Reinforcement Learning

2018-04-10

Poll

- Who is familiar with reinforcement learning?
- Who feels able to implement Q learning?

https://www.technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learning/
https://www.youtube.com/watch?v=V1eYniJ0Rnk

Announcements

- Extra Credit 1: MOBA competition: Due Apr 22, 23:55
- Extra Credit 2
  - The plan is to run the study the weeks of April 16th and 23rd.
  - https://www.cc.gatech.edu/~surban6/2018sp-gameAI/extra-credit.html
  - Email mguzdial@gmail.com if interested, cc me
- HW7 done; HW 8 is posted (also due Apr 22, 23:55)

N-2: Player models

1. What is a player model? What does it allow?
2. What are two high-level categories of modeling?
3. What are a couple major types within the first category?
4. What are ways to get a player model?

N-1: PCG concluded

1. What are the 4 high-level forms of PCG we discussed?
2. We described cellular automaton and agents as _____
3. Discuss the relationship of the evaluation/fitness function and (a) player models, and (b) designer preferences
4. L-systems are a form of ____, and are particularly useful for ____
5. What is the rewriting order of L-systems, and why does this matter?
6. Explain collapse and propagation in CSP, and the effect on search
7. What class of techniques have we seen before that seem particularly suited to quest and story generation?

Reinforcement Learning: An introduction.
Richard Sutton and Andrew Bartow, 2nd ed, MIT Press, 2017.
http://incompleteideas.net/book/the-book-2nd.html

REINFORCEMENT LEARNING

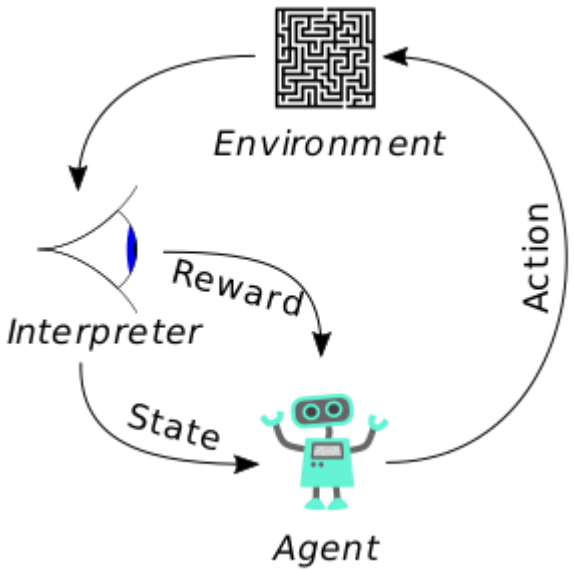
Situating this Lecture

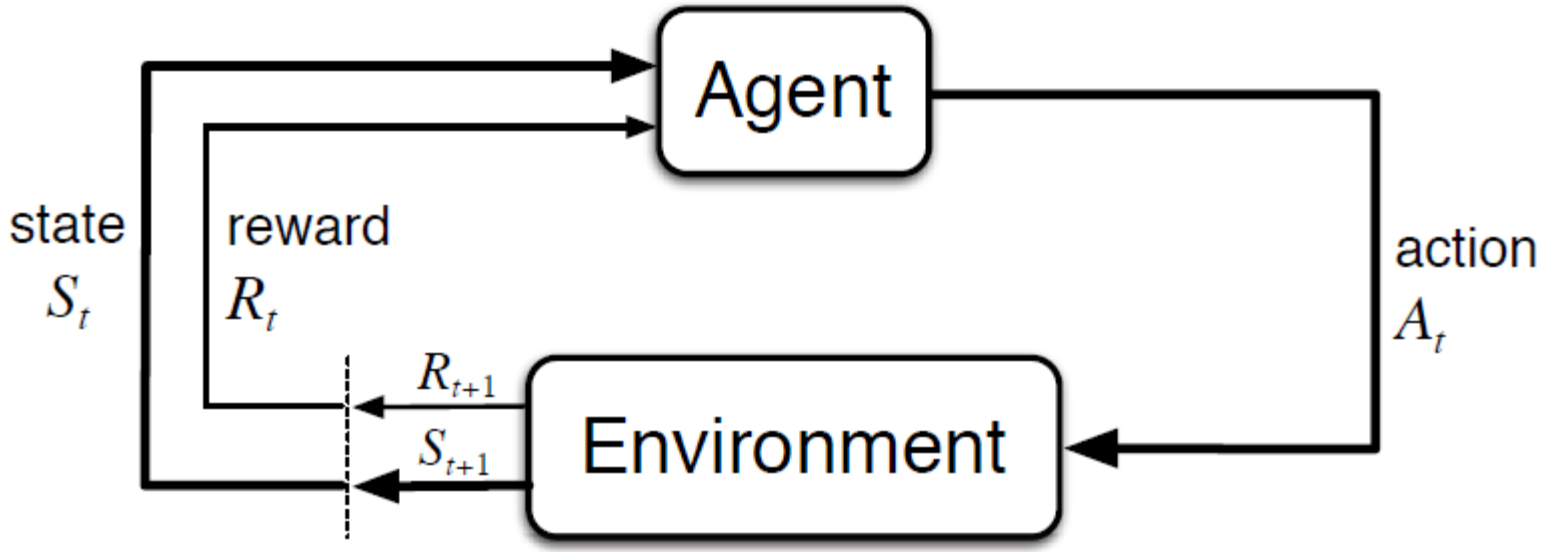
- At the beginning of the semester I told you there were three major branches of Game AI
  - AAA
  - Indie
  - Academic

- Besides some occasional weird stuff (Black & White), we are now totally in the realm of academic Game AI.

Situating RL

- The BLUF: RL is about learning from interaction about how to map situations to actions to maximize reward. Key elements:
  - Trial-and-error search: must discover which actions yield most reward by trying them
  - Delayed reward: actions may affect both immediate reward and also next situation and all subsequent rewards
- Different from:
  - Supervised learning: training set of labeled examples
  - Unsupervised learning: finding latent structure in unlabeled data
- RL: maximize reward signal rather than discover hidden structure





By Megajuice - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=57895741

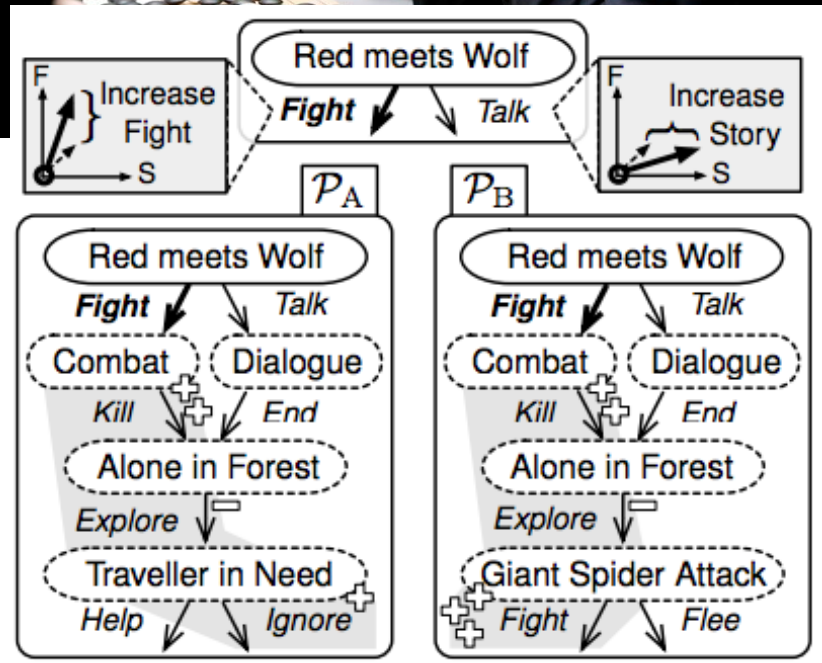
Sutton & Bartow, Fig 3.1

Fundamental Classes of Methods

- Dynamic programming
  - Mathematically well understood but require a complete and accurate model of the environment
- Monte Carlo methods
  - Model free and conceptually simple, but not well suited for step-by-step incremental computation
- Temporal-difference learning
  - Model free and fully incremental, but difficult to analyze

Also differ in efficiency/speed of convergence

Reinforcement Learning in Games



Reinforcement Learning: An Introduction

Richard S. Sutton
and Andrew G. Barto

Second Edition, in progress
MIT Press, Cambridge, MA, 2017

Online draft New Code Solutions Course Materials

Scholarly articles for **sutton and barto reinforcement learning**

Reinforcement learning: An introduction - **Sutton** - Cited by 25899

Reinforcement learning is direct adaptive optimal ... - **Sutton** - Cited by 348

http://incompleteideas.net/book/the-book-2nd.html

One-armed Bandit Problem

Scholarly articles for **sutton and barto reinforcement learning**

Reinforcement learning: An introduction - **Sutton** - Cited by 25899

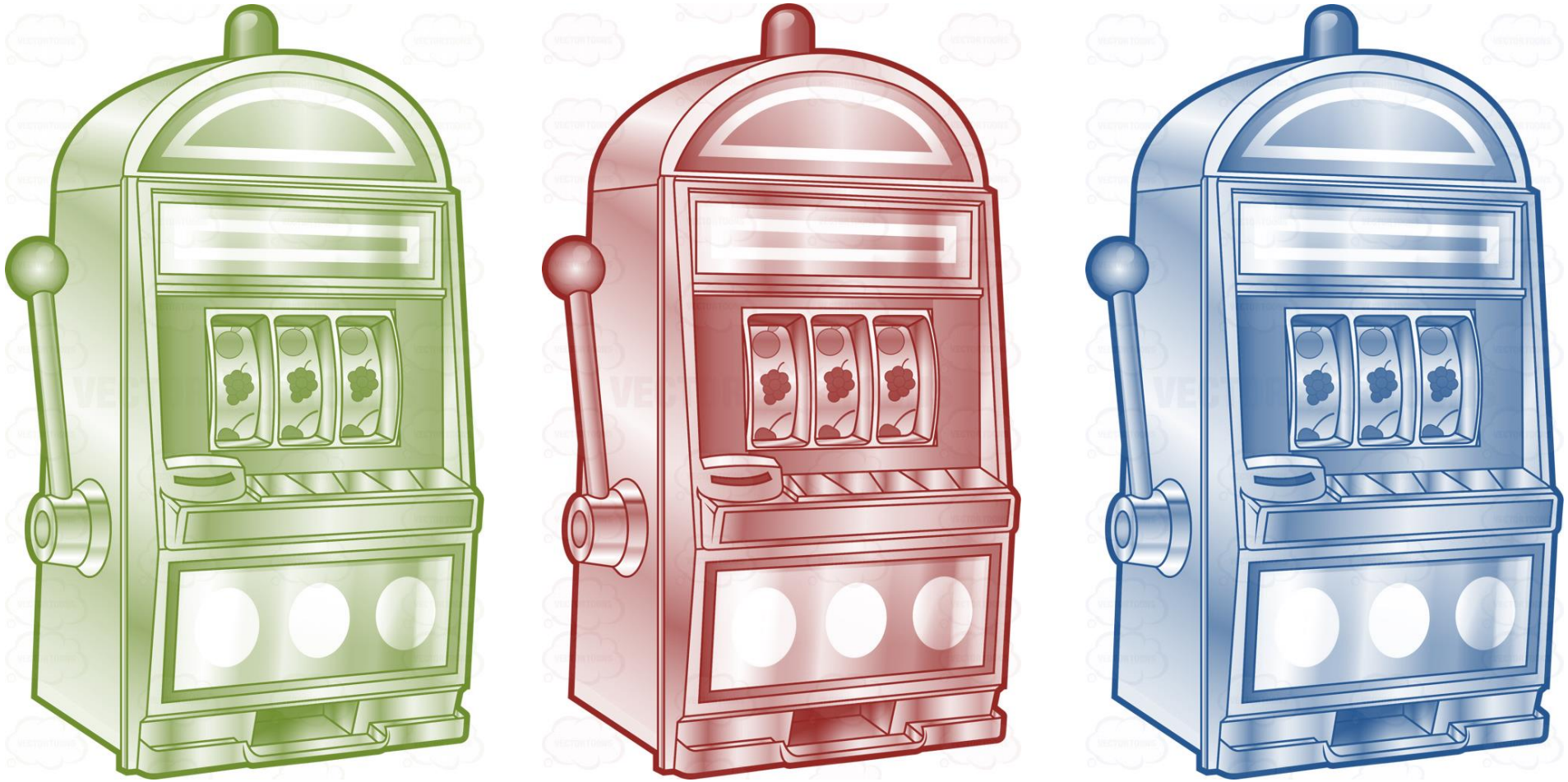
Reinforcement learning is direct adaptive optimal ... - **Sutton** - Cited by 348

One-armed Bandit Process

Pull #	Response	Believed Probability of Jackpot
1	WIN (1.0)	1.0
2	LOSS (0.0)	0.5
3	LOSS (0.0)	0.33…
4	LOSS (0.0)	0.25
…	…	…
N.	LOSS (0.0)	0.1

Unknown True Probability

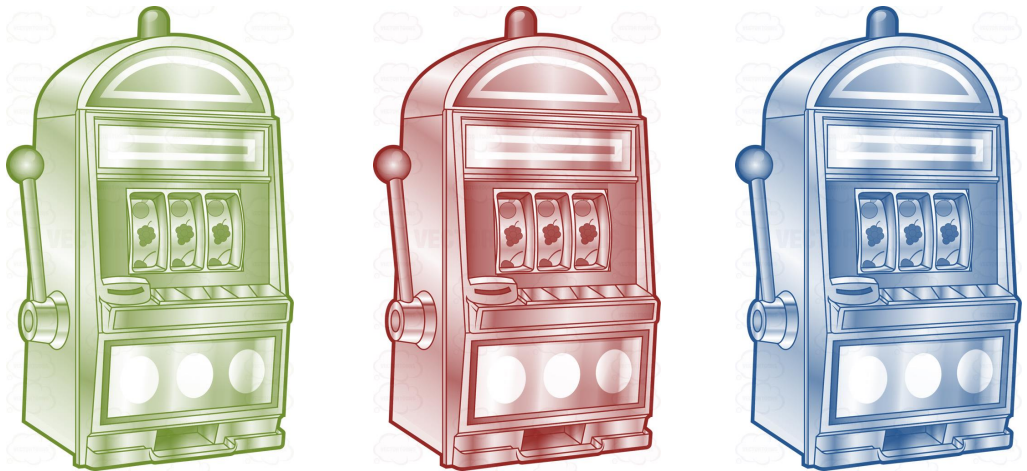
Multi-armed Bandit Problem



“Through repeated action selections you are to maximize your winnings by concentrating your actions on the best levers.” – Sutton & Barto

Now what?

- We can’t just keep playing one bandit to figure out its potential for reward (money)
- Want to maximize reward across all bandits
- We need to trade off making money with current knowledge and gaining knowledge
  - *Exploitation vs. Exploration*

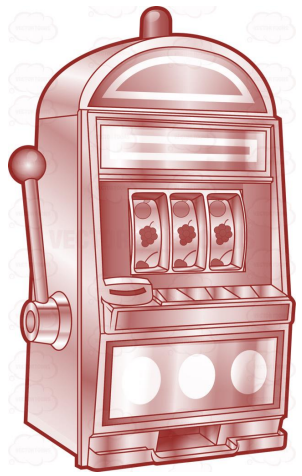


Just Exploitation

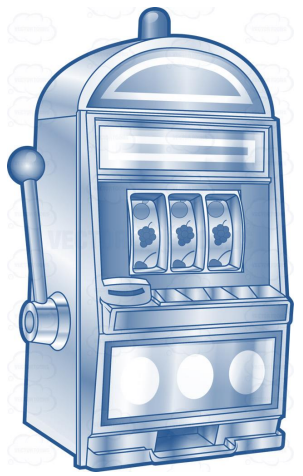
Greedy Strategy where we always pick the machine we think will give us the best reward (reminder: machines give rewards according to a probability)

Machine	**Response**	**Belief (G,R,B)**
B	LOSS	(-,-,0.0)
R	WIN	(0,1,0)
G	LOSS	(0,1,0)
R	LOSS	(0,0.5,0)
…	…	…

50%



25%



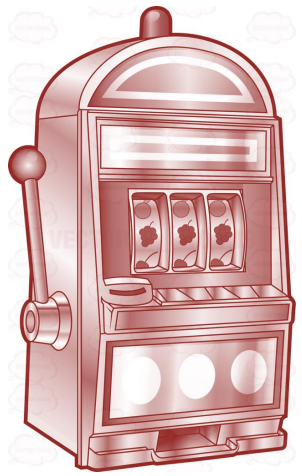
10%

Just Exploration

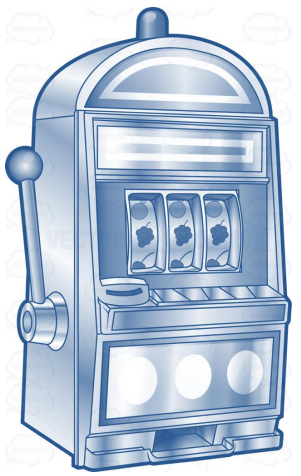
Always just pick a random machine, no matter what we believe.

Machine	Response	Belief (G,R,B)
B	LOSS	(-,-,0.0)
R	WIN	(0,1,0)
G	LOSS	(0,1,0)
B	**LOSS**	**(0,1,0)**
…	…	…

50%



25%



10%

Epsilon (ϵ) Greedy
(ϵ-Greedy)

(ϵ=0.1)

- 90% of the time try the best machine according to our current beliefs
- 10% of the time take random action

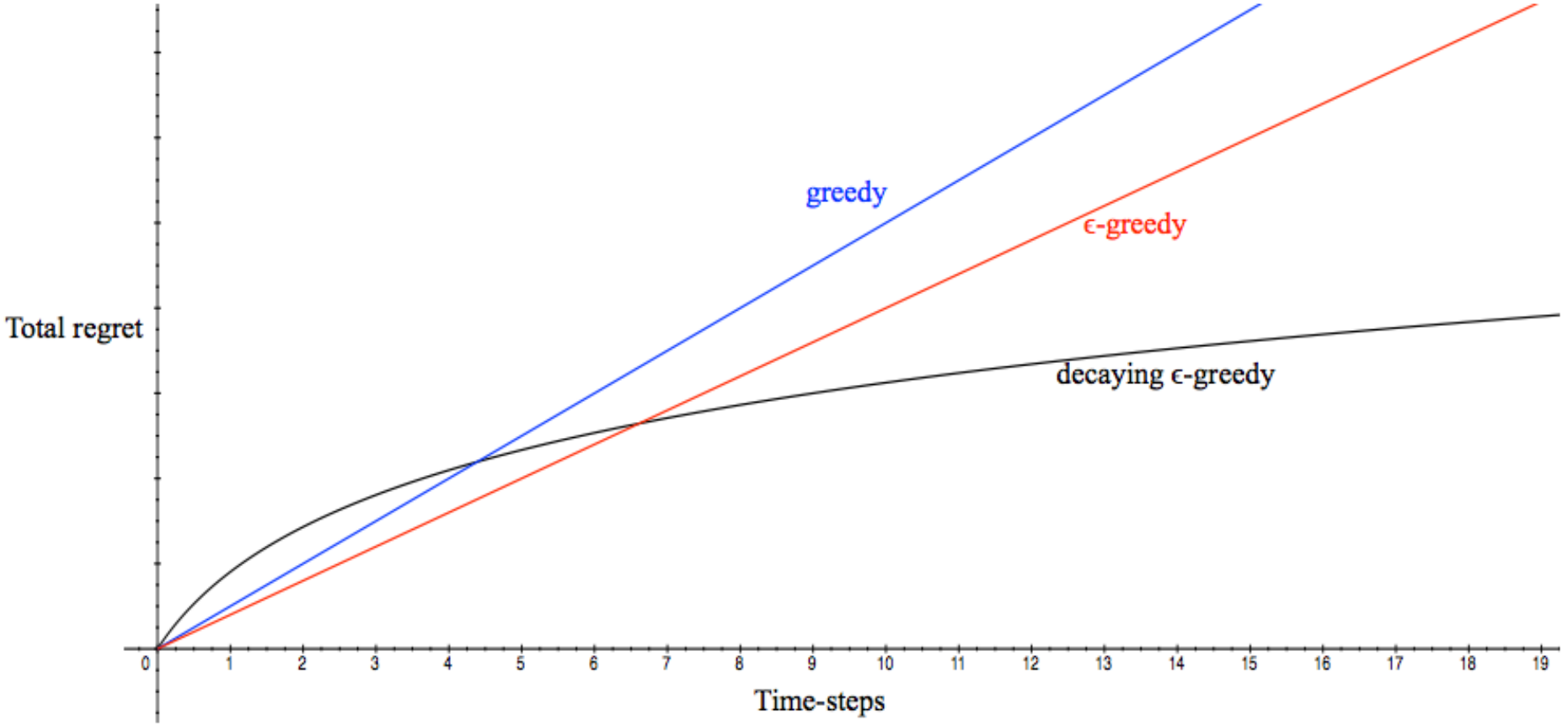
Now we can “escape” from early greedy mistakes!

Regret

- Imagine a perfect agent, who always takes the best action
- We can compare that agent's payout with our agent's payout for each action to get the *regret* for that step
- Summing regret across all time steps gives us the *total regret*

- Maximize total reward == minimize total regret

Different Strategies and Regret



Decaying ϵ-Greedy

- Drop ϵ lower and lower according to some schedule
- Now: logarithmic asymptotic regret

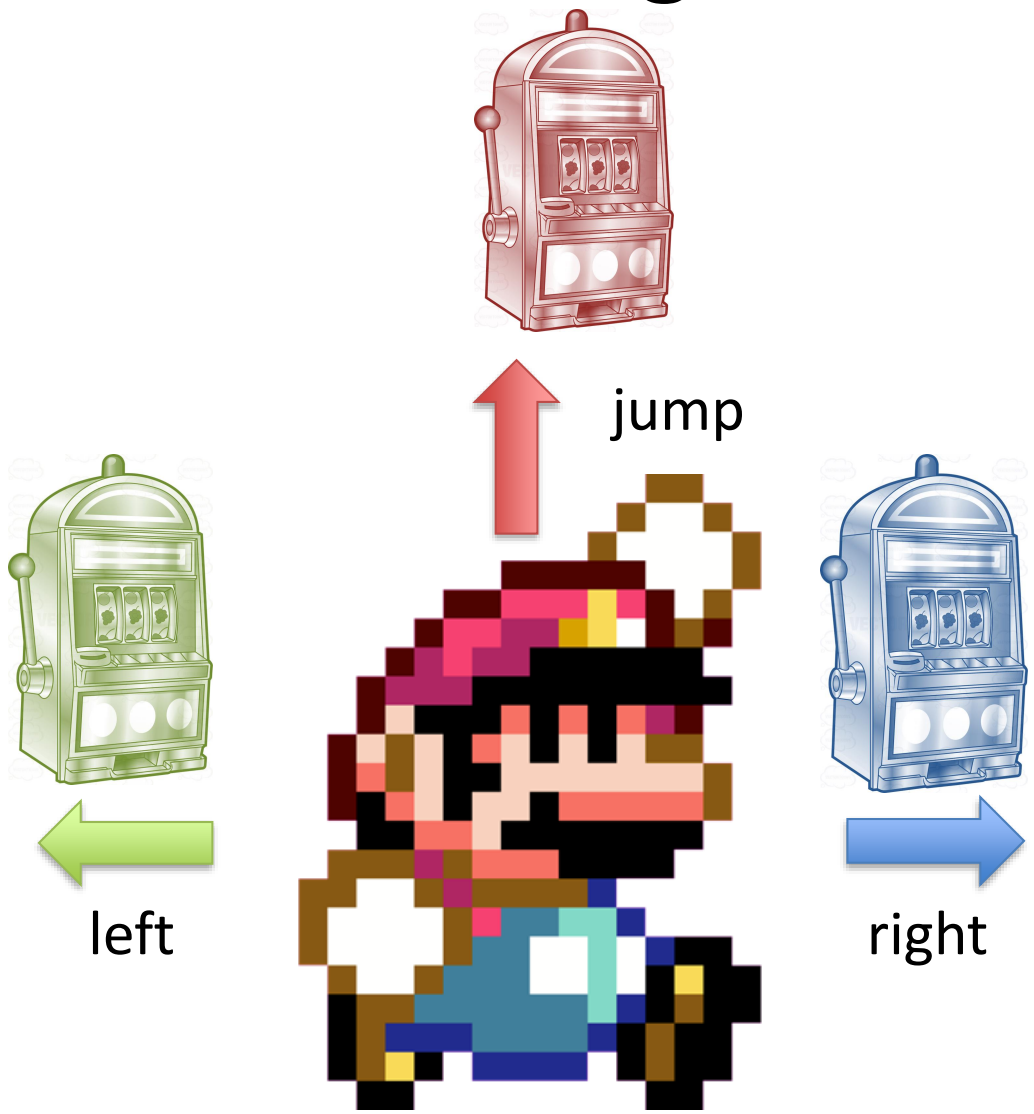
- Downside: we need to have enough domain knowledge to come up with a schedule beforehand

What does this have to do with games?

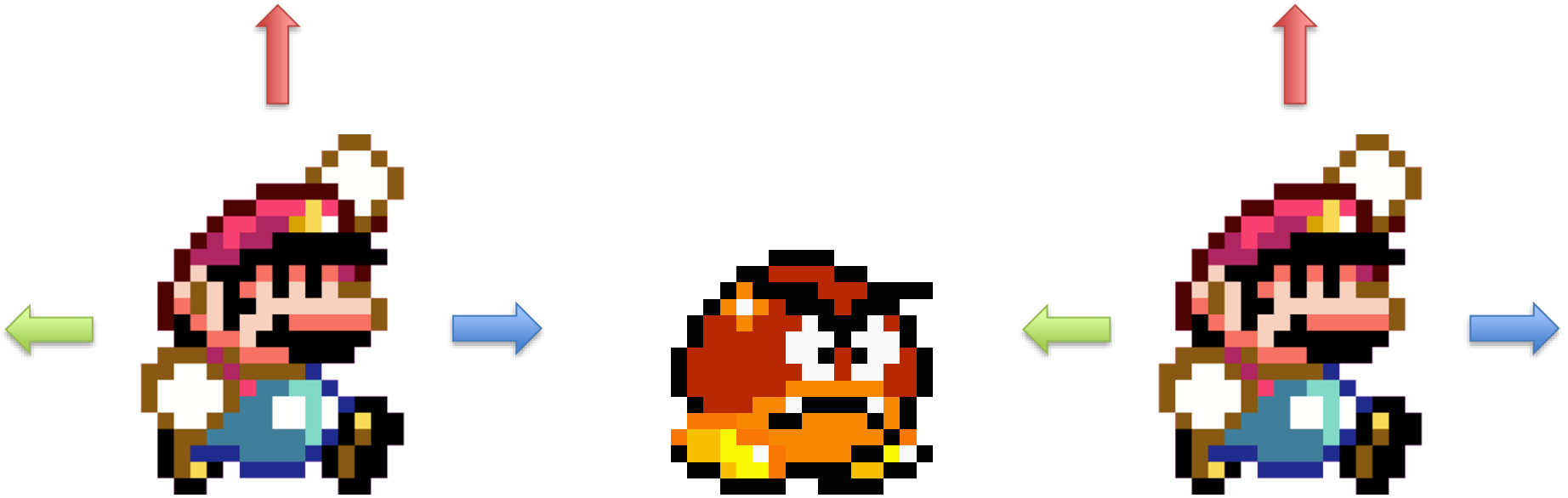
We can imagine mapping this onto a video game domain

Each distinct machine is now a distinct action

Reward: Not dying, end of level, etc.



But these values will probably differ based on context…



Value of going left versus right will be different if an enemy is to the left or the right.

State

- Multi-armed bandit problem is a “single state” or “stateless” **Markov Decision Process**

- But in games (and most cases we care about) there is more than one state

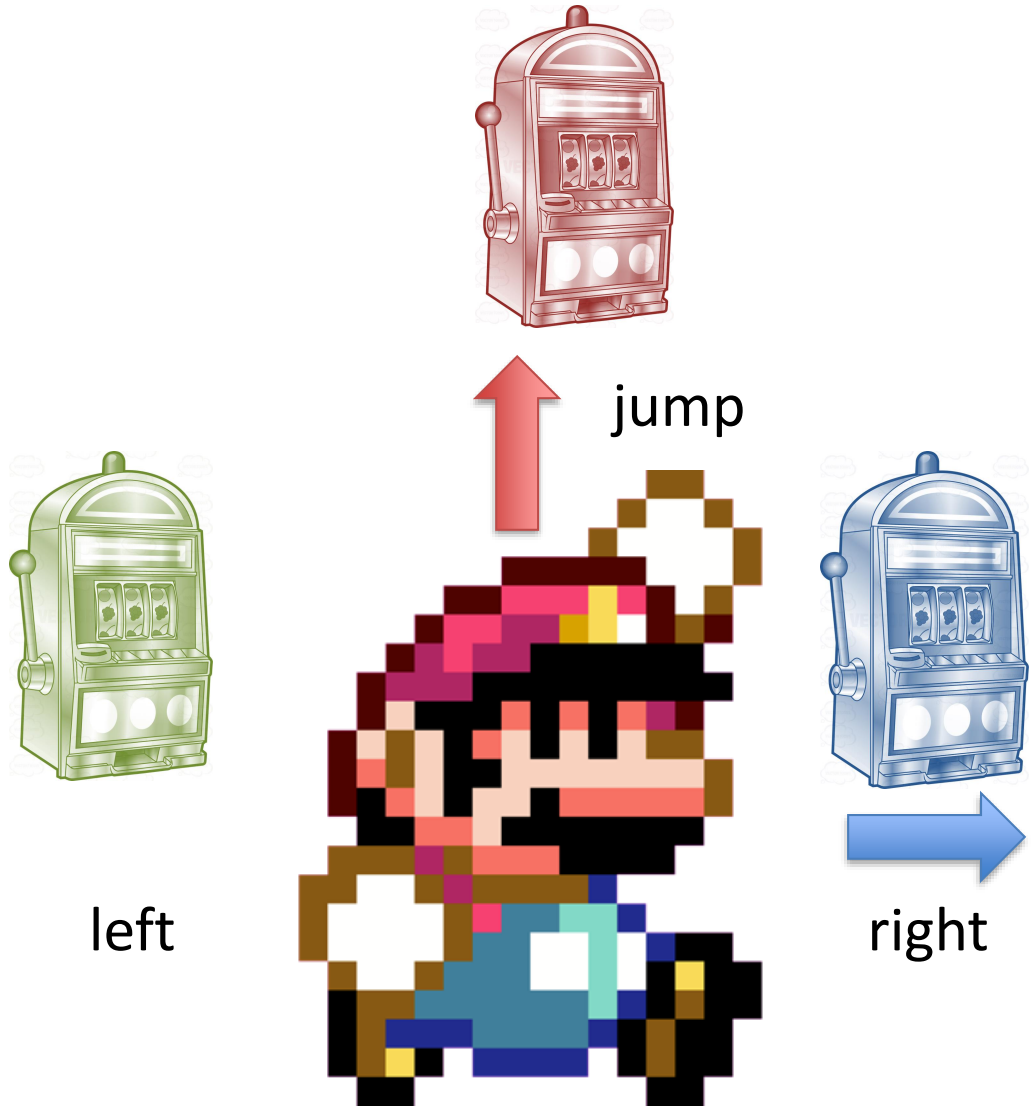
Andrey Markov

Markov Decision Process

- *S*: finite set of states
- *A*: finite set of actions
- $P(s,s_1)$: Probability that action a takes us from state s to state s_1
- $R(s,s_1)$: Reward for transitioning from state s to state s_1
- *y:* Discount factor ([0-1]). Demonstrates the difference in importance between future awards and present awards

High-level Idea

If the multi-armed bandit problem was a single state MDP, we can think of learning a strategy to play a game as solving this problem for *every state of the game*



Markovian State

- We call a state representation *Markovian* if it has all the information we need to make an optimal decision

State Representation Examples

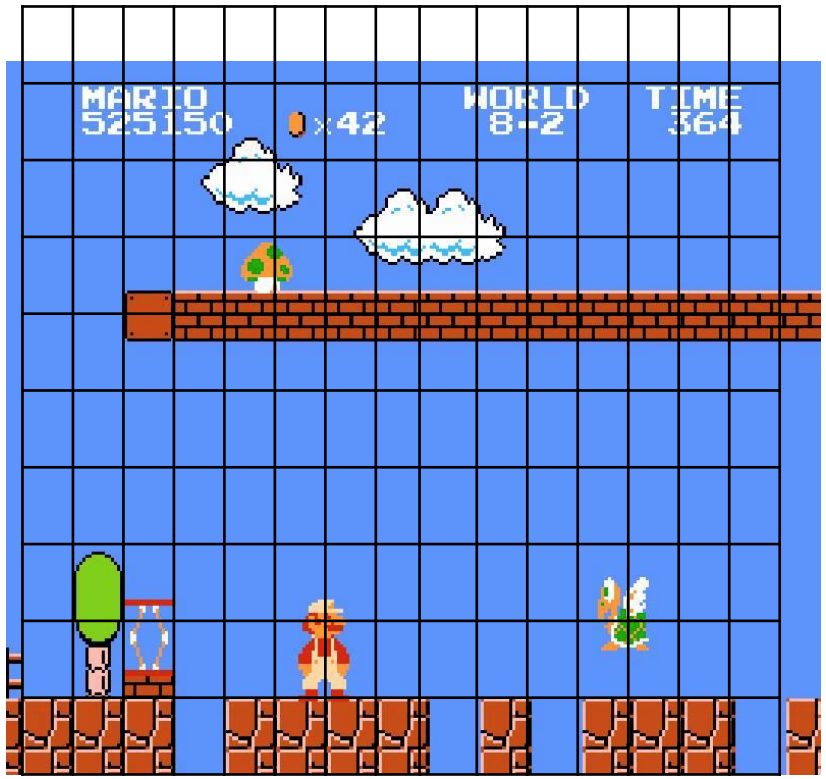
List of facts

- -enemy to right
- -powerup above
- -fire mario
- -on ground
- -etc…

State Representation Examples

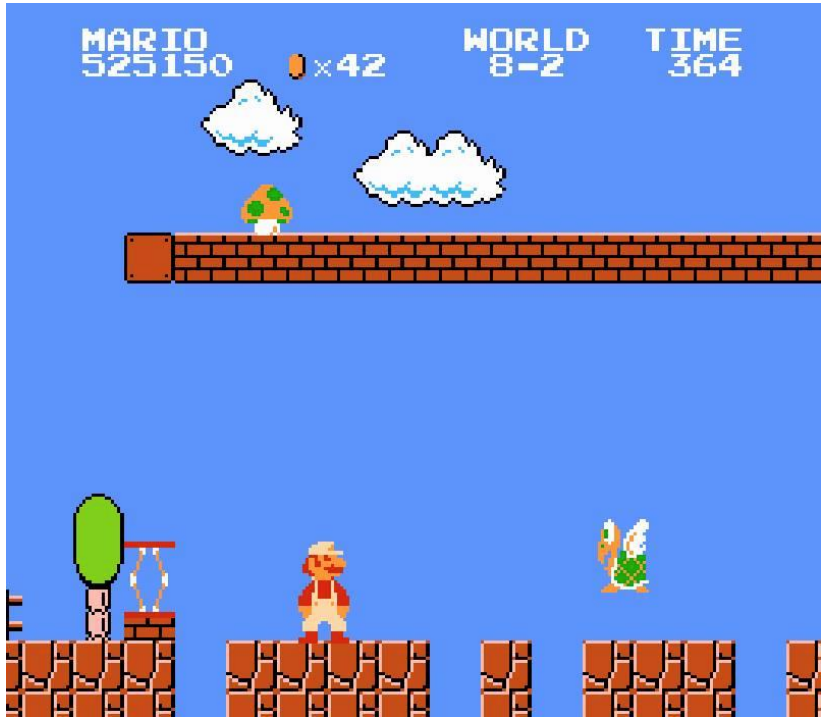
Grid Representations

- Segment all locations to tiles
- All enemies represented individually? Or just as “enemy”
- How much of the level to include in the state? Just screen? Smaller? Larger?



State Representation Examples

What pixels are present on the screen at this time?



State Representation Tradeoffs

- More complex state representations ensure that an agent has all info needed
- Less complex state representations speed up training (more states “look” the same)

- Goal: find the middle ground. Simplest state representation that still allows for optimal performance

Question

What state representations would you use for the following? Why?

- Tic-Tac-Toe
- A simple platformer
- A real time strategy game (Civilization/Starcraft)