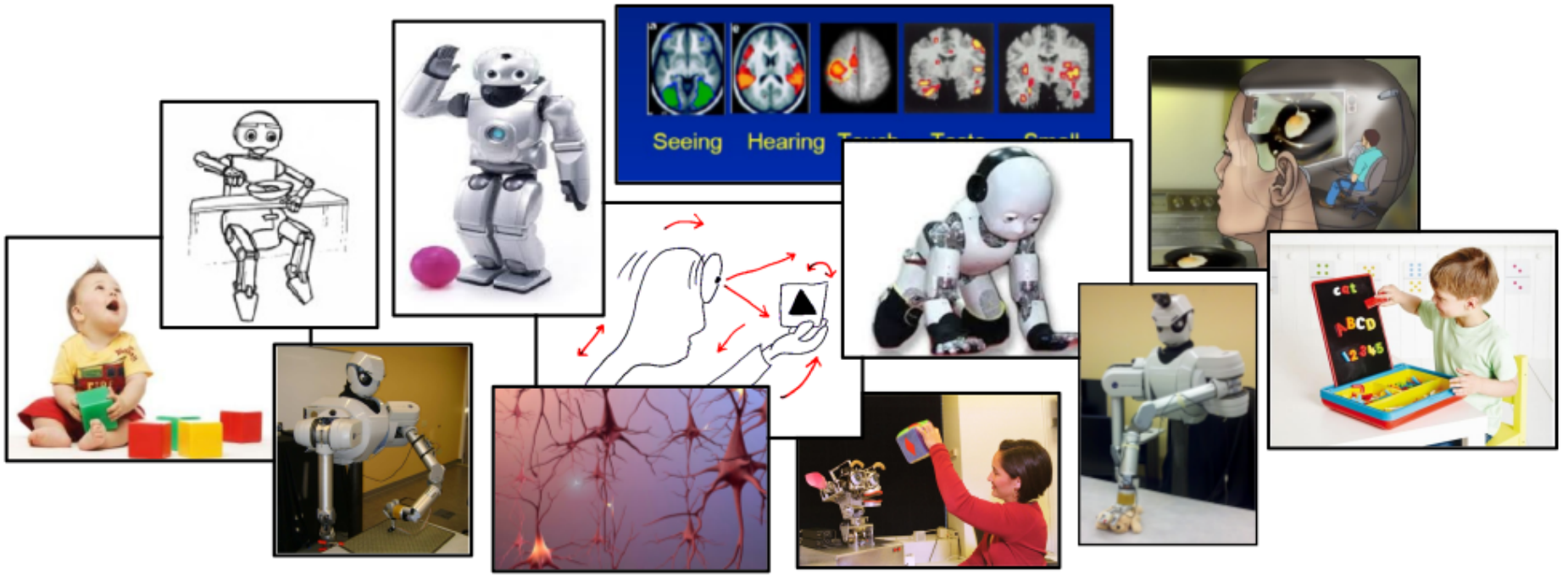
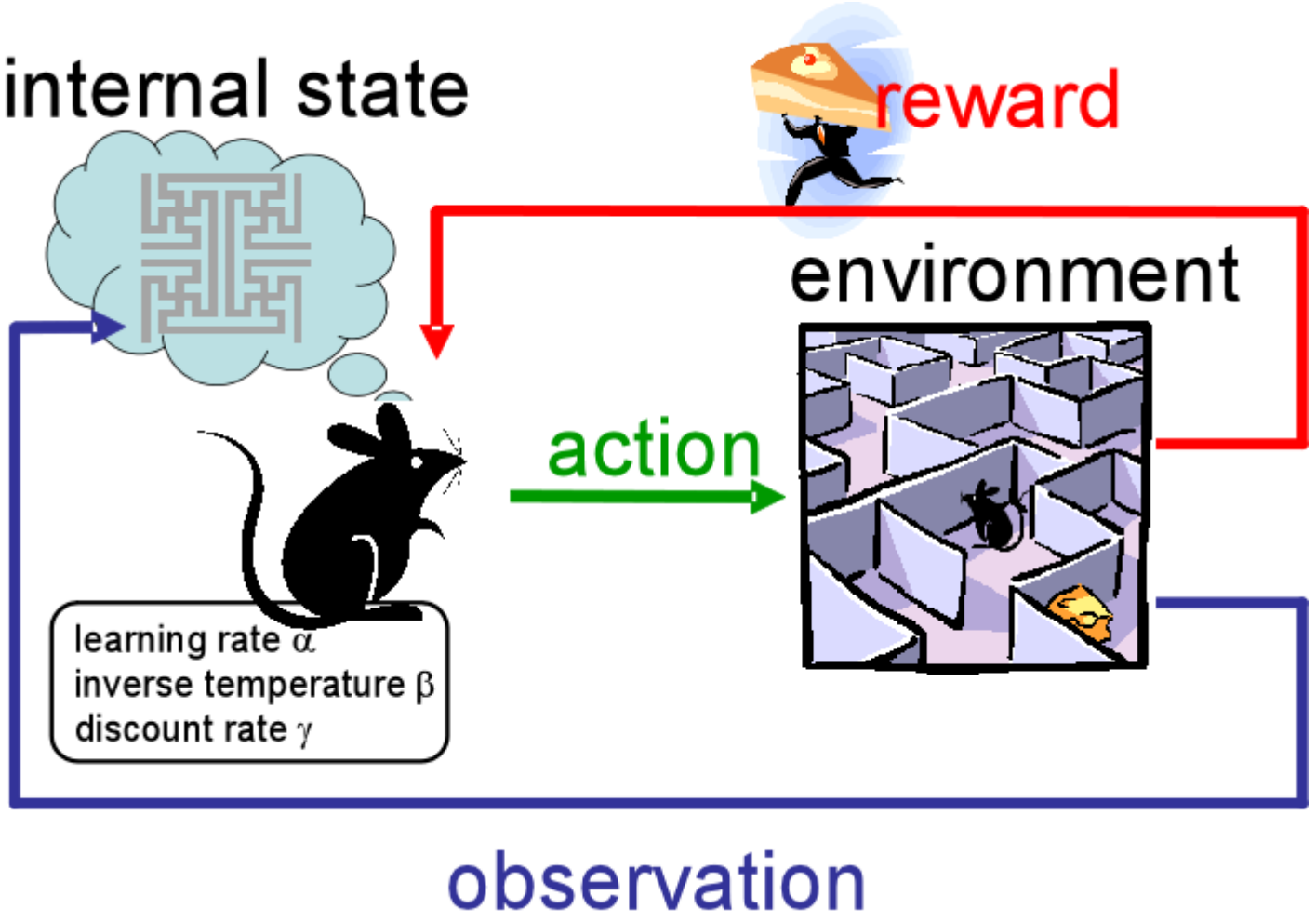


COMP 150: Developmental Robotics

Instructor: Jivko Sinapov
www.cs.tufts.edu/~jsinapov

Reinforcement Learning



Main Reference

Sutton and Barto, (2012). Reinforcement Learning: An Introduction, Chapter 1-3

http://ufal.mff.cuni.cz/~straka/courses/npfl114/2016/sutton-bookdraft2016sep.pdf

Activity: You are the Learner

At each time step, you receive an observation (a color)

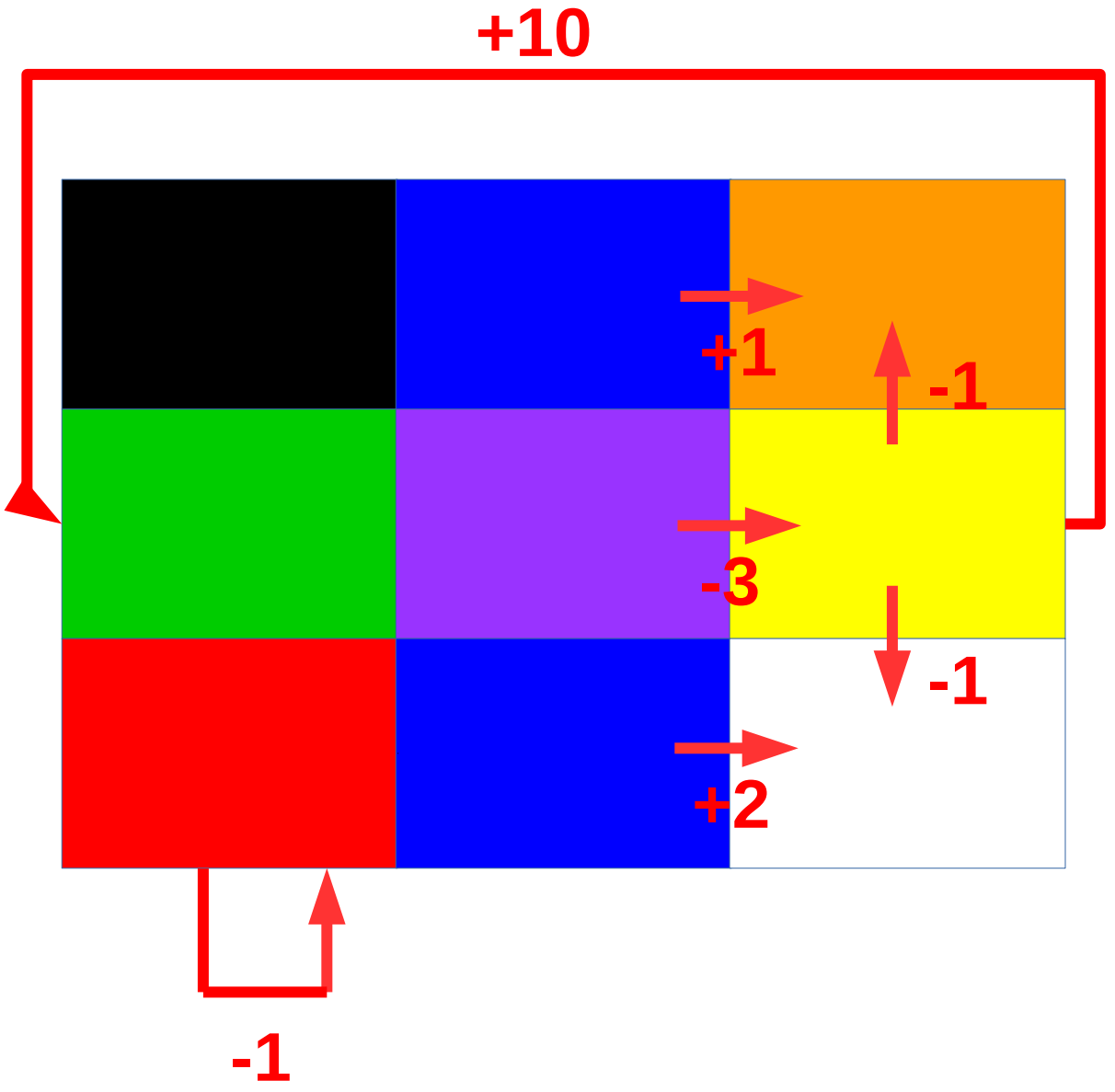
You have three actions: “clap”, “wave”, and “stand”

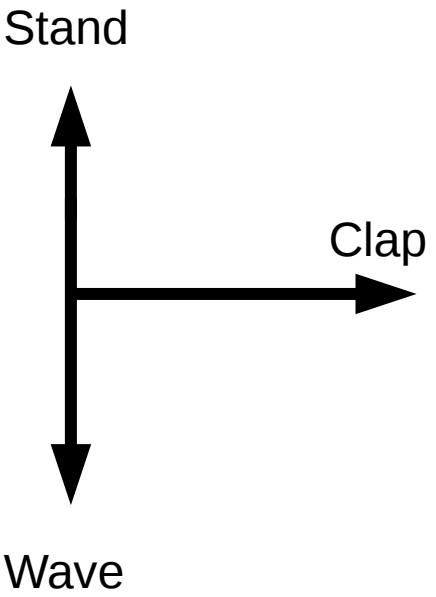
After performing an action, you may receive a reward (which could be negative :P)

How did you solve it?

The Actual Problem

The Actual Problem

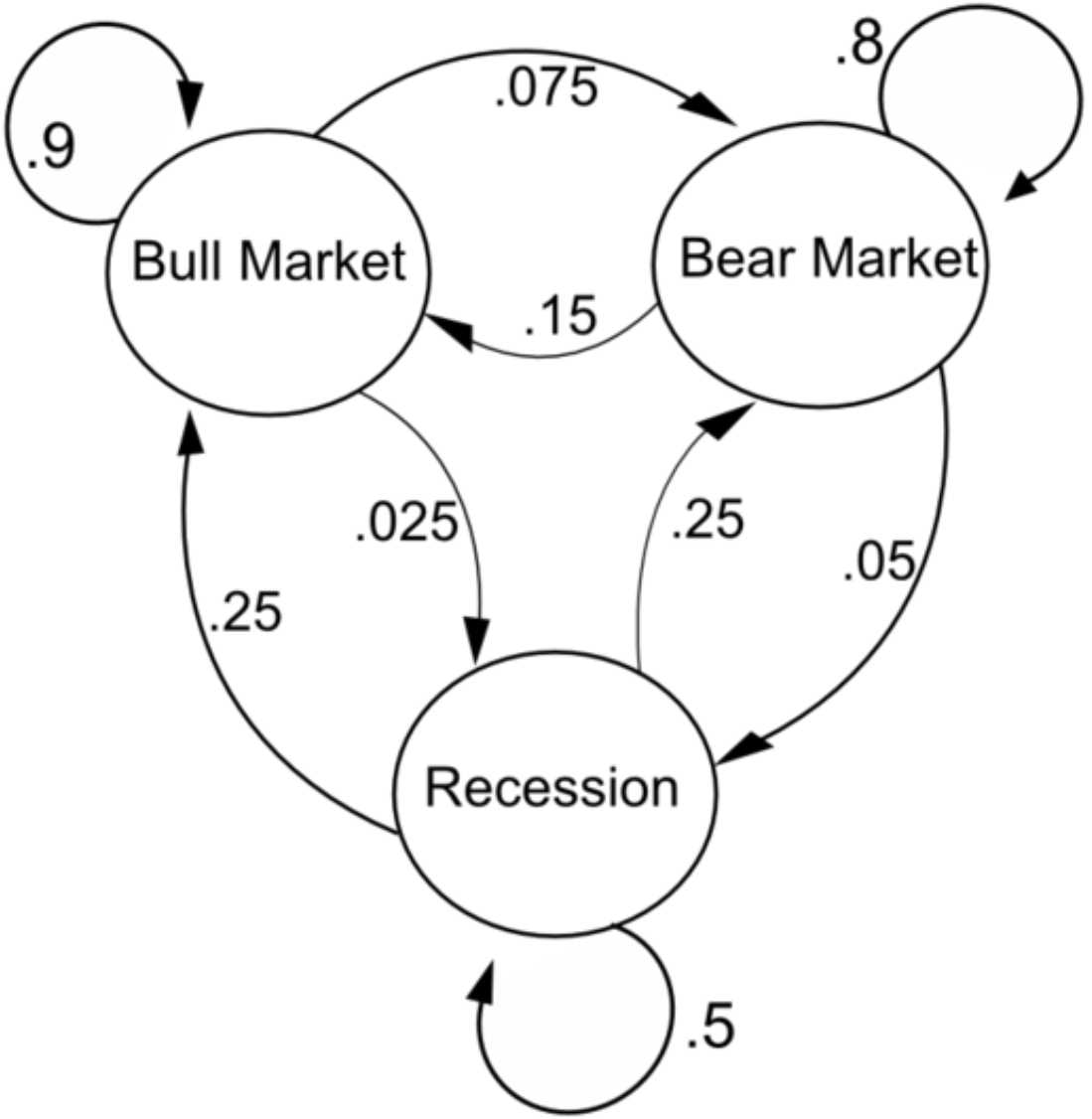




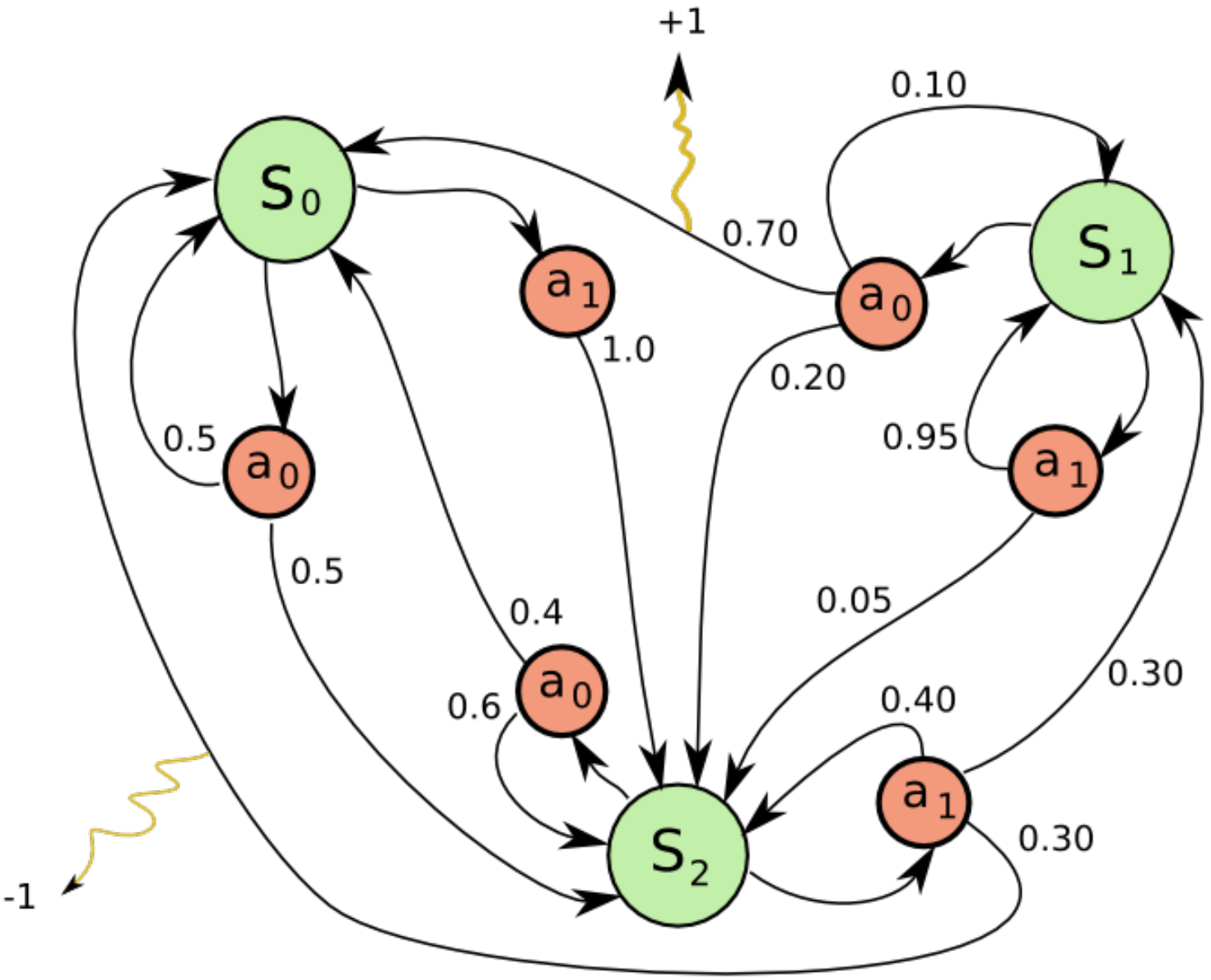
Andrey Andreyevich Markov (1856 – 1922)

[http://en.wikipedia.org/wiki/Andrey_Markov]

Markov Chain



Markov Decision Process



The Multi-Armed Bandit Problem

a.k.a. how to pick between Slot Machines (one-armed bandits) so that you walk out with the most $$$ from the Casino

Arm 1

Arm 2

. . . .

Arm k

How should we decide which slot machine to pull next?

How should we decide which slot machine to pull next?

0 1 3 0 1

0 0 0 50 0

How should we decide which slot machine to pull next?

1 with prob = 0.6 and 0 otherwise

50 with prob = 0.01 and 0 otherwise

Value Function

A value function encodes the “value” of performing a particular action (i.e., bandit)

Rewards observed when performing action *a*

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a}.$$

Value function Q

# of times the agent has picked action *a*

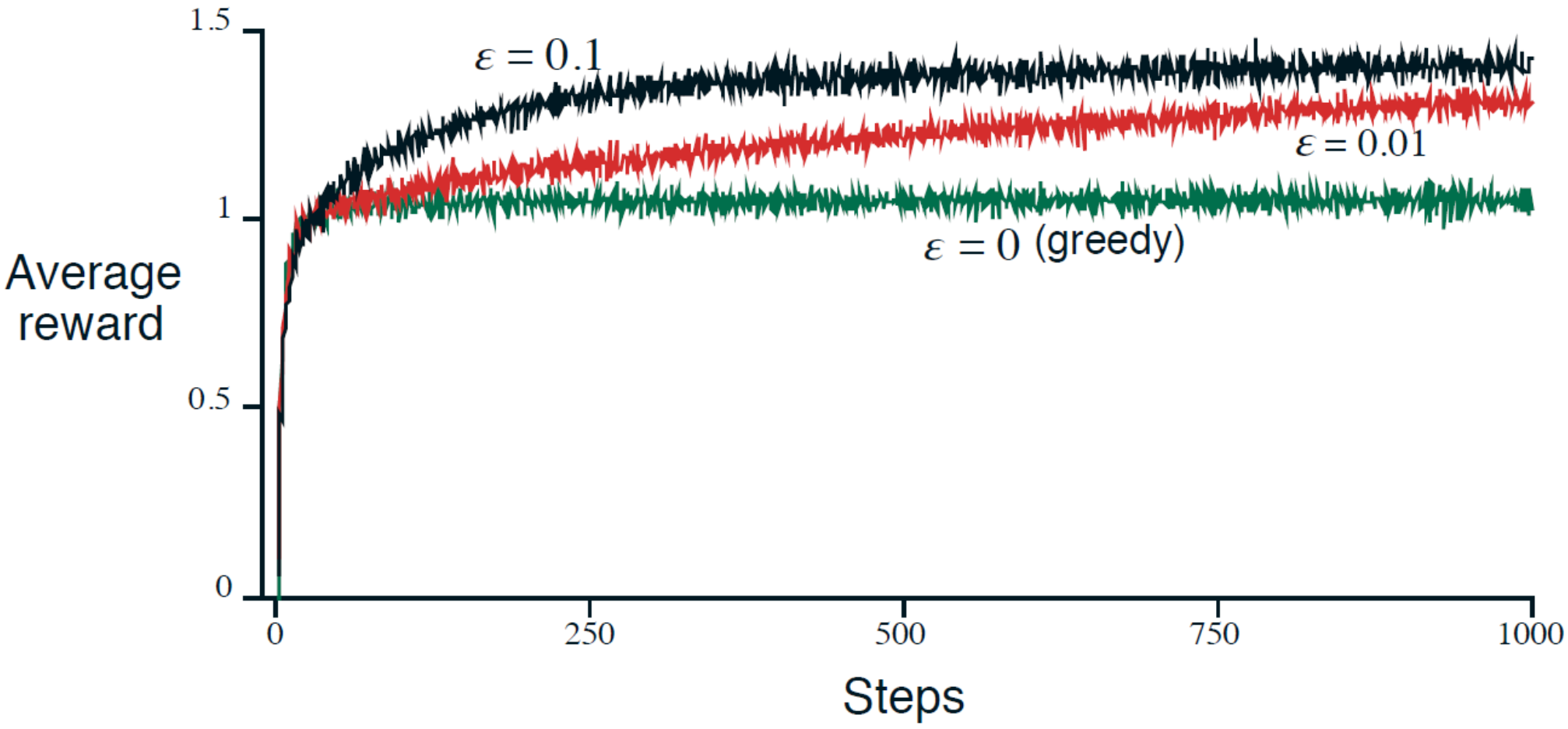
How do we choose next action?

- Greedy: pick the action that maximizes the value function, i.e.,

$$Q_t(A_t^*) = \max_a Q_t(a)$$

- ε-Greedy: with probability ε pick a random action, otherwise, be greedy

10-armed Bandit Example



Soft-Max Action Selection

Exponent of natural logarithm (~ 2.718)

$$\frac{e^{Q_t(a)/\tau}}{\sum_{i=1}^{n} e^{Q_t(i)/\tau}}$$

"temperature"

As temperature goes up, all actions become nearly equally likely to be selected; as it goes down, those with higher value function outputs become more likely

What happens after choosing an action?

Batch:

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a}$$

Incremental:

$$\begin{aligned}
Q_{k+1} &= \frac{1}{k}\sum_{i=1}^{k} R_i \\
&= \frac{1}{k}\left(R_k + \sum_{i=1}^{k-1} R_i\right) \\
&= \frac{1}{k}\Big(R_k + (k-1)Q_k + Q_k - Q_k\Big) \\
&= \frac{1}{k}\Big(R_k + kQ_k - Q_k\Big) \\
&= Q_k + \frac{1}{k}\Big[R_k - Q_k\Big],
\end{aligned}$$

Updating the Value Function

$$NewEstimate \leftarrow OldEstimate + StepSize \left[Target - OldEstimate \right]$$

What happens when the payout of a bandit is changing over time?

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a}$$

What happens when the payout of a bandit is changing over time?

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a}$$

Earlier rewards may not be indicative of how the bandit performs now

What happens when the payout of a bandit is changing over time?

$$Q_{k+1} = Q_k + \alpha \Big[R_k - Q_k \Big]$$

instead of

$$Q_k + \frac{1}{k} \Big[R_k - Q_k \Big]$$

How do we construct a value function at the start (before any actions have been taken)

How do we construct a value function at the start (before any actions have been taken)

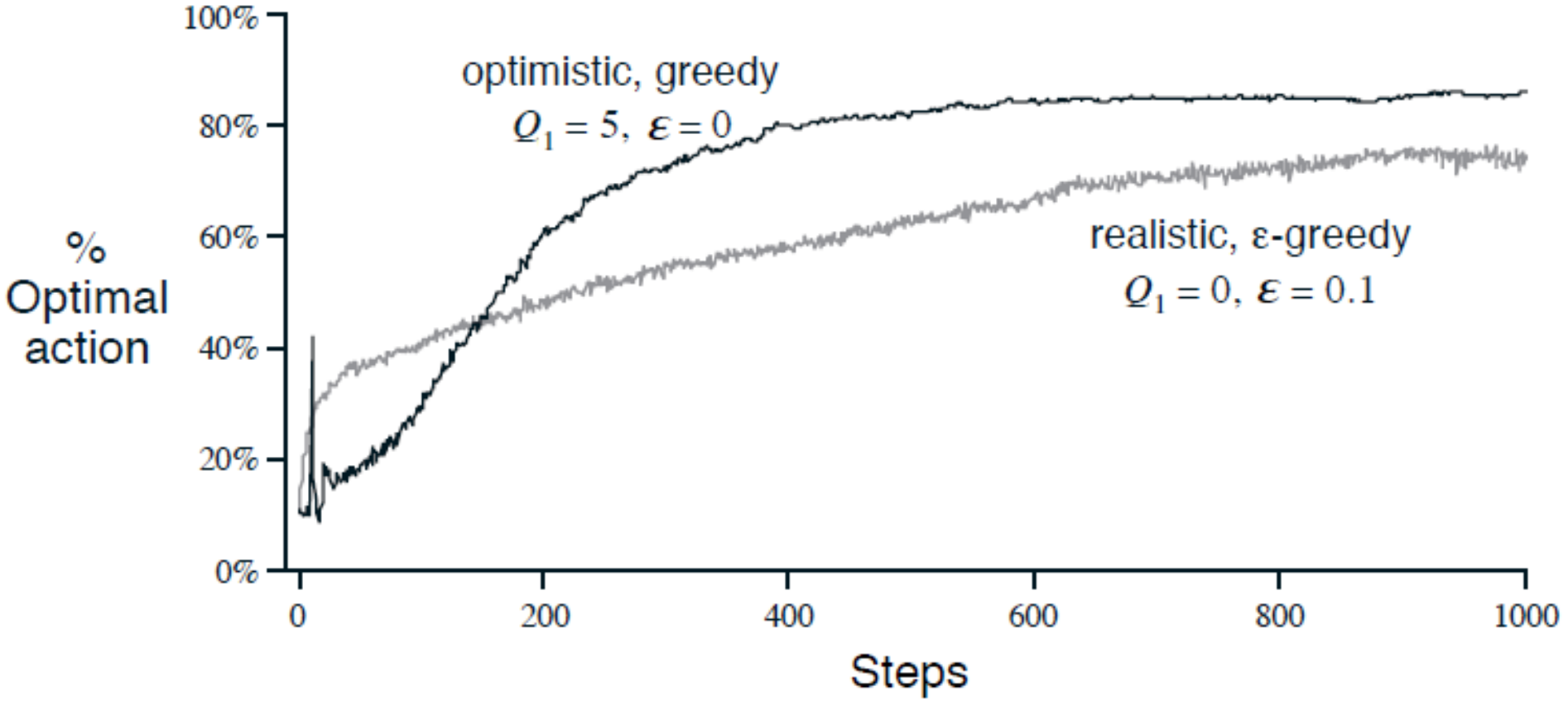
Zeros:	0	0	0
Random:	-0.23	0.76	-0.9
Optimistic:	+5	+5	+5

Arm 1

Arm 2

. . . .

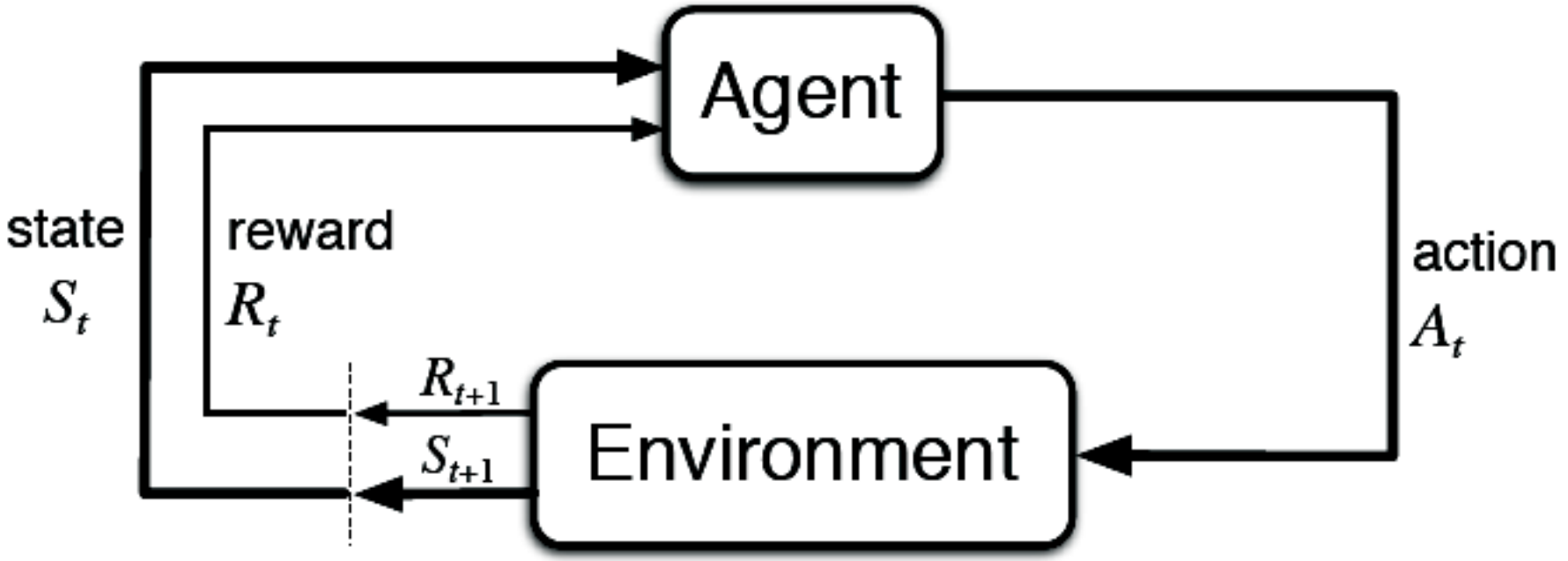
Arm k


100%
80%
60%
40%
20%
0%
% Optimal action
optimistic, greedy
$Q_1 = 5,\ \varepsilon = 0$
realistic, ε-greedy
$Q_1 = 0,\ \varepsilon = 0.1$
0
200
400
600
800
1000
Steps

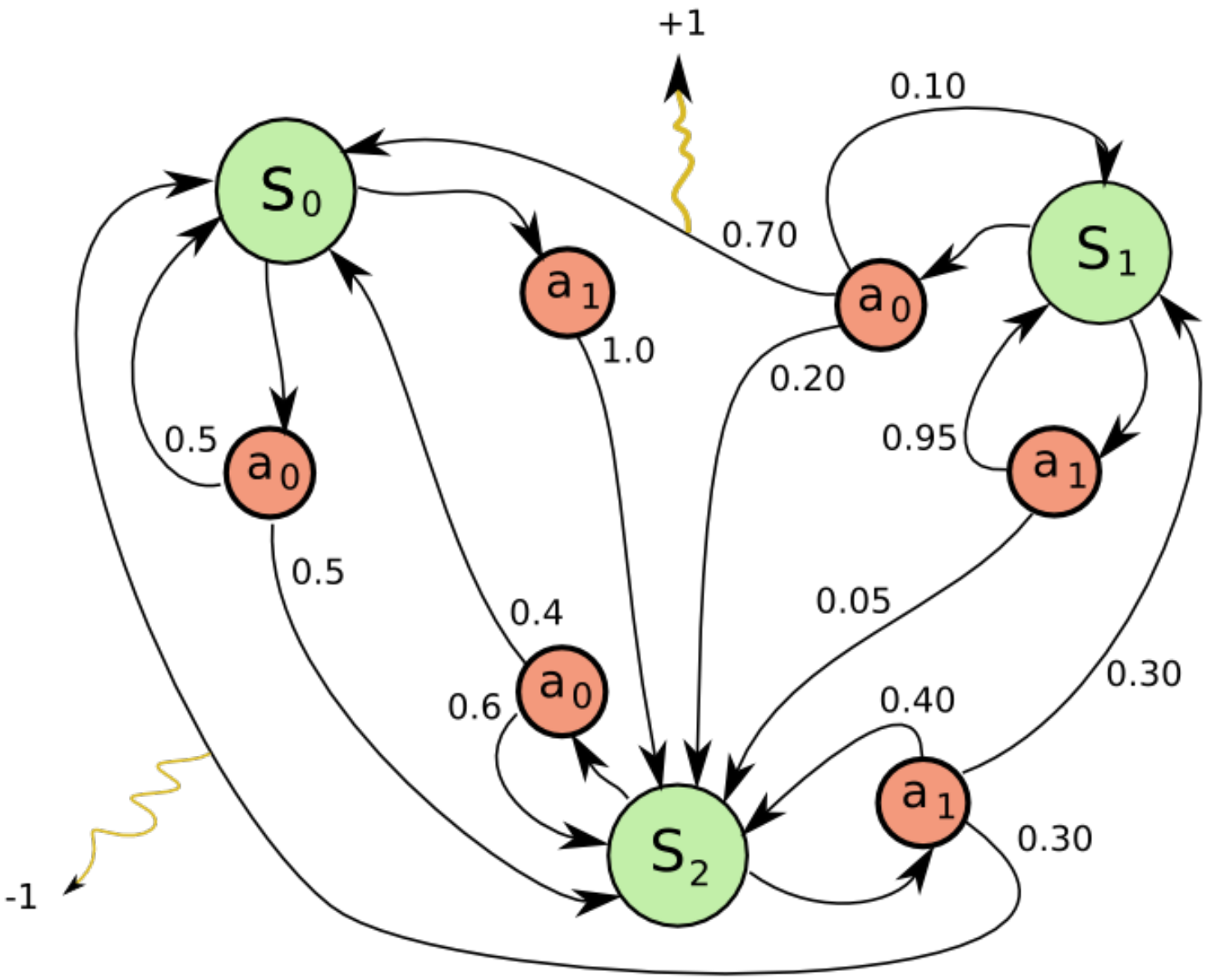
The Multi-Armed Bandit Problems

The casino always wins – so why is this problem important?

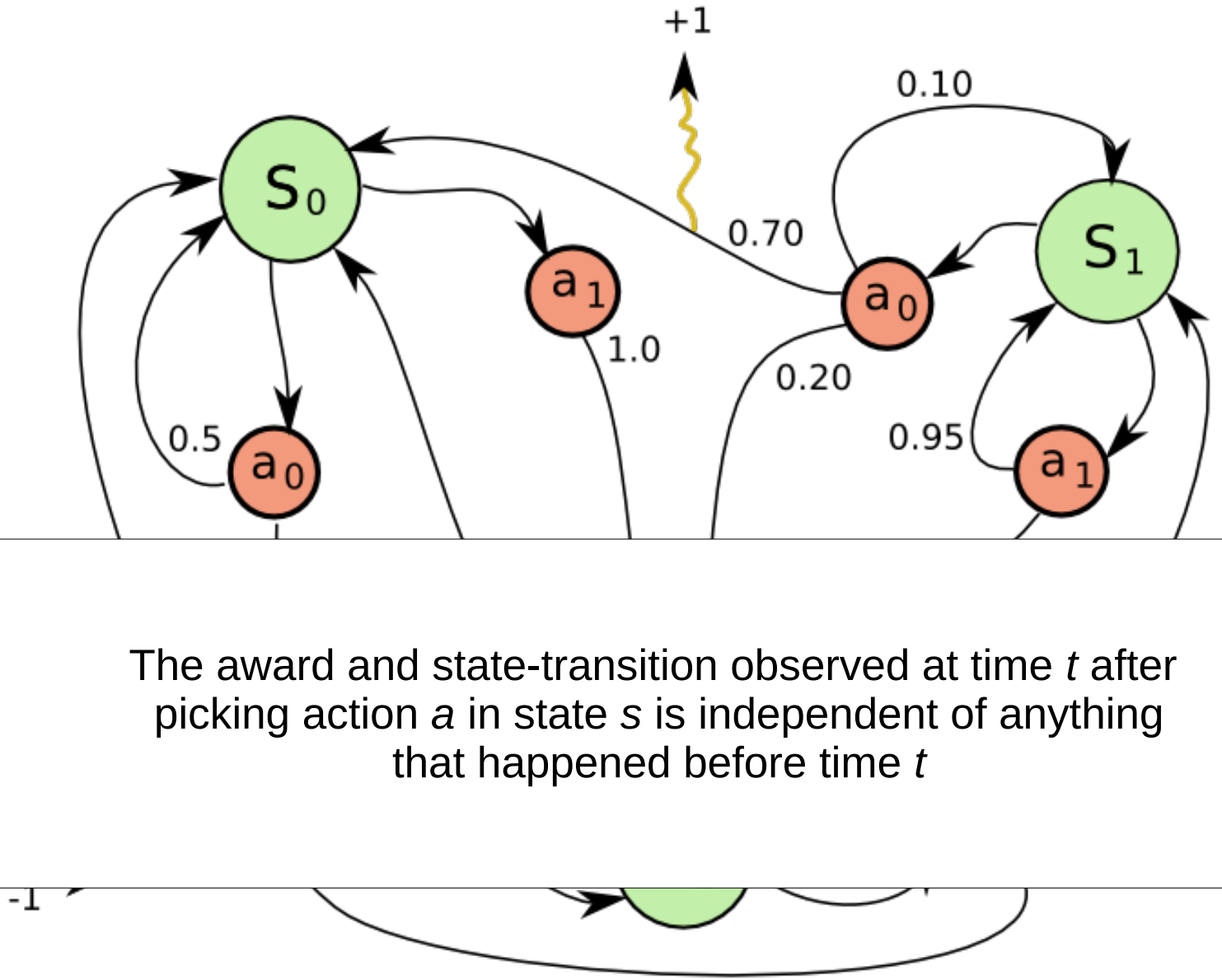
The Reinforcement Learning Problem



RL in the context of MDPs

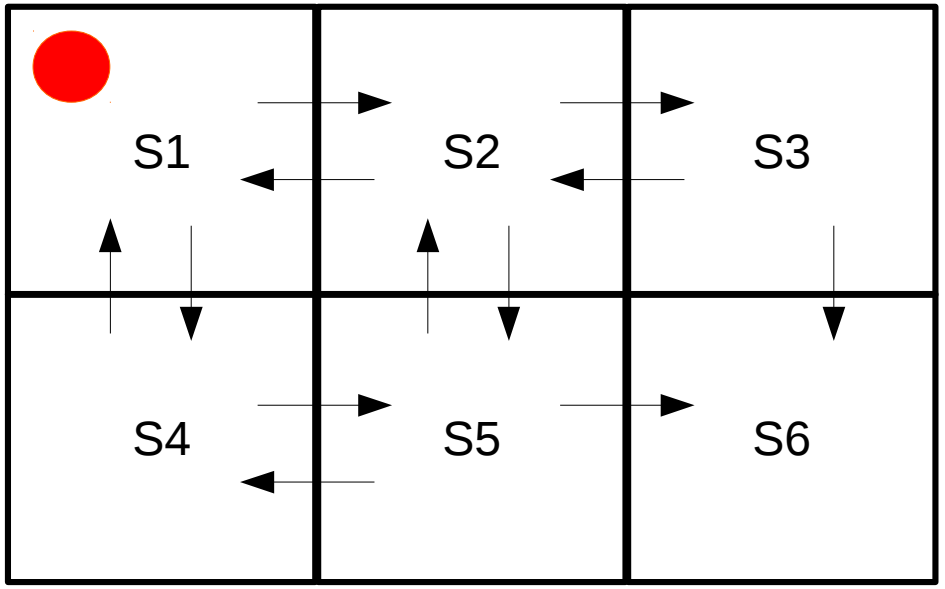


The Markov Assumption



The award and state-transition observed at time *t* after picking action *a* in state *s* is independent of anything that happened before time *t*

Q-Learning



+ 100 reward for getting to S6
0 for all other transitions

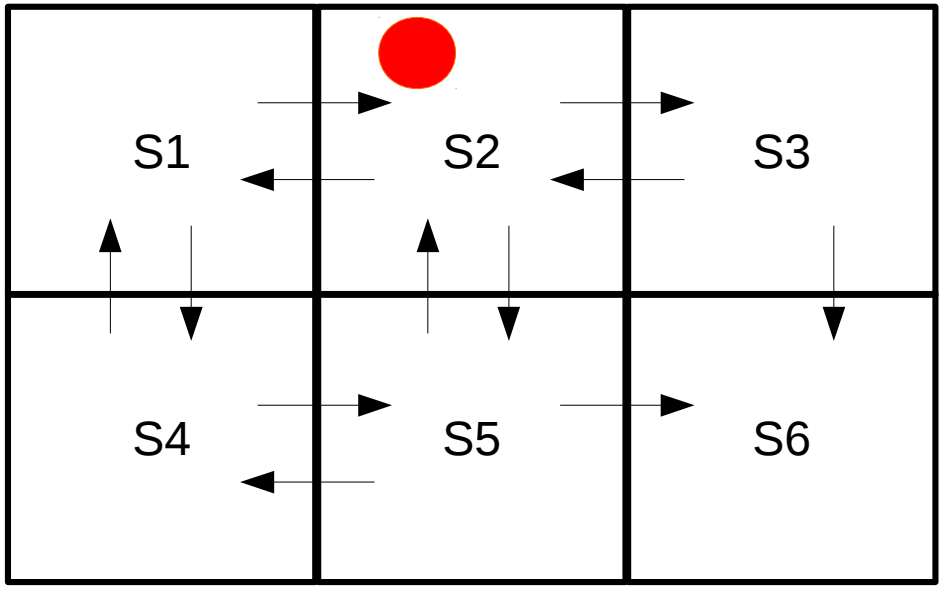
Update rule upon executing action a in state s, ending up in state s' and observing reward r :

$$Q(s , a)=r +\gamma \max a' Q(s' ,a ')$$

γ = 0.5 (discount factor)

Q-Table

S1	right	25
S1	down	0
S2	right	50
S2	left	0
S2	down	0
S3	left	0
S3	down	100
S4	up	0
S4	right	0
S5	left	0
S5	up	0
S5	right	0



+ 100 reward for getting to S6
0 for all other transitions

Update rule upon executing action a, ending up in state s' and observing reward r :

$$Q(s , a)=r +\gamma \max a' Q(s' ,a ')$$

γ = 0.5 (discount factor)

Q-Table

S1	right	25
S1	down	25
S2	right	50
S2	left	12.5
S2	down	25
S3	left	25
S3	down	100
S4	up	12.5
S4	right	50
S5	left	25
S5	up	25
S5	right	100

Q-Learning Properties

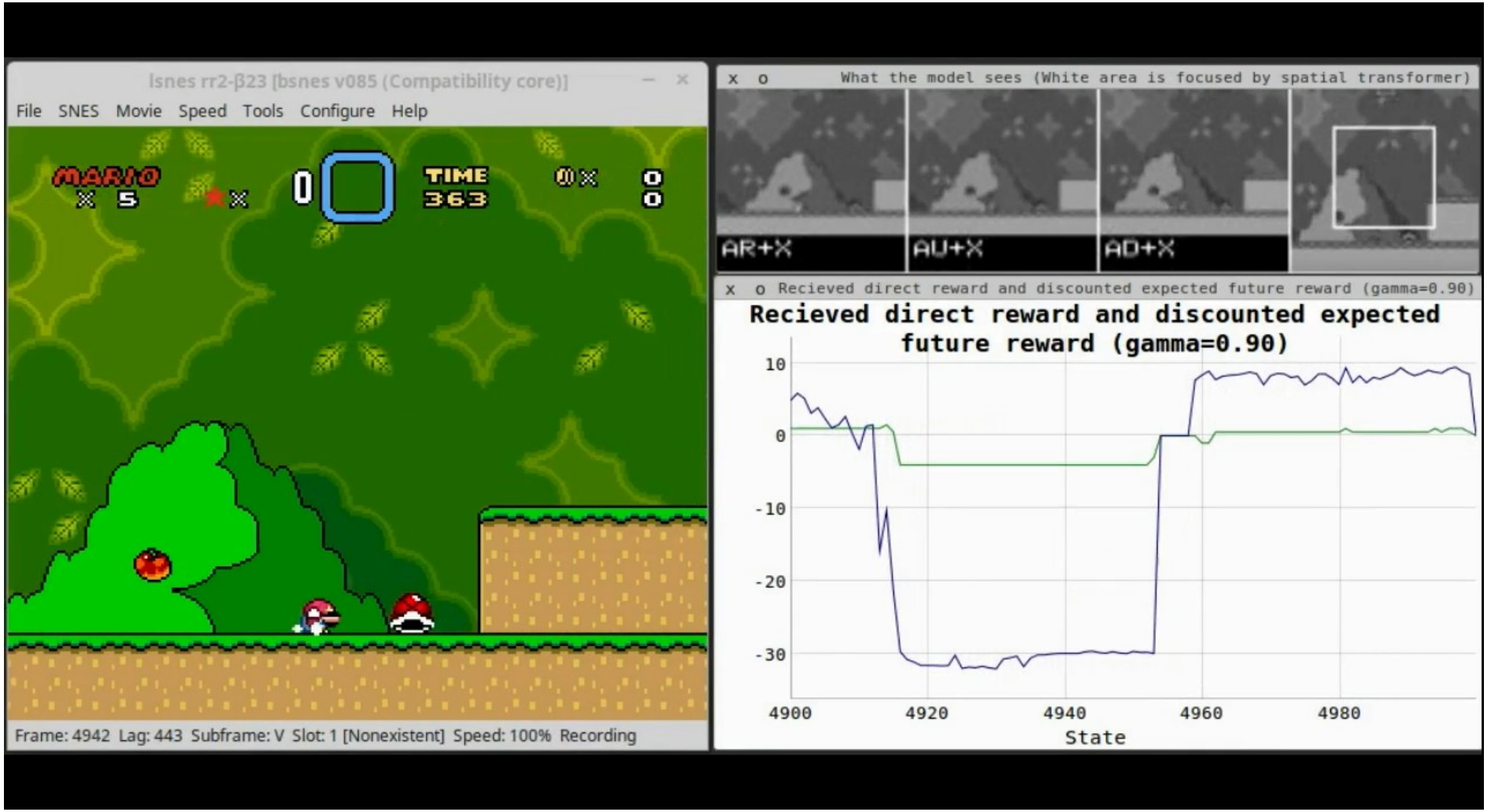
- Convergence to the true Q-function is guaranteed...as long as we visit every state-action pair infinitely many times!
- Table size can be very large for complex problems
- We cannot estimate unseen values
- How do we fix these problems?

What are some situations where a robot may need to use RL?

(discussion)

Flipping Pancakes using RL

Playing Mario using Deep RL



Learning to Shoot Penalty Kicks

https://www.youtube.com/watch?v=mRpX9DFCdwI

Resources

- BURLAP: Java RL Library: http://burlap.cs.brown.edu/
- Reinforcement Learning: An Introduction

  http://people.inf.elte.hu/lorincz/Files/RL_2006/SuttonBook.pdf