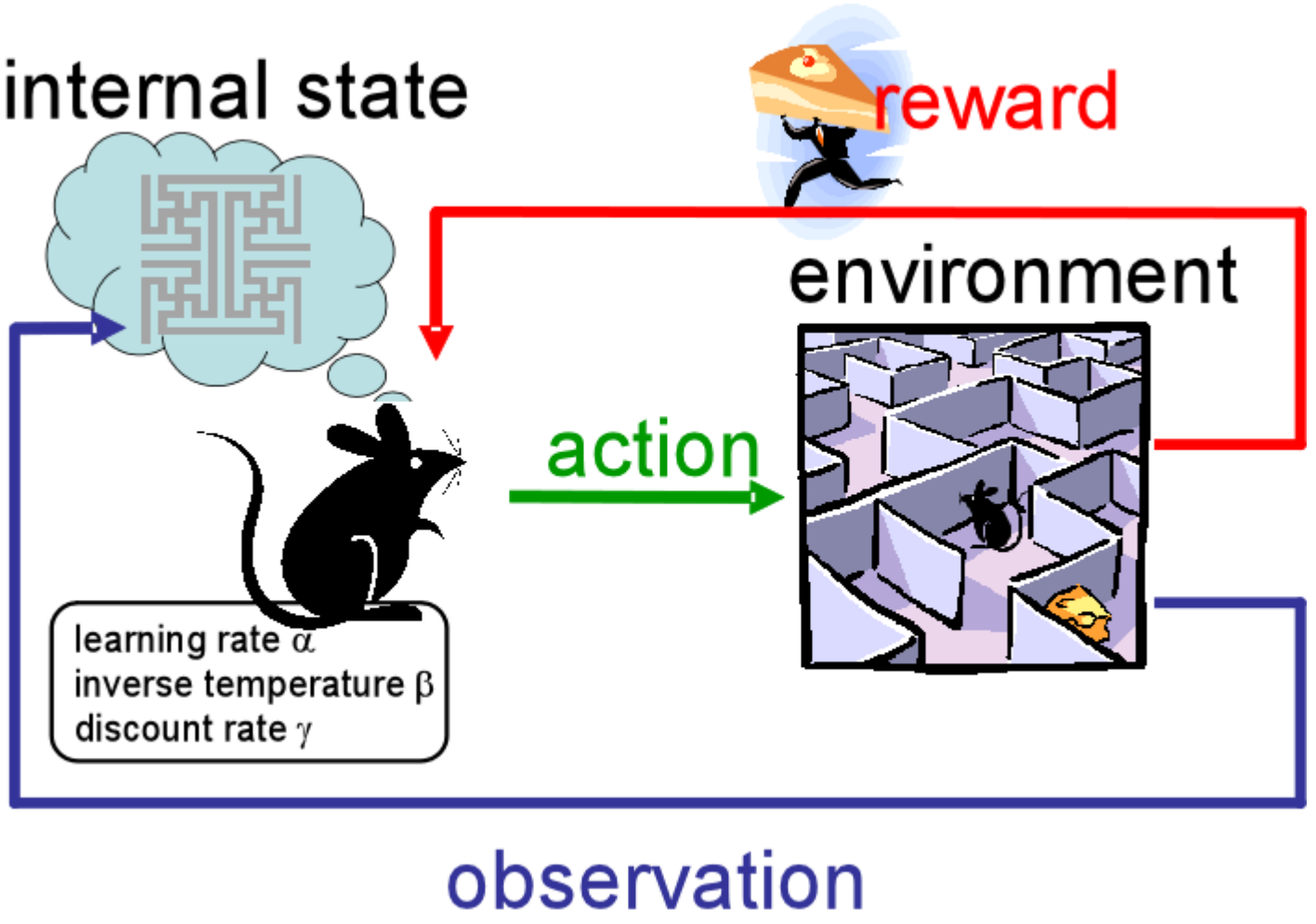




COMP 141: Probabilistic Robotics for Human-Robot Interaction

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

Today: Reinforcement Learning



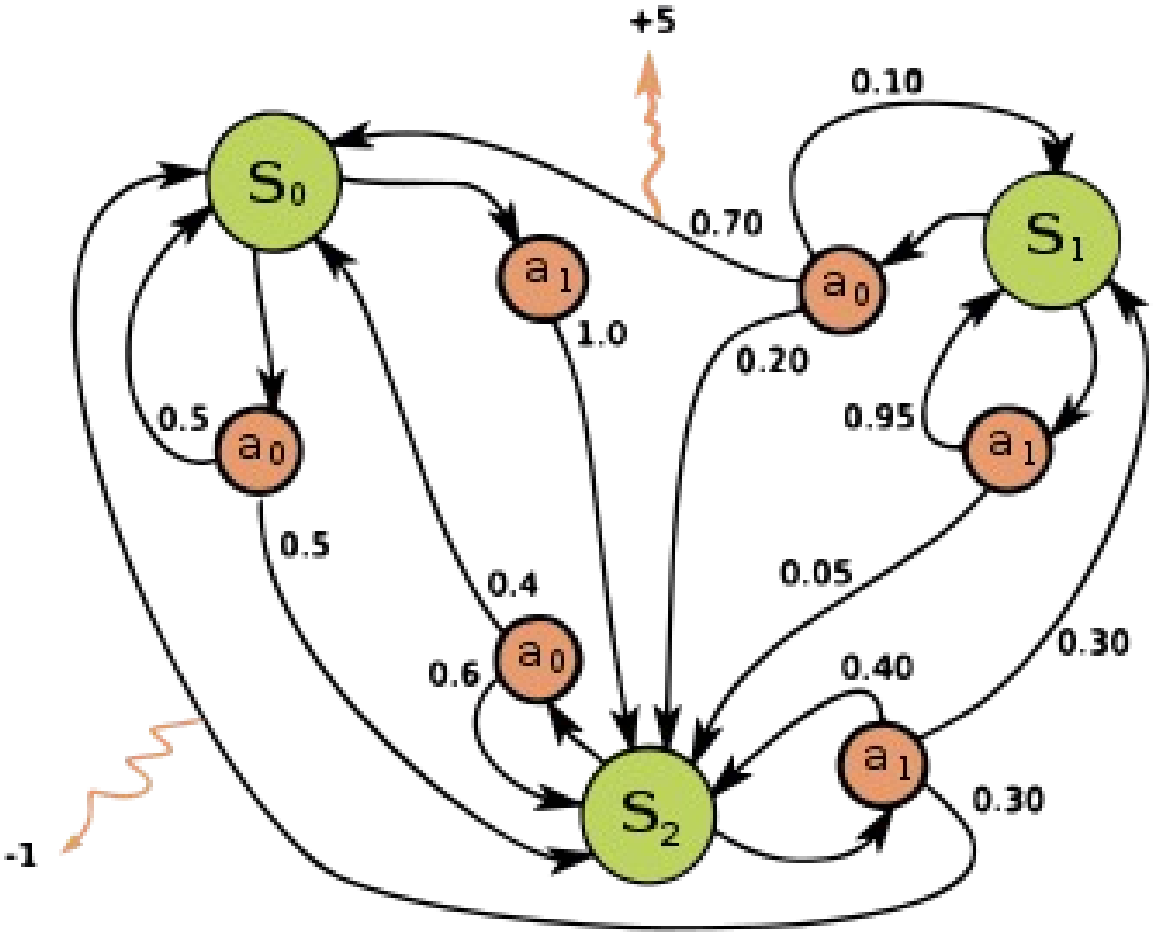
Announcements

Reading Assignments

Today

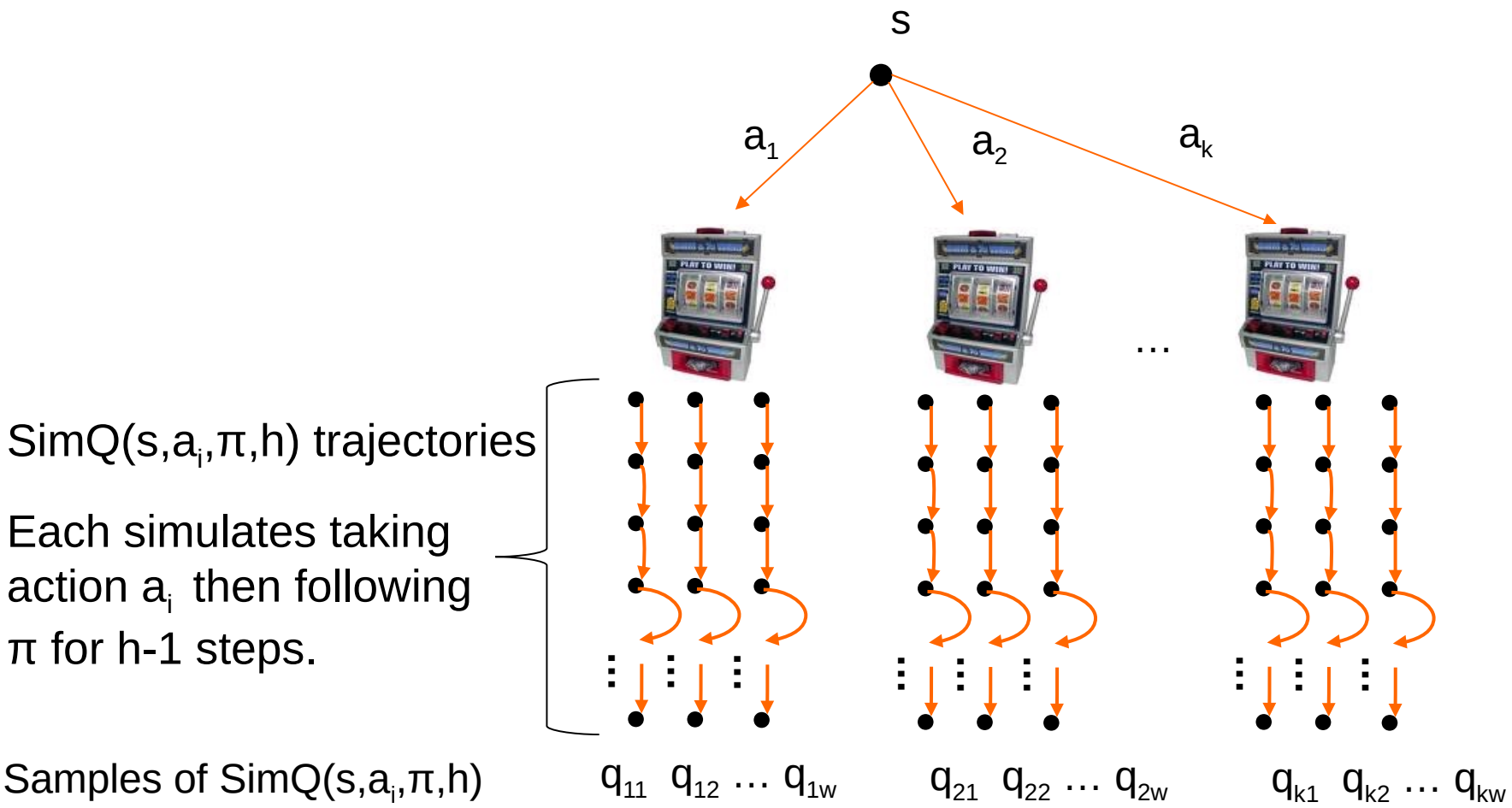
- Reinforcement Learning Applications
- Student-led Paper Presentation

Monte Carlo Planning Demo



Policy Rollout Algorithm

1. For each a_i run SimQ(s,a_i,π,h) **w** times
2. Return action with best average of SimQ results

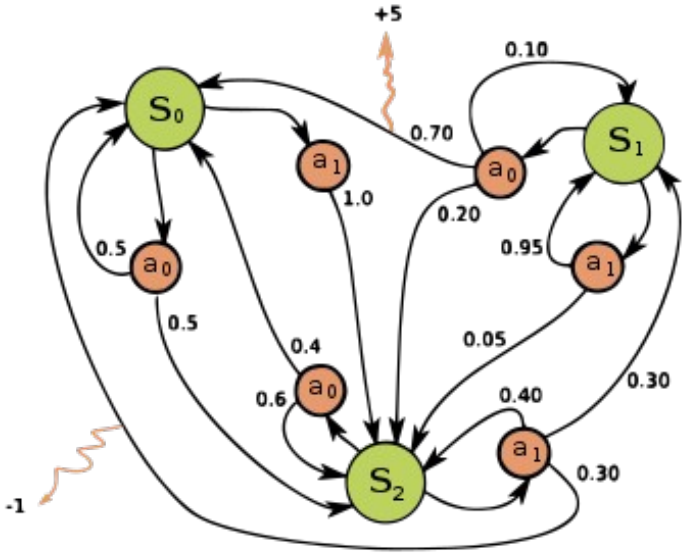


Policy Rollout Algorithm

```
SimQ(s,a,π,h)
    r = R(s,a)                          } simulate a in s
    s = T(s,a)
    for i = 1 to h-1                    } simulate h-1 steps
        r = r + β^i R(s, π(s))          } of policy
        s = T(s, π(s))
    Return r
```

- Simply simulate taking **a** in **s** and following policy for h-1 steps, returning discounted sum of rewards
- Expected value of SimQ(s,a,π,h) is $Q_{\pi}(s,a,h)$

Python Demo of Policy Improvement



S0	a0	27.16
S0	a1	27.81
S1	a0	29.83
S1	a1	27.66
S2	a0	25.77
S2	a1	25.94

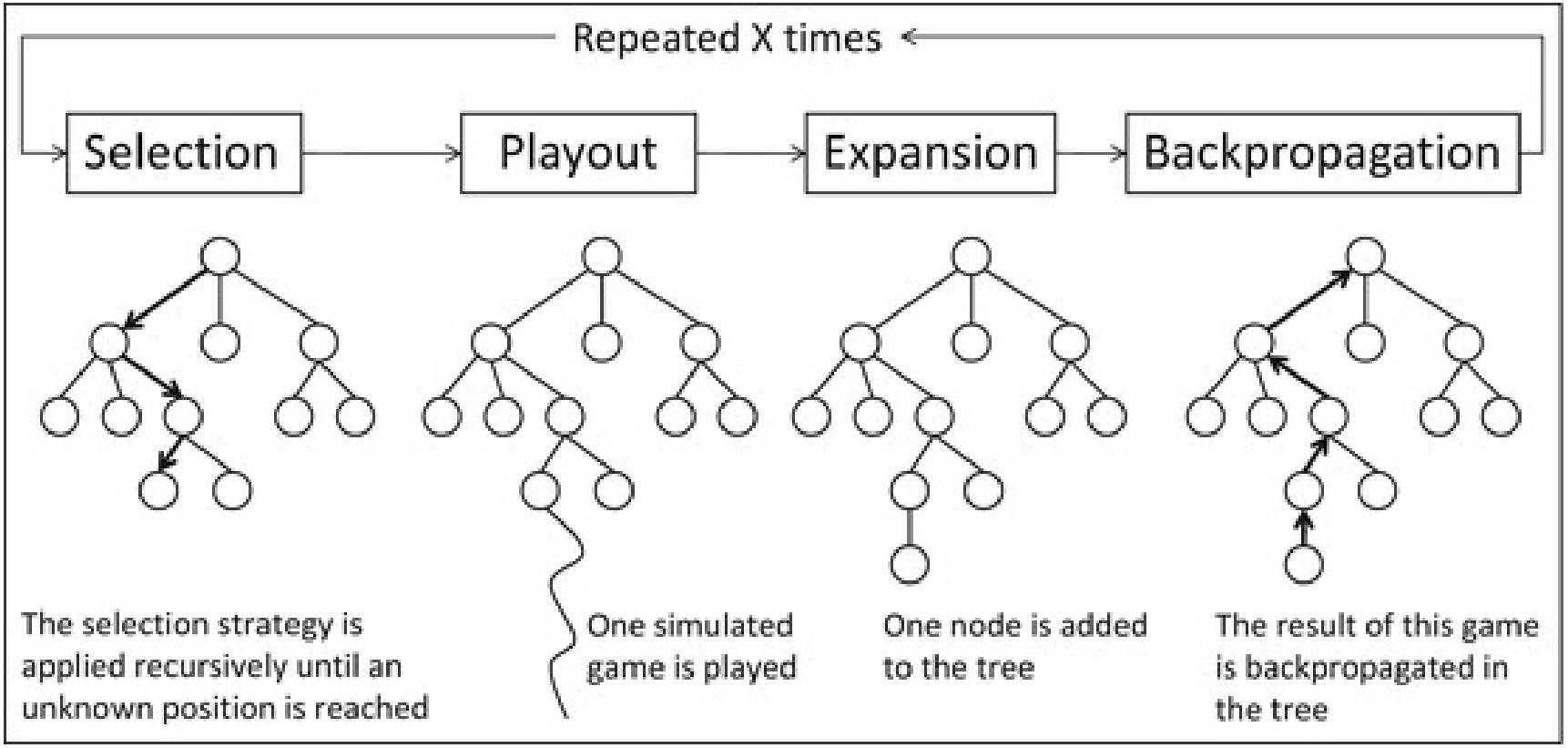
S0	a0	46.67
S0	a1	47.37
S1	a0	48.86
S2	a1	47.14
S3	a0	44.29
S3	a1	44.95

Horizon = 30 time steps
Discount factor = 1.0

State-action value table with random policy

State-action value table with policy derived from table 1

Monte Carlo Tree Search

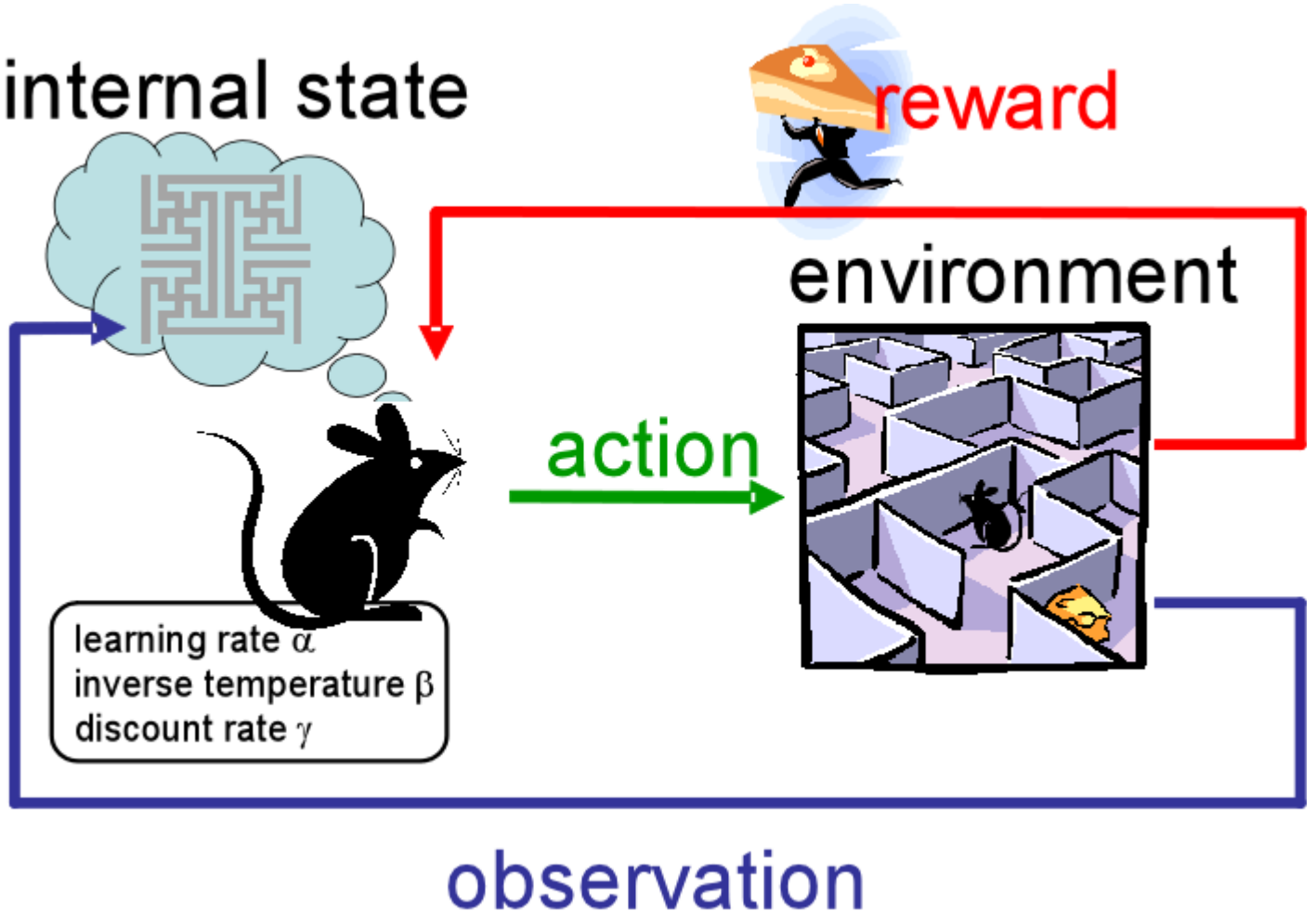


https://miro.medium.com/max/2982/1*Ntm0xHhJ5jOgsL9AdB2kNw.jpeg

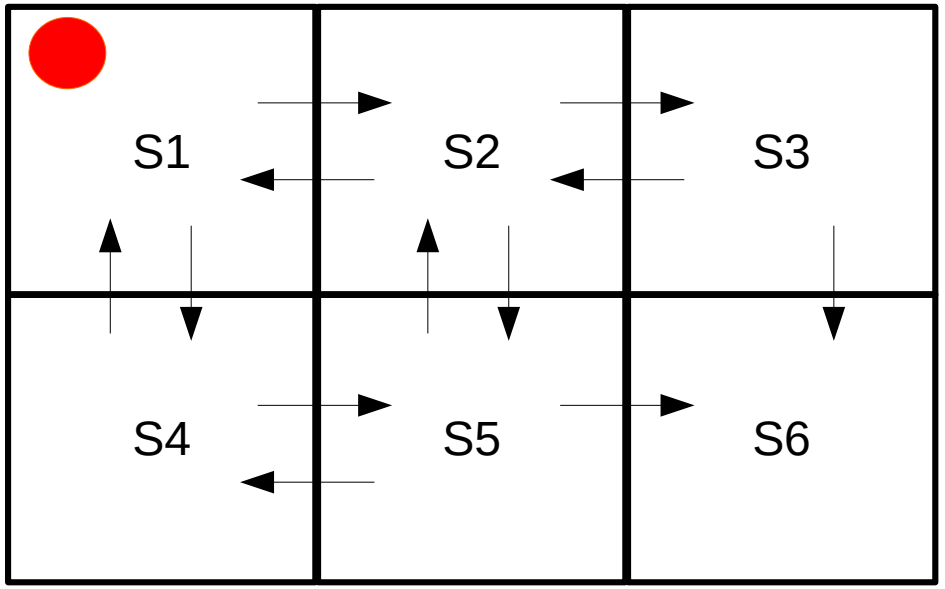
Applications

- https://www.youtube.com/watch?v=NscV-RpRSRw
- https://www.youtube.com/watch?v=gDG1WgZqGKU
- https://www.youtube.com/watch?v=fS5tTa_Tl1Y

Reinforcement Learning



A Simple RL Algorithm: Q-learning



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

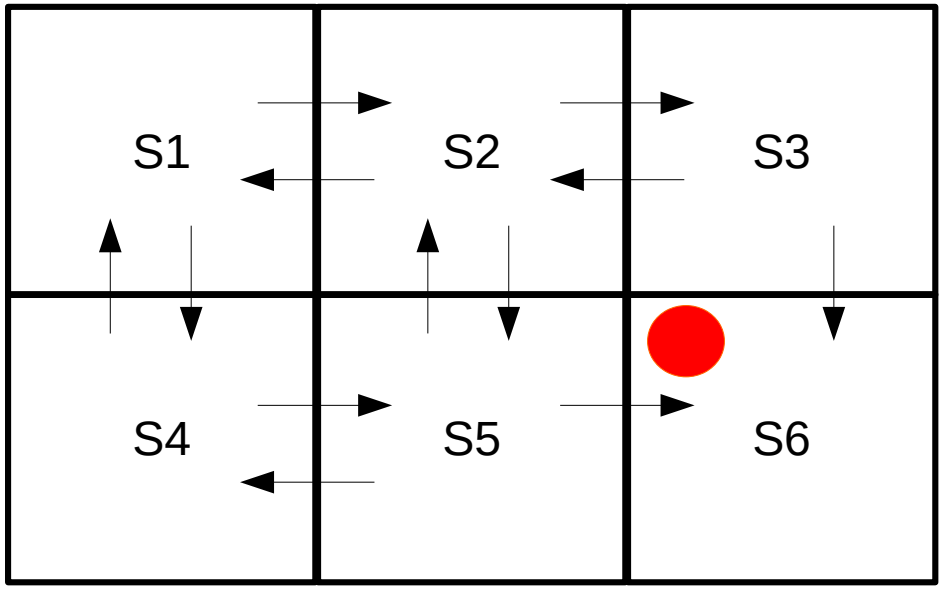
Update rule upon executing action a, ending up in state s' and observing reward r :

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

γ = 0.5 (discount factor)

Q-Table

S1	right	
S1	down	
S2	right	
S2	left	
S2	down	
S3	left	
S3	down	
S4	up	
S4	right	
S5	left	
S5	up	
S5	right	



+ 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 :

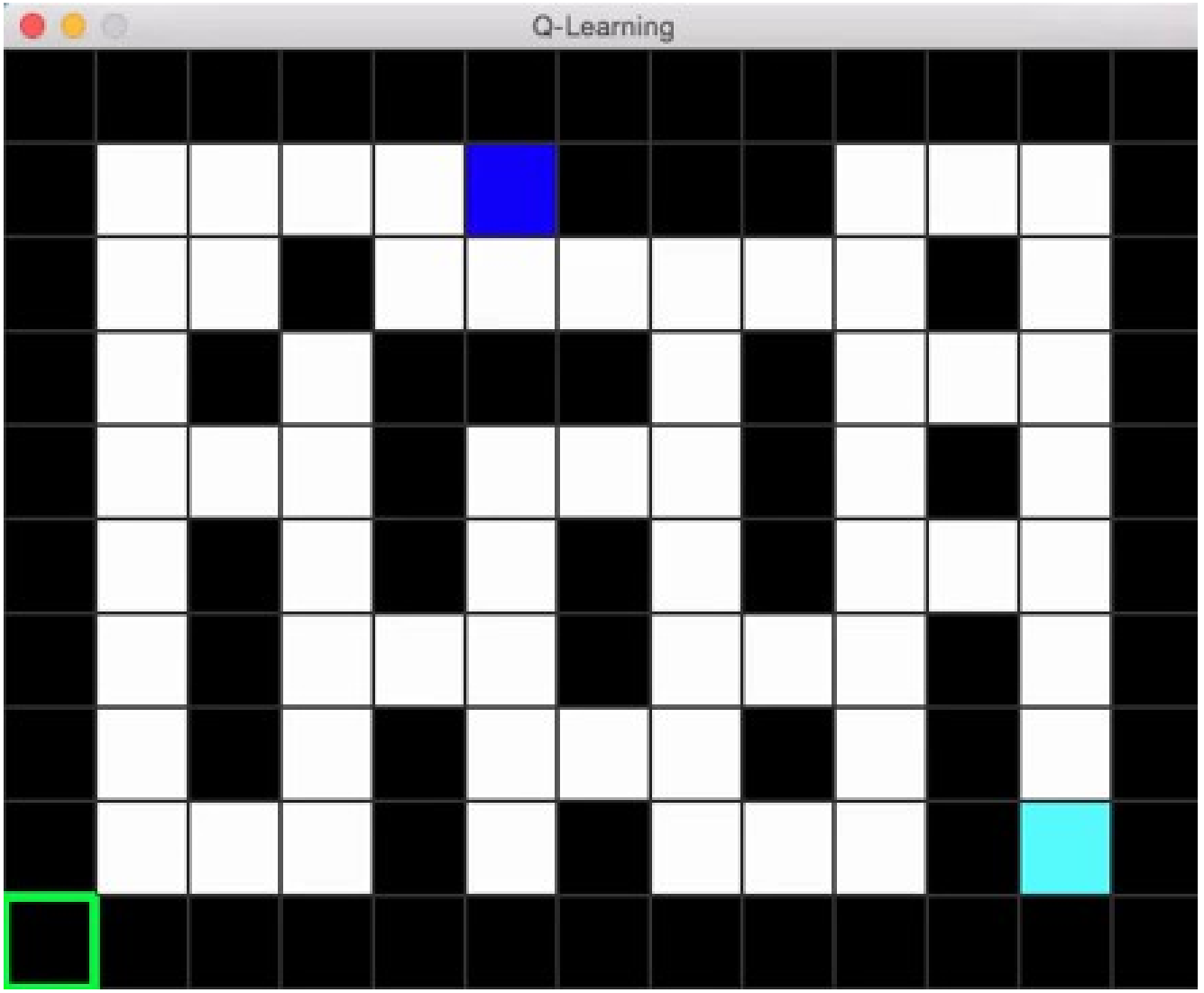
$$Q(s , a)=r +\gamma \max a' Q(s' ,a ')$$

γ = 0.5 (discount factor)

Q-Table

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

Example with larger board



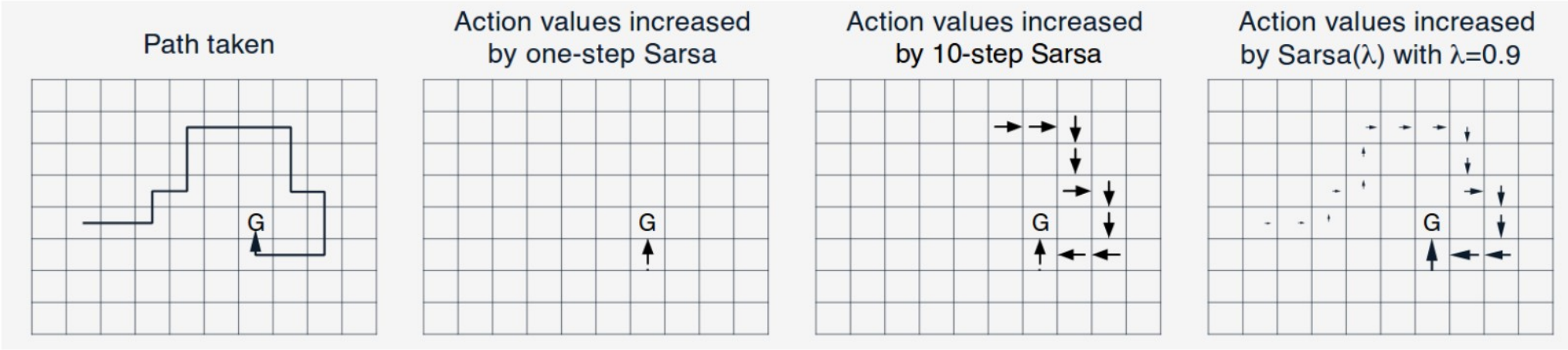
Q-Learning Algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

[Sutton and Barto]

Other Approaches



Some Applications of RL in Robotics

Robot Navigation

Robot Navigation

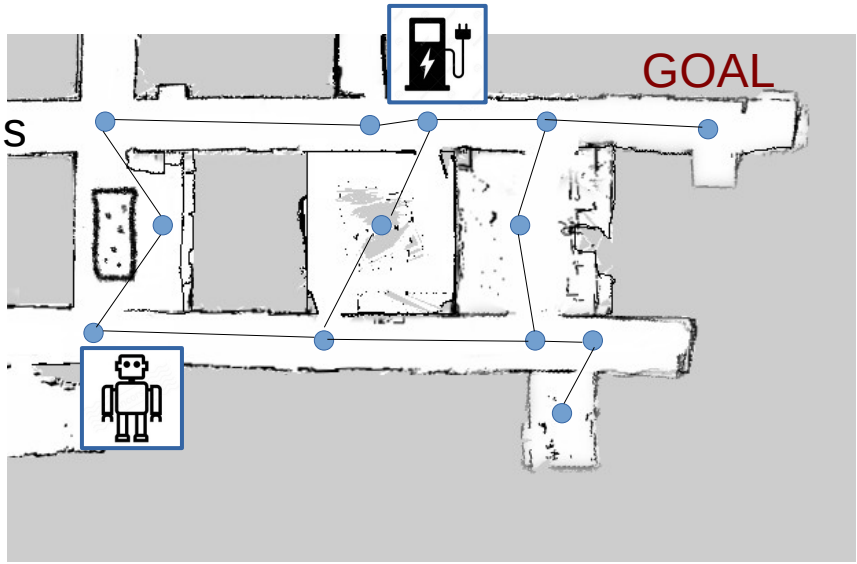
Reward Function:
+100 if it got to the goal
-1 otherwise
X where X is proportional to how much closer we got to the goal

State-space definition:
K nodes of topological graph
X, Y, Theta

Actions:
Move to a neighbor node

Robot Navigation

Reward Function:
+100 if we transition to the goal state
-100 if the robot transitions to a state where its battery is dead
+ 10 reward if the robot charges when its battery is low
- 10 reward if the robot charges when batter is high
- 1 per for all other action
-10 reward for failed action



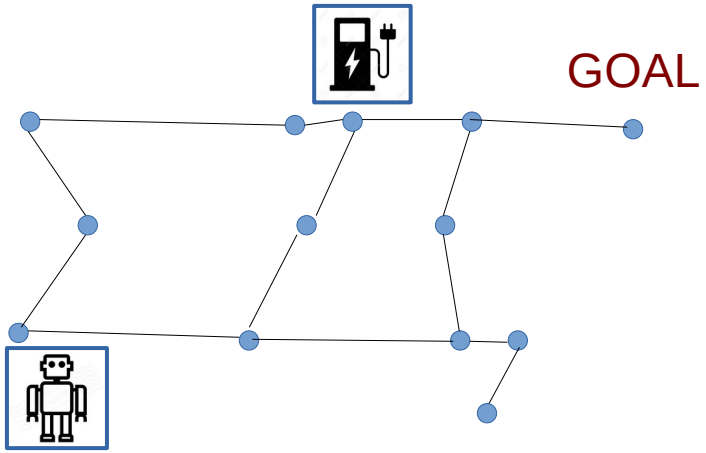
State-space definition:

Battery level: {low,medium,high}
Position in topological graph (i.e., which node are we currently at)
For each edge e_i, have value f_i \in [0,1] indicating the presence of obstacles/people
Time of day {morning,noon,evening}

Actions:
- for each adjacent node, an action to go there
- at one particular node, there is also a charge action

Robot Navigation

Reward Function:
+100 if reached goal
-100 if battery dies
-x for navigation action where x is the # of seconds
+20 if batter level is low
+10 if level is medium



State-space definition:

- Location: 1 of K nodes in topological graph
- Battery level: {low, medium, high}
- Workday = {yes, no}

Actions:

- Move to adjacent nodes
- Charge action (only available at one particular node)

Continuous RL

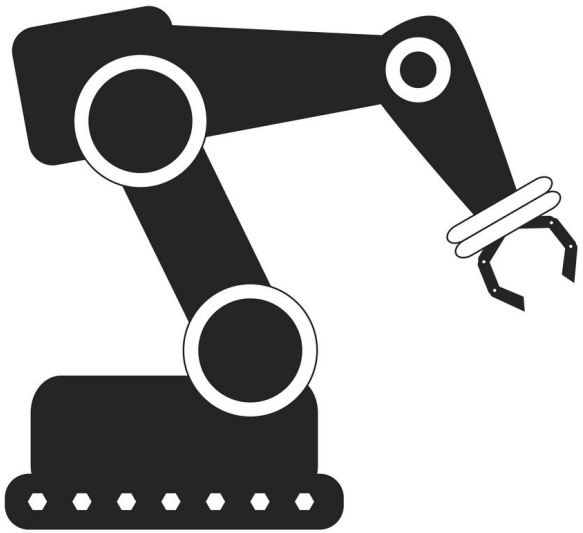
- States may be encoded as real-valued feature vectors

- Actions may also consist of real-valued signals (e.g., joint velocities/torques for each joint)

Example Application: Reaching

Reward Function:

TARGET



State-space definition:

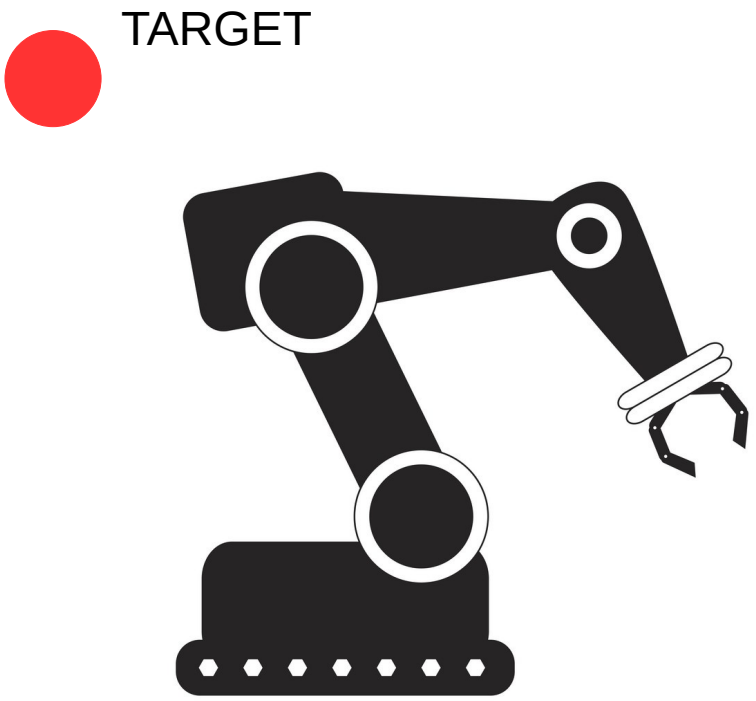
Actions:

https://cdn1.vectorstock.com/i/1000x1000/11/40/isolated-industrial-robot-arm-icon-vector-21491140.jpg

Example Application: Reaching

Reward Function:
+ 100 for getting to the goal

d_{t-1} – d_t
-1 for each time step



State-space definition:
- XY position of gripper
- Joint positions (angles)
- Joint velocities (angles/s)
- XY position of target
- dX and dY

Actions:
- Joint velocities x 3
- (Joint torques x 3)
- (Joint positions – at each time step, we can say joint x should go to angle y)

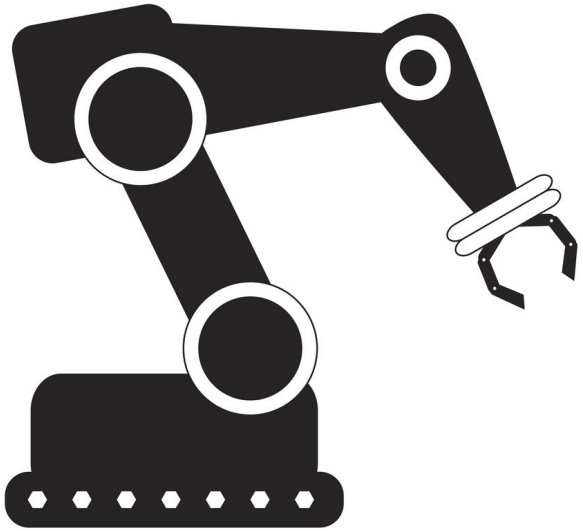
https://cdn1.vectorstock.com/i/1000x1000/11/40/isolated-industrial-robot-arm-icon-vector-21491140.jpg

Example Application: Reaching

Reward Function:
+ 100 when reaching reward

TARGET

State-space definition:
- Joint positions (3 numbers)
- Joint velocities (3 numbers)

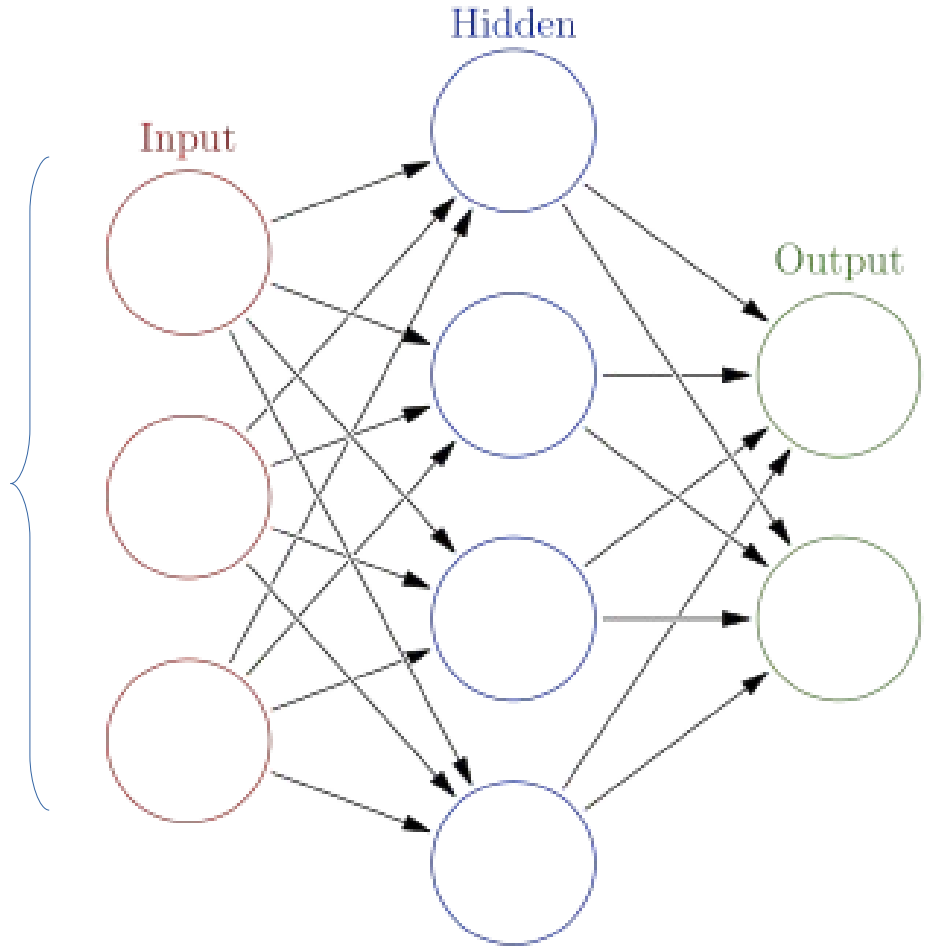


Actions:
- Apply torque or current on each joint (3 numbers)

https://cdn1.vectorstock.com/i/1000x1000/11/40/isolated-industrial-robot-arm-icon-vector-21491140.jpg

RL Networks

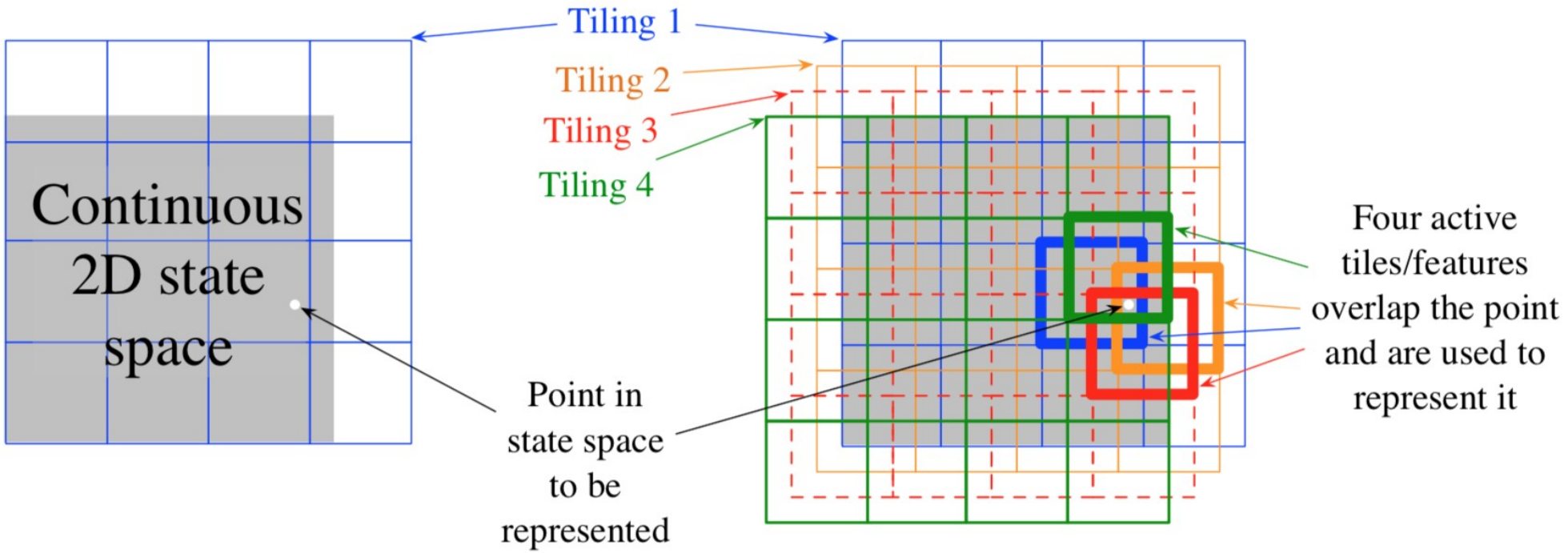
State feature vector



Pr(take action 1)

Pr(take action 2)

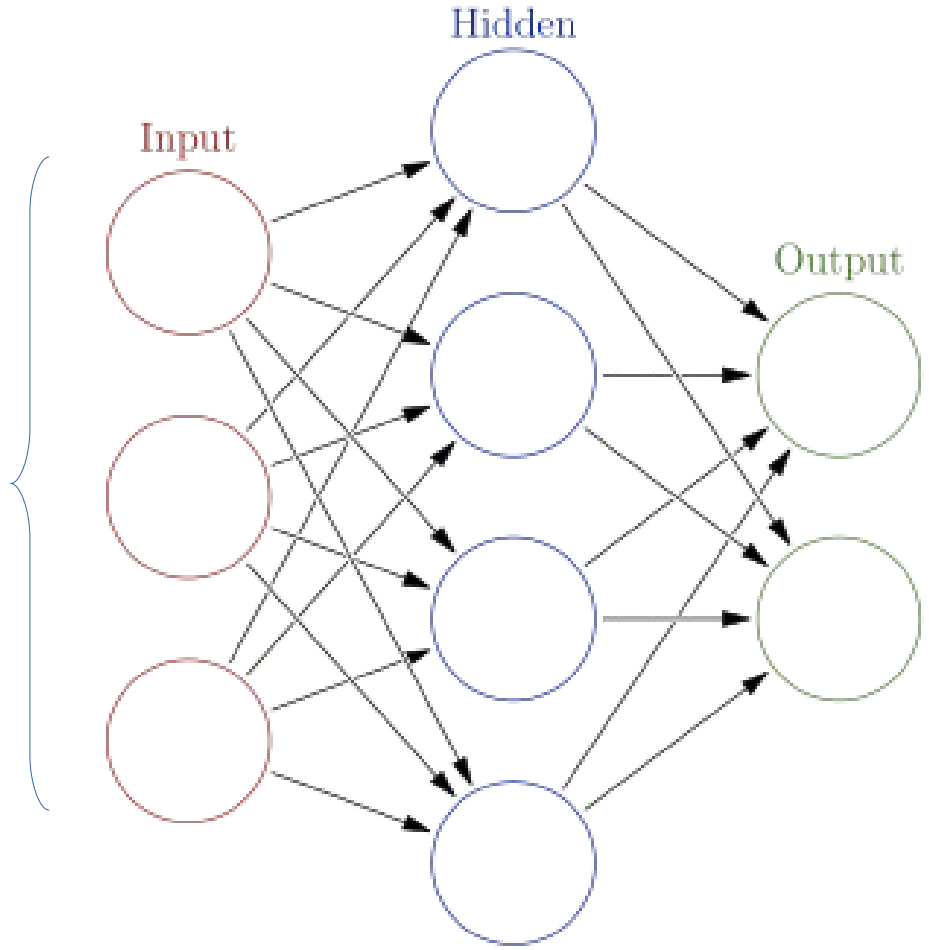
One approach: discretize continuous space



Sherstov, Alexander A., and Peter Stone. "Function approximation via tile coding: Automating parameter choice." International Symposium on Abstraction, Reformulation, and Approximation. Springer, Berlin, Heidelberg, 2005.

RL Networks

State feature vector



Joint velocity for J1

Joint velocity for J2

Applications: Reaching

- Reaching: https://www.youtube.com/watch?v=ZVIxt2rt1_4

- Paper:

  https://arxiv.org/abs/1803.07067

Applications: Flipping a Pancake

- Video: https://www.youtube.com/watch?v=W_gxLKSsSIE
- Paper:

  http://kormushev.com/papers/Kormushev-IROS2010.pdf

Applications: locomotion

- Video: https://www.youtube.com/watch?v=2iNrJx6IDEo

If you want to know more about RL...

- take COMP 138: Reinforcement Learning in the Fall