Lecture 15b: Reinforcement Learning

CSCI 360

Introduction to Artificial Intelligence

USC

Here is where we are…

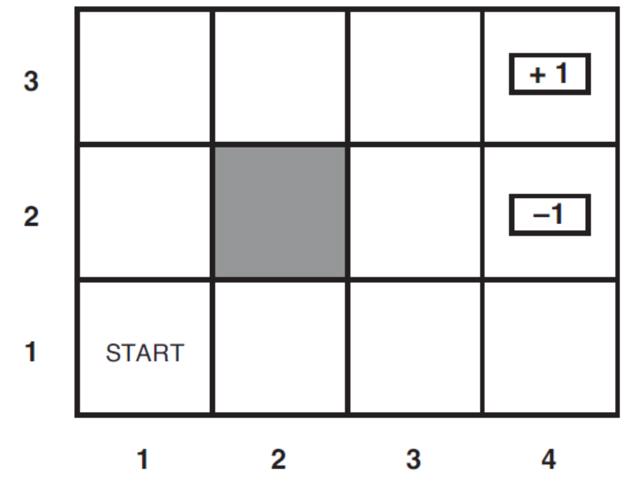
	3/1		Project 2 Out	
9	3/4 3/6	3/5 3/7	Quantifying Uncertainty Bayesian Networks	[Ch 13.1-13.6] [Ch 14.1-14.2]
10	3/11 3/13	3/12 3/14	*(spring break, no class)* *(spring break, no class)*	
11	3/18 3/20	3/19 3/21	Inference in Bayesian Networks Decision Theory	[Ch 14.3-14.4] [Ch 16.1-16.3 and 16.5]
	3/23		Project 2 Due	
12	3/25 3/27	3/26 3/28	***Advanced topics*** *(Chao traveling to National Science Foundation)* ***Advanced topics*** *(Chao traveling to National Science Foundation)*	
	3/29		Homework 2 Out	
13	4/1 4/3	4/2 4/4	Markov Decision Processes Decision Tree Learning	[Ch 17.1-17.2] [Ch 18.1-18.3]
	4/5 4/5		Homework 2 Due Project 3 Out	
14	4/8 4/10	4/9 4/11	Perceptron Learning Neural Network Learning	[Ch 18.6] [Ch 18.7]
15	4/15 4/17	4/16 4/18	Statistical Learning Reinforcement Learning	[Ch 20.2.1-20.2.2] [Ch 21.1-21.2]
16	4/22 4/24	4/23 4/25	Artificial Intelligence Ethics Wrap-Up and Final Review	
	4/26		Project 3 Due	
	5/3	5/2	**Final Exam** (2pm-4pm)	

Outline

- What is AI?
- Part I: Search
- Part II: Logical reasoning
- Part III: Probabilistic reasoning
- **Part IV: Machine learning**
  - Decision Tree Learning
  - Perceptron Learning
  - Neural Network Learning
  - Statistical Learning
  - **Reinforcement Learning**

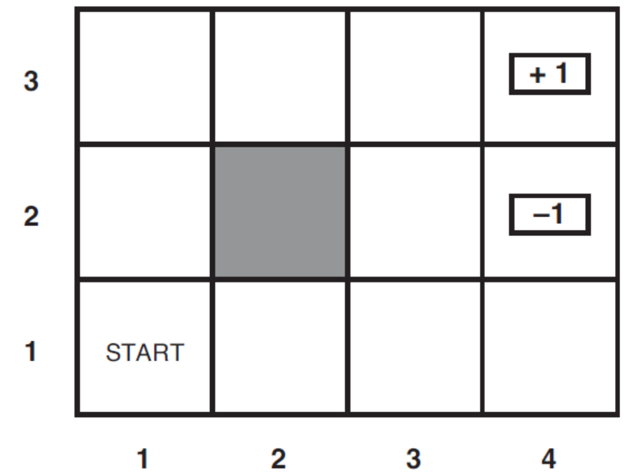
Example search problem

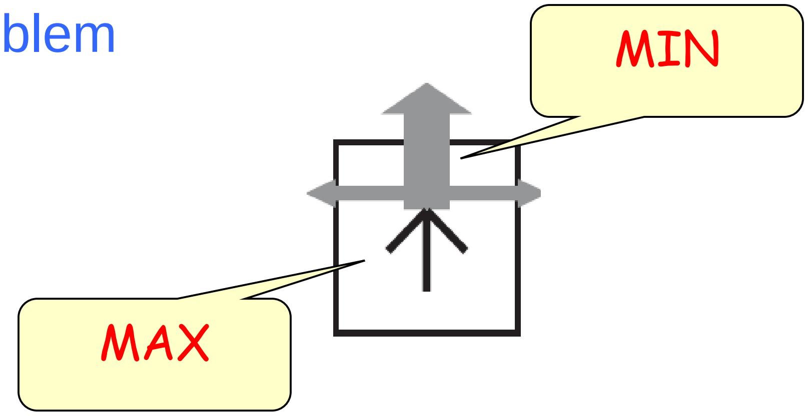
- Initial state: (1,1)
- Goal state: (4,3) utility +1
  (4,2) utility -1
- Actions:
  - Up, Down, Left, Right reward -0.04
  - *(won't move it running into the wall)*
- Transition model:
  - If RESULT(s, a) is deterministic, then it's a search problem



Example adversarial search problem

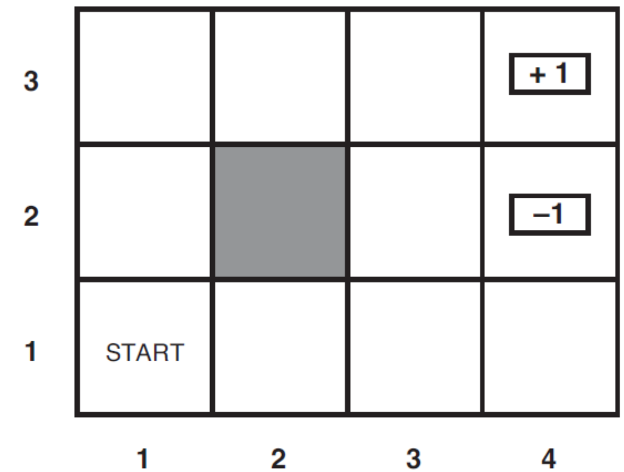
- Initial state: (1,1)
- Goal state: (4,3) utility +1
  (4,2) utility -1
- Actions:
  – Up, Down, Left, Right reward -0.04
  – *(won't move it running into the wall)*
- Transition model:
  – If RESULT(s, a) is non-deterministic, then it's an adversarial search problem

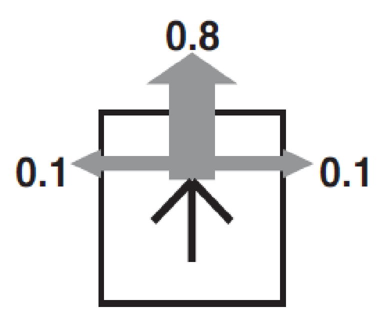




Example MDP (Markov Decision Process) problem

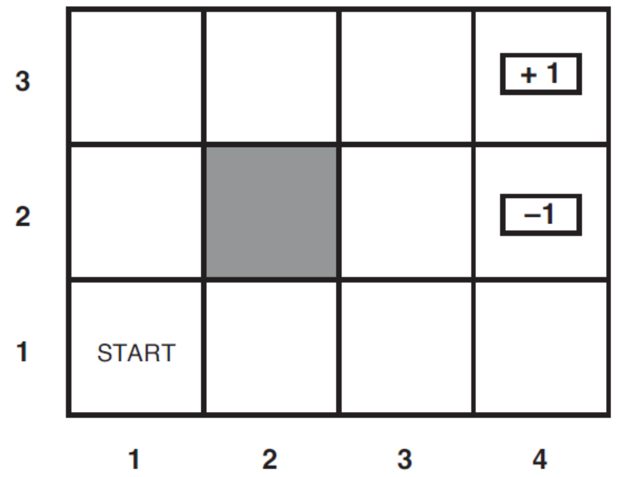
- Initial state: (1,1)
- Goal state: (4,3) utility +1
  (4,2) utility -1
- Actions:
  - Up, Down, Left, Right reward -0.04
  - *(won't move it running into the wall)*
- Transition model:
  - If RESULT(s, a) is **non-deterministic**, and **probabilistic**, then it's MDP (Markov decision process)

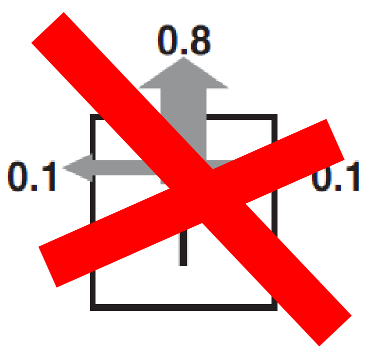




Example reinforcement learning problem

- Initial state: (1,1)
- Goal state: (4,3) utility +1
  (4,2) utility -1
- Actions:
  - Up, Down, Left, Right reward -0.04
  - ~~(won't move it running into the wall)~~
- Transition model:
  - If RESULT(s, a) is unknown, then it's reinforcement learning





Markov decision process (MDP)

- MDP is a sequential decision problem that has
  - a **fully observable, stochastic** environment
  - a **Markovian** transition model, and
  - the **additive** rewards

- State space formalism
  - Initial state (s0)
  - Actions(s) = {a1, a2, … } for each state
  - Transition model P(s' | s, a) for each state and each action
  - Reward function R(s)

Reinforcement learning: *Compared to MDP*

- MDP is a sequential decision problem that has
  - a **fully observable, stochastic** environment
  - ~~a **Markovian** transition model, and~~
  - the **additive** rewards

- State space formalism
  - Initial state (s0)
  - Actions(s) = {a1, a2, … } for each state
  - ~~Transition model P(s’ | s, a) for each state and each action~~
  - ~~Reward function R(s)~~

The Story So Far: MDPs and RL

Known MDP: Offline Solution

Goal	Technique
Compute U^*, Q^*, π^*	Value / policy iteration
Evaluate a fixed policy π	Policy evaluation

Unknown MDP: Model-Based

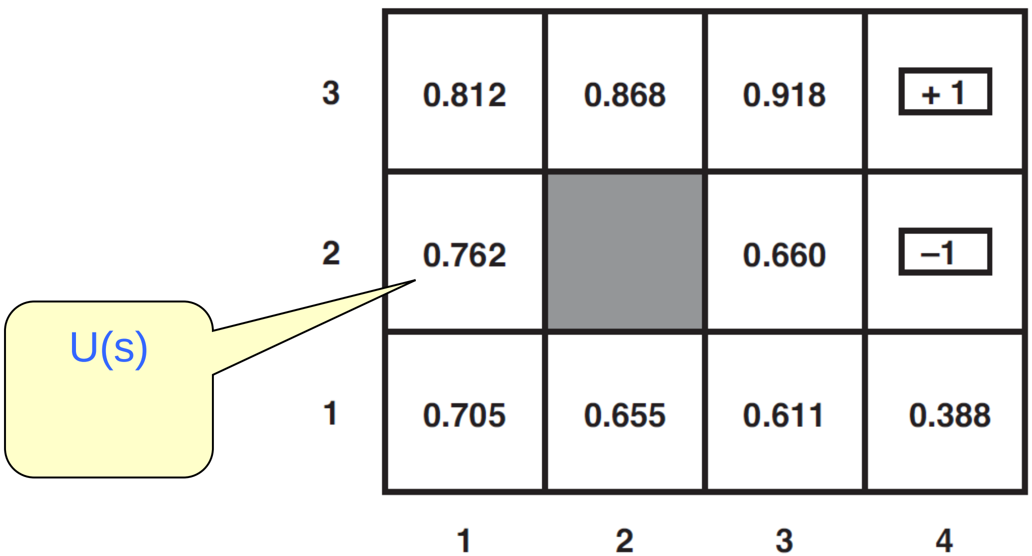
Goal	Technique
Compute U^*, Q^*, π^*	VI/PI on approx. MDP
Evaluate a fixed policy π	PE on approx. MDP

Unknown MDP: Model-Free

Goal	Technique
Compute U^*, Q^*, π^*	Q-learning
Evaluate a fixed policy π	Value Learning

Difference between R(s) and U(s)

- ***R(s)*** – the "**short-term**" reward for being in state (s)
- ***U(s)*** – the "**long-term**" **total** reward from (s) onward

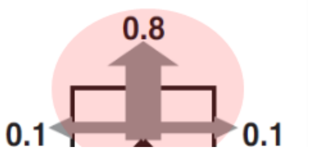


$$R(s) = \begin{cases} +1 \\ -0.04 \\ -1 \end{cases}$$

Given U(s), compute optimal policy π*(s)

- Pick the action with the maximal expected utility (MEU)

$$\pi^*(s) = \operatorname*{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$



	1	2	3	4
3	0.812	0.868	0.918	$\boxed{+1}$
2	0.762		0.660	$\boxed{-1}$
1	0.705	0.655	0.611	0.388

Up:

Down:

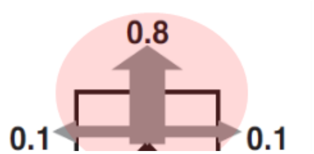
Left:

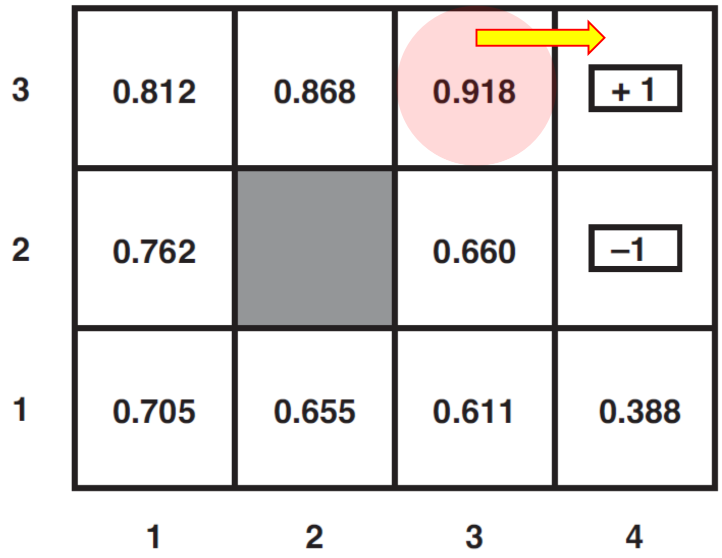
Right:

Given U(s), compute optimal policy π*(s)

- Pick the action with the maximal expected utility (MEU)

$$\pi^*(s) = \operatorname*{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$





Up:

*0.8*0.918 + 0.1*0.868 + 0.1*1 = 0.9212*

Down:

*0.8*0.660 + 0.1*0.868 + 0.1*1 = 0.7148*

Left:

*0.8*0.868 + 0.1*0.660 + 0.1*0.918 = 0.8522*

Right:

*0.8*1 + 0.1*0.918 + 0.1*0.660 =* $\boxed{0.9578}$

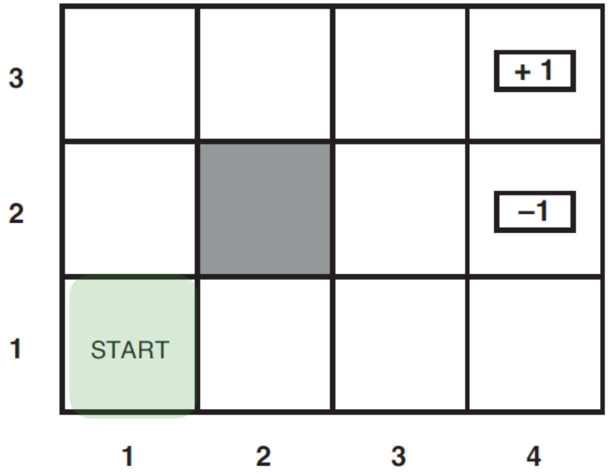
The problem is that "**we don't have** *U(s)*" in the first place

Bellman equation (1957)

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

Example:

$$\begin{aligned}
U(1,1) = -0.04 + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up) \\
& 0.9U(1,1) + 0.1U(1,2), && (Left) \\
& 0.9U(1,1) + 0.1U(2,1), && (Down) \\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]. && (Right)
\end{aligned}$$



Similarly, write down U(1,2), U(1,3), U(1,4),
U(2,1), U(2,2), U(2,3), U(2,4),
U(3,1), U(3,2), U(3,3), U(3,4)

Solutions to these equations are unique!

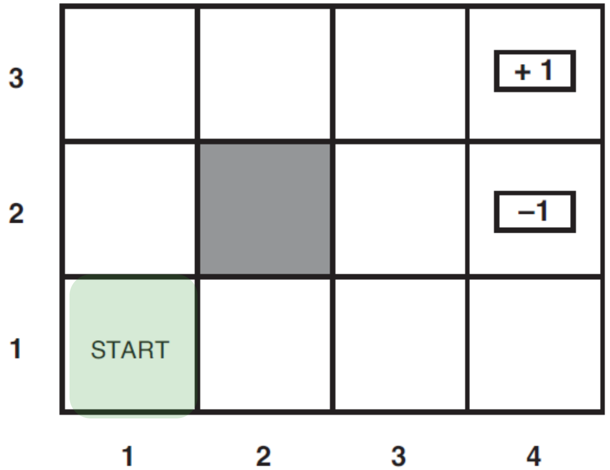
Bellman equation (1957)

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

You can't directly solve these equations due to the non-linear (**max**) operation

Example:

$$\begin{aligned} U(1,1) = -0.04 + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up) \\ & 0.9U(1,1) + 0.1U(1,2), && (Left) \\ & 0.9U(1,1) + 0.1U(2,1), && (Down) \\ & 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]. && (Right) \end{aligned}$$



Similarly, write down U(1,2), U(1,3), U(1,4),
U(2,1), U(2,2), U(2,3), U(2,4),
U(3,1), U(3,2), U(3,3), U(3,4)

Solutions to these equations are unique!

Policy evaluation

- Given a policy *π(s)*, compute *U(s)*

$$\boxed{U_i(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s')}$$

This is a set of *linear* equations, which is polynomial time solvable (by LP solver); in contrast, the Bellman equation $U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$ is nonlinear.

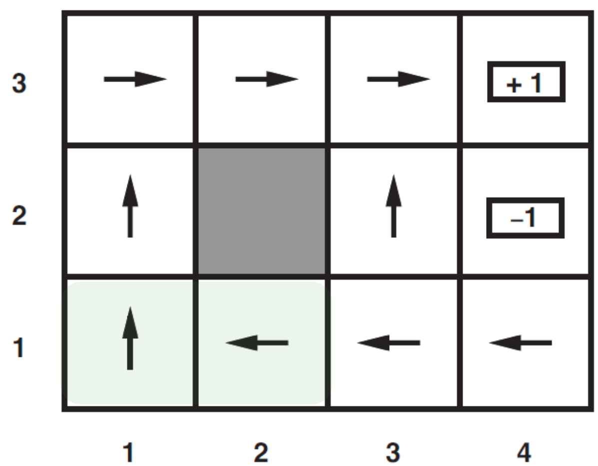
Policy evaluation: *example*

But you do not know the **transition model** *P(s' |s, a)*

- Given a policy *π(s)*, compute *U(s)*

$$\boxed{U_i(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s')}$$

This is a set of *linear* equations, which is polynomial time solvable (by LP solver); in contrast, the Bellman equation $U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$ is nonlinear.



$$\begin{aligned} U_i(1,1) &= -0.04 + 0.8U_i(1,2) + 0.1U_i(1,1) + 0.1U_i(2,1) \,, \\ U_i(1,2) &= -0.04 + 0.8U_i(1,3) + 0.2U_i(1,2) \,, \\ &\vdots \end{aligned}$$

You could have directly solved these equations

Reinforcement learning

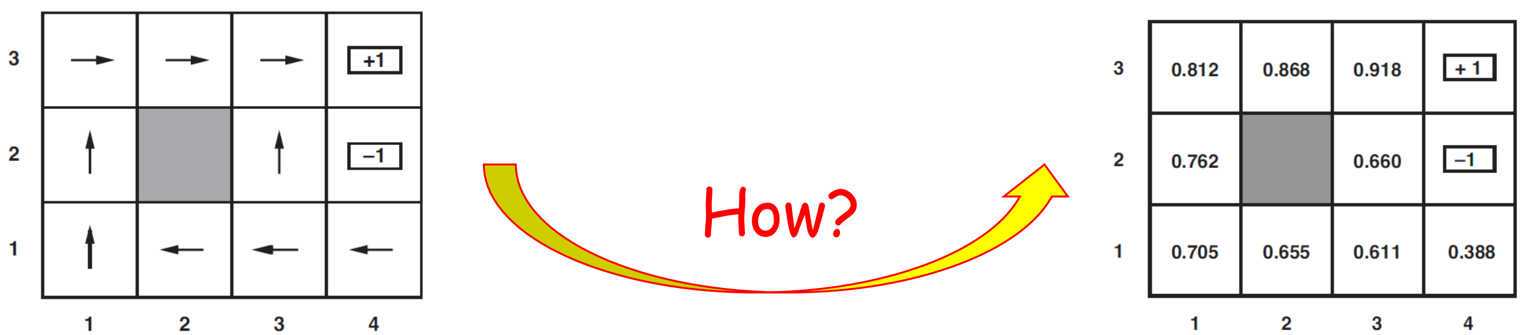
- Passive RL
  - The policy is fixed, but the transition model is unknown, and we want to learn the utility U(s)
- Active RL
  - Need to learn a policy (i.e., deciding what actions to take)

Reinforcement learning

- Passive RL
  - The policy is fixed, but the transition model is unknown, and we want to learn the utility U(s)
- Active RL
  - Need to learn a policy (i.e., deciding what actions to take)

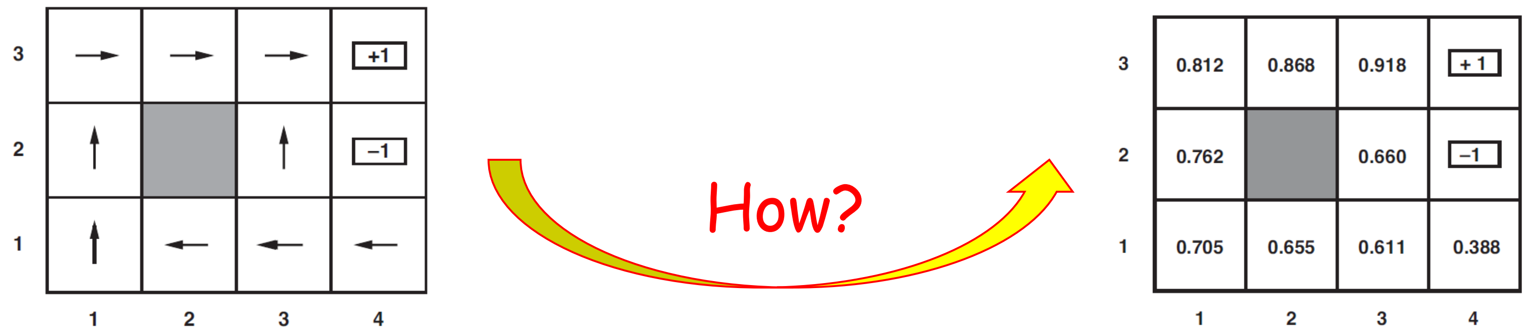
Passive Reinforcement Learning

- Objective
  - Assume policy π(s) is fixed, but transition model is unknown, and we would like to directly learn the utility U(s)



Passive Reinforcement Learning

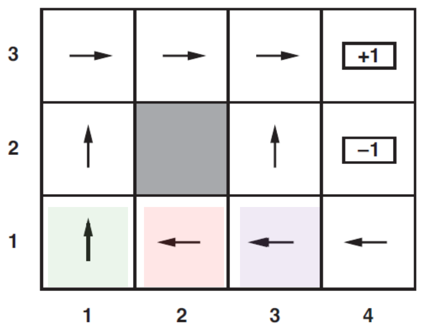
- Objective
  - Assume policy π(s) is fixed, but transition model is unknown, and we would like to directly learn the utility U(s)



- Executes a set of trials

$$(1,1)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (2,3)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (4,3)_{\text{+1}}$$
$$(1,1)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (2,3)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (3,2)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (4,3)_{\text{+1}}$$
$$(1,1)_{\text{-.04}} \rightsquigarrow (2,1)_{\text{-.04}} \rightsquigarrow (3,1)_{\text{-.04}} \rightsquigarrow (3,2)_{\text{-.04}} \rightsquigarrow (4,2)_{\text{-1}} \ .$$

Direct utility estimation *[Widrow & Hoff, 1960]*

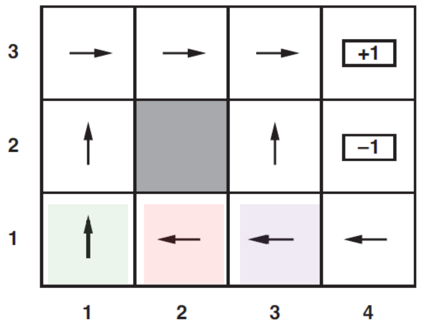


- Learn a map from states to utilities
  - (1,1) →
  - (1,2) →
  - (1,3) →
  - …

- Executes a set of trials

$$(1,1)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (2,3)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (4,3)_{\text{+1}}$$
$$(1,1)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (2,3)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (3,2)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (4,3)_{\text{+1}}$$
$$(1,1)_{\text{-.04}} \rightsquigarrow (2,1)_{\text{-.04}} \rightsquigarrow (3,1)_{\text{-.04}} \rightsquigarrow (3,2)_{\text{-.04}} \rightsquigarrow (4,2)_{\text{-1}} \;.$$

Direct utility estimation *[Widrow & Hoff, 1960]*



- Learn a map from states to utilities
  - (1,1) → 0.76
  - (1,2) →
  - (1,3) →
  - …

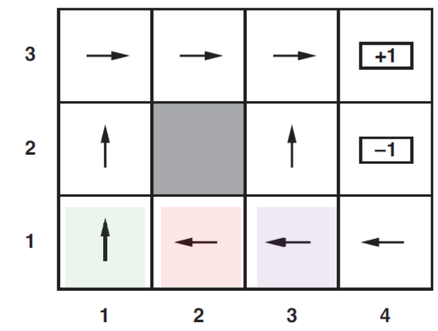
- Executes a set of trials

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}} .$$

Direct utility estimation *[Widrow & Hoff, 1960]*

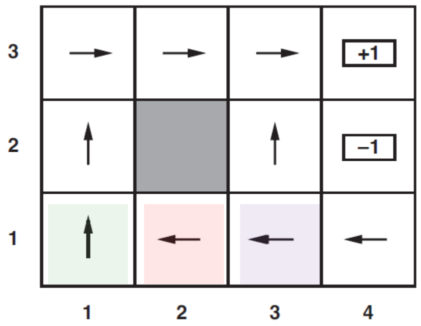


- Learn a map from states to utilities
  - (1,1) → 0.72
  - (1,2) → (0.76 + 0.84) / 2 = 0.80
  - (1,3) →
  - …

- Executes a set of trials

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}} \ .$$

Direct utility estimation *[Widrow & Hoff, 1960]*

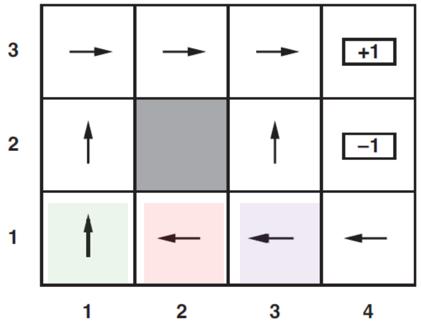


- Learn a map from states to utilities
  - (1,1) → 0.72
  - (1,2) → (0.76 + 0.84) / 2 = 0.80
  - (1,3) → (0.80 + 0.88) / 2 = 0.84
  - …

- Executes a set of trials

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}} .$$

Direct utility estimation *[Widrow & Hoff, 1960]*



- Learn a map from states to utilities
  - (1,1) → 0.76
  - (1,2) → (0.76 + 0.84) / 2 = 0.80
  - (1,3) → (0.80 + 0.88) / 2 = 0.84
  - …

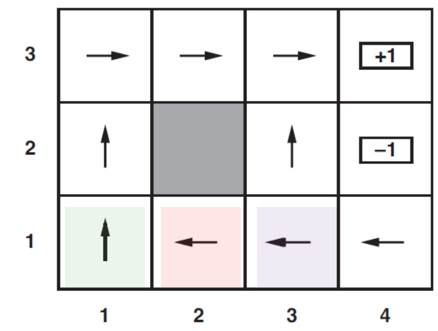
Problem: misses important information – *utilities of states are NOT independent!*

$$\boxed{U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^{\pi}(s')}$$

- Executes a set of trials

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}}\ .$$

Direct utility estimation *[Widrow & Hoff, 1960]*



- Learn a map from states to utilities
  - (1,1) → 0.76
  - (1,2) → (0.76 + 0.84) / 2 = 0.80
  - (1,3) → (0.80 + 0.88) / 2 = 0.84
  - …

Problem: misses important information – *utilities of states are NOT independent!*

Could have updated U(1,1) based on previous value of U(1,2) but it didn't…

$$\boxed{U^\pi(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^\pi(s')}$$

- Executes a set of trials

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}} \; .$$

Question

- While learning ***U(s)***, how to leverage state dependency (e.g., Bellman equation) to speed up the iterations?
  - Without it, “**Direct Utility Estimation**” method is slow to converge

Adaptive dynamic programming (ADP)

- Learn the transition model **P(s' | s,a)** first
  - **Input**: a "state-action" pair ***(s,a)***
  - **Output**: a new state ***s'***
  - Treat it as a table of probabilities
    - Count **how often** each action outcome ***s'*** occurs, given ***(s,a)***

$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$$
$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$$
$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (2,1)_{\mathbf{-.04}} \rightsquigarrow (3,1)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (4,2)_{\mathbf{-1}} \; .$$

$P((2,3) \mid (1,3), Right)$ is estimated to be 2/3

$$U^\pi(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^\pi(s')$$

Question

- How do you count?
  - Often times, the results need to be **normalized**
  - In other words,we want frequency (e.g., **P(s’ | s,a)**)

Example: Expected Age

Goal: Compute expected age of this **CS 360** students

Known P(A)

$$E[A] = \sum_a P(a) \cdot a \qquad = 0.35 \times 20 + \ldots$$

Without P(A), instead collect samples [a_1, a_2, … a_N]

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Why does this work? Because eventually you learn the right model.

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Why does this work? Because samples appear with the right frequencies.

[from Dan Klein and Pieter Abbeel, UC Berkeley]

Question

- How to compute the “**moving average**”?
  - Don’t want to wait for all samples to come in
  - Need to have the average of samples seen so far

Exponential Moving Average

- Exponential moving average
  - The running interpolation update: $\bar{x}_n = (1-\alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
  - Makes recent samples more important:

$$\bar{x}_n = \frac{x_n + (1-\alpha) \cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \dots}{1 + (1-\alpha) + (1-\alpha)^2 + \dots}$$

  - Forgets about the past (distant past values were wrong anyway)
- Decreasing learning rate (alpha) can give converging averages

Temporal-difference learning

- Idea: learn from every experience
  - Update *U(s)* each time we experience a transition *(s, a, s', r)*
  - Likely outcome *(s')* will contribute more often

- Adjust value based on the value of successor state
  - Moving average

Sample of U(s): $sample = R(s) + \gamma\, U^{\pi}(s')$

Update to U(s): $U^{\pi}(s) = (1 - \alpha)\, U^{\pi}(s) + (\alpha)\, sample$

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha\big(R(s) + \gamma\, U^{\pi}(s') - U^{\pi}(s)\big)$$

Temporal-difference learning *(example)*

- Suppose that, after the 1st trial, we have
  - U(1,3) = **0.84**
  - U(2,3) = 0.92
- In the 2nd trial, since transition (1,3) → (2,3) occurs again
  - U(1,3) = -0.04 + U(2,3) = -0.04 + 0.92 = **0.88**
  - Assume (γ=1) for ease of understanding

$$\boxed{U^\pi(s) \leftarrow U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))}$$

$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$$
$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$$
$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (2,1)_{\mathbf{-.04}} \rightsquigarrow (3,1)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (4,2)_{\mathbf{-1}} \;.$$

Temporal-difference learning *(example)*

- Suppose that, after the 1st trial, we have
  - U(1,3) = **0.84**
  - U(2,3) = 0.92
- In the 2nd trial, since transition (1,3) → (2,3) occurs again
  - U(1,3) = -0.04 + U(2,3) = -0.04 + 0.92 = **0.88**
  - Assume (γ=1) for ease of understanding

But we could do better

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

(0.88 - 0.84)

$$\begin{aligned}
&(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}} \\
&(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}} \\
&(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}} \; .
\end{aligned}$$

Reinforcement learning

- Passive RL
  - The policy is fixed, but the transition model is unknown, and we want to learn the utility U(s)
- Active RL
  - Need to learn a policy (i.e., deciding what actions to take)

Trade-off: *exploitation vs. exploration*

- When to explore?
  - **Random actions**: explore a fixed amount
  - **Better idea**: try to explore areas whose badness is not (yet) established, but eventually stop exploring

- Exploration function
  - Takes a value estimate **u** and a visit count **n**, and returns an optimistic utility, e.g.

$$\boxed{f(u,n) = u + k/n}$$

Q-learning

- Learn an "action-utility" representation, denoted ***Q(s,a)***
  - Q(s,a) denotes the value of doing action (a) in state (s)

$$\boxed{U(s) = \max_a Q(s,a)}$$

- **Insight:** Agent with ***Q(s,a)*** no longer needs ***P(s' | s,a)*** for action selection
  - Called "model-free" method

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

Detour: Q-Value Iteration

- Value iteration: find successive (depth-limited) values
  - Start with $U_0(s)$
  - Given U_k, calculate the depth k+1 values for all states:

$$\boxed{U^\pi(s) \leftarrow U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))}$$

- But Q-values are more useful, so compute them instead
  - Start with $Q_0(s,a)$
  - Given Q_k, calculate the depth k+1 Q-values for all Q-states:

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

Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you’re acting sub optimally!
- This is called off-policy learning
- Caveats:
  - You have to explore enough…
  - In the limit, it doesn’t matter how you select actions (!)

The Story So Far: MDPs and RL

Known MDP: Offline Solution

Goal	Technique
Compute U^*, Q^*, π^*	Value / policy iteration
Evaluate a fixed policy π	Policy evaluation

Unknown MDP: Model-Based

Goal	Technique
Compute U^*, Q^*, π^*	VI/PI on approx. MDP
Evaluate a fixed policy π	PE on approx. MDP

Unknown MDP: Model-Free

Goal	Technique
Compute U^*, Q^*, π^*	Q-learning
Evaluate a fixed policy π	Value Learning