LEARNING TO ACT

Oliver Schulte

Simon Fraser University

OUTLINE

- What is Reinforcement Learning?
- Key Definitions
- Key Learning Tasks
- Reinforcement Learning Techniques
- Reinforcement Learning with Neural Nets

OVERVIEW

LEARNING TO ACT

- So far: learning to predict
- Now: learn to **act**
  - In engineering: control theory
  - Economics, operations research: decision and game theory

EXAMPLES

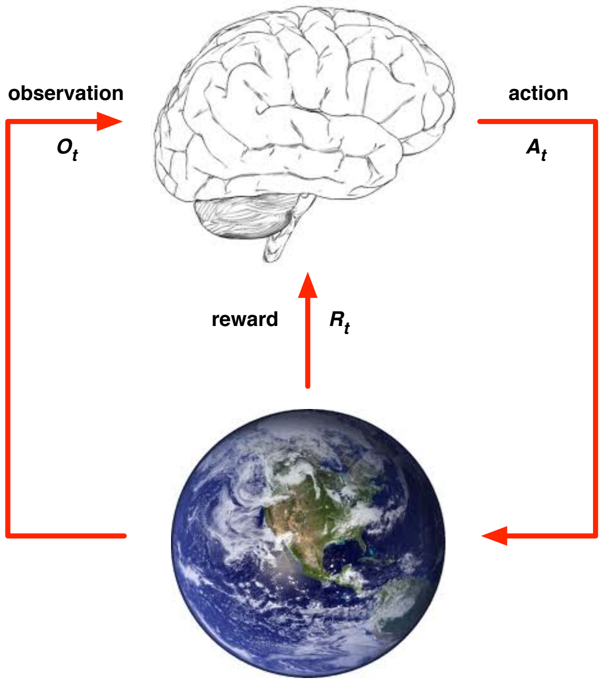
- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon, Go
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play Starcraft, Atari games better than humans
- Drive a car
- Play hockey

A NEW KIND OF LEARNING

- There is no supervisor, only a reward signal
  - No labels "wrong choice, right choice"
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

RL FRAMEWORK

- At each step t the **agent:**
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The **environment:**
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}



ACTING IN ACTION

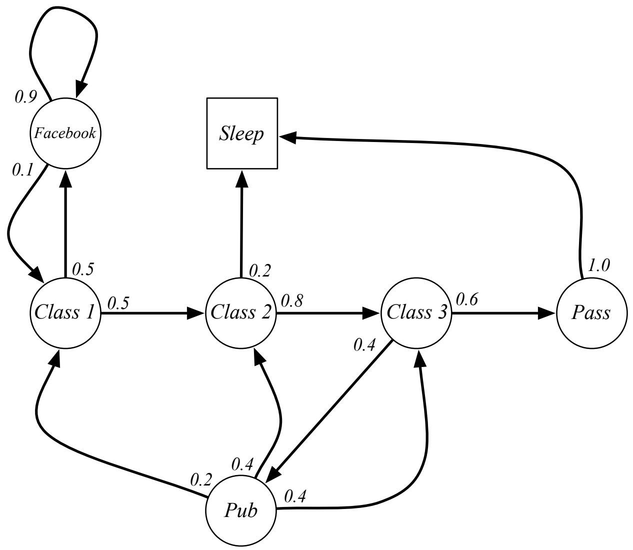
- Autonomous Helicopter
  - An example of **imitation learning**: start by observing human actions
- Learning to play video games
  - “Deep Q works best when it lives in the moment”
- Learn to flip pancakes

MARKOV DECISION PROCESSES

MARKOV PROCESS

- Aka Markov Chains
- Think about atomic representation of environment state (Russell and Norvig)
- Like state space in problem search
- A Markov process moves from one state to another with a certain probability
- Transition probability: $P(s_{t+1} = s' | s_t = s)$
- Demo

EXAMPLE: STUDENT LIFE



Sample episodes for Student Markov Chain starting from $S_1 = C1$

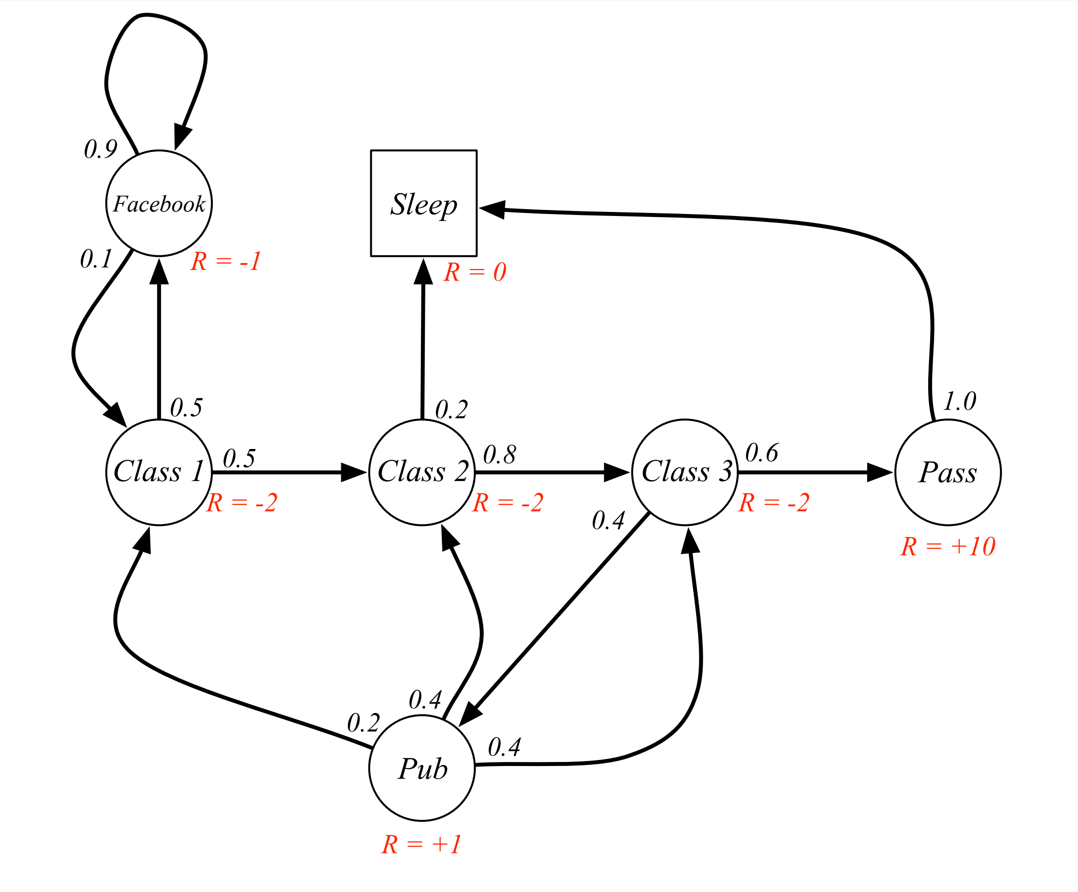
$$S_1, S_2, ..., S_T$$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

Source:
David Silver

MARKOV REWARD PROCESS

- Markov Process + Reward R_s associated with state
- More generally reward for *transition* R(s,s')



MARKOV DECISION PROCESSES

- Markov decision process (MDP) = Markov reward process + **actions**
- Transition probabilities, rewards depend on actions
- Markov game = MDP with actions, rewards for > 1 agent

EXAMPLE: STUDENT MARKOV DECISION PROCESS

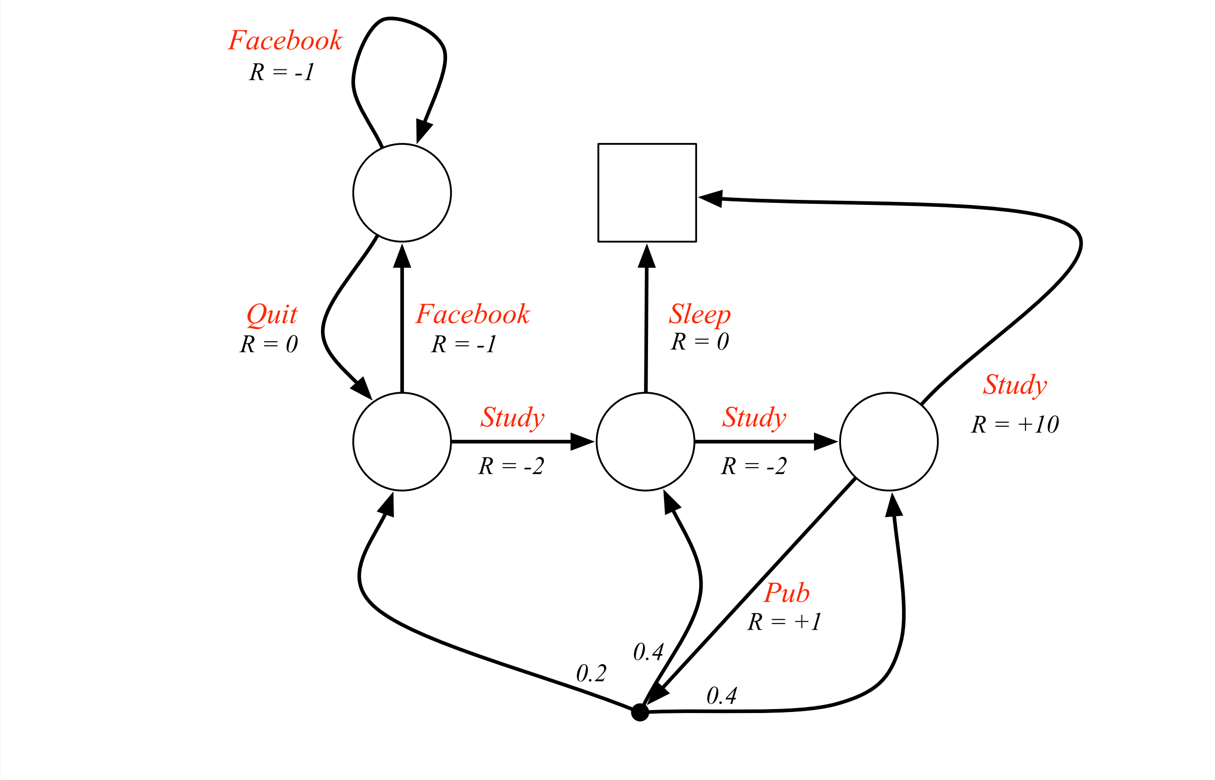
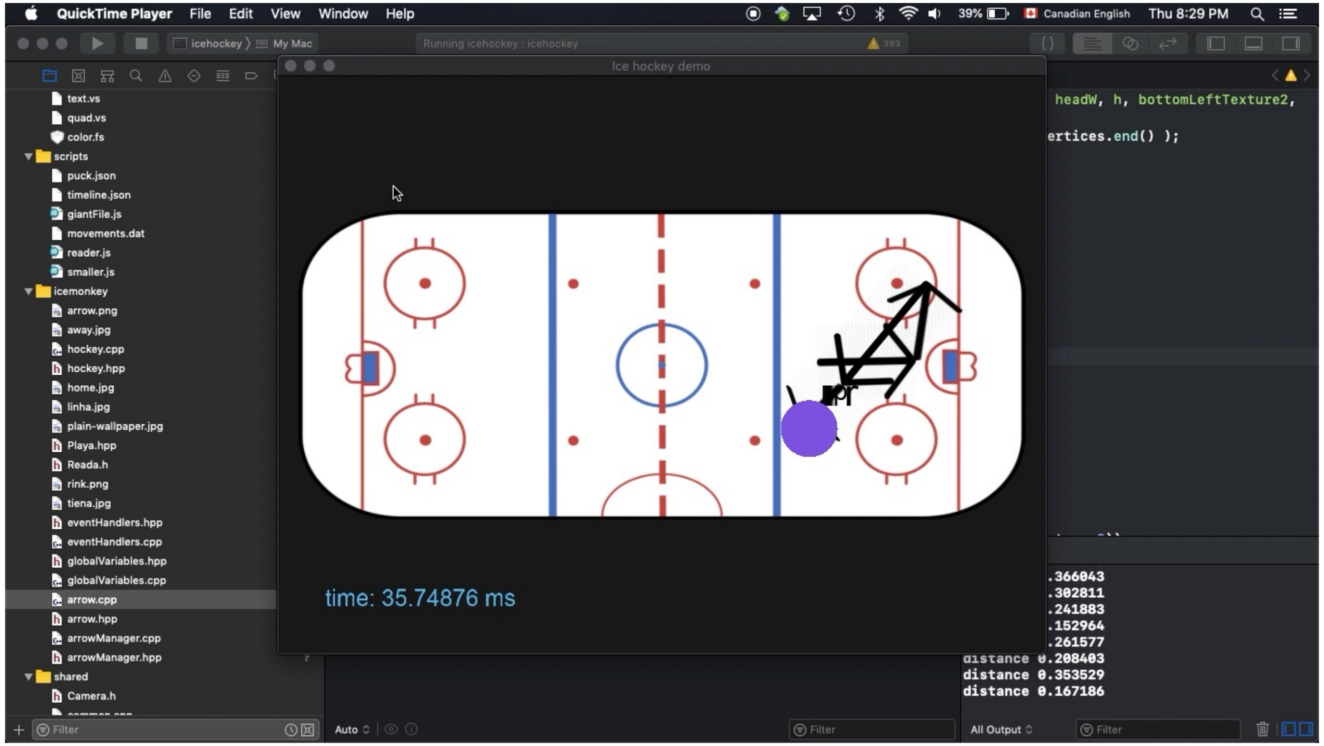


Figure from David Silver, Lectures on Reinforcement Learning

HOCKEY EXAMPLE

What are the states?
What are the rewards?



MARKOV CHAINS

Theory and Algorithms

EXERCISES

- Consider a Markov chain like the one shown in this demo
- What is the probability of the sequence AABB?
- What is the longest possible sequence of observations?

MULTI-STEP TRANSITIONS

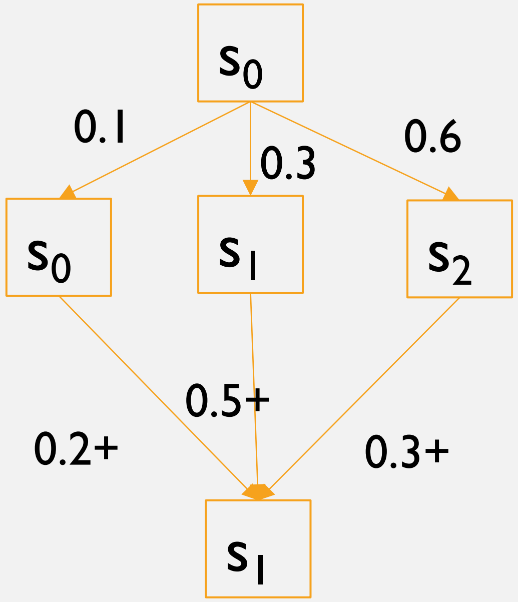
- What is the chance that if we start in state s we will reach state s' after a fixed number of n steps?
  - Think: from initial state, what is the chance of reaching a goal state in n steps?
- E.g. in this demo, what is the chance that we reach state 3 from 0 after 3 steps?
- What we want is a step-n transition matrix – how can we compute this efficiently?
- Notation: $P_n(s'|s)$

DYNAMIC PROGRAMMING

- Think Iterative Deepening: Build up transition matrices for 1, 2,…,n-1, n steps.
- For n = 1: Use given transition matrix $P(s'|s) = P_1(s'|s)$
- For n+1: $P_{n+1}(s'|s)=\sum_{s*} P(s'|s*) \times P_n(s'|s*)$

TREE VISUALIZATION FOR N=3

To compute $P_3(s_1|s_0)$



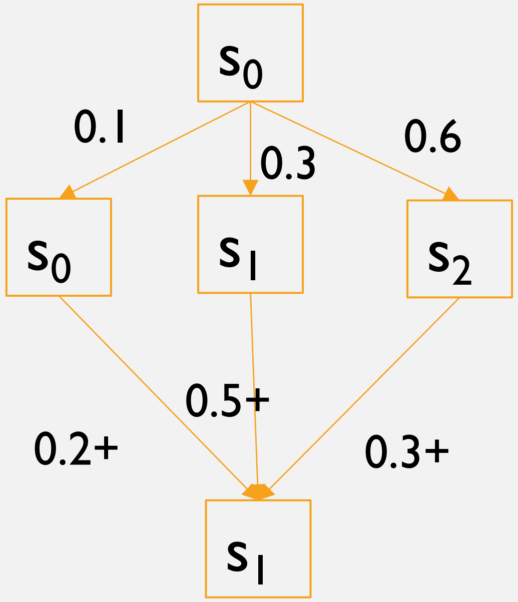
	s_0	s_1	s_2
s_0	0.1	0.3	0.6

Uses 1-step transition probability $P(s^*|s_0)$

Uses 2-step transition probability $P_2(s_1|s^*)$
e.g. $P_2(s_1|s_0) = 0.2$

TREE VISUALIZATION

To compute $P_n(s_1|s_0)$



	s_0	s_1	s_2
s_0	0.1	0.3	0.6

Uses 1-step transition probability $P(s^*|s_0)$

Uses (n-1)-step transition probability $P_{(n-1)}(s_1|s^*)$

INFINITE CHAINS

- What if we let the number of steps *n* go to infinity?
- It can be shown that under certain conditions on the chain, there is a limit transition probability matrix $P_{\infty}(s'|s)$
- This is the **stationary** transition matrix

PERFORMANCE METRIC FOR MDPS

FACTORED STATES

- In practice, RL uses a **factored state representation**

➢The state is defined by a list of values for a set of variables.

  - E.g. in hockey, can include score, game time, locations of players, location of puck

- If we have only 2 integer variables *x* and *y*, we can visualize states in a grid world

GRID WORLD EXAMPLE

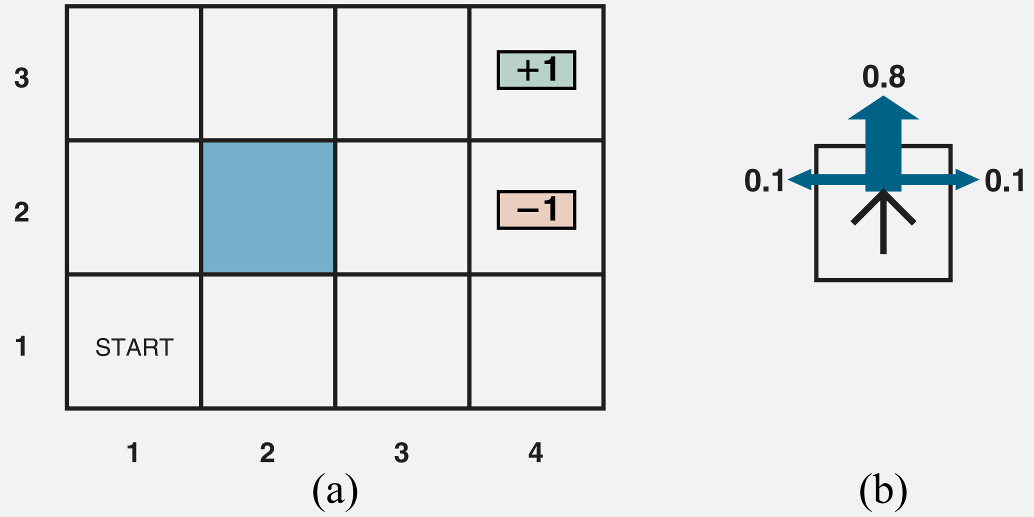


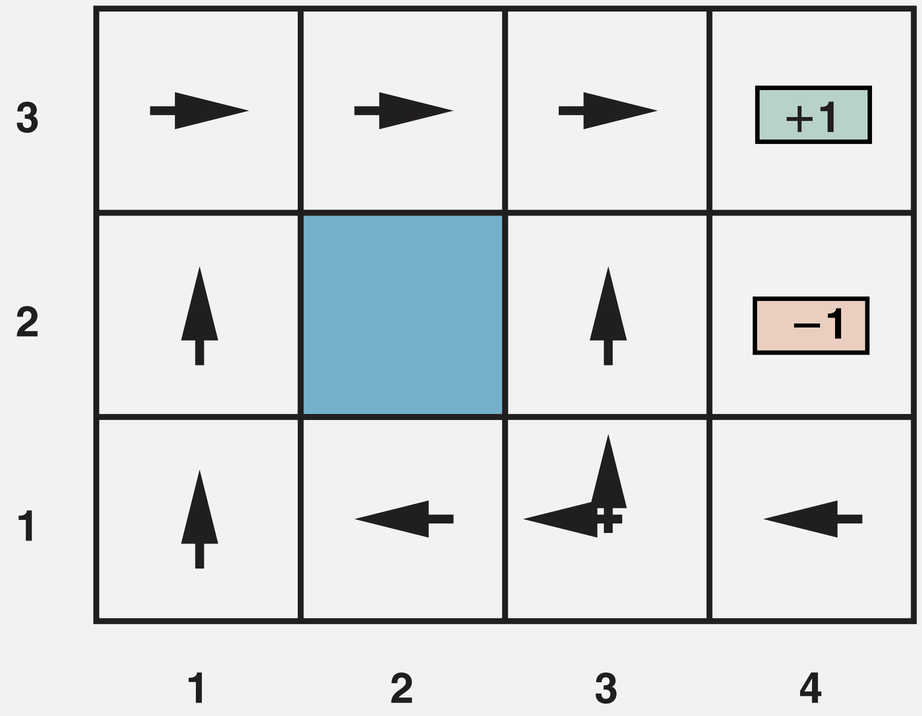
Figure 17.1 (a) A simple, stochastic 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward +1 and –1, respectively, and all other transitions have a reward of –0.04.

Fig. Russell and Norvig 2010

POLICIES

- A deterministic **policy** π is a function that maps states to actions
  - $\pi(s)=a$
  - i.e. tells us how to act
- Can also be probabilistic $\pi(a|s)$
- Can be implemented using neural nets.

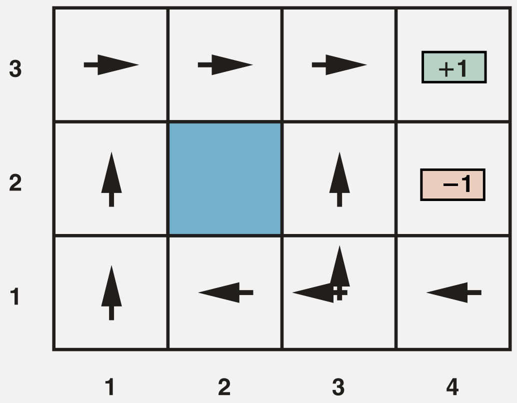
POLICY EXAMPLE

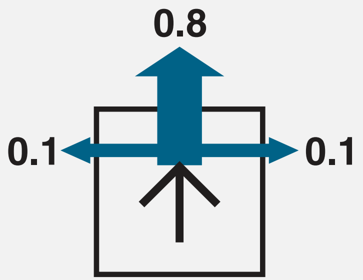


TRAJECTORIES

- A trajectory/episode is a sequence $s_1,a_1,r_1,s_2, a_2, r_2,\ldots,s_n,a_n,r_n$
- Length of trajectory = n
- A policy π and MDP transition probabilities p(s'|s,a) determine a probability for every trajectory

EXAMPLE + EXERCISE 1

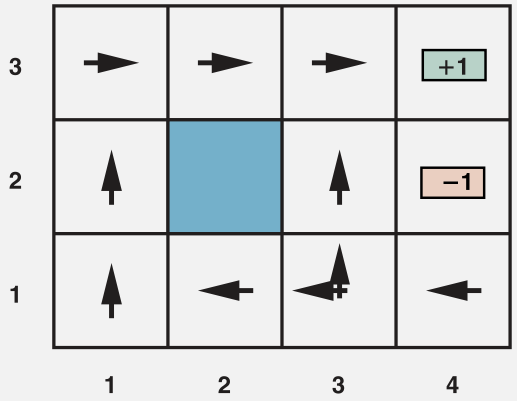


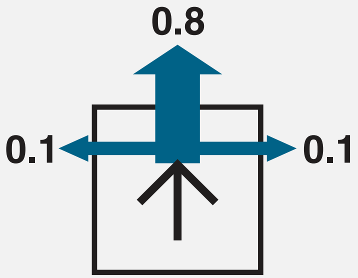


- Note that the trajectory probability depends on both the policy and the MDP
- Last action-reward not shown (partial trajectory)

State	Action	Reward	State	Action	Reward	State	Probability
(1,1)	Up	-0.04	(1,2)	Up	-0.04	(1,3)	0.8x0.8
(1,1)	Up	-0.04	(1,2)	Up	-0.04	(1,2)	?
(1,1)	Up	-0.04	(1,2)	Right	-0.04	(1,2)	?

EXAMPLE + EXERCISE II



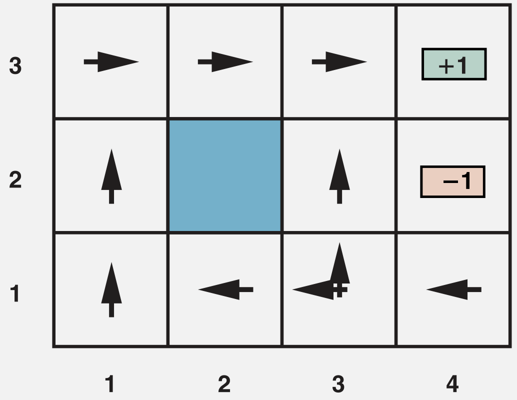


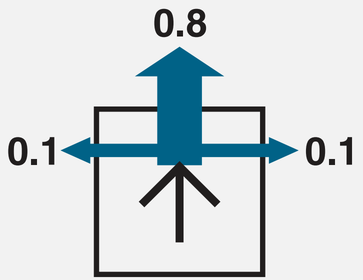
- Starting at state (1,1), how many trajectories are there of length
  - 1
  - 2
  - 3

RETURNS AND DISCOUNTING

- The return of a trajectory is the total sum of rewards.
- Typically rewards are weighted by a *discount factor* γ between 0 and 1.
- Return = $r_0+\gamma r_1+\gamma^2 r_2+\ldots$

RETURN EXAMPLE + EXERCISE





γ =0.5

What if γ = *1*?

State	Action	Reward	State	Action	Reward	State	Probability	Return
(1,1)	Up	-0.04	(1,2)	Up	-0.04	(1,3)	0.8x0.8	-0.04-0.5x0.04
(1,1)	Up	-0.04	(1,2)	Up	-0.04	(1,2)	0.8x0.2	?
(1,1)	Up	-0.04	(1,2)	Right	-0.04	(1,2)	0	?

WHY DISCOUNT?

- Most Markov reward and decision processes are discounted. Why?
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Mathematically convenient to discount rewards for infinite trajectories (more below)
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
  - There may be a small probability that process ends
- Animal/human behaviour shows preference for immediate reward
- If all trajectories are guaranteed to terminate, we can use undiscounted sum ($\gamma = 1$)

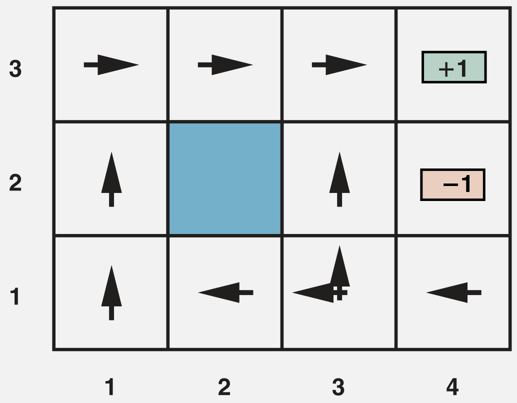
THE VALUE FUNCTION: PERFORMANCE METRIC FOR POLICIES

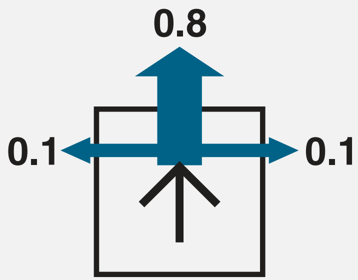
- Maximize **expected return (total reward)** of policy π from state *s* $= \sum_{\text{trajectories } \tau} p(\tau|s,\pi) \times \text{return}(\tau)$
- We write $V^{\pi}(s)$ for the **expected return** of policy π from state s
- A policy π is optimal if for every state s, the policy achieves the maximum expected return

➢ A policy π* is **optimal** if for any other policy π and for all states s $V^{\pi^*}(s) \geq V^{\pi}(s)$

- The value of an optimal policy is written as $V^*(s)$.

EXAMPLE + EXERCISE III





- Add up the contributions to $V^{\pi}(1,1)$ for the three trajectories shown
- $\gamma = 0.5$

State	Action	Reward	State	Action	Reward	State	Probability	Return
(1,1)	Up	-0.04	(1,2)	Up	-0.04	(1,3)	0.8x0.8	-0.04-0.5x0.04
(1,1)	Up	-0.04	(1,2)	Up	-0.04	(1,2)	0.8x0.2	-0.04-0.5x0.04
(1,1)	Up	-0.04	(1,2)	Right	-0.04	(1,2)	0	-0.04-0.5x0.04

OPTIMAL POLICIES: EXAMPLE

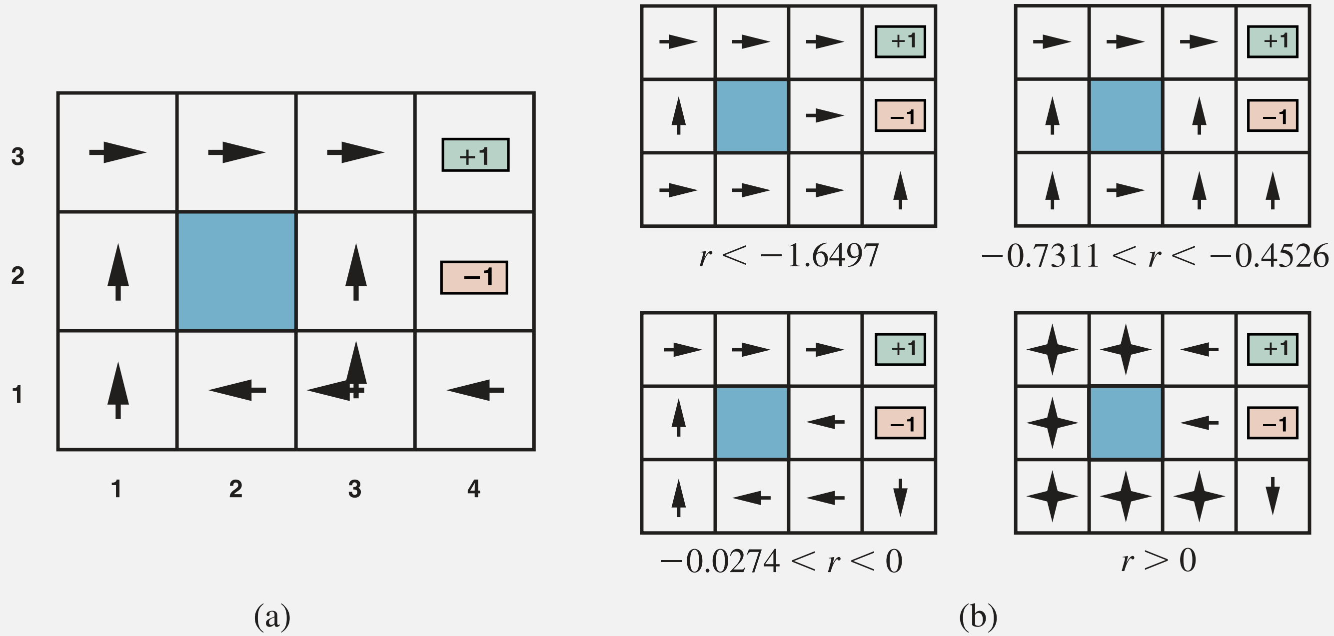


Figure 17.2 (a) The optimal policies for the stochastic environment with $r = -0.04$ for transitions between nonterminal states. There are two policies because in state (3,1) both *Left* and *Up* are optimal. (b) Optimal policies for four different ranges of r.

COMMENTS ON THE VALUE FUNCTION

- A powerful look-ahead concept.
  - Like searching through an entire search tree for expected success
- Game example: chance of winning, expected total score.
- Dr. Strange looks ahead
- Can also be computed by a neural network

OPTIMAL VALUE FUNCTION: EXAMPLE

3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	–1
1	0.7453	0.6953	0.6514	0.4279
	1	2	3	4

Figure 17.3 The utilities of the states in the 4×3 world with $\gamma = 1$ and $r = -0.04$ for transitions to nonterminal states.

DYNAMIC PROGRAMMING

- Searching through the space of policies is infeasible
- Instead use a dynamic programming approach: Find an optimal policy for $1, 2, \ldots, n, n+1$ steps.
- Eventually can consider letting *n* go to infinity

POLICY EVALUATION

Finite Horizon Case

INITIALIZATION

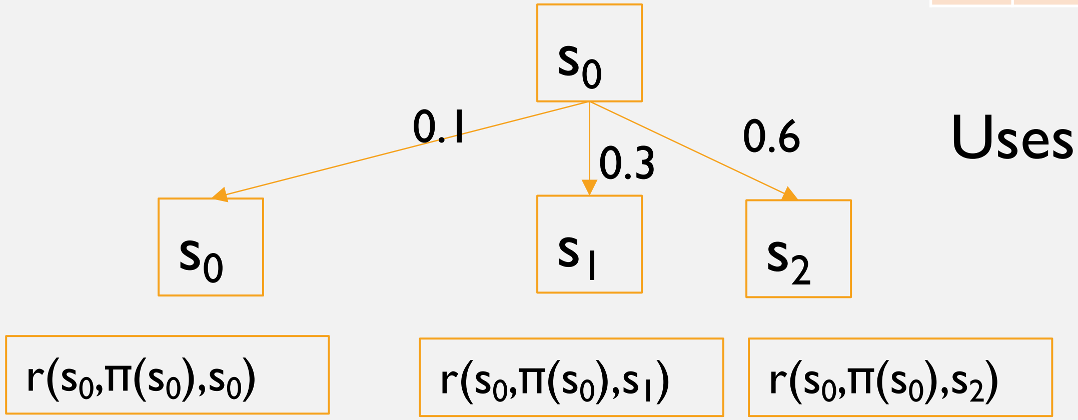
- We start by computing the value function $V^{\pi}(s)$ for a fixed policy π (not necessarily optimal).
- How can we compute the values $V_1^{\pi}(s)$ = expected return after 1 step?
- Directly from MDP:
  $V_1^{\pi}(s)=\sum_{s'} p(s'|\pi(s),s) \times r(s,a,s')$

Probability of next state given current state and policy action

Reward associated with transition

TREE VISUALIZATION

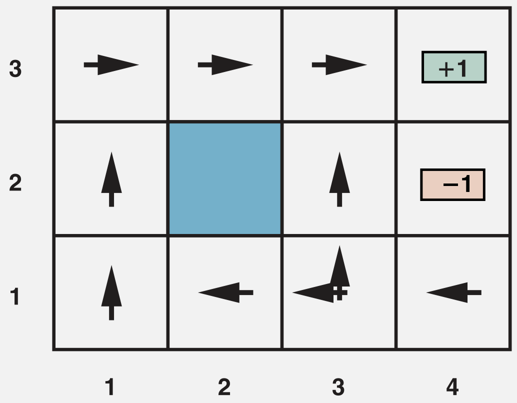
To compute $V_1^{\pi}(s_0)$

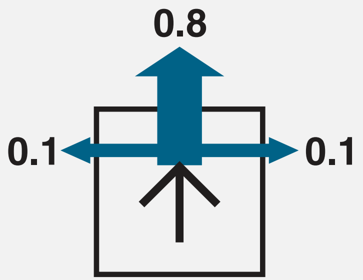


	s_0	s_1	s_2
s_0	0.1	0.3	0.6

Uses 1-step transition probability $P(s^*|s_0)$

EXAMPLE + EXERCISE





- Compute $V_1^{\pi}(1,1)$
- Exercise: what if $\pi(1,1)$=Right?
- So which move is better Up or Right?

Next State	Reward	Probability	XReward	Sum =
(1,2)	-0.04	0.8	0.8 x -0.04	
(2,1)	-0.04	0.1	0.1 x -0.04	
(1,1)	-0.04	0.1	0.1 x -0.04	

POLICY EVALUATION: BELLMAN UPDATE

- Suppose we have computed $V_n^\pi(s)$ = expected return after n steps
- How can we update to compute $V_{n+1}^\pi(s)$?
- $V_{n+1}^\pi(s) = \sum_{s'} P(s'|s, \pi(s)) \times [r(s, \pi(s), s') + \gamma V_n^\pi(s')]$

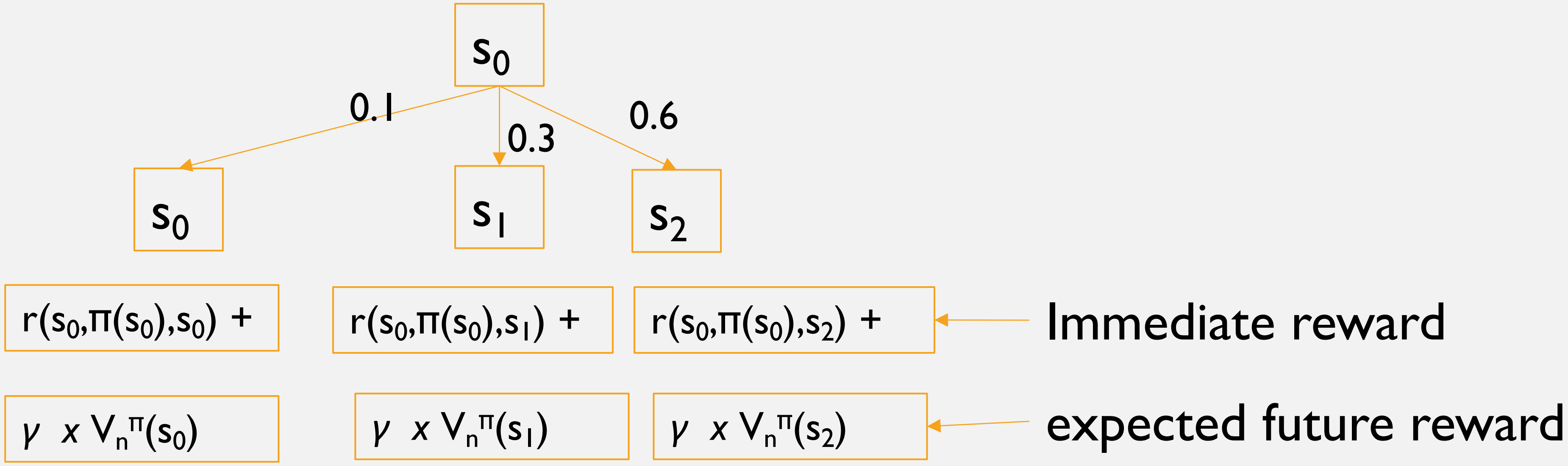
Immediate reward expected future reward

TREE VISUALIZATION

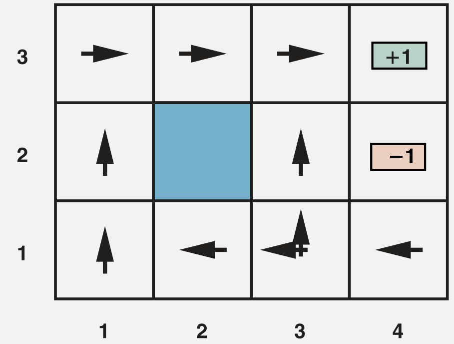
To compute $V_{(n+1)}^{\pi}(s_0)$

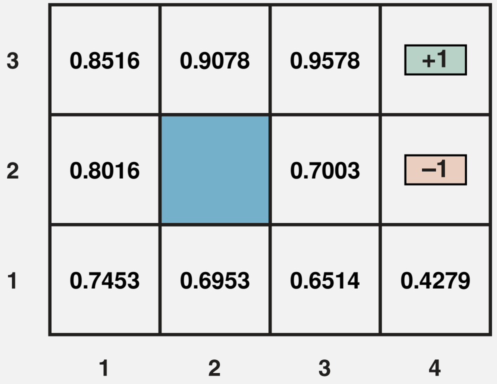
	s_0	s_1	s_2
s_0	0.1	0.3	0.6

1-step transition probability $P(s^*|s_0)$

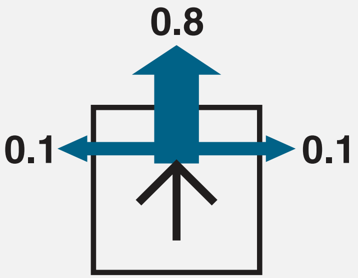


EXAMPLE + EXERCISE





- Suppose that $V_{(n)}^{\pi}$ is as shown
- Compute $V_{(n+1)}^{\pi}(1,1)$
- Assume no discounting



Next State	Reward	Probability	XReward	Future Reward	Sum = ?
(1,2)	-0.04	0.8	0.8 x -0.04		
(2,1)	-0.04	0.1	0.1 x -0.04		
(1,1)	-0.04	0.1	0.1 x -0.04		

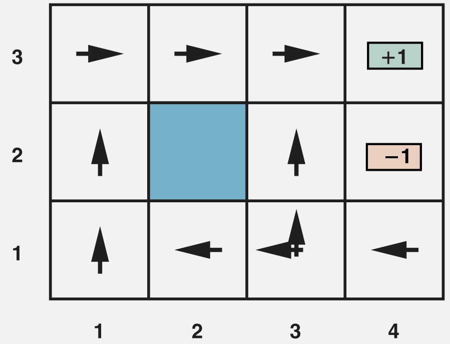
VALUE ITERATION: POLICY EVALUATION

- Input: MDP, policy π, depth d
- $V^{\pi}(s) := 0$ for all s
- For $i = 1$ to d
  - For all s do
    $V^{\pi}(s) = \sum_{s'} P(s'|s, \pi(s)) \times [r(s, \pi(s),s') + \gamma V^{\pi}(s')]$
- End for
- Return V^{π}

grid world demo

VALUE ITERATION FOR SOLVING AN MDP

EXERCISE



	1	2	3	4
3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	−1
1	0.7453	0.6953	0.6514	0.4279

- Given the value function shown, what is the best move at
  - (1,1)
  - (2,3)?

FROM VALUE TO POLICY

- It is easy to **extract** a policy from a value function:
- At each state, choose an action that maximizes expected future return
- $\pi^*(s) = \text{argmax}_a \sum_{s'} P(s'|s, a) \times [r(s, a,s') + \gamma V(s')]$
  $= \text{argmax}_a \; Q^*(s,a)$
- $Q^*(s,a)$ is known as the **action-value** function
  - = the expected total return if we choose action *a* in state *s*

VALUE ITERATION: OPTIMAL VALUE FUNCTION

- Input: MDP, ~~policy π,~~ depth *d*
- $V^*(s) := 0$ for all s
- For *i* = 1 to *d*
  - For all s do
    $V^*(s) = \max_a \sum_{s'} P(s'|s, a) \times [r(s, a,s') + \gamma V^\pi(s')]$
    $= \max_a Q^*(s,a)$
- End for
- Return V^*

Demo

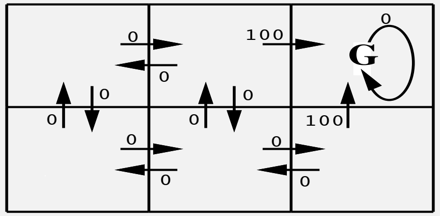
EXTENSION TO INFINITE HORIZON

- It is often useful to let the process run to any depth
- MDP may run forever ("neverending learning")
- Even if each trajectory is guaranteed to be finite, we may not know a definite upper bound in advance (termination uncertainty)
- Even if we know an upper bound in advance, it can introduce undesirable complications
  - E.g. every video game ends within 10 hours but at the beginning players don't think about the end
- Typically the value function changes very little at a modest depth (e.g. $d = 13$ for the NHL)

VALUE ITERATION: INFINITE HORIZON

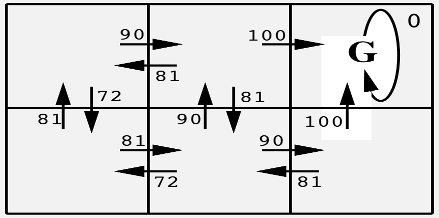
- Input: MDP, ~~policy π, depth *d*~~
- $V^*(s) := 0$ for all s
- Repeat until convergence
  - For all s do
    $V^*(s) = \max_a \sum_{s'} P(s'|s, a) \times [r(s, a,s') + \gamma V^{\pi}(s')]$
- Return V^*

RL CONCEPTS

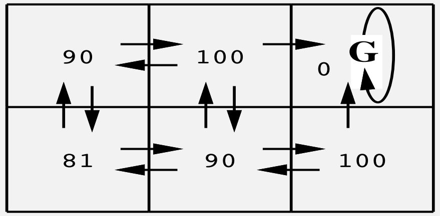


$r(s, a)$ (immediate reward) values

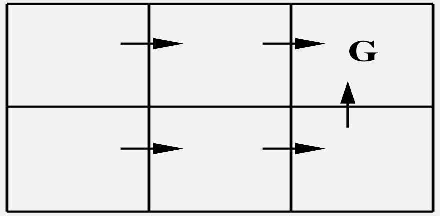
These 3 functions can be computed by neural networks



$Q(s, a)$ values



$V^*(s)$ values



One optimal policy

SUMMARY

- Reinforcement Learning: learning to act
- Adds *actions* and *rewards* to a temporal Markov model
- Inference/Planning: find optimal policy given fully specified MDP
  - Value iteration: find optimal value function, extract policy
  - Policy iteration: alternate policy evaluation and policy extraction
- Learning problems (next)
  - Value function: Estimate the expected cumulative reward given a state for a given policy/ an optimal policy
  - Agent discovery: Learn an optimal policy