STOCHASTIC MODELS AND OPTIMIZATION

Gergely Neu
(Univ. Pompeu Fabra)
Hrvoje Stojic
(University College London / Prowler.io)

INTRODUCTION

Gergely Neu [ˈgɛrgɛj nɔɣ] / Gergo / Gergő

2003-2008: MSc in Electrical Engineering
Budapest Univ. of Technology and Economics

2008-2013: PhD in Computer Science
Budapest Univ. of Technology and Economics
(12 months at University of Alberta, Canada)

2013-2015: INRIA Lille, France

2015-: UPF DTIC

INTRODUCTION

Gergely Neu [ˈgɛrgɛj nɔɣ] / Gergo / Gergő

2003-2008: MSc in Electrical Engineering
Budapest Univ. of Technology and Economics

2008-2013: PhD in Computer Science
Budapest Univ. of Technology and Economics
(12 months at University of Alberta, Canada)

2013-2015: postdoc @ INRIA Lille, France

2015-: postdoc / asst. prof. / etc. @ UPF DTIC

Working in reinforcement learning since 2006

COURSE HISTORY

–2019

- focus on Dynamic Programming
- applications in management science
- by Mihalis Markakis

- focus on Reinforcement Learning
- from the broader (?) perspective of contemporary AI / Machine Learning

2020–

THE COURSE SINCE 2020

Two main parts:

Reinforcement Learning
in Markov decision processes
by Gergely

Bandit problems
by Hrvoje

MORE SPECIFICALLY…

1. Reinforcement learning preliminaries
2. Dynamic programming basics
3. Multi-armed bandits
4. Multi-armed bandits: extensions and applications
5. Contextual multi-armed bandits
6. Bayesian optimization and contextual bandits applications
7. Basic RL algorithms
8. Foundations of deep RL
9. Regularization in RL
10. Exploration vs. exploitation in RL

MORE SPECIFICALLY…

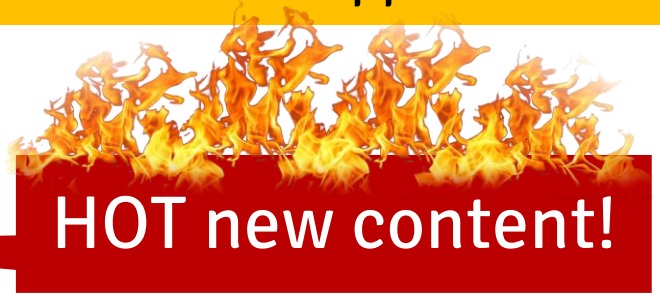
1. Reinforcement learning preliminaries
2. Dynamic programming basics
3. Multi-armed bandits
4. Multi-armed bandits: extensions and applications
5. Contextual multi-armed bandits
6. Bayesian optimization and contextual bandits applications
7. Basic RL algorithms
8. Foundations of deep RL
9. Regularization in RL
10. Exploration vs. exploitation in RL

by Hrvoje

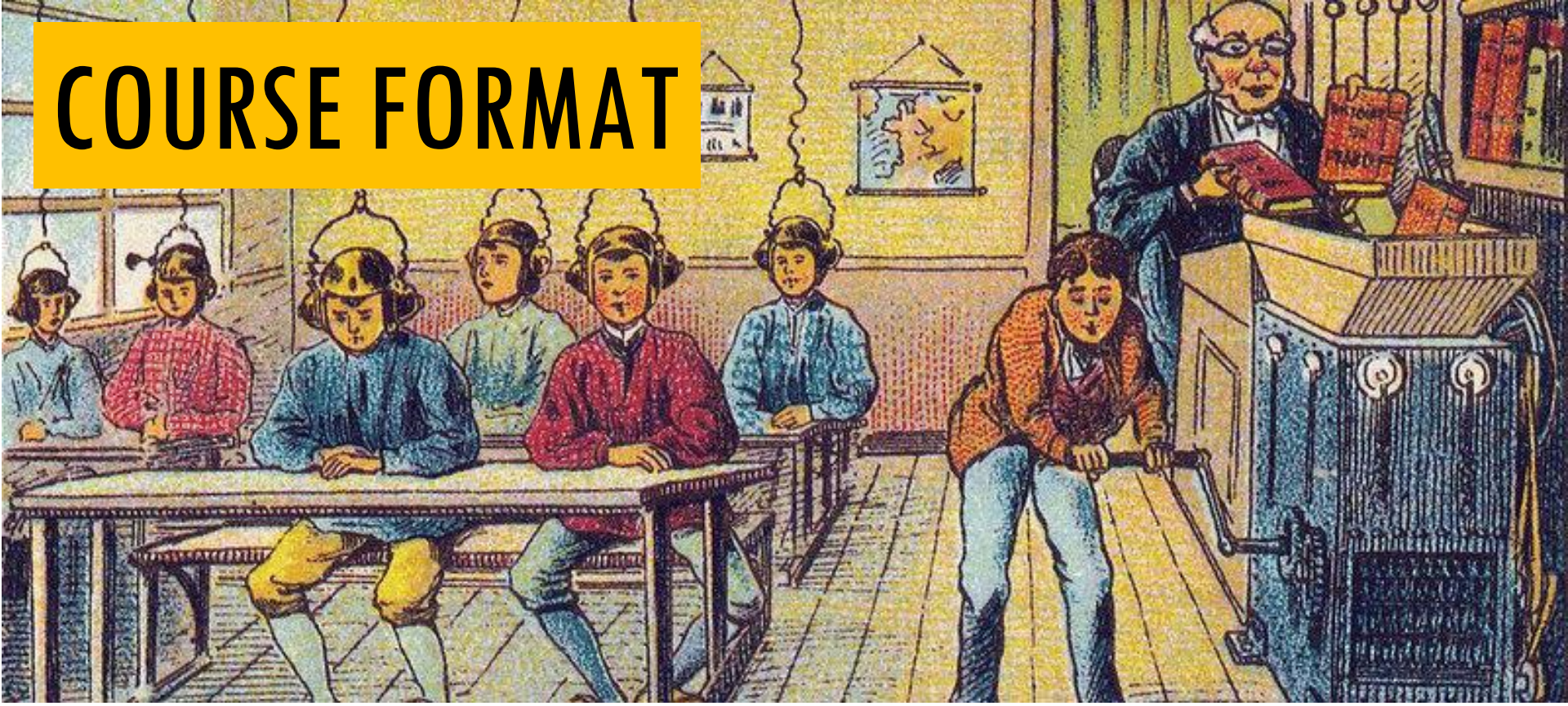
MORE SPECIFICALLY…

1. Reinforcement learning preliminaries
2. Dynamic programming basics
3. Multi-armed bandits
4. Multi-armed bandits: extensions and applications
5. Contextual multi-armed bandits
6. Bayesian optimization and contextual bandits applications
7. Basic RL algorithms
8. Foundations of deep RL
9. Regularization in RL
10. Exploration vs. exploitation in RL

by Hrvoje



COURSE FORMAT



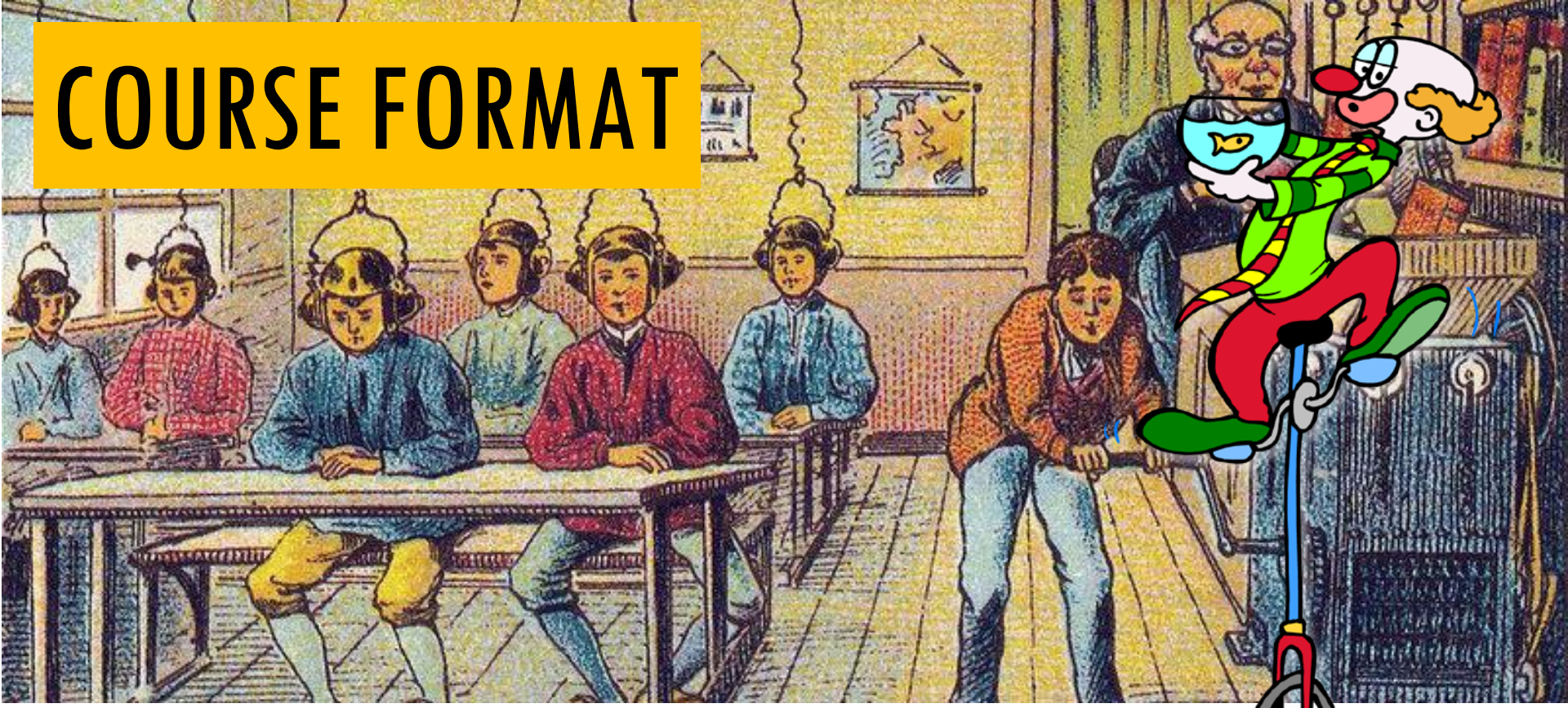
Face-to-face with Gergely (hopefully)

Online with Hrvoje

For further questions:

- gergely.neu@gmail.com
- hrvoje.stojic@protonmail.com

COURSE FORMAT



Face-to-face with Gergely (hopefully)

Online with Hrvoje

For further questions:

- gergely.neu@gmail.com
- hrvoje.stojic@protonmail.com

COURSE REQUIREMENTS

50% of score: problem sets

- individual work
- 3 from Gergely, 2 from Hrvoje

50% of score: final project

- programming (e.g., implementing an RL algorithm on a benchmark) or literature review
- in groups of at most 3 students

NECESSARY BACKGROUND

Mathematics:

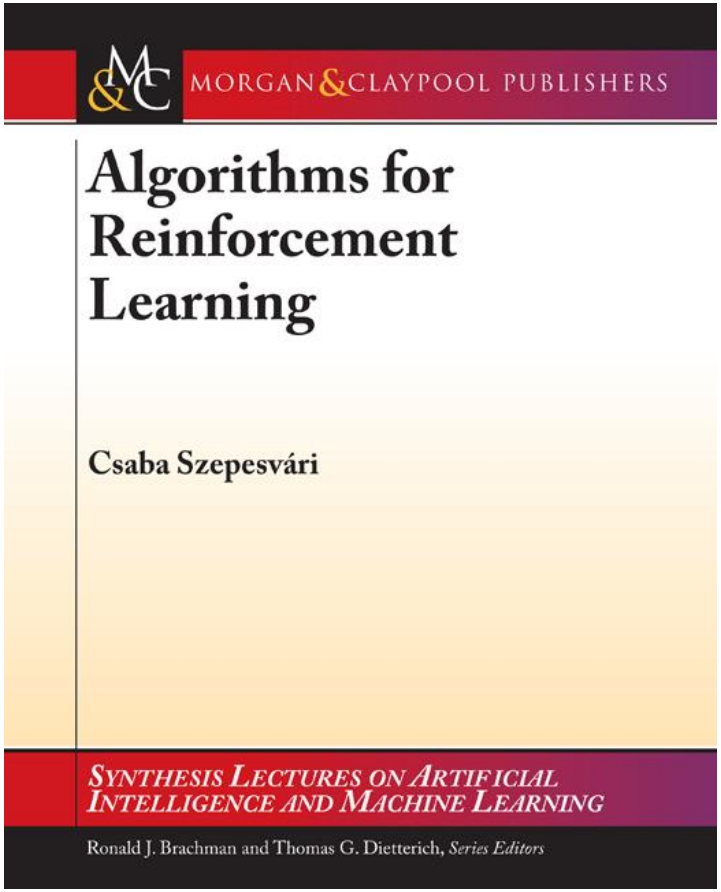
- Important: basic probability, linear algebra
- Useful: stochastic processes, multivariate calculus, constrained optimization…
- Advice: prepare for tons of definitions!

Coding:

- Problem sets: simple coding exercises
- Final project: more complex, but there's a lot of good material online

RECOMMENDED READING ON RL

- Richard Sutton and Andrew Barto (2018): "Reinforcement Learning: An Introduction (2nd edition)"
  - For an enjoyable (but not very rigorous) introduction
- Dimitri Bertsekas (2012): "Dynamic Programming and Optimal Control"
  - For a rigorous treatment of the basics
- Csaba Szepesvári (2012): "Algorithms for RL"
  - For a rigorous description of basic RL algorithms



WANT TO LEARN MORE?

Edit profile

RL Theory Virtual Seminars

@RLtheory

Virtual seminar series featuring the latest advances in theoretical reinforcement learning. Seminars (approximately) every Tuesday at 5pm UTC.

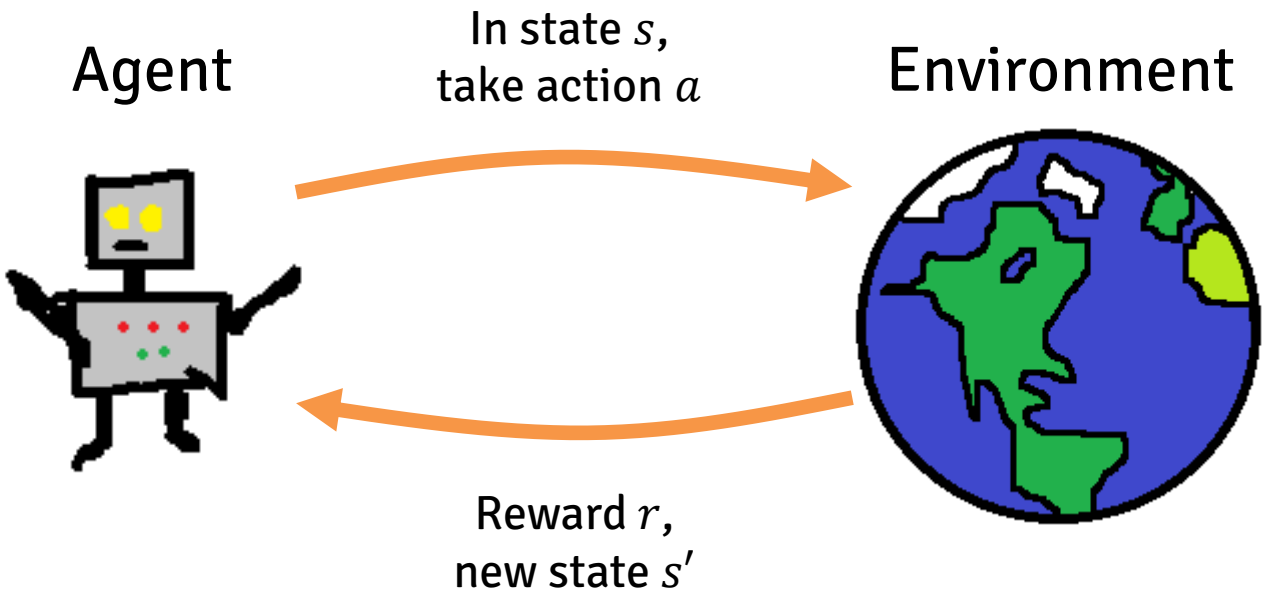
sites.google.com/view/rltheorys... Joined April 2020

0 Following **3,003** Followers

REINFORCEMENT LEARNING

Gergely Neu
Univ. Pompeu Fabra

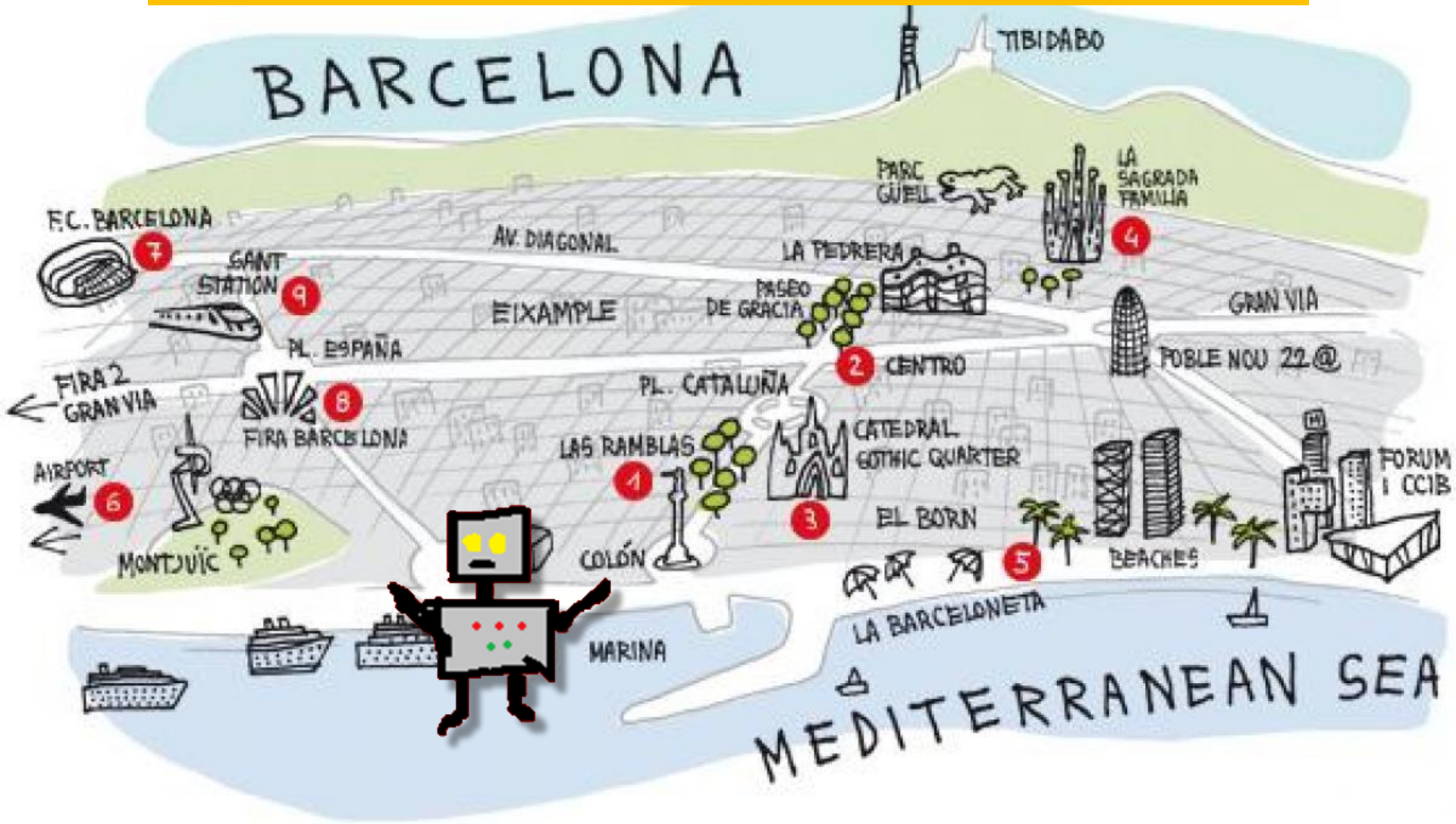
WHAT IS REINFORCEMENT LEARNING?



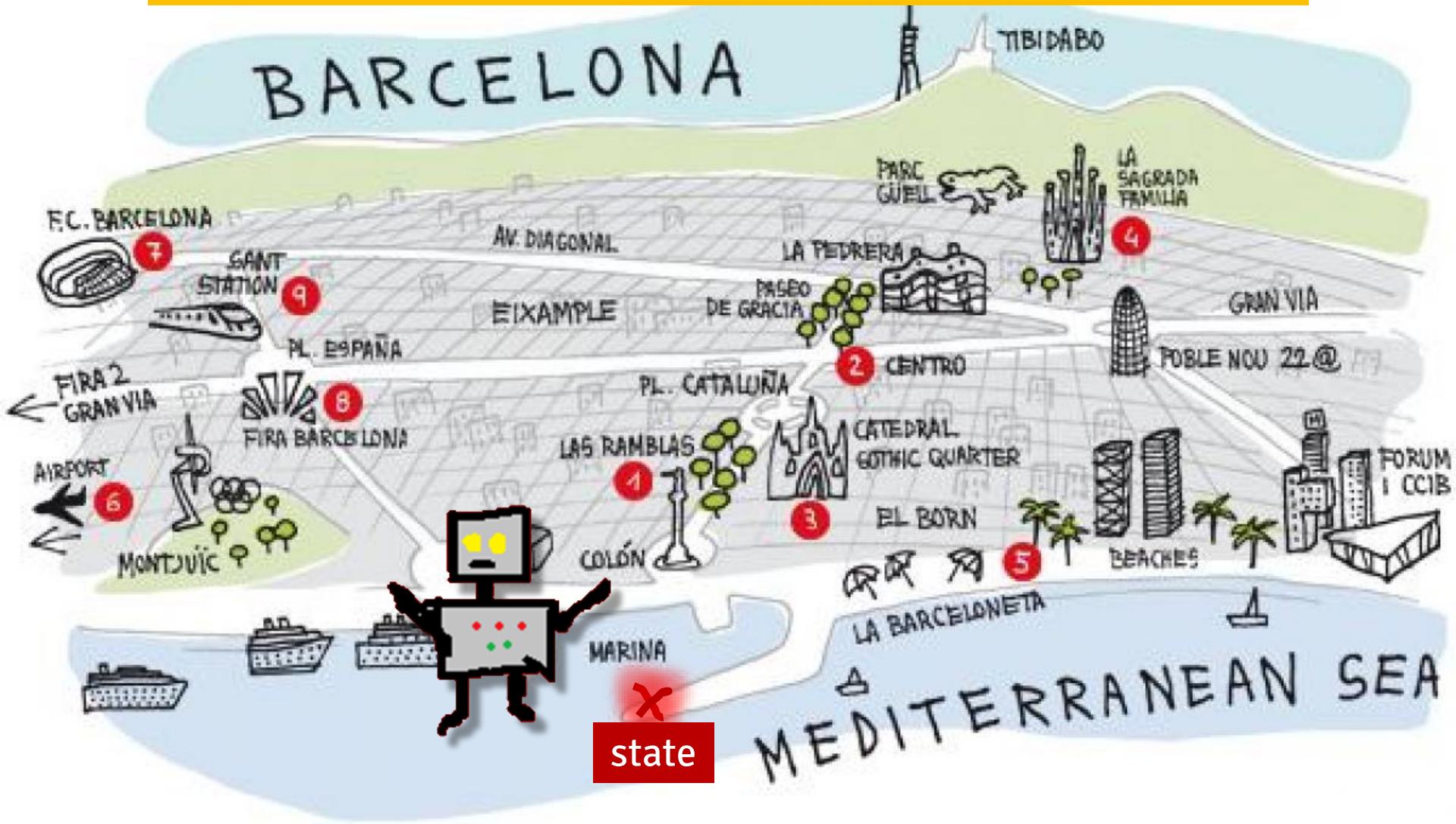
Learning to

- maximize reward
- in a reactive environment
- under partial feedback

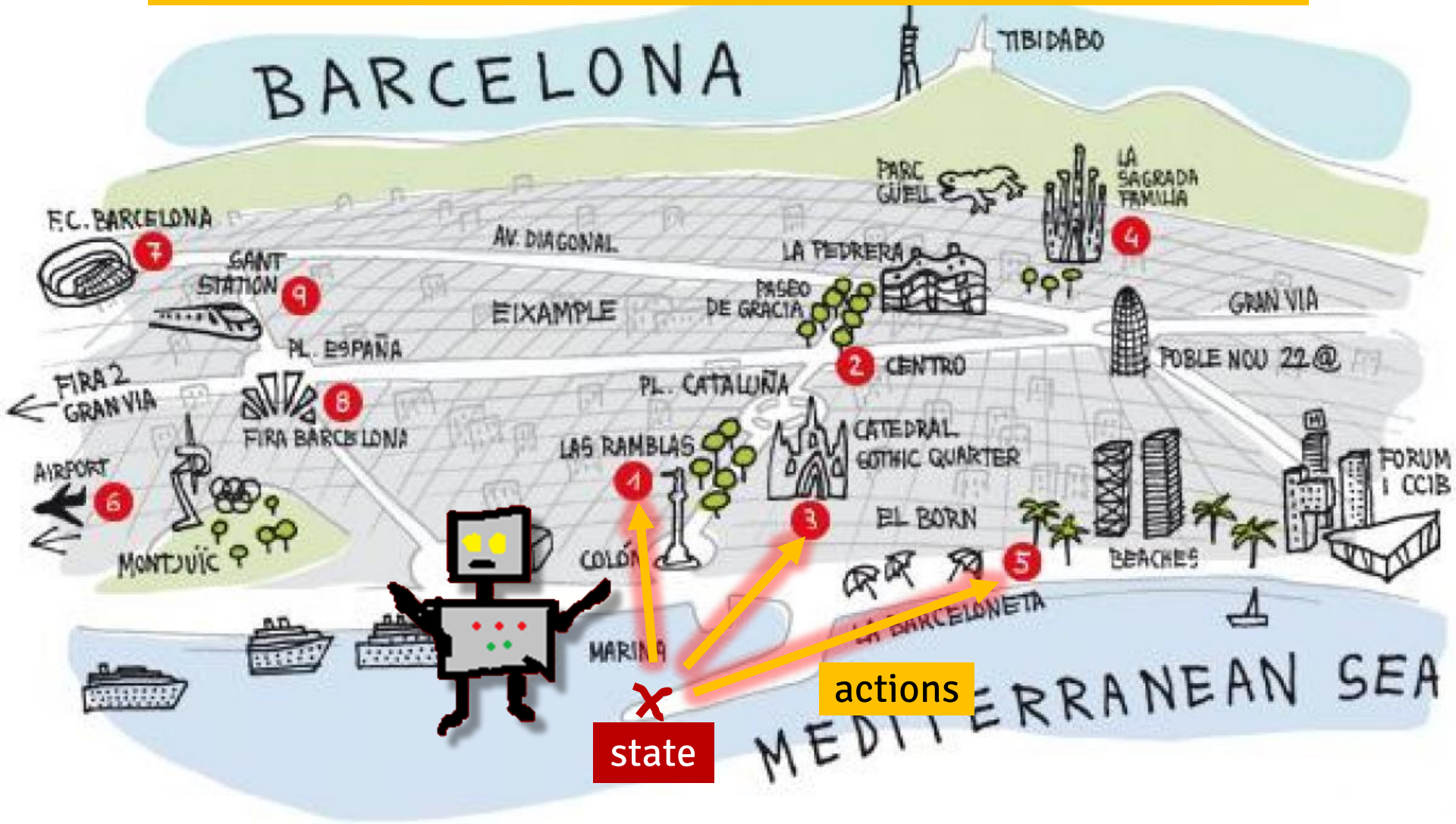
RL EXAMPLE 0.


BARCELONA
TIBIDABO
PARC GÜELL
LA SAGRADA FAMILIA
4
F.C. BARCELONA
7
AV. DIAGONAL
LA PEDRERA
SANT STATION
9
EIXAMPLE
PASEO DE GRACIA
GRAN VIA
PL. ESPAÑA
2
CENTRO
POBLE NOU 22@
FIRA 2 GRAN VIA
PL. CATALUÑA
8
FIRA BARCELONA
CATEDRAL GOTHIC QUARTER
LAS RAMBLAS
1
AIRPORT
6
3
EL BORN
FORUM I CCIB
MONTJUÏC
COLÓN
5
BEACHES
LA BARCELONETA
MARINA
MEDITERRANEAN SEA

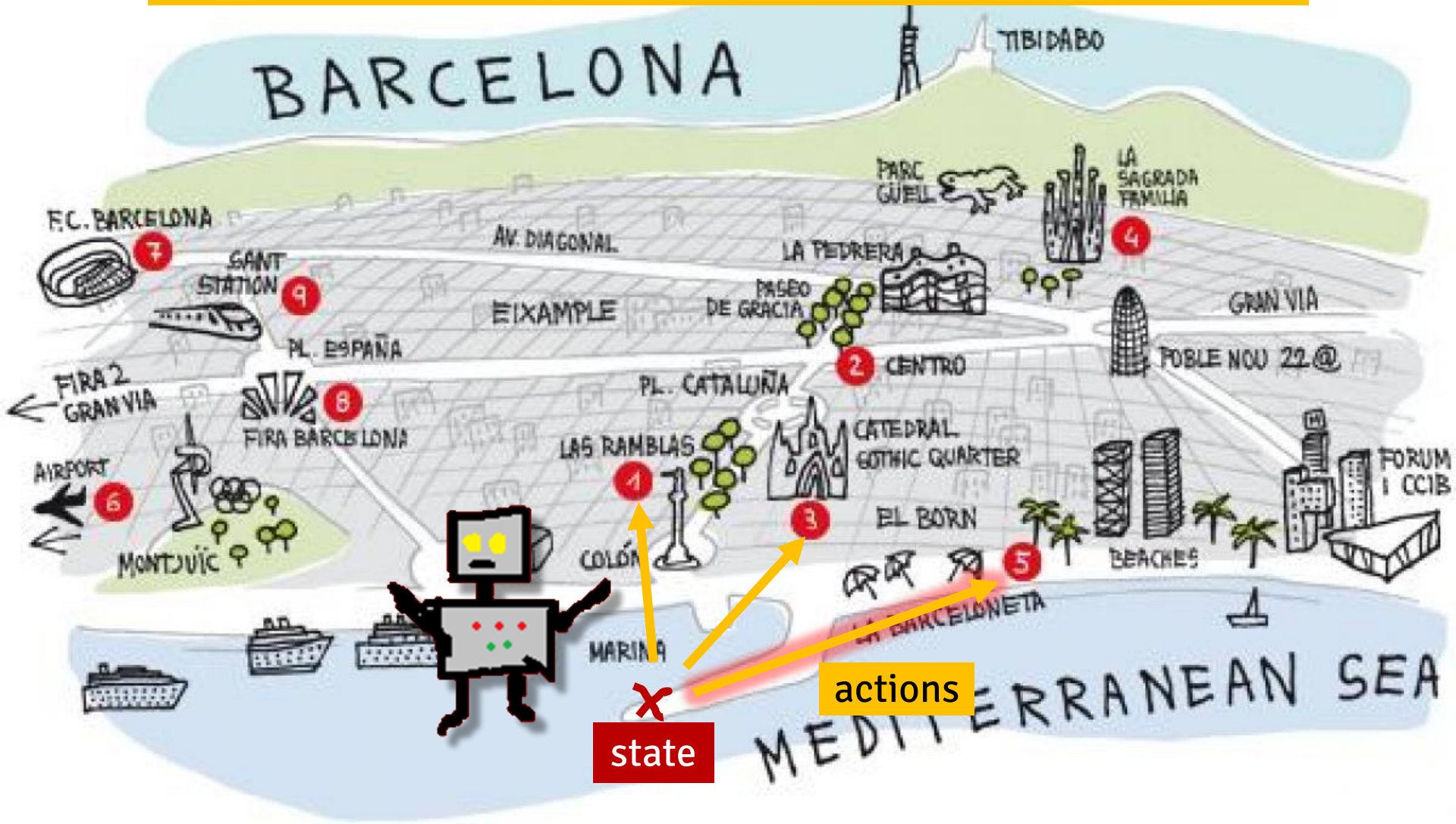
RL EXAMPLE 0.


BARCELONA
TIBIDABO
PARC GÜELL
LA SAGRADA FAMILIA
F.C. BARCELONA
7
AV. DIAGONAL
LA PEDRERA
SANT STATION
9
EIXAMPLE
PASEO DE GRACIA
4
GRAN VIA
PL. ESPAÑA
2
CENTRO
POBLE NOU 22@
FIRA 2 GRAN VIA
PL. CATALUÑA
8
FIRA BARCELONA
CATEDRAL GOTHIC QUARTER
LAS RAMBLAS
AIRPORT
1
3
EL BORN
FORUM I CCIB
6
5
BEACHES
MONTJUÏC
COLÓN
LA BARCELONETA
MARINA
MEDITERRANEAN SEA
state

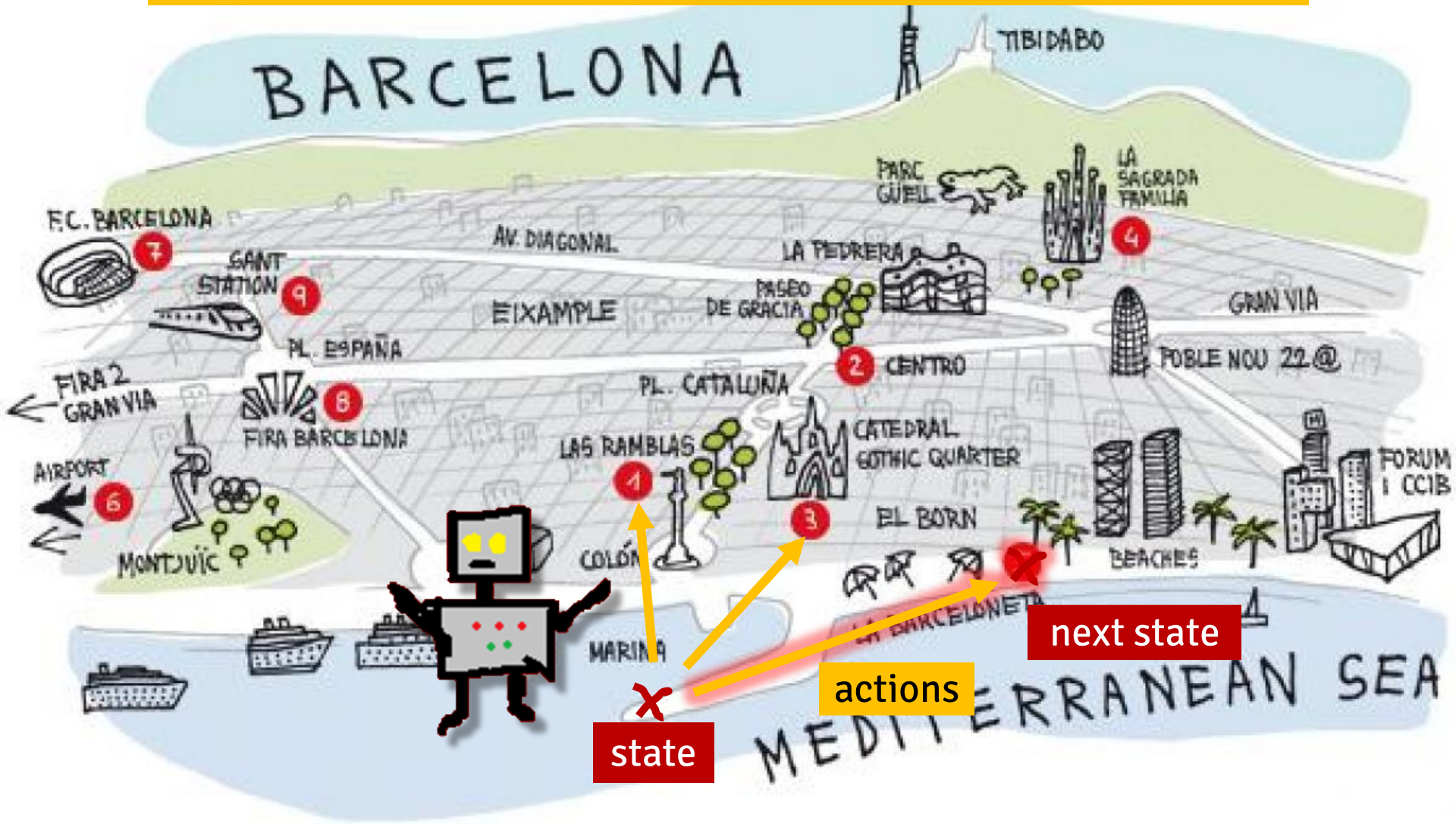
RL EXAMPLE 0.


BARCELONA
TIBIDABO
PARC GÜELL
LA SAGRADA FAMILIA
F.C. BARCELONA
AV. DIAGONAL
LA PEDRERA
SANT STATION
EIXAMPLE
PASEO DE GRACIA
GRAN VIA
PL. ESPAÑA
CENTRO
POBLE NOU 22@
FIRA 2 GRAN VIA
PL. CATALUÑA
FIRA BARCELONA
CATEDRAL GOTHIC QUARTER
LAS RAMBLAS
AIRPORT
EL BORN
FORUM I CCIB
MONTJUÏC
COLÓN
BEACHES
MARINA
LA BARCELONETA
MEDITERRANEAN SEA
actions
state

RL EXAMPLE 0.


BARCELONA
TIBIDABO
PARC GÜELL
LA SAGRADA FAMILIA
F.C. BARCELONA
AV. DIAGONAL
LA PEDRERA
SANT STATION
EIXAMPLE
PASEO DE GRACIA
GRAN VIA
PL. ESPAÑA
CENTRO
POBLE NOU 22@
FIRA 2 GRAN VIA
PL. CATALUÑA
FIRA BARCELONA
CATEDRAL GOTHIC QUARTER
LAS RAMBLAS
AIRPORT
EL BORN
FORUM I CCIB
MONTJUÏC
COLÓN
BEACHES
MARINA
actions
state
MEDITERRANEAN SEA

RL EXAMPLE 0.

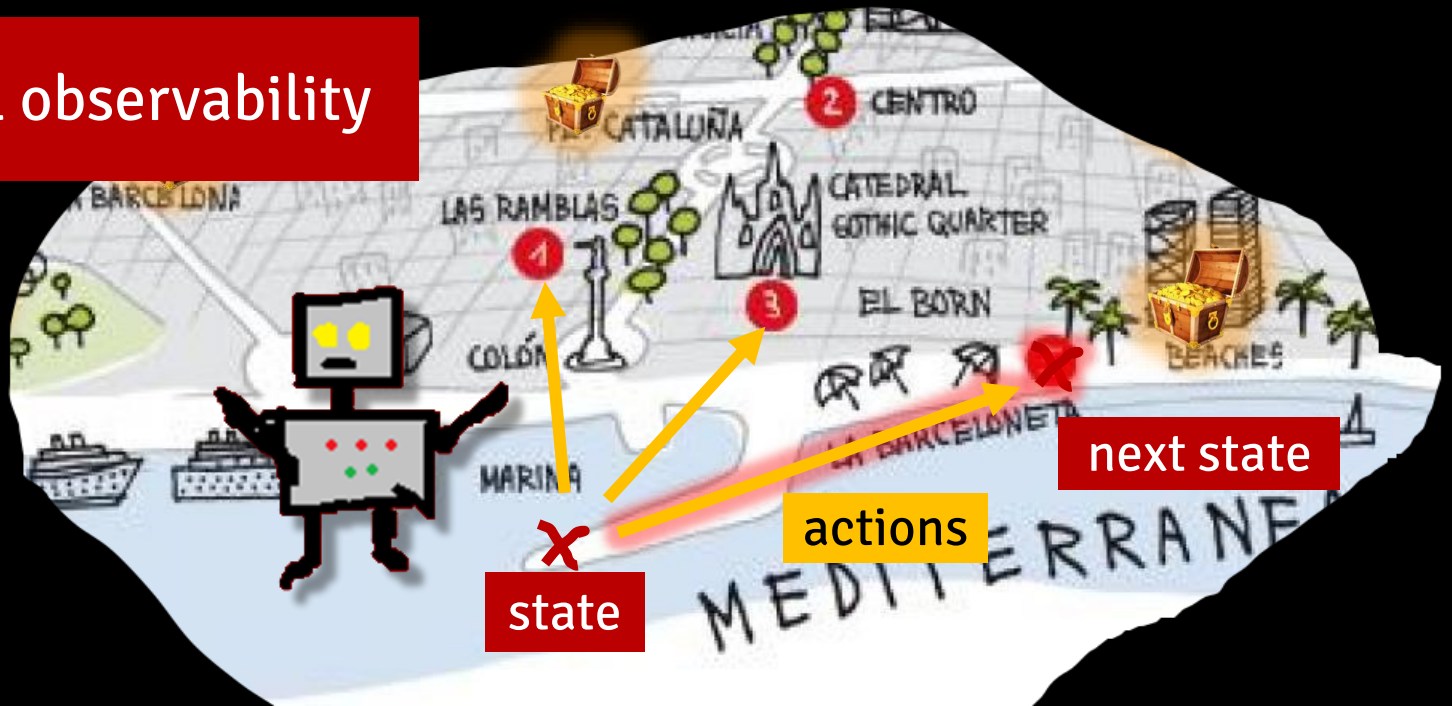

BARCELONA
TIBIDABO
PARC GÜELL
LA SAGRADA FAMILIA
F.C. BARCELONA
AV. DIAGONAL
LA PEDRERA
SANT STATION
EIXAMPLE
PASEO DE GRACIA
GRAN VIA
PL. ESPAÑA
CENTRO
POBLE NOU 22@
FIRA 2 GRAN VIA
PL. CATALUÑA
FIRA BARCELONA
CATEDRAL GOTHIC QUARTER
LAS RAMBLAS
AIRPORT
EL BORN
FORUM I CCIB
MONTJUÏC
COLÓN
BEACHES
MARINA
next state
actions
state
MEDITERRANEAN SEA

RL EXAMPLE 0.


BARCELONA
TIBIDABO
reward
F.C. BARCELONA
SANT STATION
AV. DIAGONAL
LA PEDRERA
SAGRADA FAMILIA
EIXAMPLE
PASEO DE GRACIA
GRAN VIA
PL. ESPAÑA
CENTRO
FIRA 2 GRAN VIA
CATALUÑA
POBLE NOU 22@
FIRA BARCELONA
AIRPORT
LAS RAMBLAS
CATEDRAL GOTHIC QUARTER
FORUM I CCIB
EL BORN
COLÓN
BEACHES
MARINA
next state
actions
state
MEDITERRANEAN SEA

RL EXAMPLE 0.

partial observability



RL EXAMPLE 0.

reward?

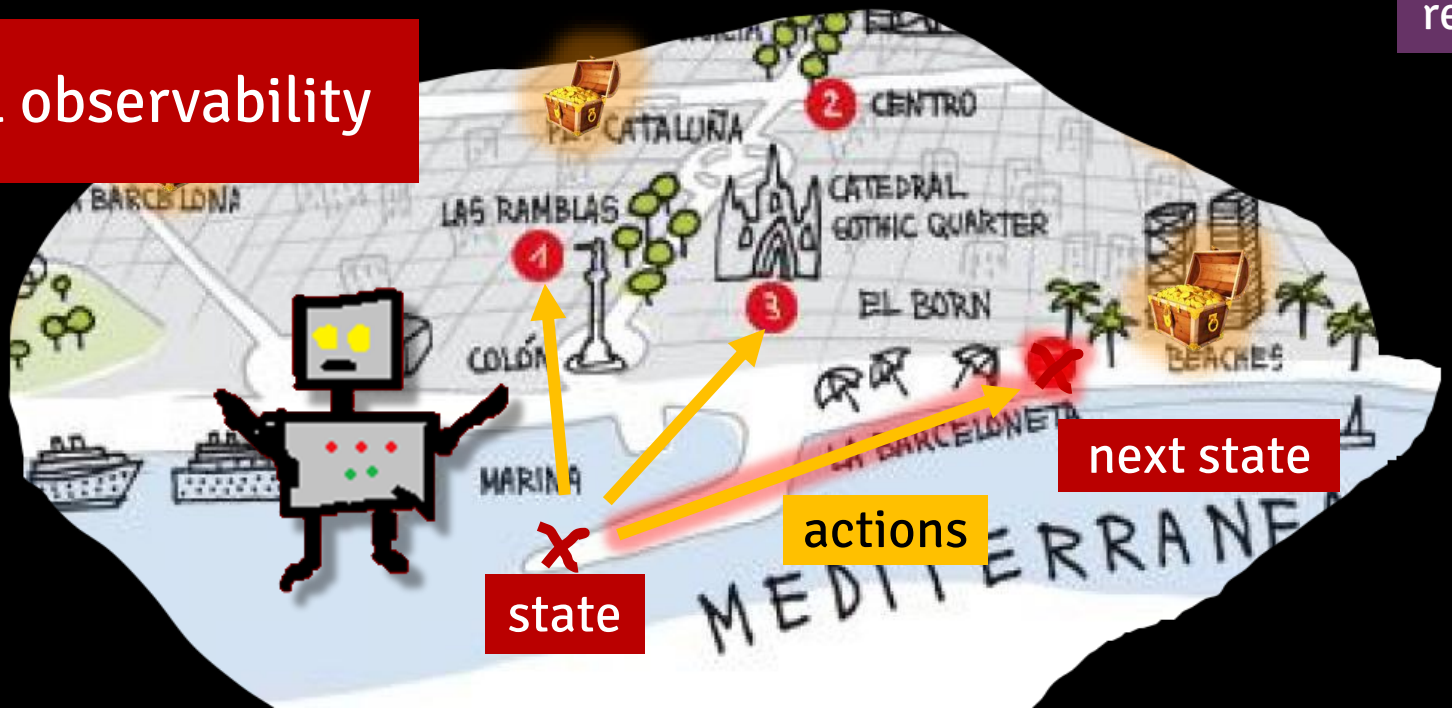
reward?

reward?

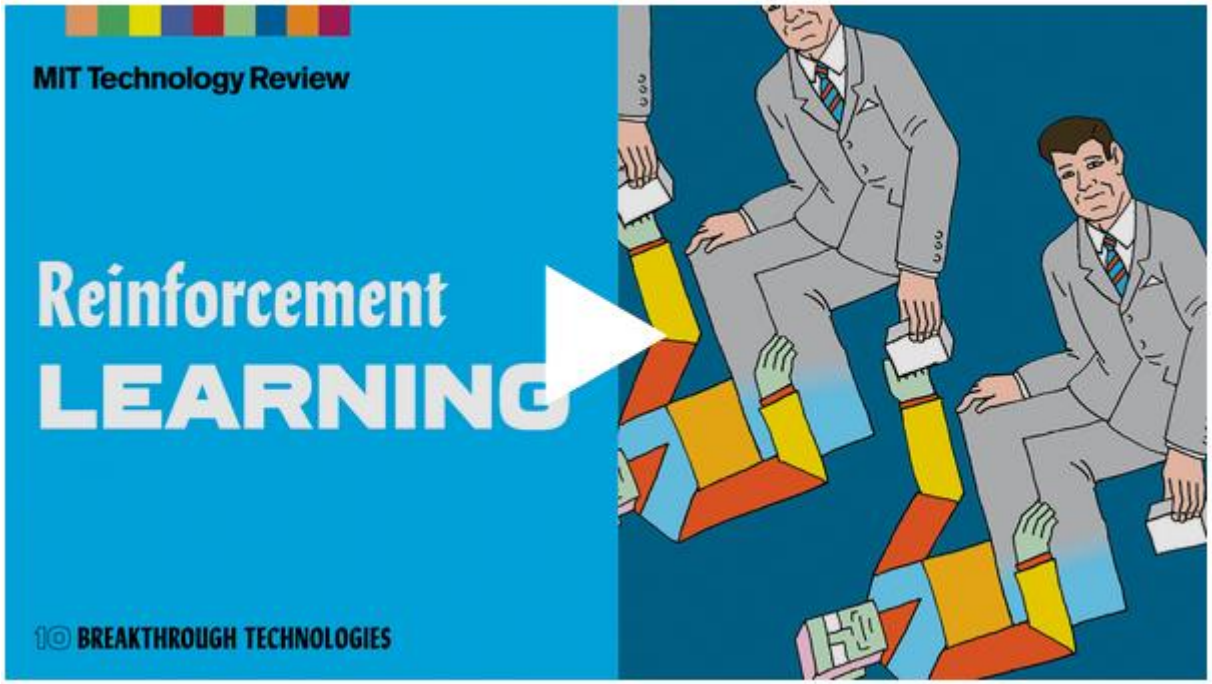
reward?

reward?

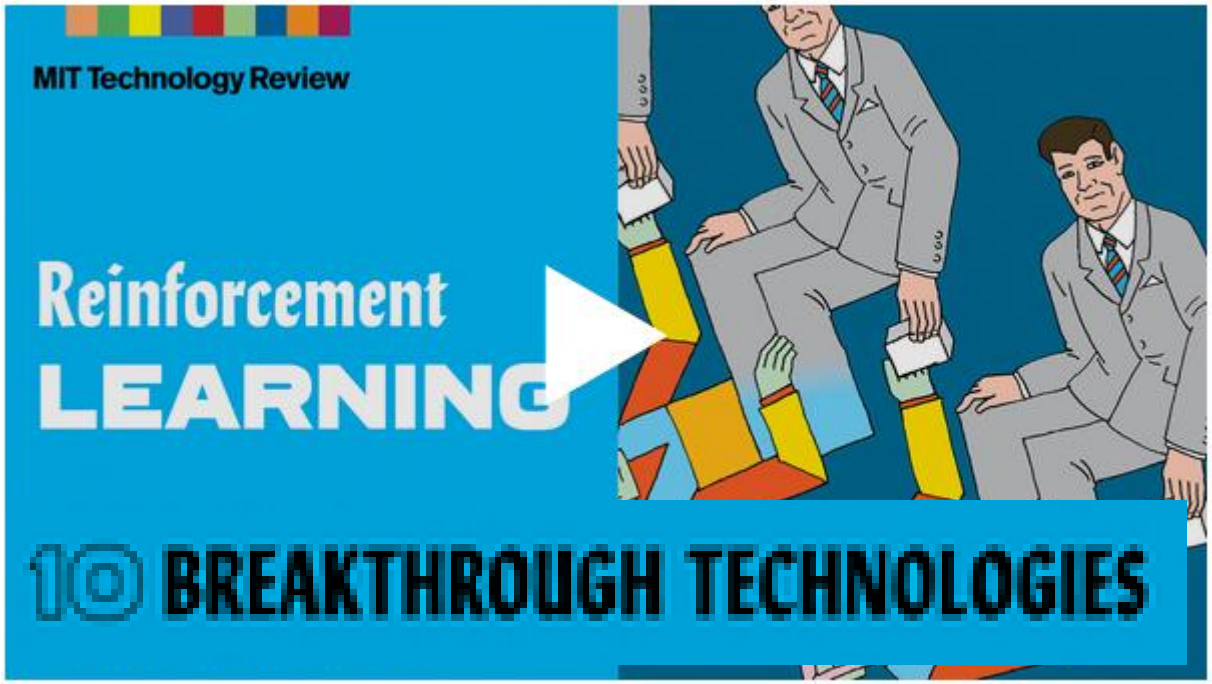
partial observability



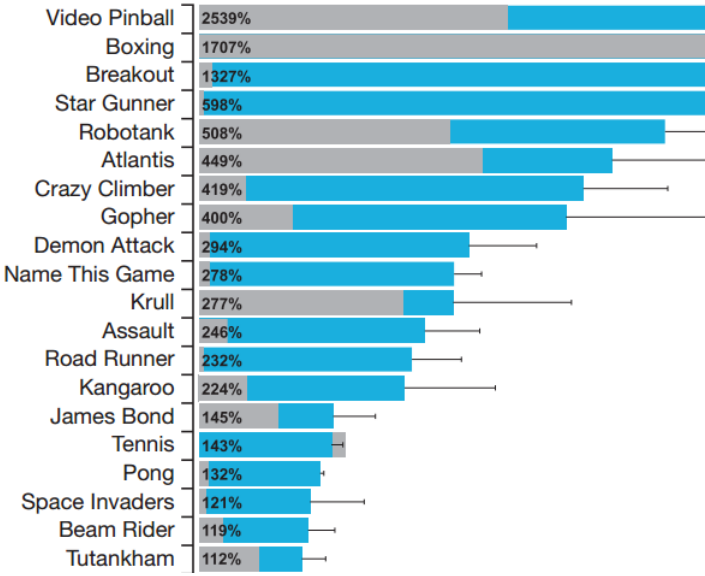
WHY SHOULD I CARE?



WHY SHOULD I CARE?



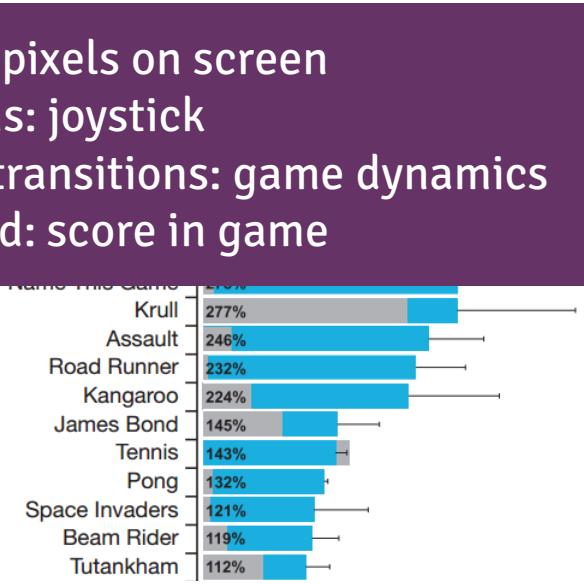
WHY SHOULD I CARE?



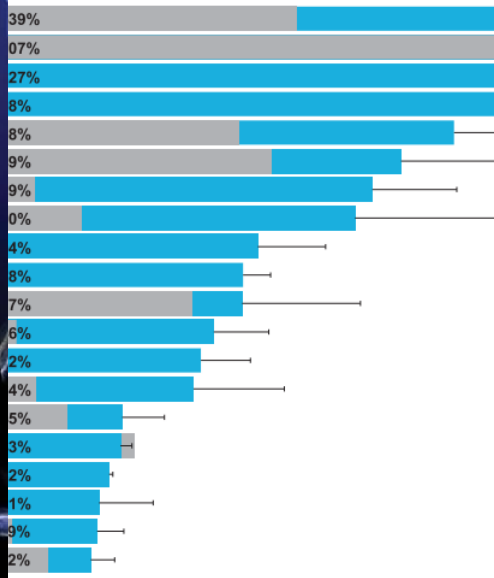
WHY SHOULD I CARE?

- State: pixels on screen
- Actions: joystick
- State transitions: game dynamics
- Reward: score in game

Breakthrough in Atari game playing

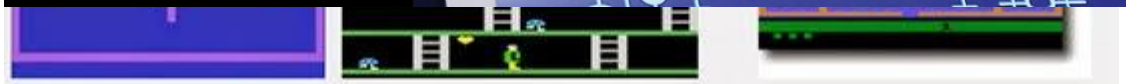


WHY SHOULD I CARE?



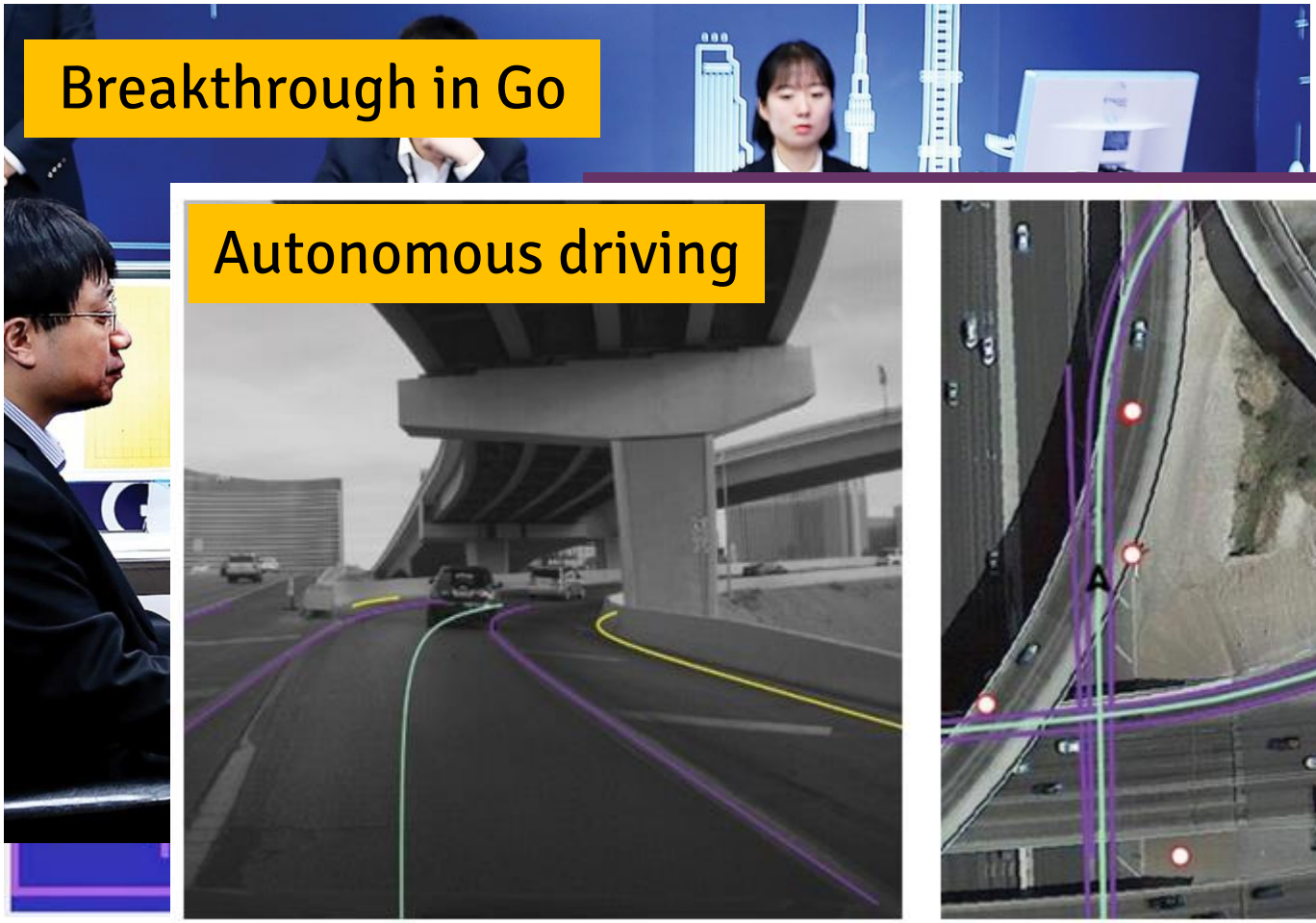
WHY SHOULD I CARE?

- State: stones currently on board
- Actions: place stone on board
- State transitions: own move + adversary's move
- Reward: +1 for winning the game



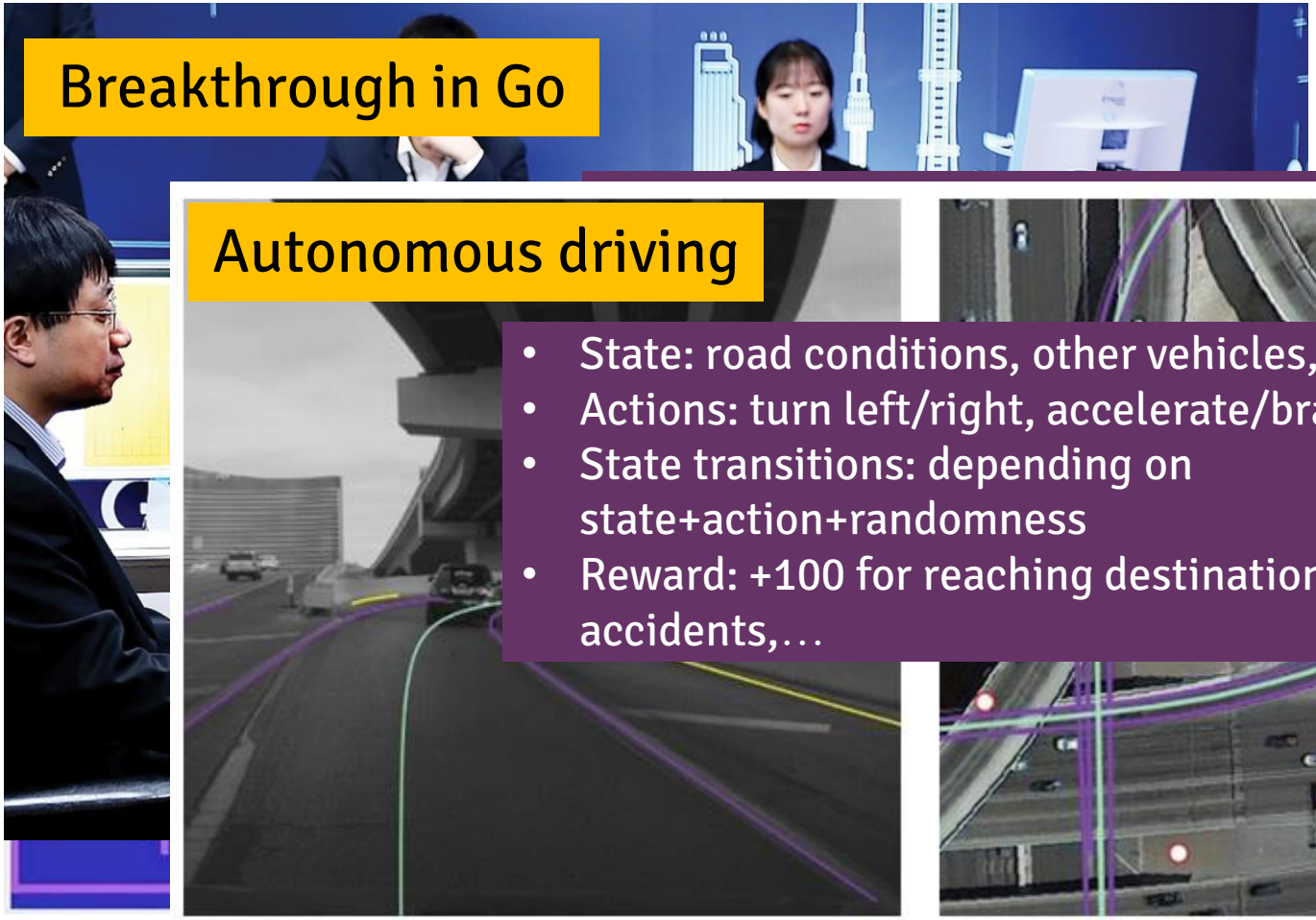
WHY SHOULD I CARE?

Breakthrough in Go



Autonomous driving

WHY SHOULD I CARE?

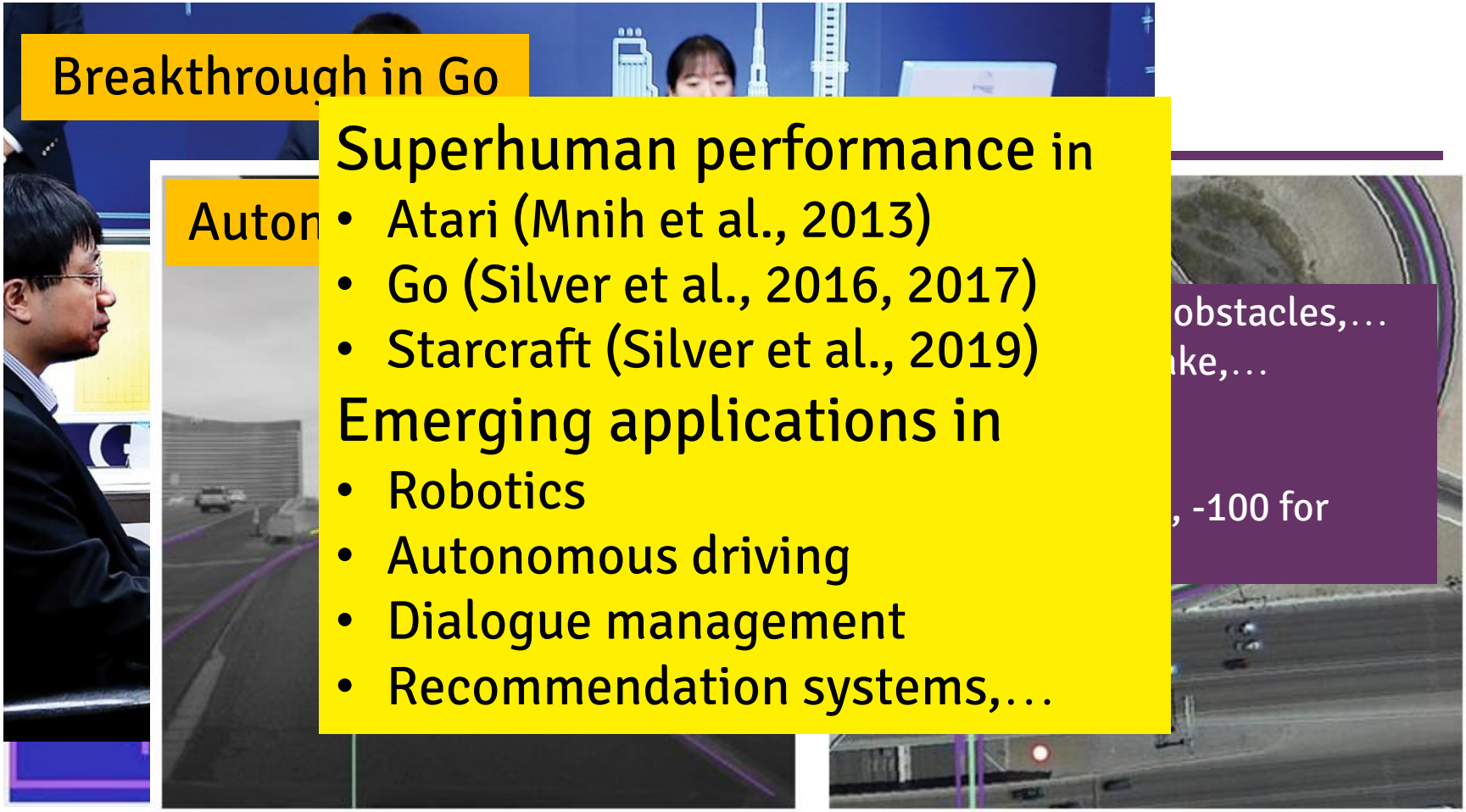


Breakthrough in Go

Autonomous driving

- State: road conditions, other vehicles, obstacles,…
- Actions: turn left/right, accelerate/brake,…
- State transitions: depending on state+action+randomness
- Reward: +100 for reaching destination, -100 for accidents,…

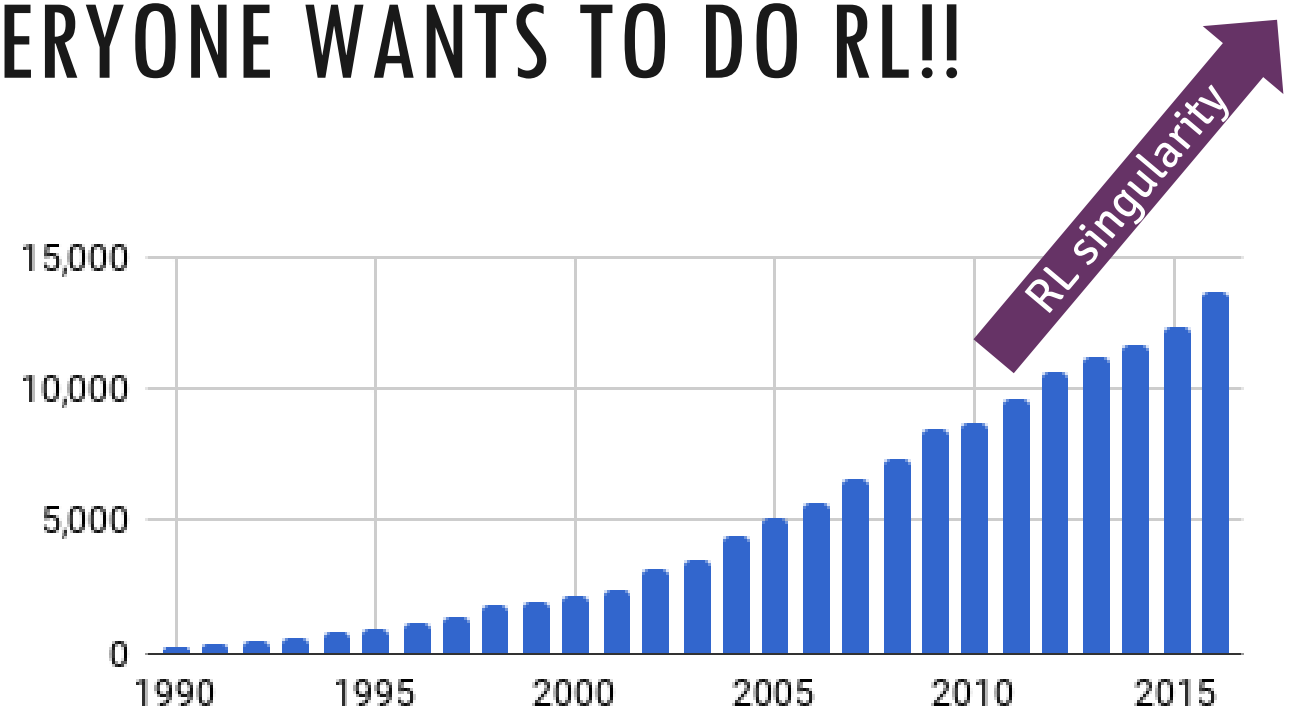
WHY SHOULD I CARE?



Superhuman performance in
- Atari (Mnih et al., 2013)
- Go (Silver et al., 2016, 2017)
- Starcraft (Silver et al., 2019)

Emerging applications in
- Robotics
- Autonomous driving
- Dialogue management
- Recommendation systems,…

EVERYONE WANTS TO DO RL!!



# of RL papers / year

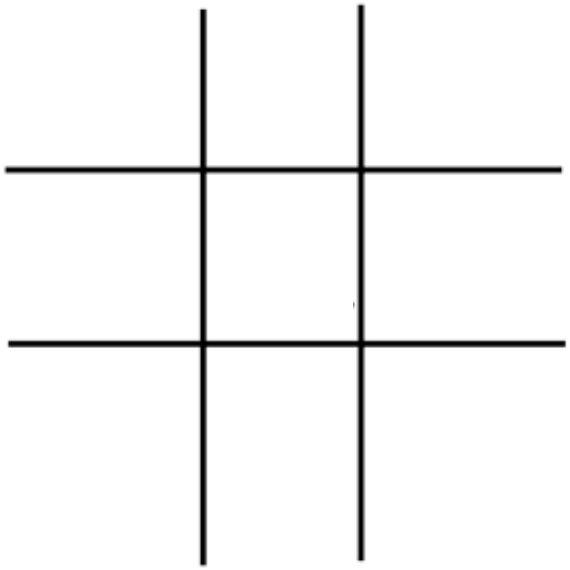
Facebook AI Research

DeepMind

OpenAI

Microsoft Research

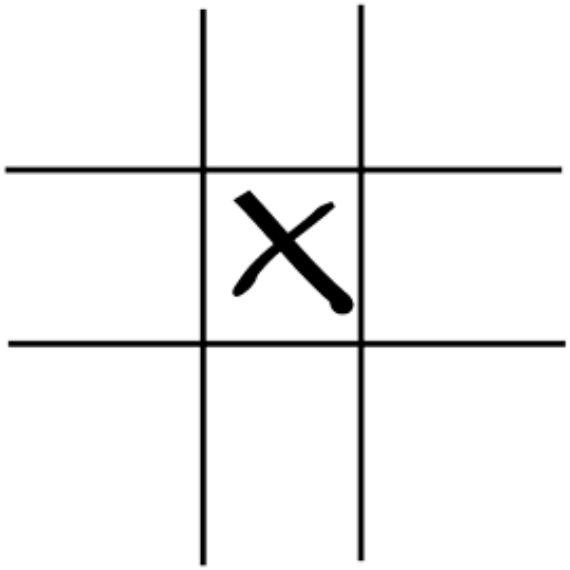
A CONCRETE EXAMPLE: TIC-TAC-TOE



In a sequence of games

- observe state of the board

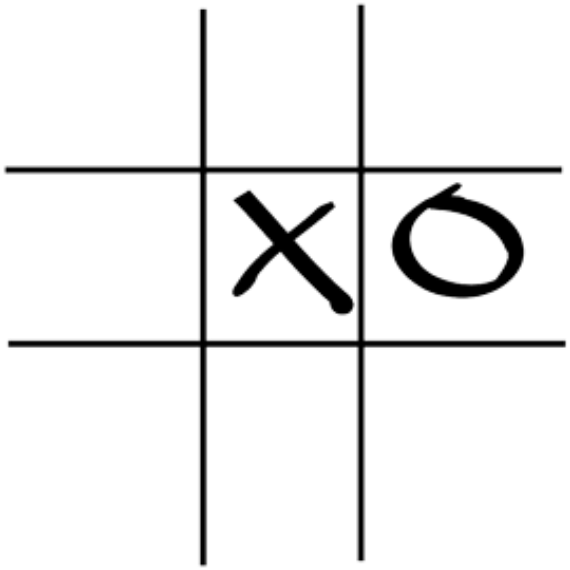
A CONCRETE EXAMPLE: TIC-TAC-TOE



In a sequence of games

- observe state of the board
- take action

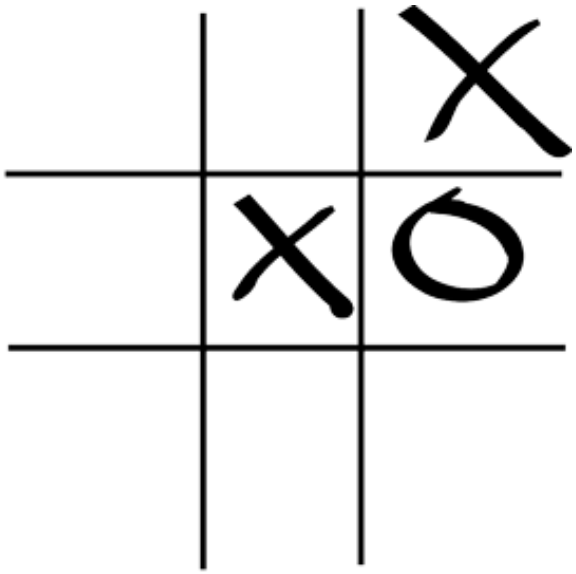
A CONCRETE EXAMPLE: TIC-TAC-TOE



In a sequence of games

- observe state of the board
- take action
- observe opponent's move

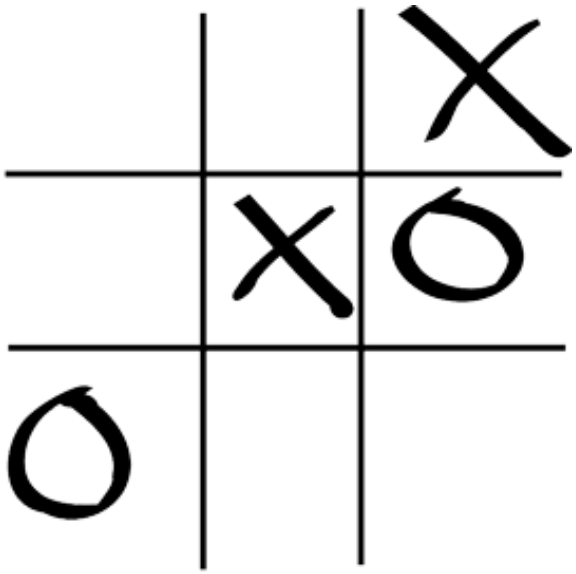
A CONCRETE EXAMPLE: TIC-TAC-TOE



In a sequence of games

- observe state of the board
- take action
- observe opponent's move
- … repeat…

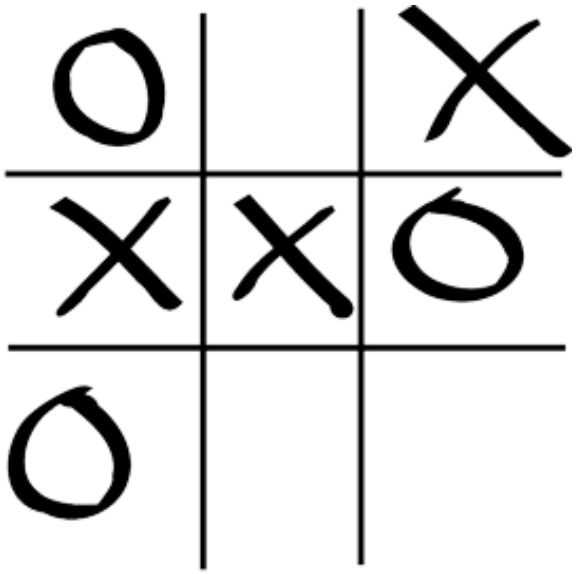
A CONCRETE EXAMPLE: TIC-TAC-TOE



In a sequence of games

- observe state of the board
- take action
- observe opponent's move
- … repeat…

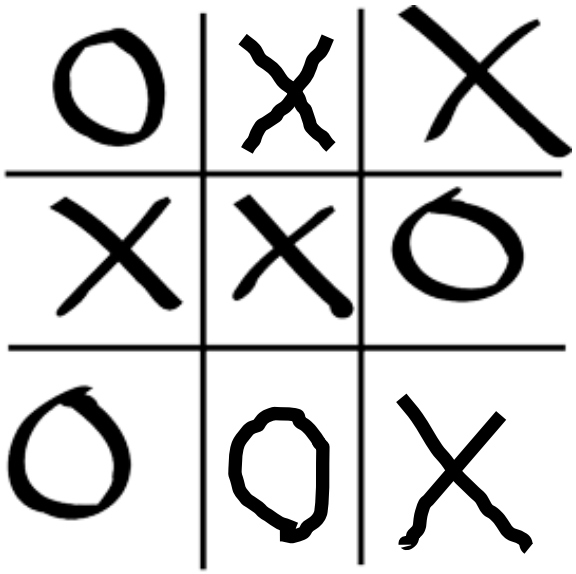
A CONCRETE EXAMPLE: TIC-TAC-TOE



In a sequence of games

- observe state of the board
- take action
- observe opponent's move
- … repeat…

A CONCRETE EXAMPLE: TIC-TAC-TOE

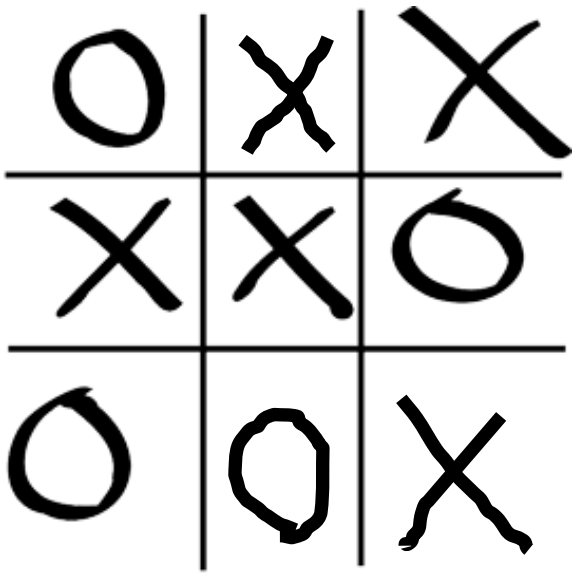


In a sequence of games

- observe state of the board
- take action
- observe opponent's move
- … repeat…
- until game is over and obtain reward

$$R = \begin{cases} +1 \text{ if win} \\ -1 \text{ if lose} \\ 0 \text{ if draw} \end{cases}$$

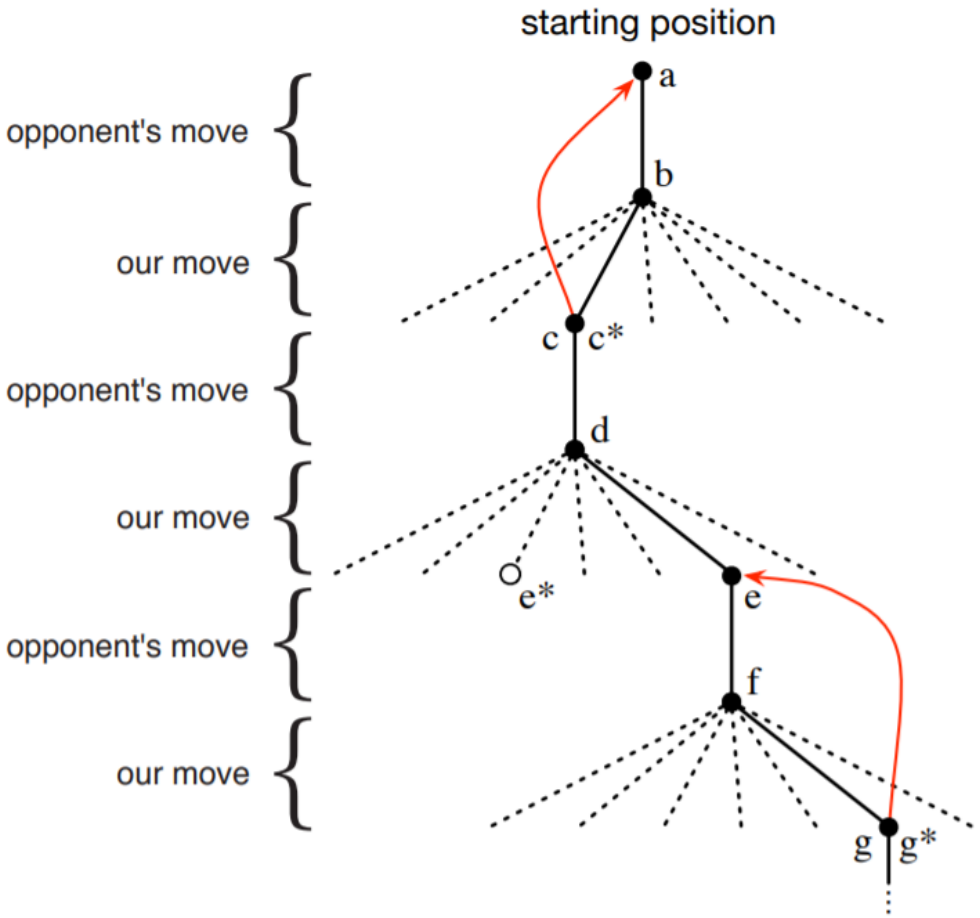
A CONCRETE EXAMPLE: TIC-TAC-TOE



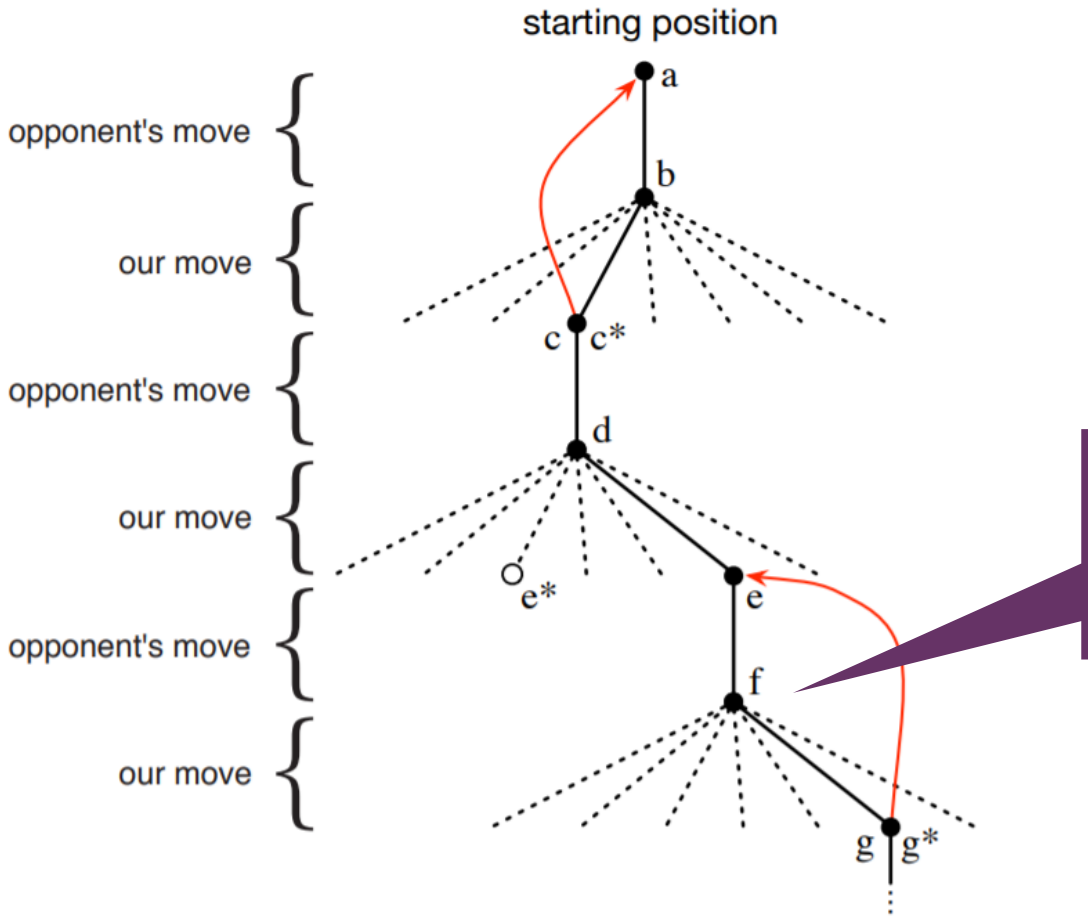
Two challenges:

1. each action influences the future evolution of the game
2. we never find out what would have happened if we took a different action

TIC-TAC-TOE DISSECTED

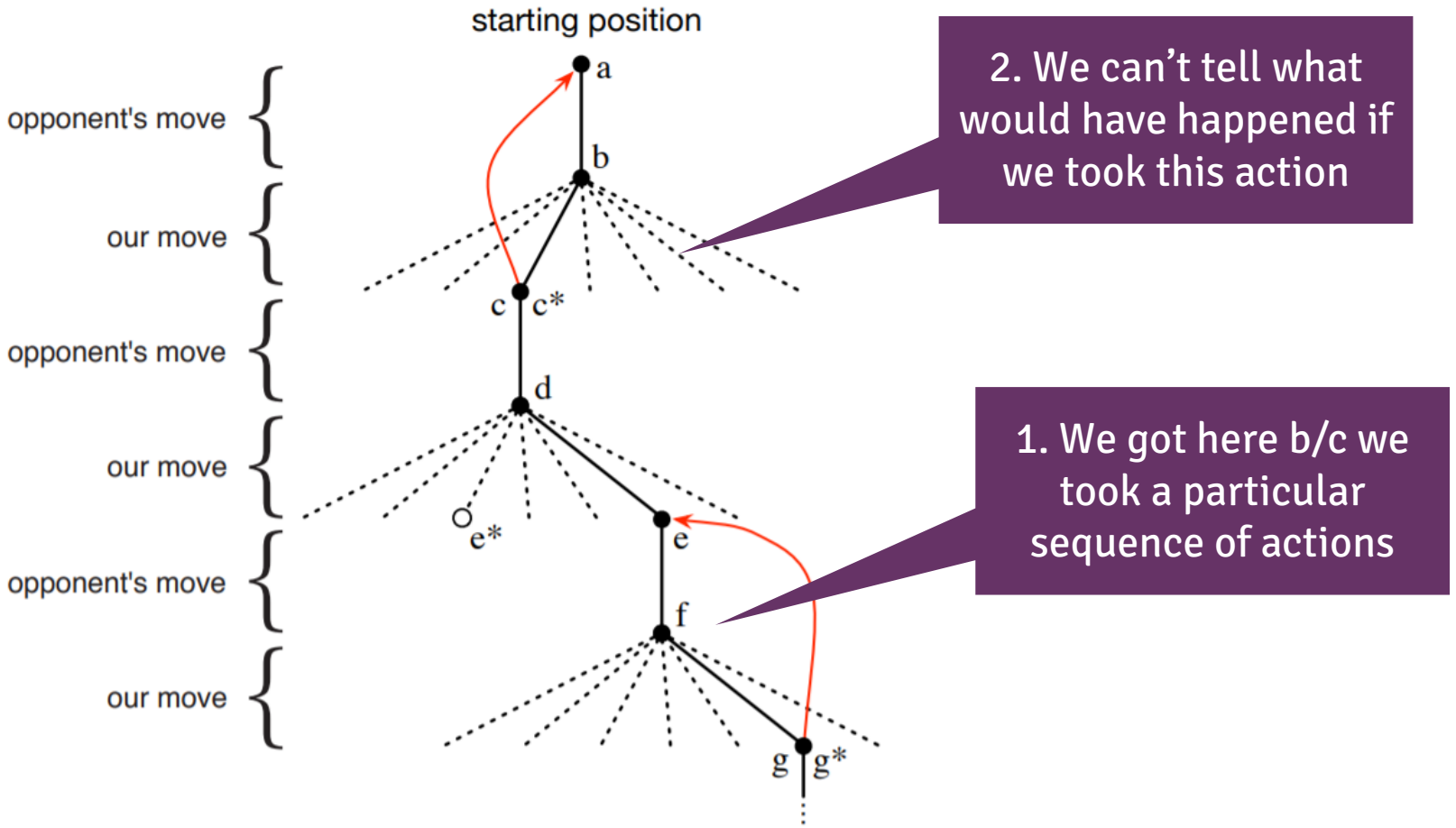


TIC-TAC-TOE DISSECTED



1. We got here b/c we took a particular sequence of actions

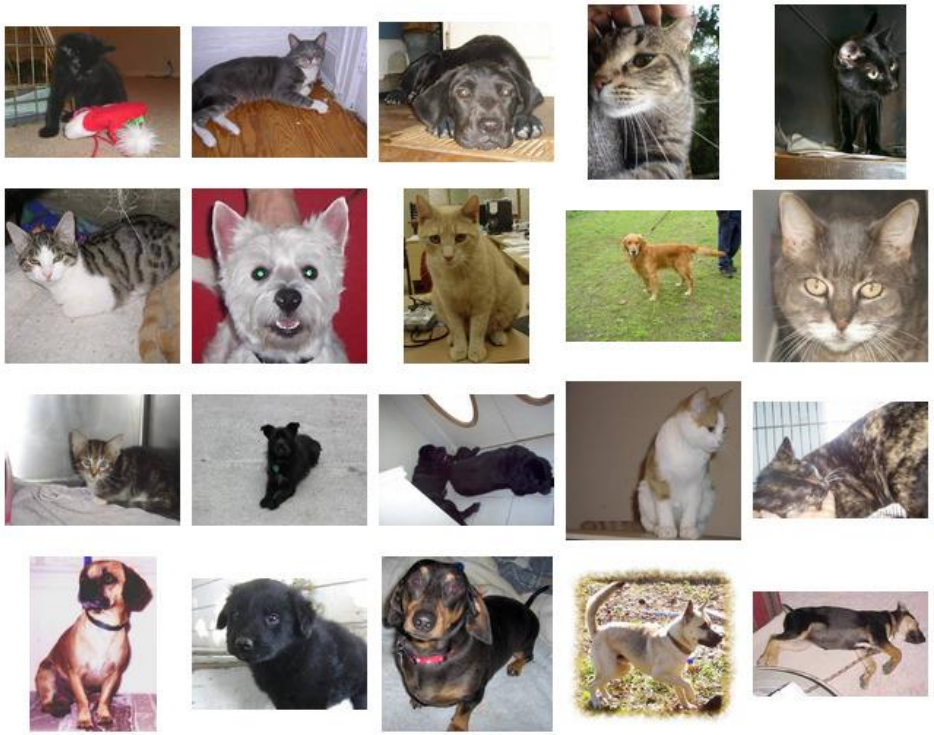
TIC-TAC-TOE DISSECTED



CONTRAST WITH SUPERVISED LEARNING

Supervised learning:

"given a labeled data set $\{x_n, y_n\}_{n=1}^{N}$, produce a function f such that $f(x) \approx y$ on unseen data (x, y)"



labeled set of cat and dog images

image classifier

CONTRAST WITH SUPERVISED LEARNING

Supervised learning:

"given a labeled data set $\{x_n, y_n\}_{n=1}^N$, produce a function f such that $f(x) \approx y$ on unseen data (x, y)"

1. Decisions made by the learning algorithm do not influence the future data
2. We always know what the true labels are

labeled set of cat and dog images

image classifier

CONTRAST WITH SUPERVISED LEARNING

Supervised learning:

"given a labeled data set $\{x_n, y_n\}_{n=1}^{N}$, produce a function f such that $f(x) \approx y$ on unseen data (x, y)"

1. Decisions made by the learning algorithm do not influence the future data
2. We always know what the true labels are

Two challenges of RL:

1. each action influences the future evolution of the game
2. we never find out what would have happened if we took a different action

THE TWO KEY CHALLENGES OF REINFORCEMENT LEARNING

1. dealing with long-term effects of actions

2. dealing with uncertainty due to partial feedback

THE TWO KEY CHALLENGES OF REINFORCEMENT LEARNING

1. dealing with long-term effects of actions

Addressed by Dynamic Programming

2. dealing with uncertainty due to partial feedback

Addressed by Multi-armed bandit theory

DYNAMIC PROGRAMMING ≈

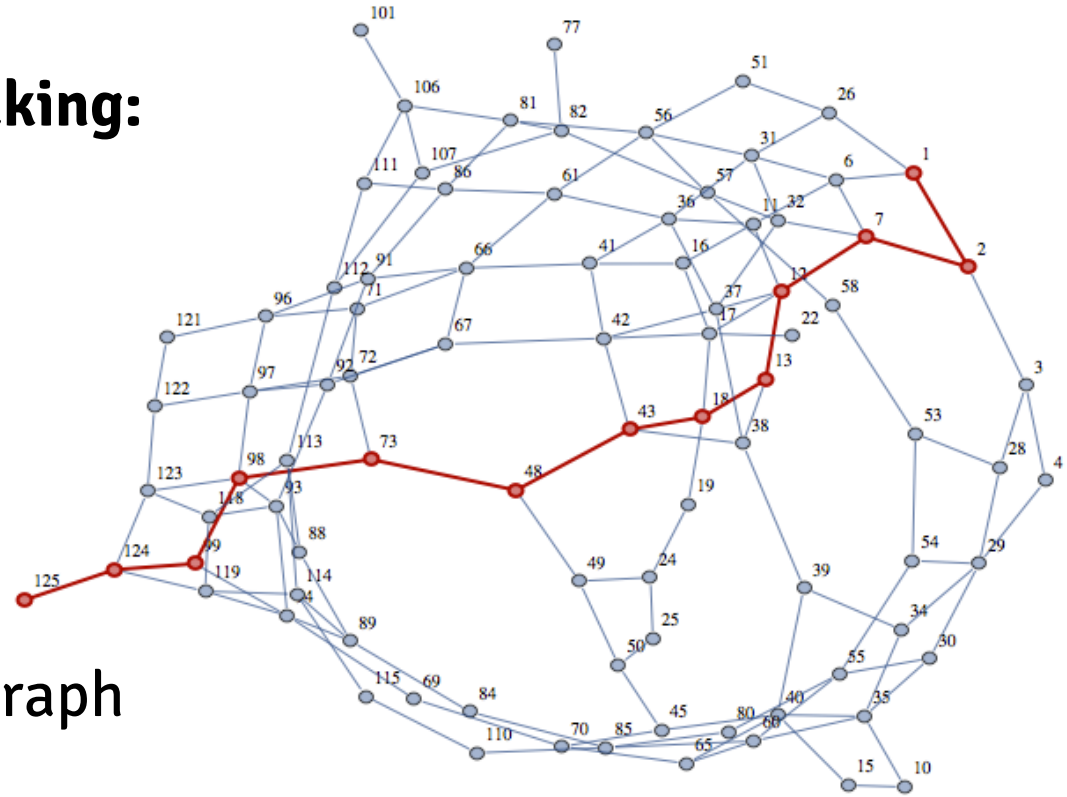
"constructing solutions to a computational problem by breaking it into subproblems"

DYNAMIC PROGRAMMING ≈

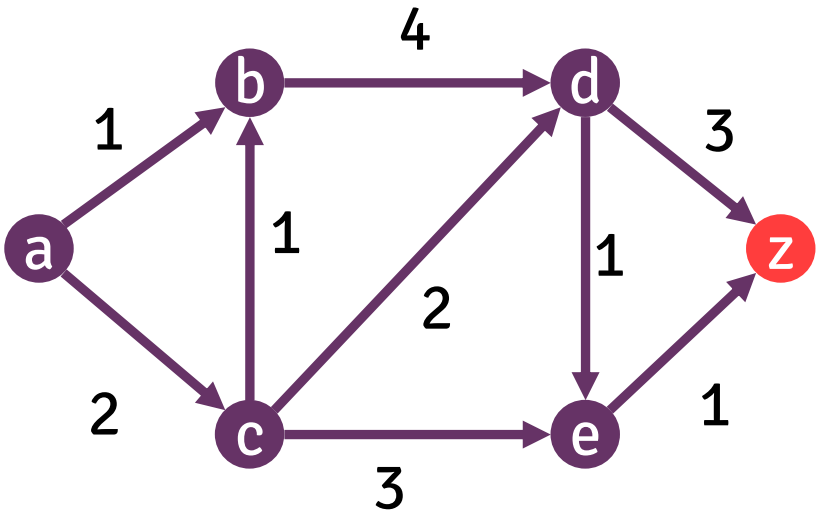
"constructing solutions to a computational problem by breaking it into subproblems"

for sequential decision-making:
"exploiting the sequential nature of the problem to decompose computation"

prime example:
Dijkstra's algorithm for computing shortest paths between two vertices in a graph



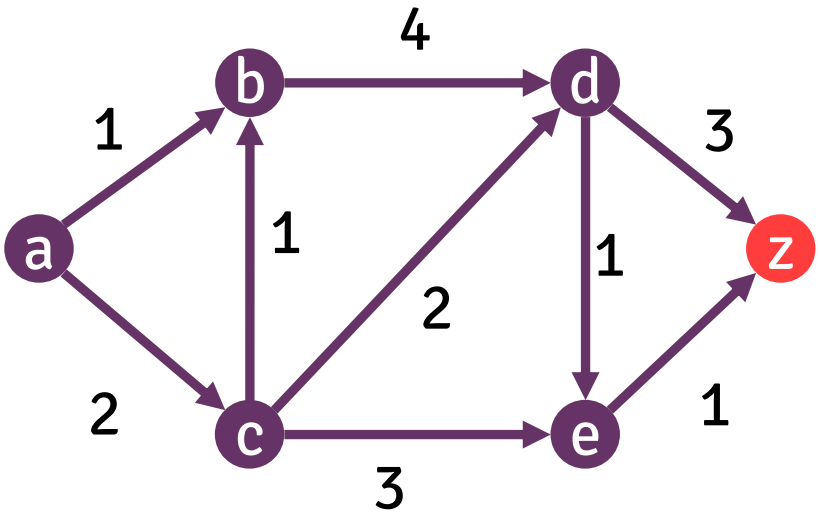
THE KEY IDEA OF DYNAMIC PROGRAMMING



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

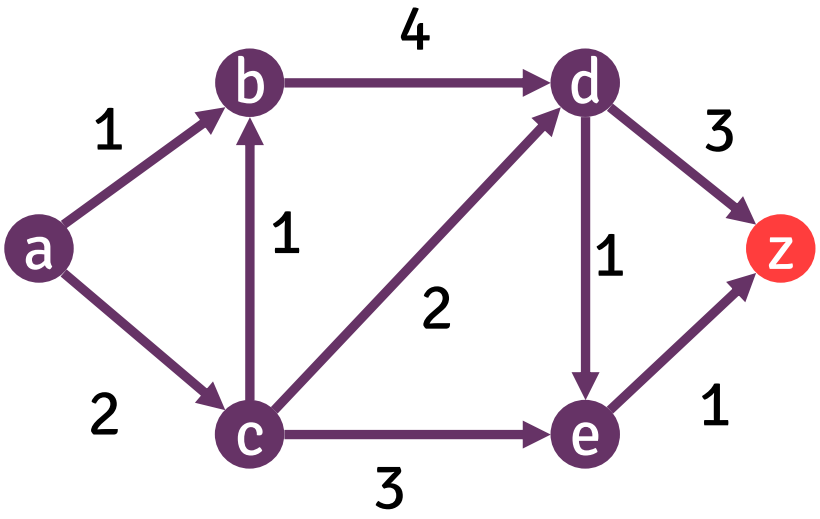
- optimal path always ends with an optimal “sub-path”



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

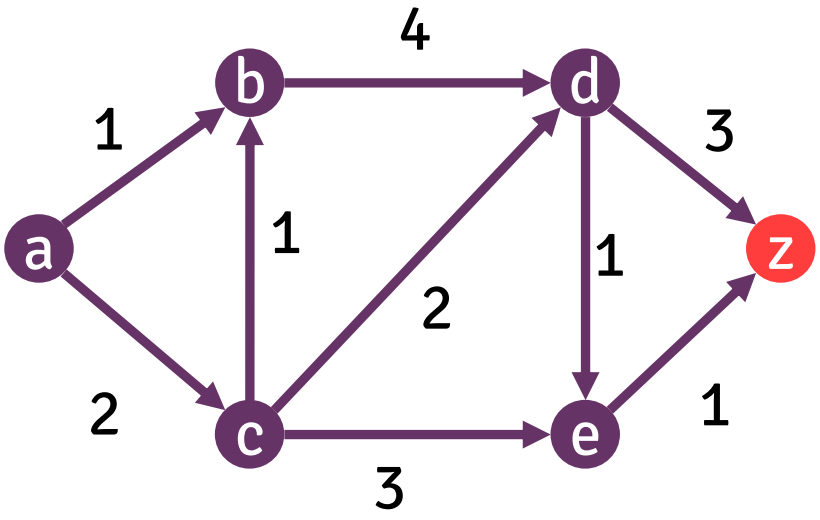
- optimal path always ends with an optimal “sub-path”
- Idea: compute optimal sub-paths from all nodes recursively



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

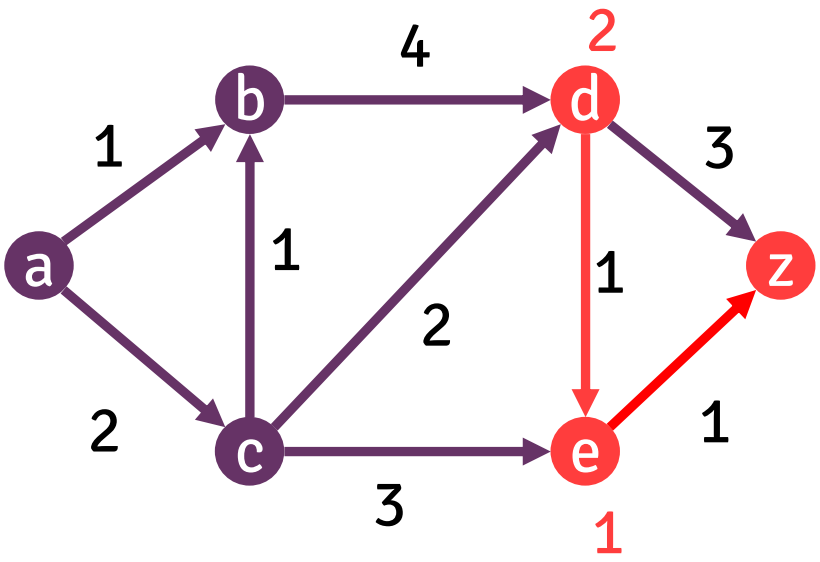
- optimal path always ends with an optimal "sub-path"
- Idea: compute optimal sub-paths from all nodes recursively
- Example:
  - Compute optimal sub-path from d and e and note down lengths



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

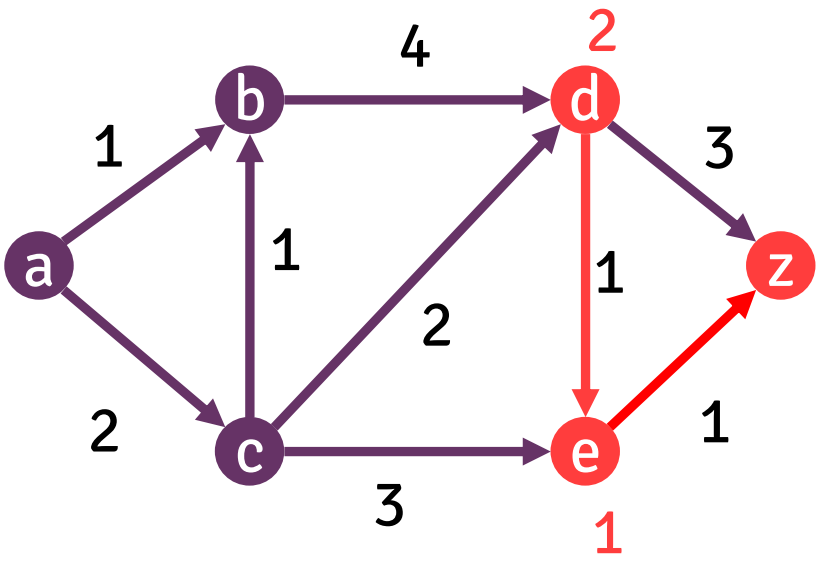
- optimal path always ends with an optimal "sub-path"
- Idea: compute optimal sub-paths from all nodes recursively
- Example:
  - Compute optimal sub-path from d and e and note down lengths



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

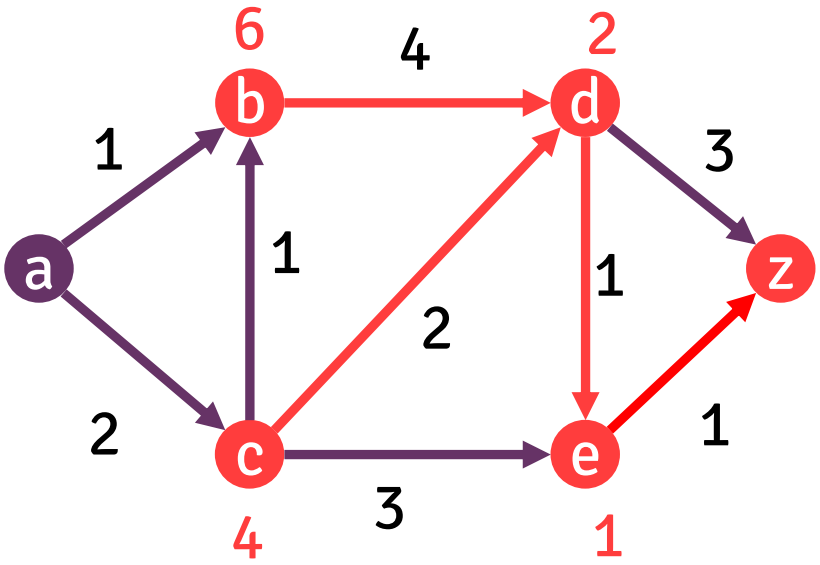
- optimal path always ends with an optimal "sub-path"
- Idea: compute optimal sub-paths from all nodes recursively
- Example:
  - Compute optimal sub-path from d and e and note down lengths
  - Do the same with b and c



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

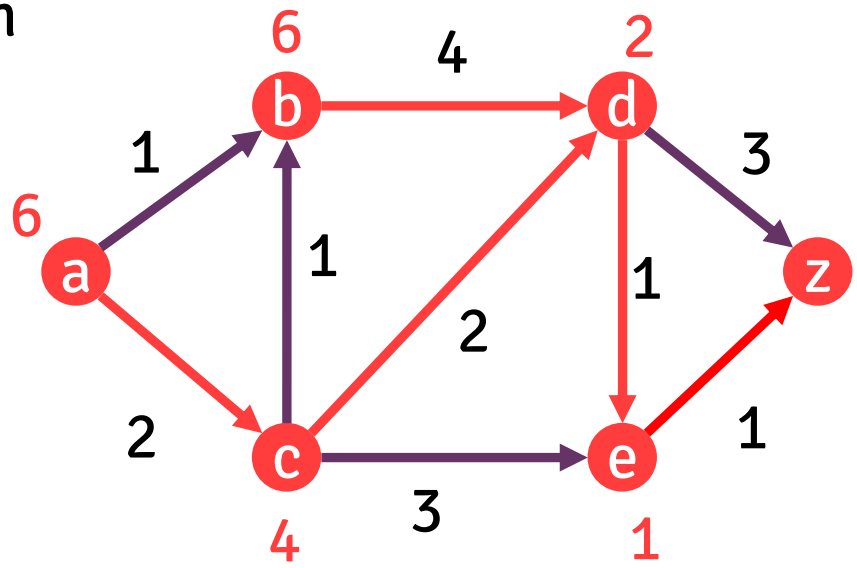
- optimal path always ends with an optimal "sub-path"
- Idea: compute optimal sub-paths from all nodes recursively
- Example:
  - Compute optimal sub-path from d and e and note down lengths
  - Do the same with b and c



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

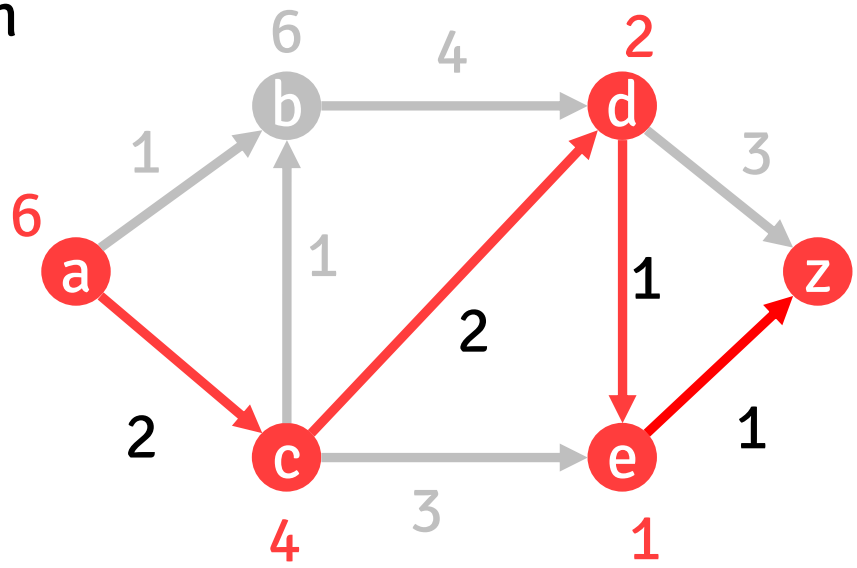
- optimal path always ends with an optimal "sub-path"
- Idea: compute optimal sub-paths from all nodes recursively
- Example:
  - Compute optimal sub-path from d and e and note down lengths
  - Do the same with b and c
  - Do the same with a



THE KEY IDEA OF DYNAMIC PROGRAMMING

Shortest path decomposes:

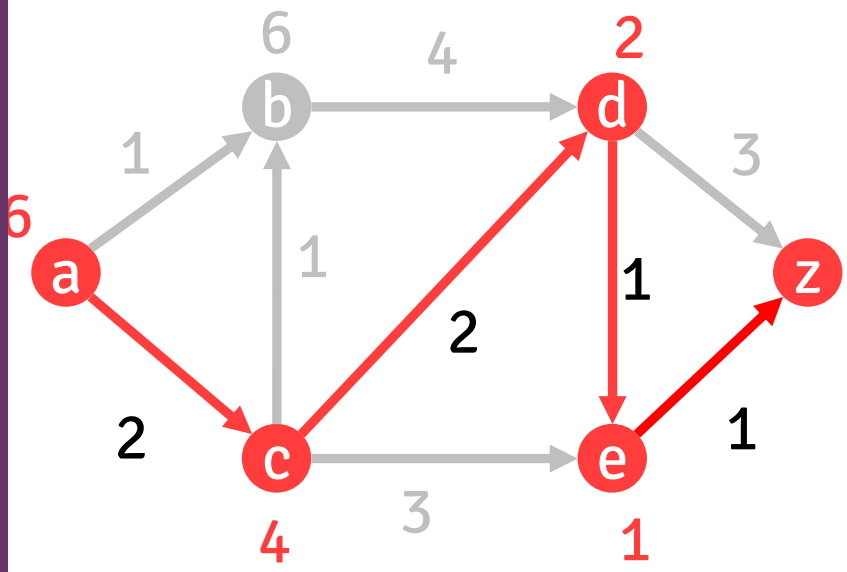
- optimal path always ends with an optimal "sub-path"
- Idea: compute optimal sub-paths from all nodes recursively
- Example:
  - Compute optimal sub-path from d and e and note down lengths
  - Do the same with b and c
  - Do the same with a
  - Extract optimal path



THE KEY IDEA OF DYNAMIC PROGRAMMING

We'll study a generalized version of this method that

1. can deal with infinite decision-making horizons,
2. can deal with randomness,
3. mixes well with machine learning techniques.



THE TWO KEY CHALLENGES OF REINFORCEMENT LEARNING

1. dealing with long-term effects of actions

Addressed by Dynamic Programming

2. dealing with uncertainty due to partial feedback

Addressed by Multi-armed bandit theory


THE BANDIT PROBLEM
Walk into the casino, play slot machines forever, try to maximize gain
GOLD SERIES
GOLD SERIES
GOLD SERIES

THE BANDIT PROBLEM

Walk into the casino, play slot machines forever, try to maximize gain

Problem: Shall I play the machines that gave good rewards so far or try something new?

THE BANDIT PROBLEM

Walk into the casino, play slot machines forever, try to maximize gain

Problem: Shall I play the machines that gave good rewards so far or try something new?

Exploration vs. exploitation

THE BANDIT PROBLEM

Walk into the casino, play slot machines forever, try to maximize gain

Problem: Shall I play the machines that gave good rewards so far or try something new?

Exploration vs. exploitation

MULTI-ARMED BANDIT THEORY

- winning probabilities $p_1, p_2, \dots, p_K$ unknown
- can only learn by taking actions and observing sample rewards
- we never observe what would have happened if we played a different action

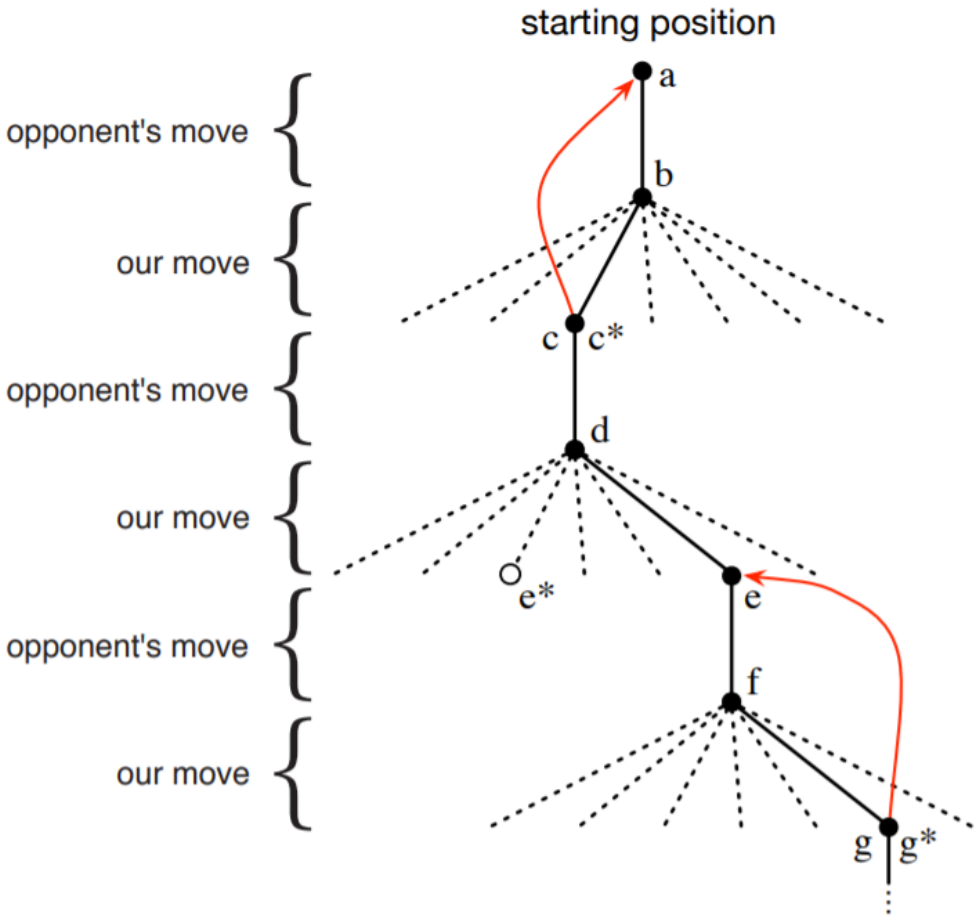
MULTI-ARMED BANDIT THEORY

p_1 p_2 p_3 $\dots$ p_K

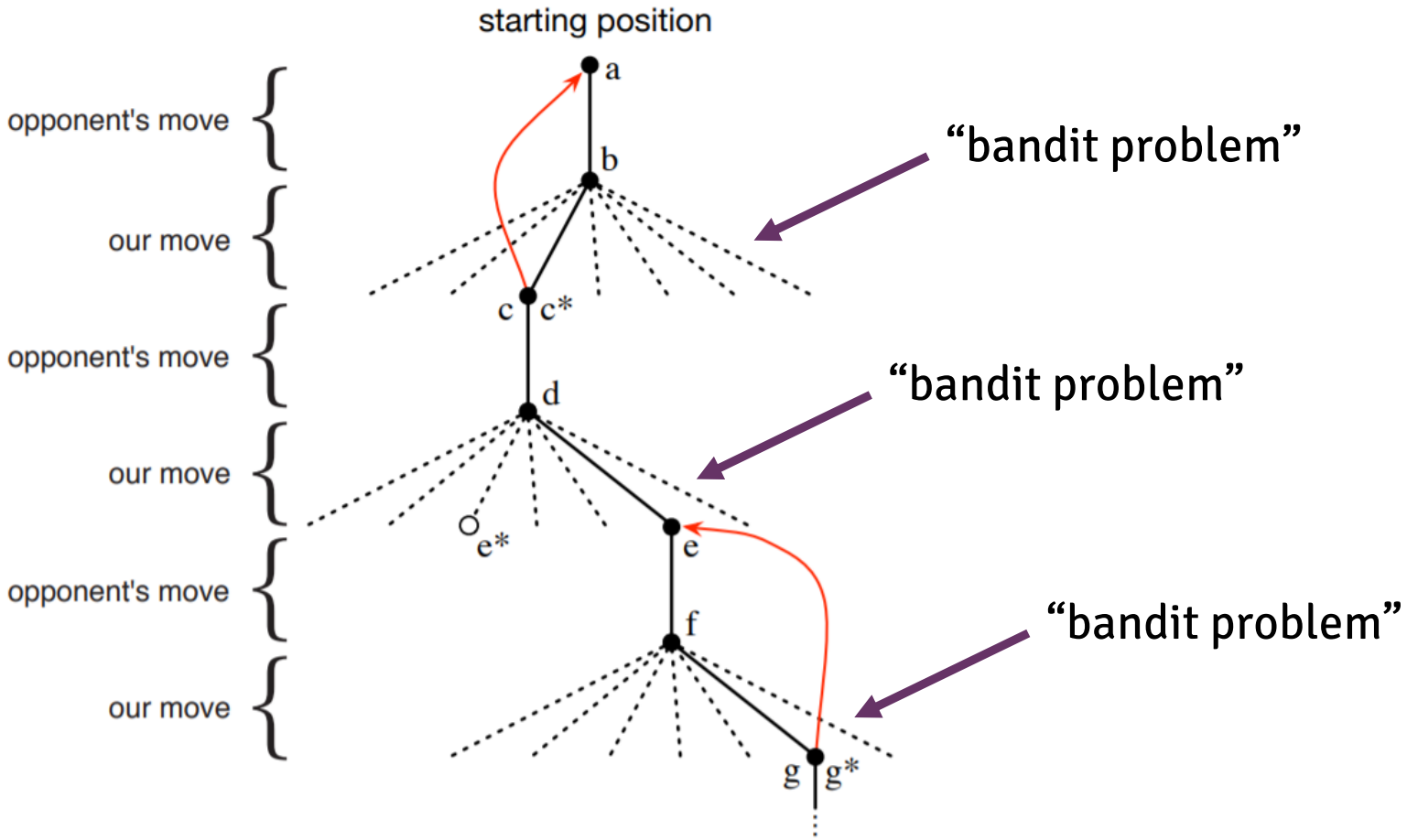
- winning probabilities $p_1, p_2, \dots, p_K$ unknown
- can only learn by taking actions and observing sample rewards
- we never observe what would have happened if we played a different action

Just like in RL!!

REINFORCEMENT LEARNING AND THE BANDIT PROBLEM

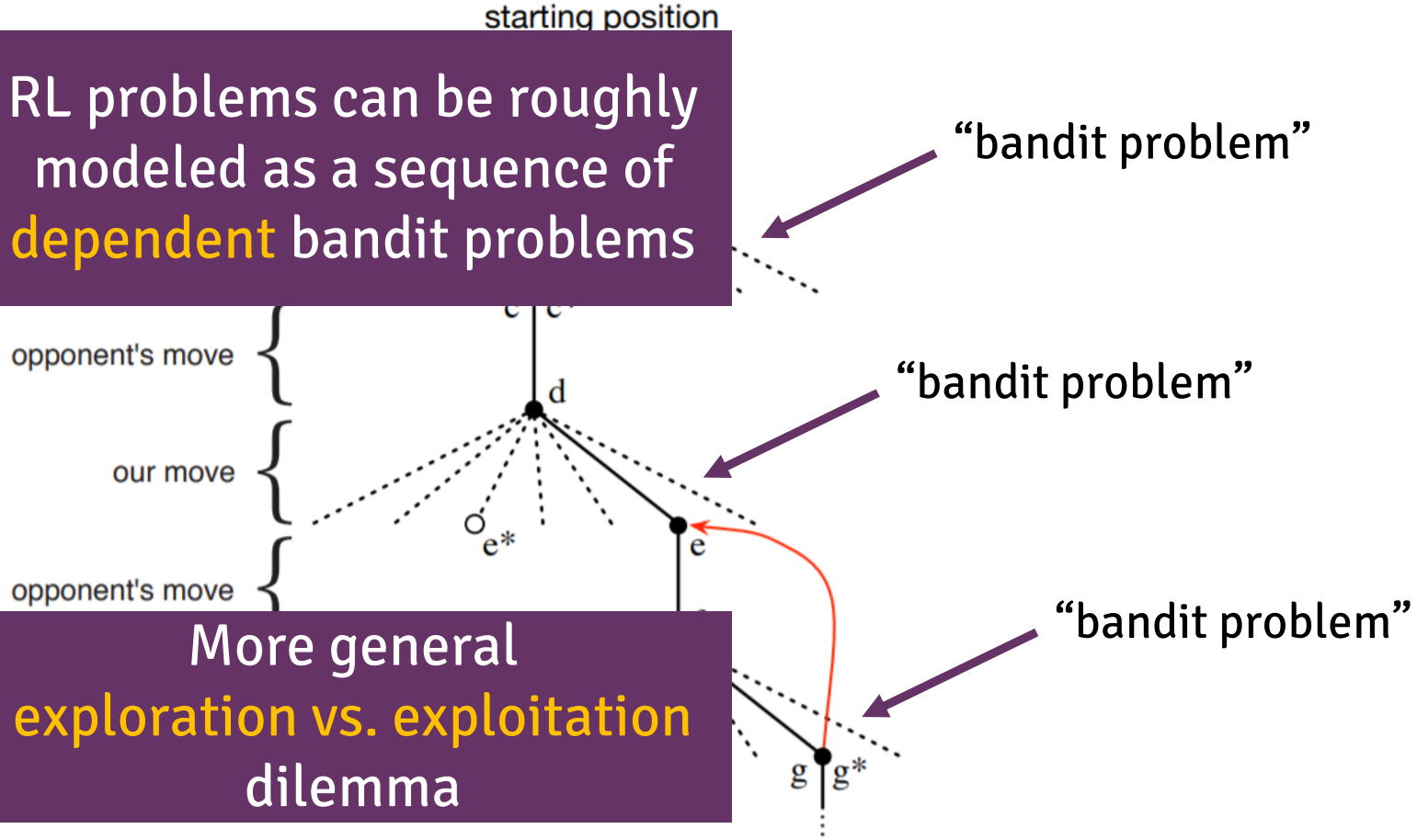


REINFORCEMENT LEARNING AND THE BANDIT PROBLEM



REINFORCEMENT LEARNING AND THE BANDIT PROBLEM

RL problems can be roughly modeled as a sequence of dependent bandit problems



More general exploration vs. exploitation dilemma

WHY MULTI-ARMED BANDIT THEORY?

Provides a comprehensive understanding of the exploration-exploitation tradeoff:

- algorithmic ideas to treat uncertainty
- precise characterization of performance limits
- extensible in numerous ways (e.g., contextual bandits, combinatorial bandits)

WHY MULTI-ARMED BANDIT THEORY?

Provides a comprehensive understanding of the exploration-exploitation tradeoff:

- algorithmic ideas to treat uncertainty
- precise characterization of performance limits
- extensible in numerous ways (e.g., contextual bandits, combinatorial bandits)

Limitation:
cannot properly model long-term effects of actions

RECIPES FOR REINFORCEMENT LEARNING

A theorist's recipe:
dynamic programming + bandit theory

A practitioner's recipe:
dynamic programming + ML tricks for scalability

The ultimate recipe:
dynamic programming + bandit theory
+ ML tricks for scalability

Next week: dynamic programming