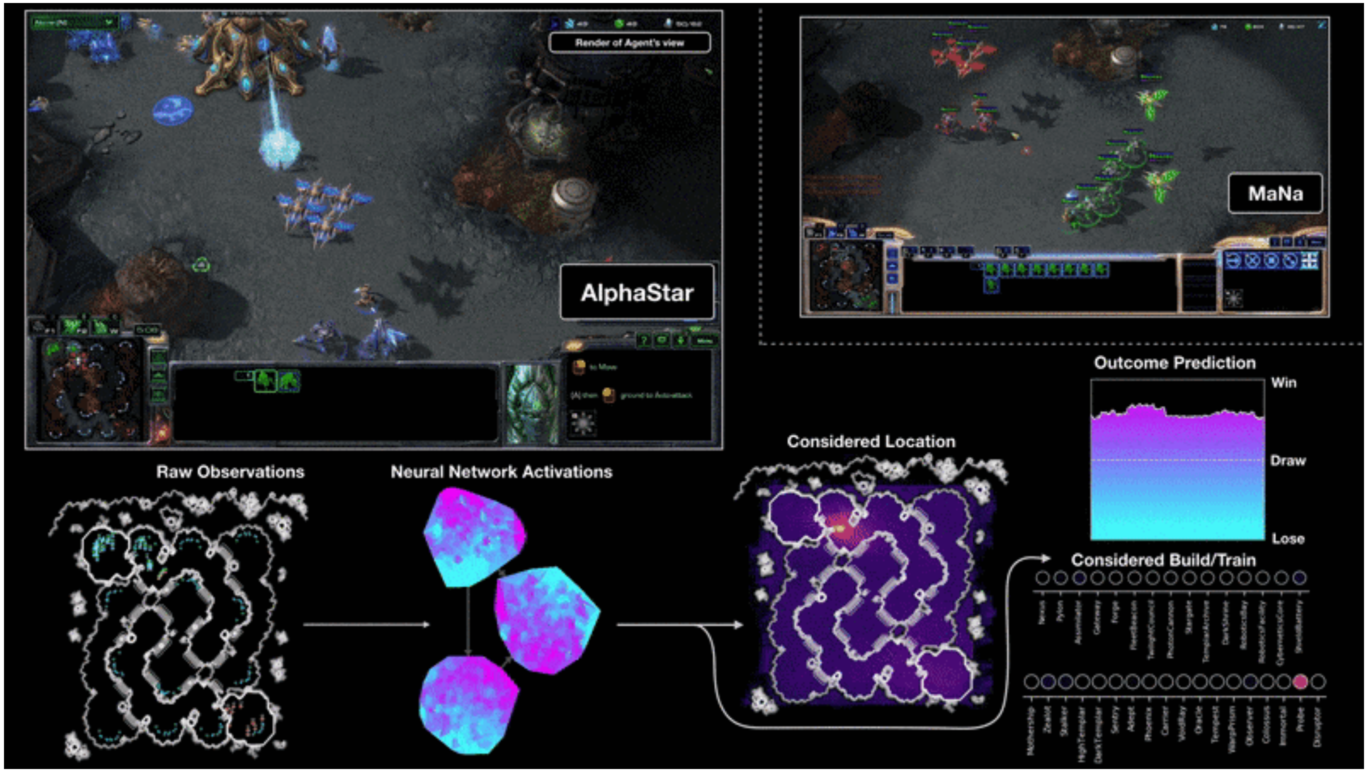

Video Pinball
2539%
Boxing
1707%
Breakout
1327%
Star Gunner
598%
Robotank
508%
Atlantis
449%
Crazy Climber
419%
Gopher
400%
Demon Attack
294%
Name This Game
278%
Krull
277%
Assault
246%
Road Runner
232%
Kangaroo
224%
James Bond
145%
Tennis
143%
Pong
132%
Space Invaders
121%
Beam Rider
119%
Tutankham
112%
Kung-Fu Master
102%
Freeway
102%
Time Pilot
100%
Enduro
97%
Fishing Derby
93%
Up and Down
92%
Ice Hockey
79%
Q*bert
78%
H.E.R.O.
76%
Asterix
69%
At human-level or above
Below human-level



H.E.R.O.
Asterix
Battle Zone
Wizard of Wor
Chopper Command
Centipede
Bank Heist
River Raid
Zaxxon
Amidar
Alien
Venture
Seaquest
Double Dunk
Bowling
Ms. Pac-Man
Asteroids
Frostbite
Gravitar
Private Eye
Montezuma's Revenge
76%
69%
67%
67%
64%
62%
57%
57%
54%
43%
42%
32%
25%
17%
14%
13%
7%
6%
5%
2%
0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%

Why does RL fail on Montezuma's Revenge?

Starcraft II



- Real-time, imperfect information, long-term planning and reward, ~100 decisions per second
- Beat 4 top human players in Dec 2018, 10 to 1
- Trained with RL and Deep networks

Relative Complexity

Game	Board Size	State-Space Complexity	Year defeated
Tic Tac Toe	9	10^{3}	1952*
Connect 4	42	10^{13}	1995*
Backgammon	28	10^{20}	1979
Chess	64	10^{47}	1997
Go (19x19)	361	10^{170}	2015
Heads up NL Holdem	N/A	10^{180}	2017
StarCraft II	N/A	10^{1685}	???

@LIV_BOEREE

Administrivia

- PS 5 due tonight
- PS 6 out today/tomorrow
- This week: Unsupervised Learning and Natural Language Processing