COMP 551 – Applied Machine Learning
Lecture 17: Reinforcement Learning

Instructor: Ryan Lowe *(ryan.lowe@cs.mcgill.ca)*

Slides by: Ryan Lowe

Class web page: *www.cs.mcgill.ca/~hvanho2/comp551*

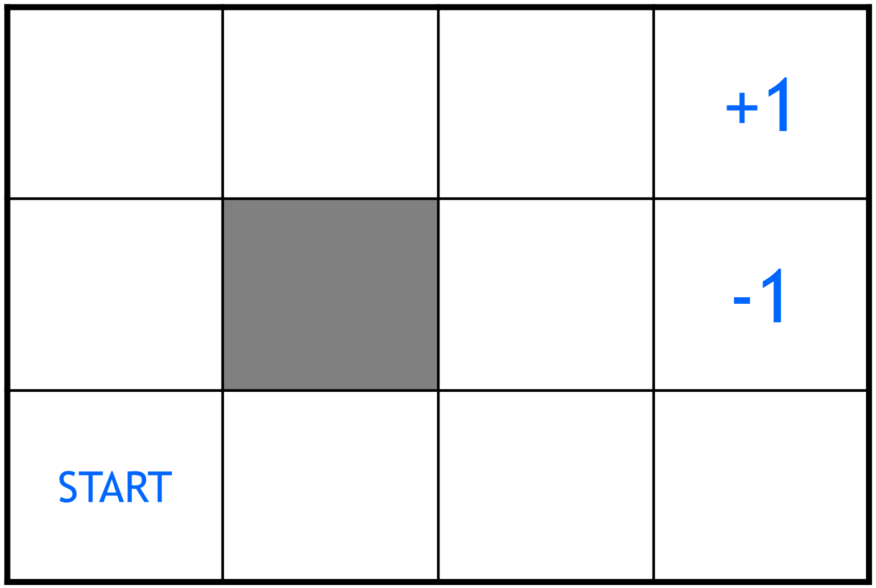
Announcements

- First round of project presentations is this Wednesday, 6-7:30pm in
  - Instructions in announcements / email
  - Covers
  - Starts **right on time**
- TAs (Prasanna + Sanjay) will host a joint office hour for assignment 3 clarifications, from **1-2pm on Thursday in TR 3104**
- My office hours today will be from 11am-12pm outside MC111N (lounge area)

Reinforcement learning

- RL is a general-purpose framework for decision-making
  - RL is for an **agent** with the capacity to **act**
  - Each **action** influences the agent's future **state**
  - Success is measured by a scalar **reward** signal
  - Goal: **select actions to maximise future reward**

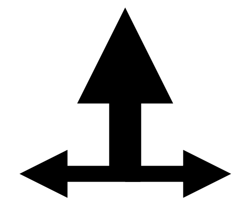
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT

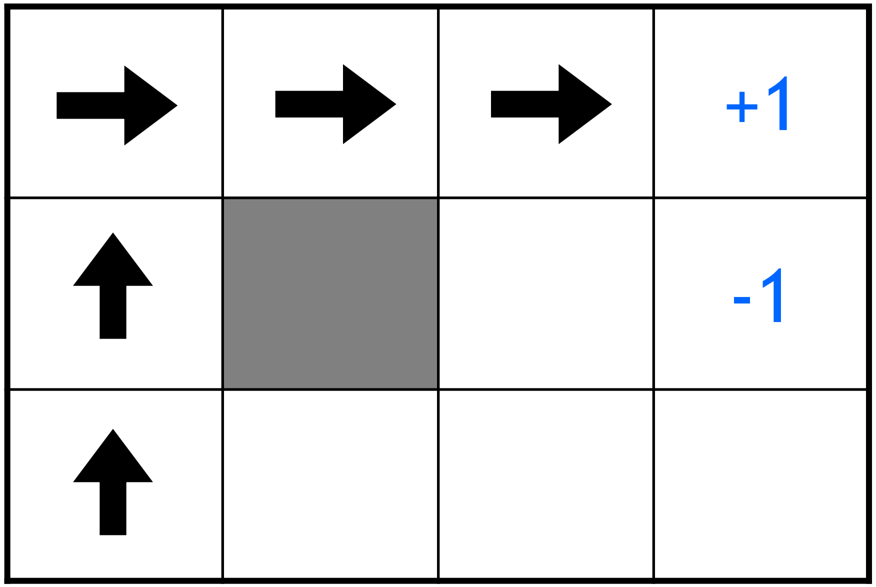


reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

- what is the solution?

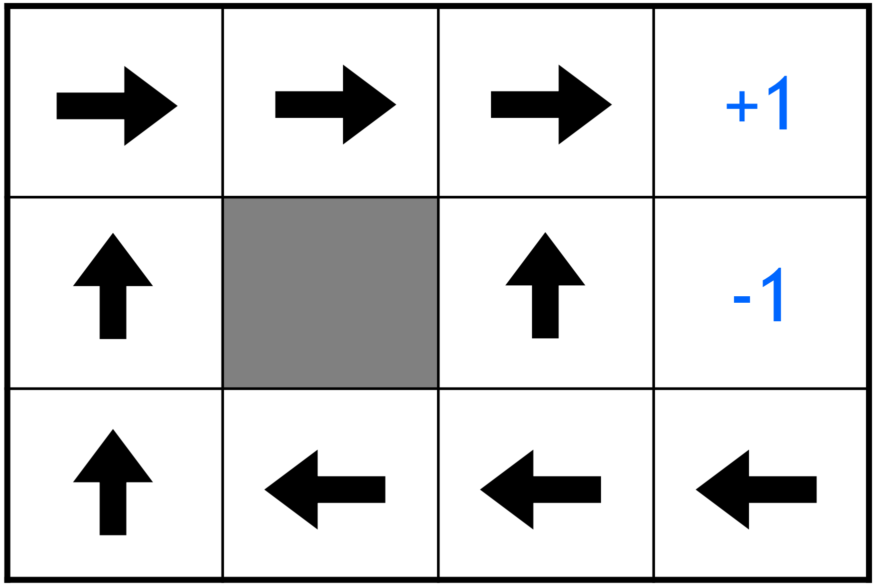
Example and slides from Peter Bodik, UC Berkeley

Is this a solution?

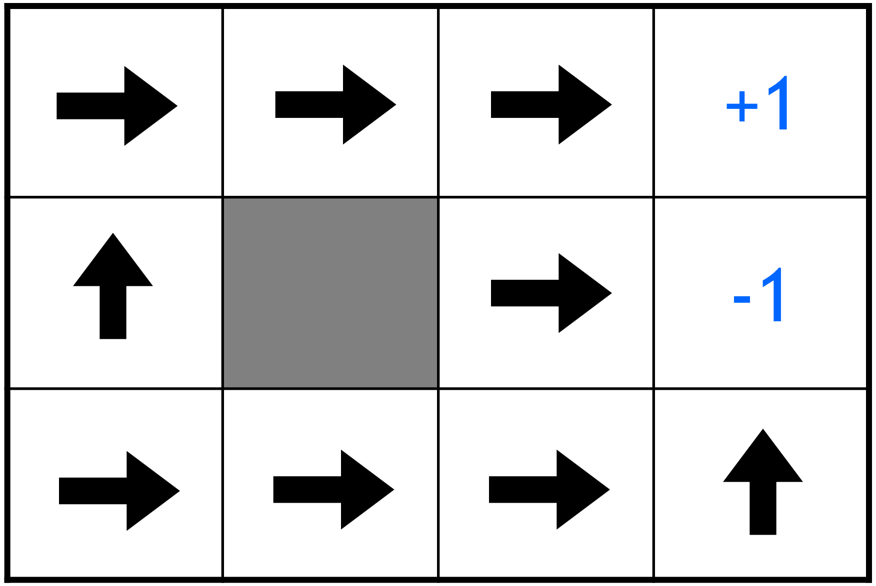


- only if actions deterministic
  - not in this case (actions are stochastic)

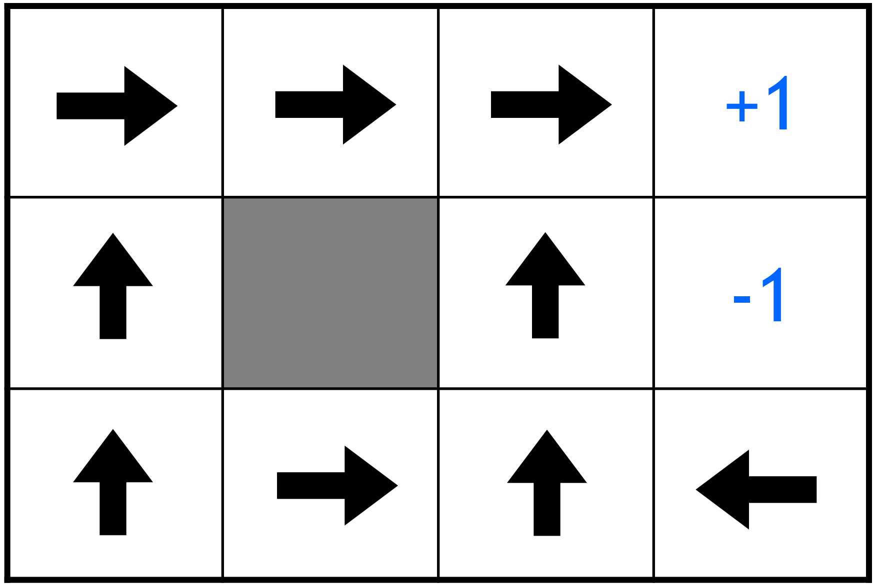
Optimal policy



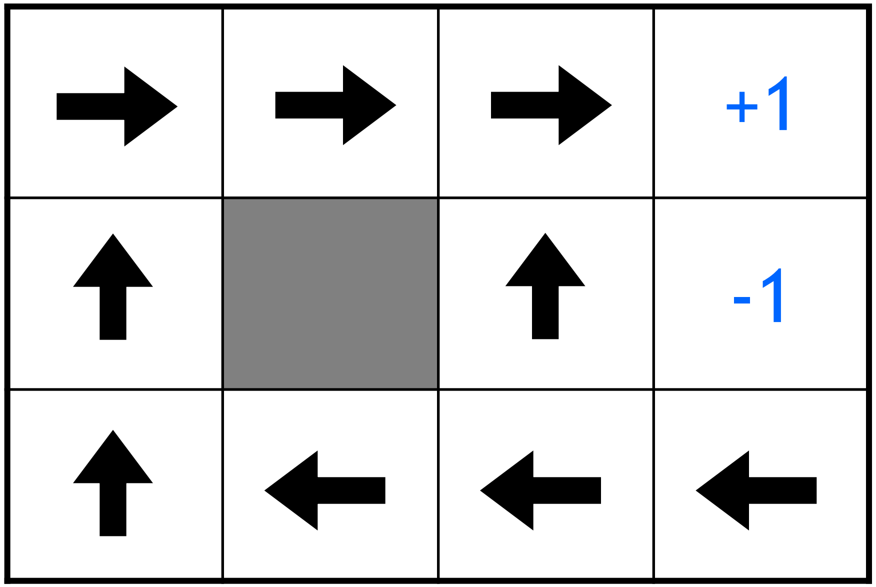
Reward for each step: -2



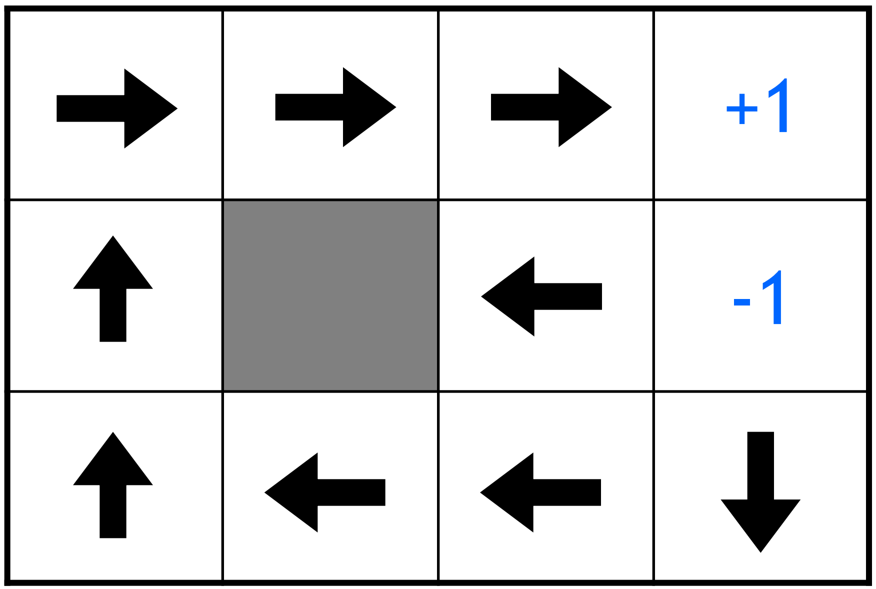
Reward for each step: -0.1



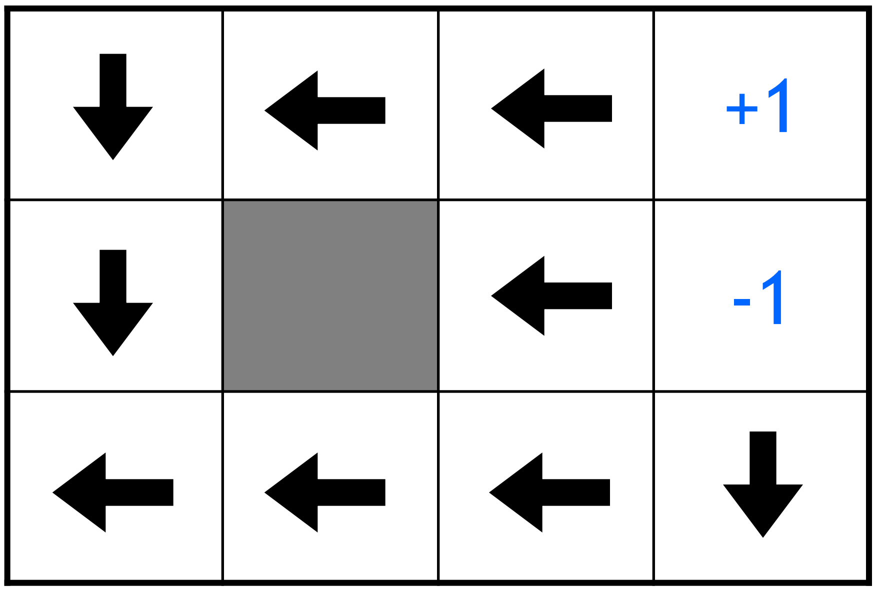
Reward for each step: -0.04



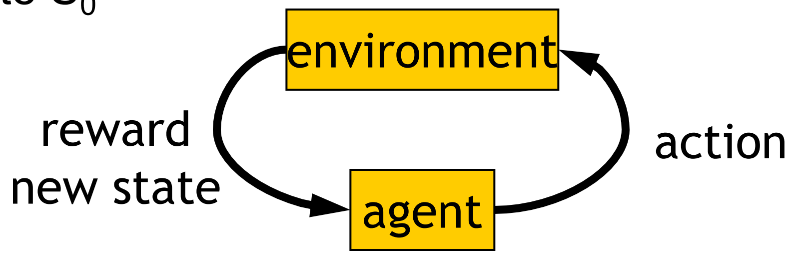
Reward for each step: -0.01



Reward for each step: +0.01



Markov Decision Process (MDP)



- set of states S, set of actions A, initial state S_0
- transition model P(s,a,s')
  - P([1,1], up, [1,2]) = 0.8
- reward function r(s)
  - r([4,3]) = +1
- **<u>goal:</u> maximize cumulative reward in the long run**
- policy: mapping from S to A
  - π(s) or π(s,a) (deterministic vs. stochastic)
- reinforcement learning
  - transitions and rewards usually not available
  - how to change the policy based on experience

Reward hypothesis

- Core assumption (reward hypothesis):

  All goals can be described by the maximization of the expected cumulative reward

- Do you agree with this?

Challenges

- Rewards may be delayed, or may seem correlated with useless actions
  - Credit assignment problem: which actions do we credit for +’ve (or –’ve) rewards?
- Entire world may not be observable --- can only see a small fraction of it
- Observations or actions may be noisy
- Actions may have long-term consequences
- Rewards may be sparse/ hard to find

Examples of RL problems

- Robotics
- Game playing (Go, Atari, Chess, etc.)
- Health care (learning treatment plans)
- Data centre cooling
- Managing investments
- Artificial general intelligence?

Reinforcement Learning

- Will present core RL ideas from David Silver's Deep RL Tutorial, ICML 2016
  - https://icml.cc/2016/tutorials/deep_rl_tutorial.pdf
  - Slides 13-50

RL: open problems

- RL isn't solved yet! There are several important problems that are under active investigation
  - Exploration
  - Model-based RL
  - Temporal abstraction
  - Multi-agent RL

Exploration

- In RL, as you move around, you update your policy and/or value function based on the rewards you get
- But, in the beginning, how do you decide to move around the environment? How do you know you've seen everything that's important?
- Most common exploration strategy in RL: **epsilon-greedy**
  - With probability epsilon, take a random action instead of the best one
- This obviously isn't very efficient!

Montezuma's Revenge

- Atari game with many rooms to explore, very hard for RL algorithms

Count-based exploration

- Better exploration strategy: 'count-based'
  - Learn to count how many times you've been in certain places, try to go where you've explored less so far
- https://www.youtube.com/watch?v=0yI2wJ6F8r0
- Explores much more than epsilon-greedy!
- But still not quite what we want (hard to scale to very large worlds, want to explore based on rewards)
- *How can we build agents that intelligently explore new environments?*

Model-based RL

- Previously discussed RL algorithms are model-free: they don't learn a model of the environment, $p(s'|s, a)$
- Intuitively, learning a model of the environment is extremely useful: allows you to predict what will happen in the future
- **Can reason about actions and their consequences before actually taking them!**
  - Much more data-efficient, don't need to try all possible actions
- How do you decide whether to take a certain action?

Model-based RL

- Humans use a mix of model-based () and model-free () RL
- Unfortunately, current model-based RL doesn't work very well
- Hard to learn exact model of the environment --- small differences with real environment accumulate!
  - Hard to deal with partial observability
- *Need to find a way of learning models that work well over longer time horizons*

Temporal abstraction

- Consider the actions taken to make a cup of coffee
  - Go to the kitchen, take out the beans
  - Grind the beans
  - Put the beans and water in the coffee machine, turn it on
  - Put cup under machine, receive coffee
- Involves a sequence of steps ('high-level actions'), each of which is composed of several 'low-level actions'

Temporal abstraction

- Regular RL just decides on a low-level action to perform at each time step
- **But if time steps are really short, this isn't very efficient!** Humans don't think of the next muscle to twitch every millisecond
- *Need a way to plan at multiple levels of abstraction*
- Some methods have been tried in RL (e.g. options), but still an open problem

Multi-agent RL

- One of the main forces pushing humans to be more intelligent is the presence of other humans
- Would it help to train RL agents in environments with many other agents
- Specifically, can use **self-play** (agent plays against itself)
- Idea: at beginning of training, agent is weak, but faces opponents who are also weak. At end of training, agent faces strong versions of itself
- Used to train strong agents to play Go, Chess, DotA

Learning from self-play in Dota 2

- https://blog.openai.com/dota-2/

Reinforcement learning: outlook

- Non-deep RL doesn't scale very well to high-dimensional problems
- Deep RL still very 'brittle' --- can be very sensitive to hyperparameters, initial conditions, etc.
  - See 'Deep RL doesn't work yet', blog post by Alex Irpan: https://www.alexirpan.com/2018/02/14/rl-hard.html
- Still many unsolved problems
- Requires a lot of compute power
- BUT very general framework that will likely be useful for

Next class

- The future of machine learning and AI
  - First half of class: cool new results in ML
  - Second half of class: safety/ ethical implications of ML. Will AI take over the world, or not? Will have time for discussion.

Other resources on RL

- Lecture series by David Silver
  - Videos: https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PL7-jPKtc4r78-wCZcQn5IqyuWhBZ8fOxT
  - Slides: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- Reinforcement Learning textbook
  - http://incompleteideas.net/book/bookdraft2017nov5.pdf