Reinforcement Learning

Machine Learning
CS5824/ECE5424
Bert Huang
Virginia Tech

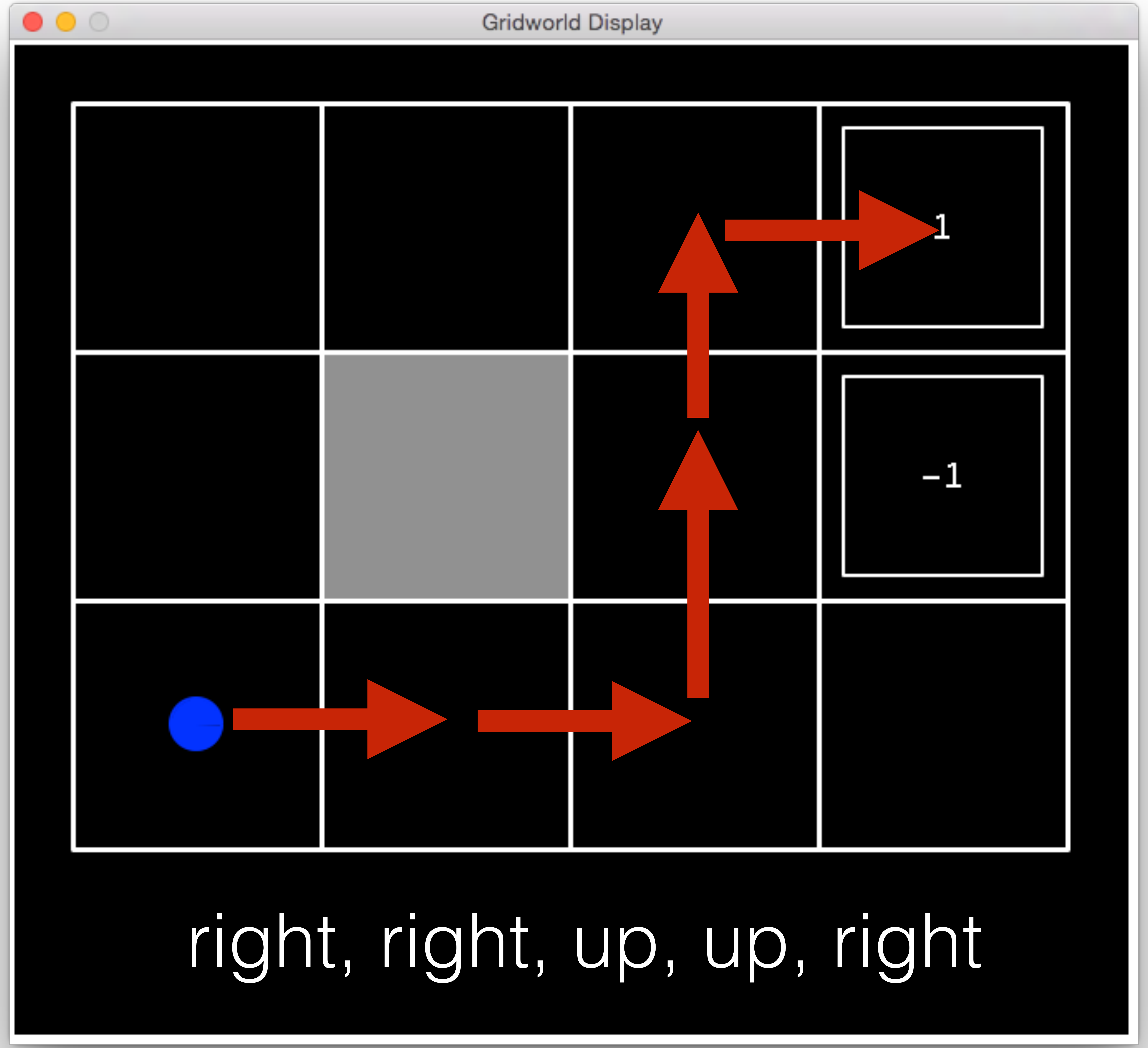
Outline

- Reinforcement learning definition
- Markov decision processes
- Q-Learning

Reinforcement Learning

- Train an AI by giving it rewards when it behaves well
- Learning algorithm gets **(1) state, (2) action, (3) result, (4) reward**
- Typical assumptions: need to learn online, randomness

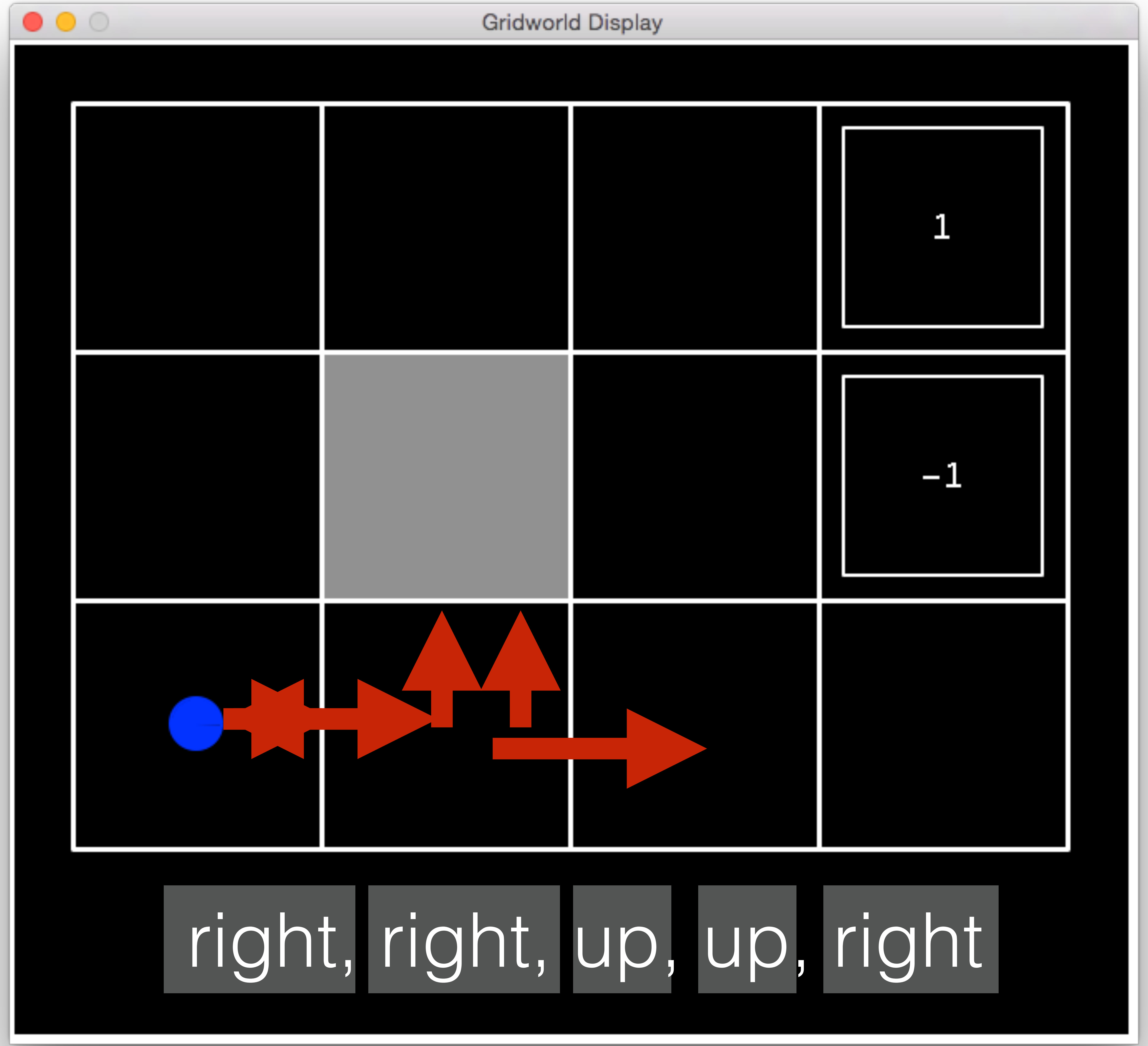
collect reward



right, right, up, up, right

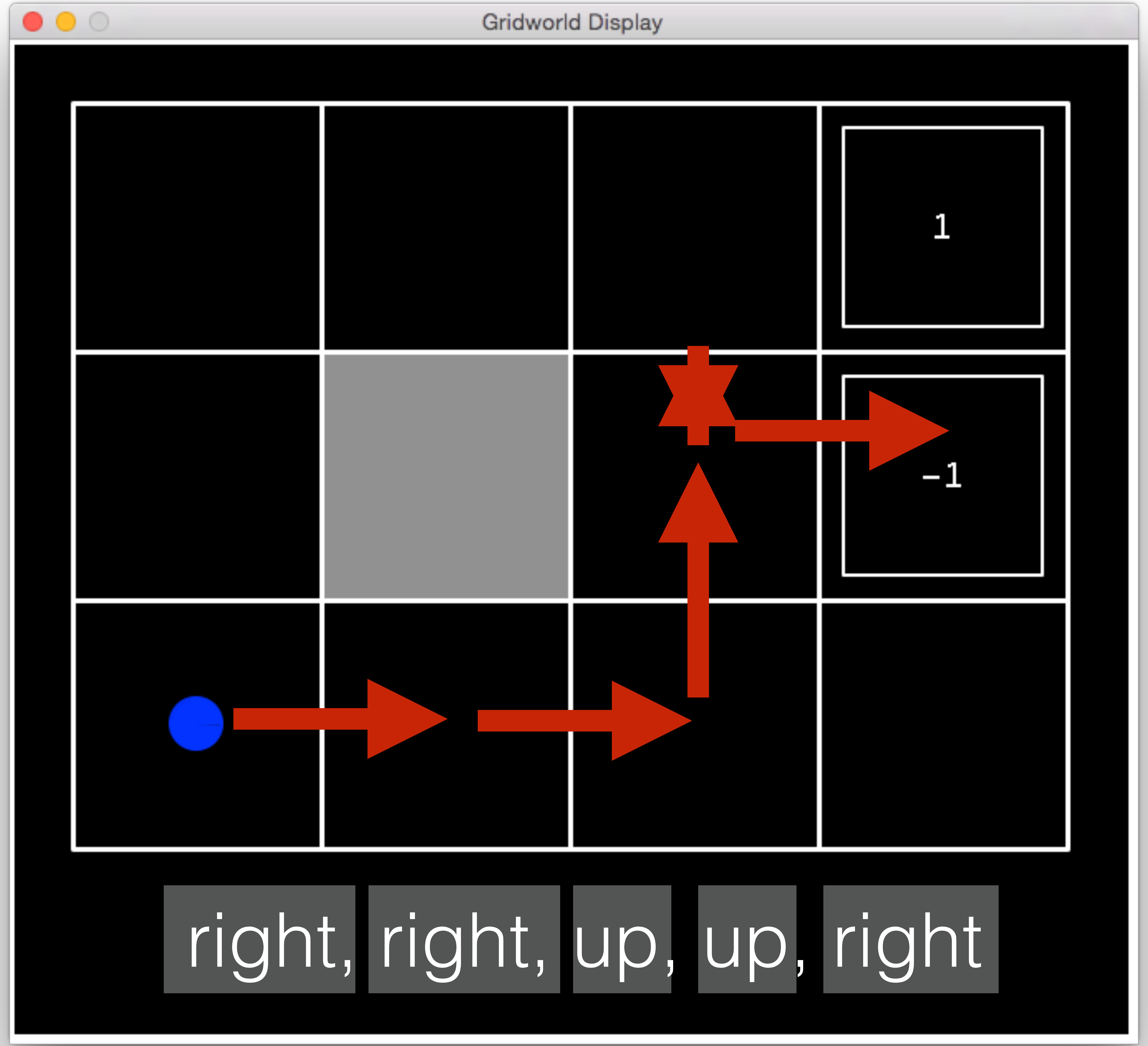
collect reward

stochastic transitions

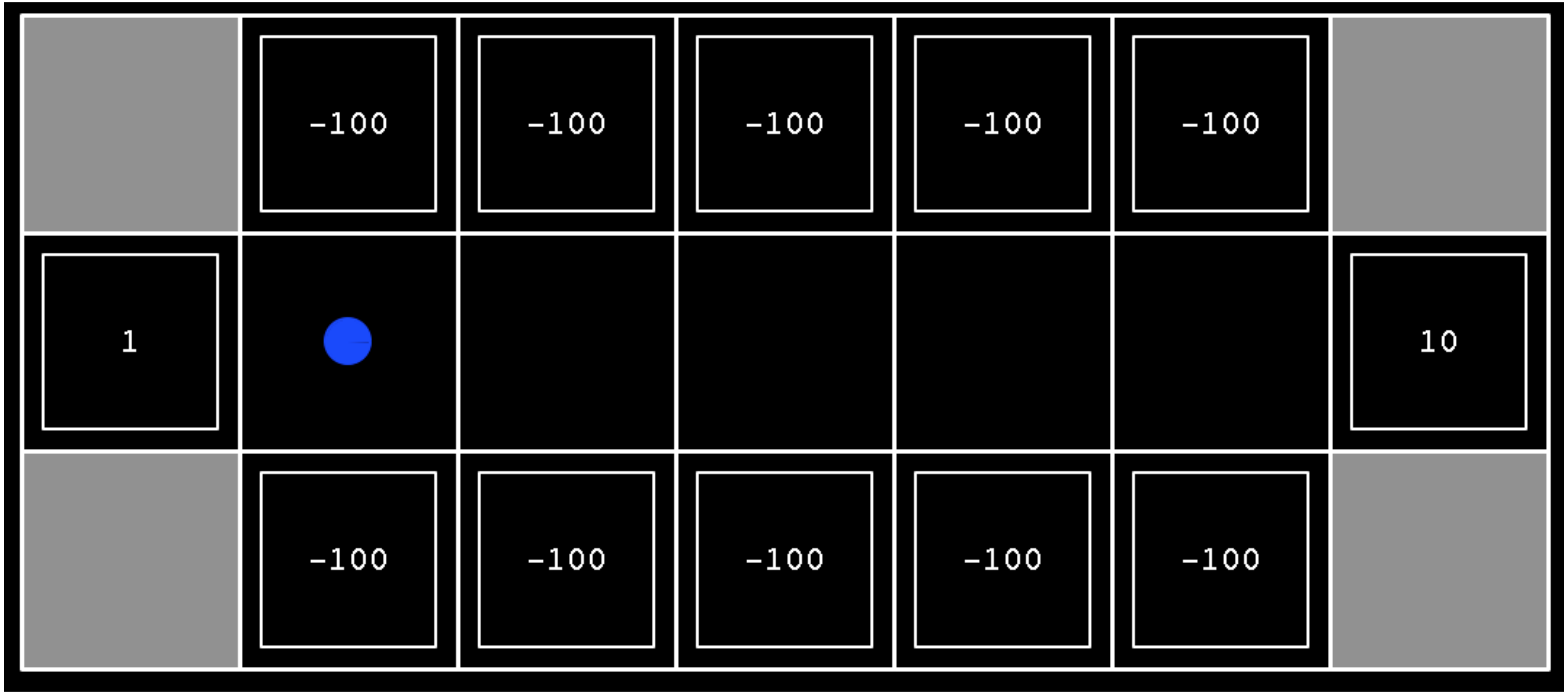


collect reward

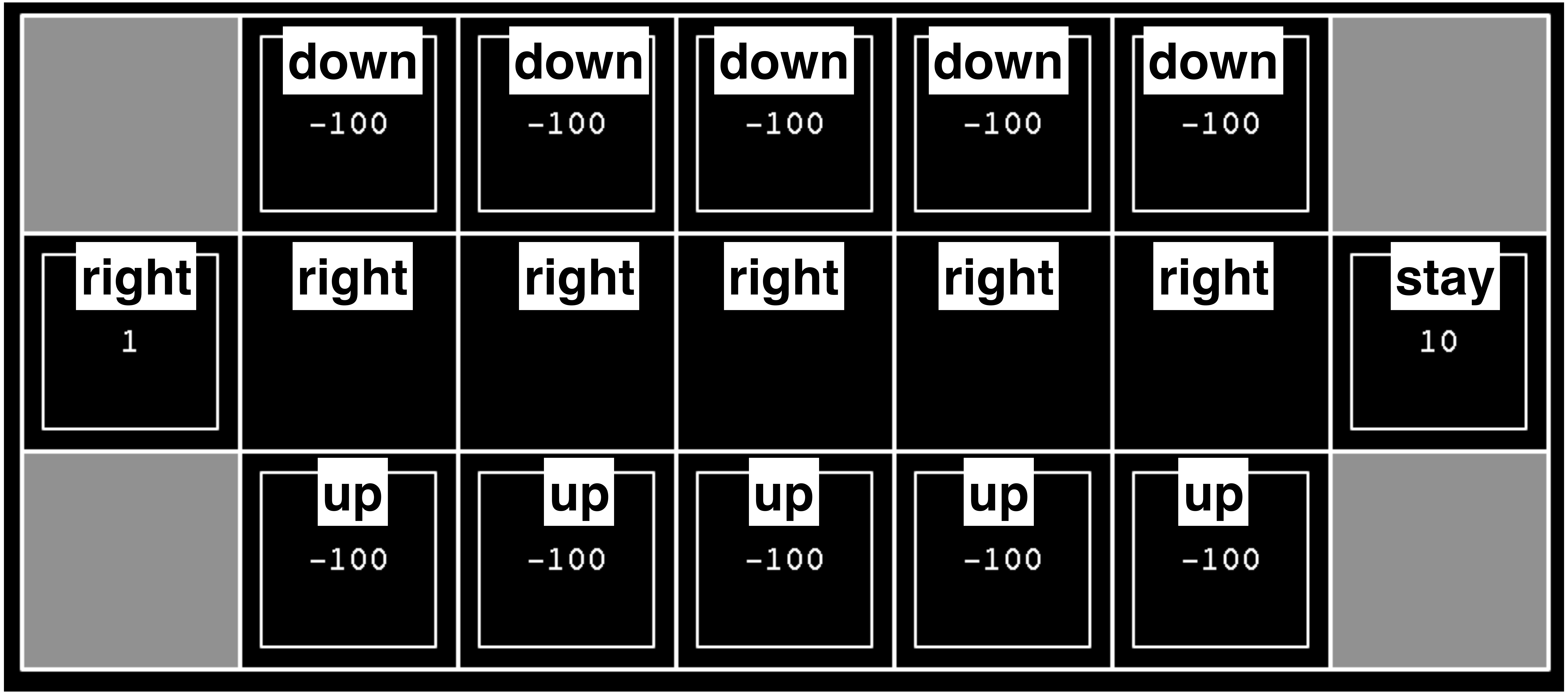
stochastic transitions



Reward function R(s)

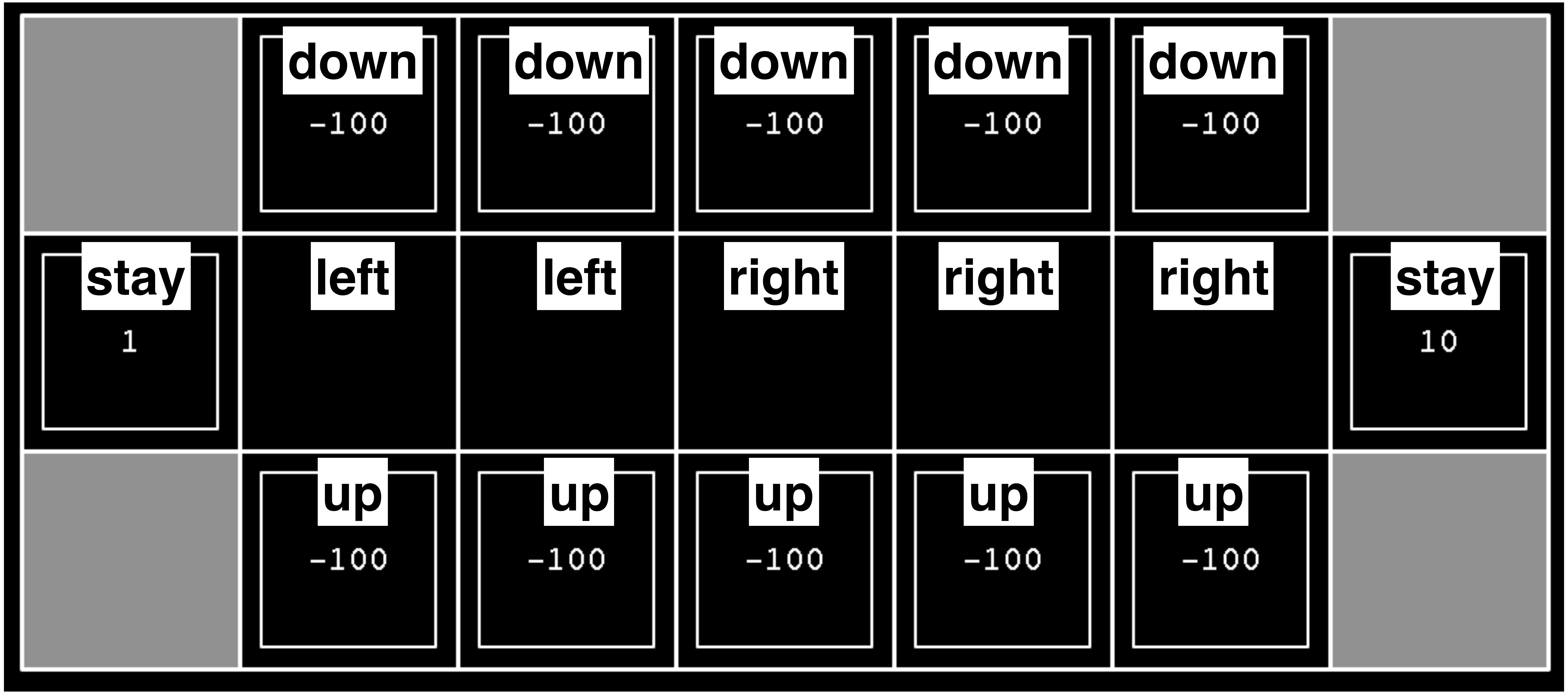


Policy π(s)



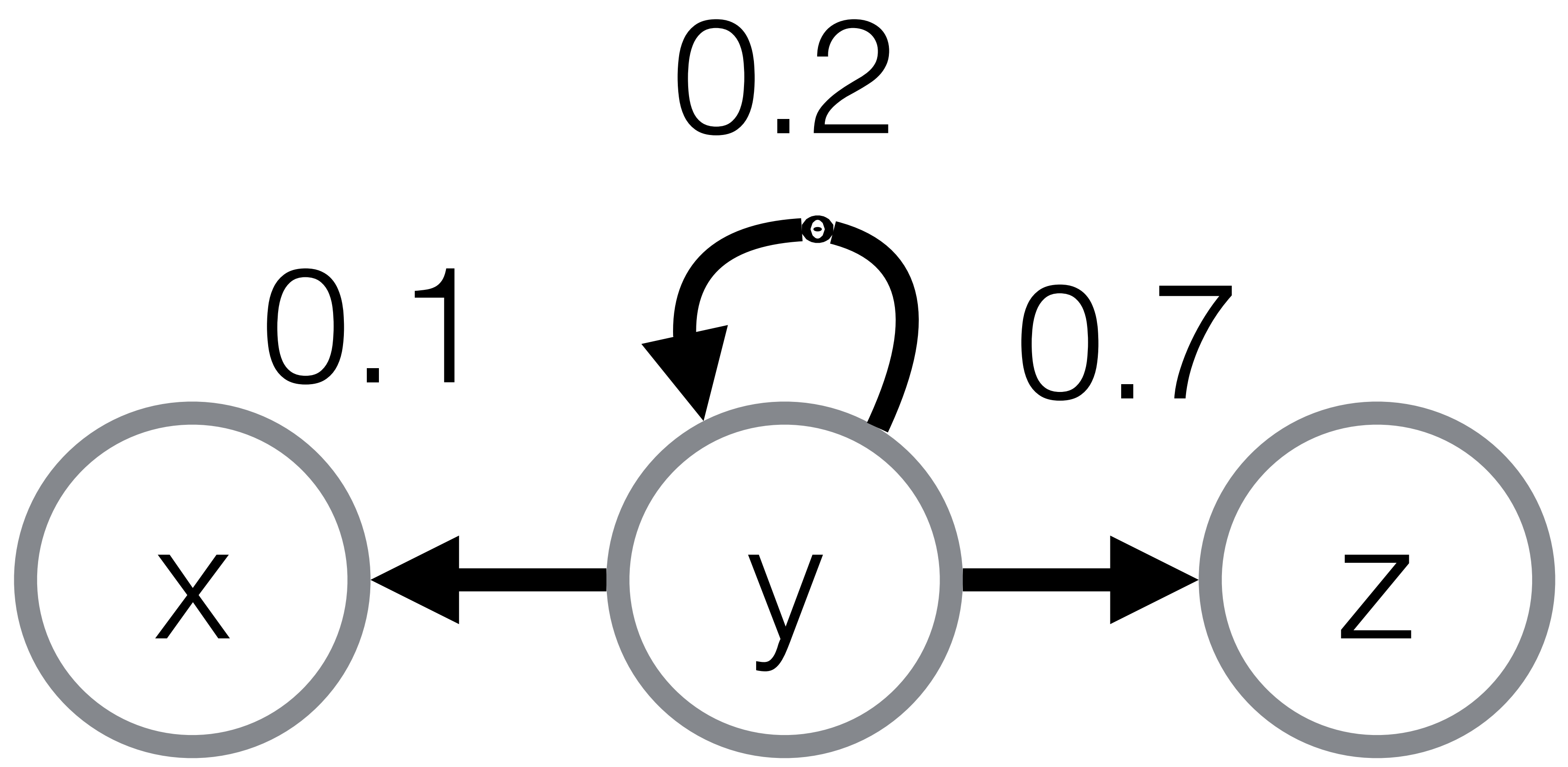
	down -100	down -100	down -100	down -100	down -100	
right 1	right	right	right	right	right	stay 10
	up -100	up -100	up -100	up -100	up -100	

Policy π(s)


down
-100
down
-100
down
-100
down
-100
down
-100
stay
1
left
left
right
right
right
stay
10
up
-100
up
-100
up
-100
up
-100
up
-100

Markov Decision Processes

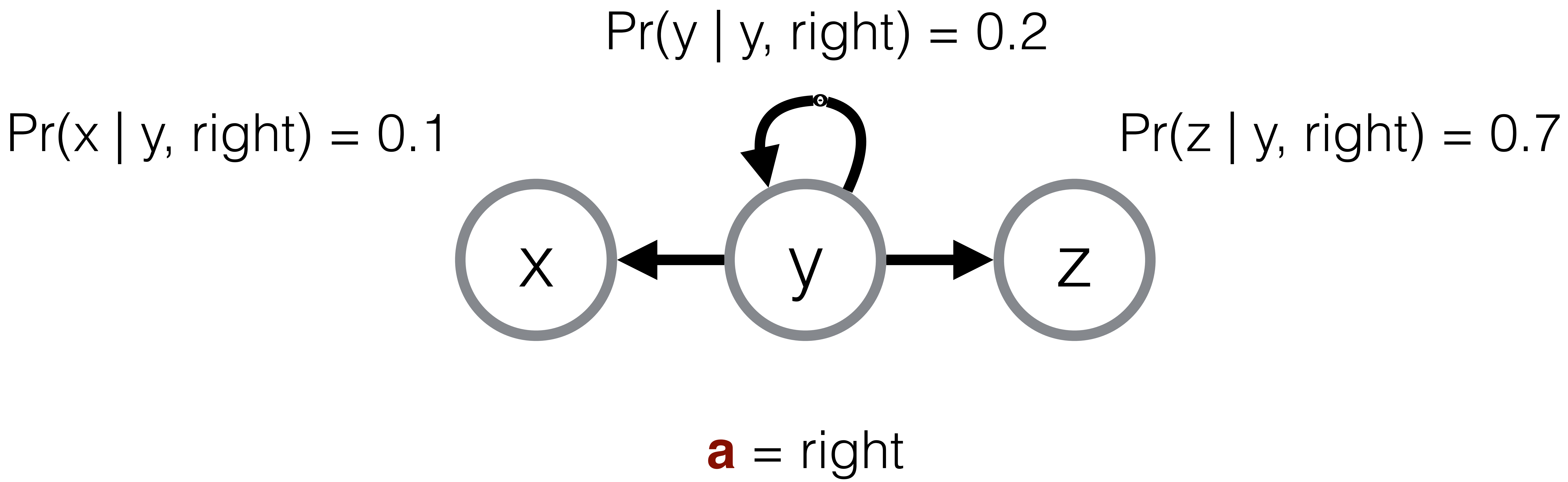
- Pr(**s'** | **s**, **a**)
    - Probability we **transition** to **s'** if we choose **action** **a** in state **s**



a = right

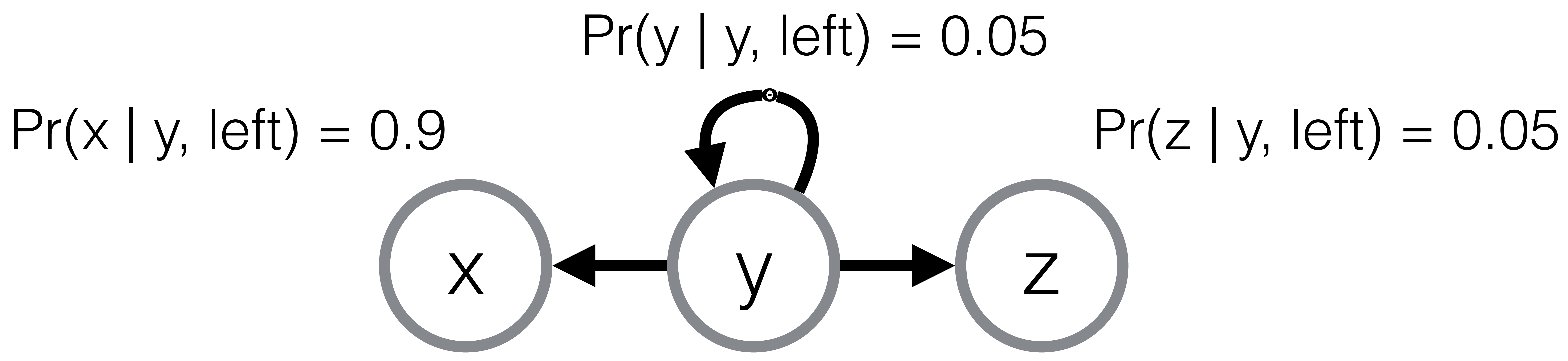
Markov Decision Processes

- Pr(**s'** | **s**, **a**)
  - Probability we **transition** to **s'** if we choose **action** **a** in state **s**



Markov Decision Processes

- Pr(**s'** | **s**, **a**)
    - Probability we **transition** to **s'** if we choose **action** **a** in state **s**



a = left

How Good is a Policy?

$$U([s_0, s_1, \ldots, s_T]) = \sum_{t=0}^{T} R(s_t)$$

$$U([s_0, s_1, \ldots, s_T]) = \sum_{t=0}^{T} \gamma^t R(s_t) \qquad \gamma \in (0, 1]$$

How Good is a Policy?

$$U([s_0, s_1, \ldots, s_T]) = \sum_{t=0}^{\cancel{T}\ \infty} \gamma^t R(s_t) \qquad \gamma \in (0, 1]$$

$$U^\pi(s) = \mathrm{E}_{\Pr([s_0, s_1, \ldots] \mid s_0 = s, \pi)} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \right]$$

$$\pi_s^* = \arg\max_\pi U^\pi(s)$$

$$U([s_0, s_1, \ldots, s_T]) = \sum_{t=0}^{\cancel{T}\ \infty} \gamma^t R(s_t) \qquad \gamma \in (0, 1]$$

$$U^\pi(s) = \mathrm{E}_{\Pr([s_0, s_1, \ldots] | s_0 = s, \pi)} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \right]$$

$$\pi^*_s = \arg\max_\pi U^\pi(s) = \pi^*_{s'}, \text{ for any } s'$$

$$U(s) = U^{\pi^*}(s)$$

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$$

U(*s*') = expected utility given optimal play from *s*'

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$$

Bellman equation

Learning a Policy

- Define an action-state utility:
  Q(s, a) = expected future reward if we choose action **a** from state **s**.

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

key challenge:
we don't know this transition probability

expected future reward
after transitioning to **s'**

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

Learning a Policy

- Define an action-state utility:
  Q(s, a) = expected future reward if we choose action **a** from state **s**.

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

Q-learning update:
Treat transition from **s to s'** as random sample from **P(s' I s, a)**

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(\underbrace{R(s) + \gamma \max_{a'} Q(s',a')}_{\text{Actual reward from next state } \mathbf{s'}} - \underbrace{Q(s,a)}_{\text{previous estimate of reward}}\Big)$$

Actual reward from next state **s'**

previous estimate of reward

Loss Minimization for Parameterized Q

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Estimation error: $$\ell(s,s') = \ell\left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

Compute $\nabla_Q \ell(s,s')$ with backpropagation, do stochastic gradient descent.

Approximate Q-Learning

$$\hat{Q}(s,a) := g(s,a,\boldsymbol{\theta}) := \theta_1 f_1(s,a) + \theta_2 f_2(s,a) + \ldots + \theta_d f_d(s,a)$$

$$\theta_i \leftarrow \theta_i + \alpha \left(R(s) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) \frac{\partial g}{\partial \theta_i}$$

$$\theta_i \leftarrow \theta_i + \alpha \left(R(s) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) f_i(s,a)$$

Policy Search

- Instead, parameterize policy as a condition probability distribution

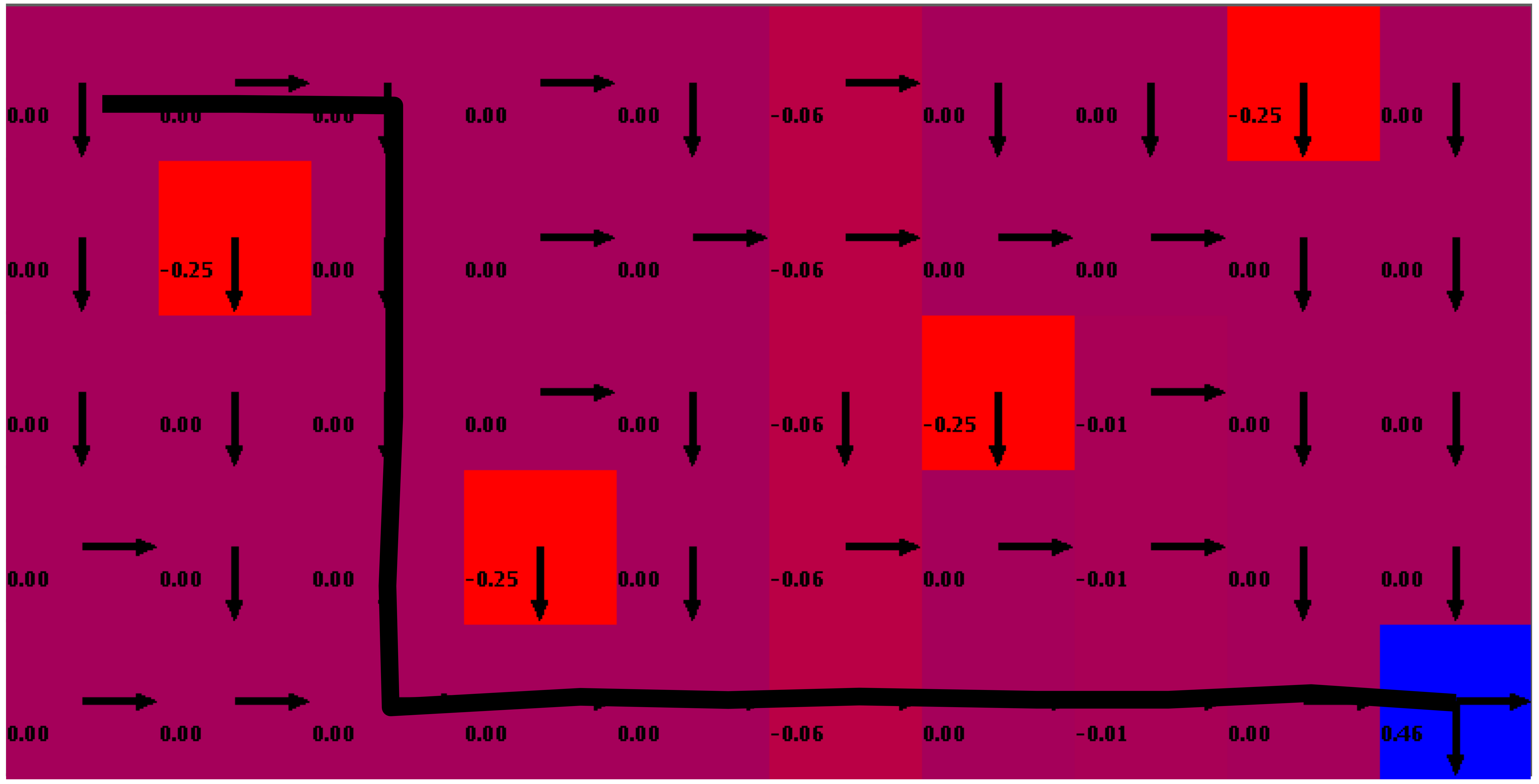
$$\pi_\theta(a \mid s)$$

- Tune parameters by approximating the **gradient** of policy
- Gradient descent
- If curious, read https://towardsdatascience.com/policy-gradients-in-a-nutshell-8b72f9743c5d

Detour Slide: Inverse Reinforcement Learning

Slide by Prof. Michael Littman

Maximum Likelihood IRL



Gradient ascent through reward parameters on likelihood function (Babes, Marivate, Littman & Subramanian 11).

Variations of Reinforcement Learning

- Passive/active reinforcement learning
- Imitation/apprenticeship learning
- Inverse reinforcement learning
- Multi-armed bandit learning

Summary

- Reinforcement learning: alternative to supervised or unsupervised
- Key challenge: delayed rewards
- Aim to get best discounted future rewards
- Q-learning: learn estimated future rewards given state and action
- Approximate Q function with your favorite function estimator