



10-301/10-601 Introduction to Machine Learning

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

# In-context Learning + Reinforcement Learning: Markov Decision Processes

Matt Gormley & Geoff Gordon

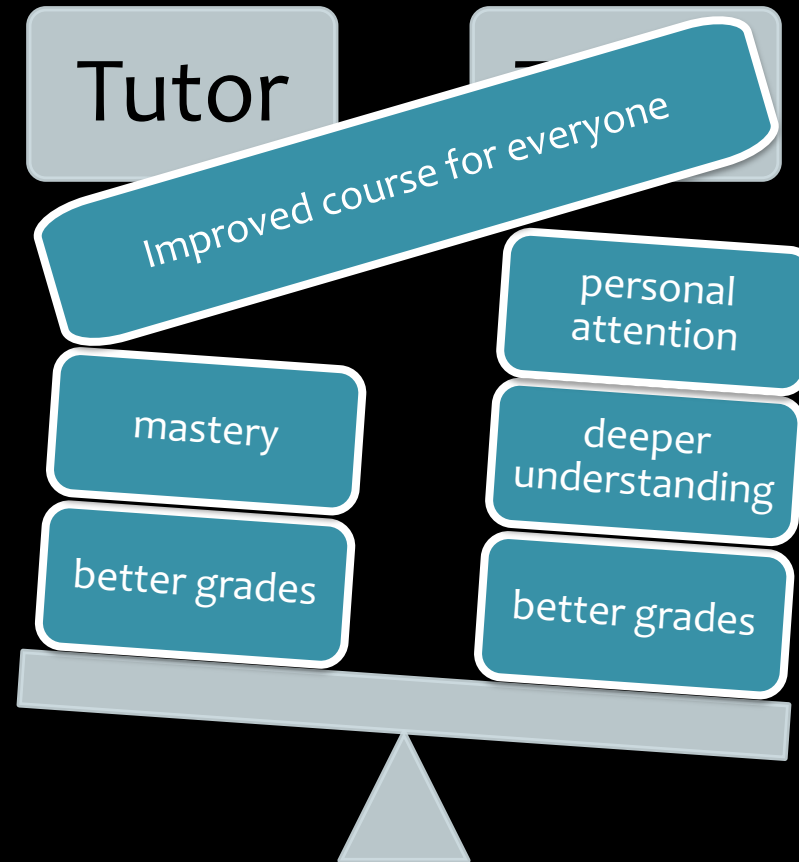
Lecture 20

Nov. 5, 2025

# Reminders

- **Exam 2: Thu, Nov 6, 7:00 pm – 9:00 pm**
  - Scope: Lectures 8 - 16
- **Homework 7: Deep Learning & LLMs**
  - Out: Thu, Nov 6
  - Due: Sun, Nov 16 11:59pm

# Peer Tutoring



# PROMPTING

# Zero-shot vs. Few-Shot

- **Definition:** in **zero-shot learning** we assume that training data does not contain any examples of the labels that appear in the test data
- **Definition:** in **few-shot learning** we assume that training data contains a handful (maybe two, three, or four) examples of each label

# Prompting

- Language models are trained to maximize the likelihood of sequences in the training data
- Most condition on the previous tokens to generate the next tokens

**Definition:** An *autoregressive language model* defines a probability distribution over sequences  $\mathbf{x}_{1:T}$  of the form:

$$p(\mathbf{x}_{1:T}) = \prod_{t=1}^T p(x_t \mid x_1, \dots, x_{t-1})$$

- Key idea behind prompting: provide a prefix string to the such that its likely completion is the answer you want

# Prompting Examples

## GPT-3

Prompt  
(i.e. context)

The City  
BY C. P. CAVAFY  
TRANSLATED BY EDMUND KEELEY  
[Poem text omitted]

SOME TREES  
John Ashbery  
[Poem text omitted]

Shadows on the Way  
Wallace Stevens

Output  
(i.e. generated poem)

I must have shadows on the way  
If I am to walk I must have  
Each step taken slowly and alone  
To have it ready made

And I must think in lines of grey  
To have dim thoughts to be my guide  
Must look on blue and green  
And never let my eye forget  
That color is my friend  
And purple must surround me too

The yellow of the sun is no more  
Intrusive than the bluish snow  
That falls on all of us. I must have  
Grey thoughts and blue thoughts walk with me  
If I am to go away at all.

## Conditional Sampling:

To draw a sample, feed in a prompt (i.e. context) and then sample from the model conditioned on that prompt

# Prompting Examples

## Llama-2-70B

Prompt

El último dueño de esta propiedad había sido un hombre soltero, que alcanzó una muy avanzada edad, y que durante gran parte de su existencia tuvo en su hermana una fiel compañera y ama de casa.

English translation:

Model Output

The last owner of this property was a bachelor man who lived to a very advanced age, and who had his sister as a faithful companion and housekeeper for most of his life.

## Conditional Sampling:

To draw a sample, feed in a prompt (i.e. context) and then sample from the model conditioned on that prompt

English source from Sense & Sensibility

The late owner of this estate was a single man, who lived to a very advanced age, and who for many years of his life, had a constant companion and housekeeper in his sister.

# Prompting Examples

## Llama-2-70B

Prompt

Rhodochiton is a genus of flowering plants within the family Plantaginaceae, native to southern Mexico and neighbouring Guatemala. They climb by means of twining leaf stalks. One of the three species, "Rhodochiton atrosanguineus", the purple bell vine, is grown as an ornamental plant. All three species are sometimes included in "Lophospermum".

Question: You can find the purple bell vine in more than one country. True, False, or Neither?

Model  
Output

Answer: True.

Explanation: The purple bell vine (Rhodochiton atrosanguineus) is native to southern Mexico and neighbouring Guatemala, which means it can be found in more than one country.

Correct Answer

True.

# Prompting Examples

## Llama-2-70B

Prompt

Story: Jason listened to the weather and heard it was going to be sunny. He thought the kids might like to go swimming. He gathered up the swimsuits, towels and sunscreen. Jason and the kids got into the truck and drove to the beach. They spent the next 2 hours playing and splashing in the surf.

One-sentence Summary:

Model Output

Jason took the kids swimming at the beach after hearing the weather forecast, gathering necessary items and driving there.

# What else can a large LM (attempt to) do?

Using the idea of **prompts**, we can apply LMs to a **variety of different problems** in natural language processing.

In the **zero-shot setting**, we simply feed the context to the model and observe how it completes the sequence. (i.e. there is no additional training)

## Answer fact-based questions:

Context →	Organisms require energy in order to do what?
Correct Answer →	mature and develop.
Incorrect Answer →	rest soundly.
Incorrect Answer →	absorb light.
Incorrect Answer →	take in nutrients.

## Complete sentences logically:

Context →	My body cast a shadow over the grass because
Correct Answer →	the sun was rising.
Incorrect Answer →	the grass was cut.

## Complete analogies:

Context →	lull is to trust as
Correct Answer →	cajole is to compliance
Incorrect Answer →	balk is to fortitude
Incorrect Answer →	betray is to loyalty
Incorrect Answer →	hinder is to destination
Incorrect Answer →	soothe is to passion

## Reading comprehension:

Context →	anli 1: anli 1: Fulton James MacGregor MSP is a Scottish politician who is a Scottish National Party (SNP) Member of Scottish Parliament for the constituency of Coatbridge and Chryston. MacGregor is currently Parliamentary Liaison Officer to Shona Robison, Cabinet Secretary for Health & Sport. He also serves on the Justice and Education & Skills committees in the Scottish Parliament. Question: Fulton James MacGregor is a Scottish politician who is a Liaison officer to Shona Robison who he swears is his best friend. True, False, or Neither?
Correct Answer →	Neither
Incorrect Answer →	True
Incorrect Answer →	False

# Zero-shot LLMs

- GPT-2 (1.5B parameters) for unsupervised prediction on various tasks
- GPT-2 models  $p(\text{output} \mid \text{input}, \text{task})$ 
  - translation: (*translate to french, english text, french text*)
  - reading comprehension: (*answer the question, document, question, answer*)
- Why does this work?

---

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile** [I'm not a fool].

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "Lie lie and something will always remain."

"I hate the word '**perfume**,'" Burr says. 'It's somewhat better in French: '**parfum**.'

If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "**- How do you get to the other side? - What side?**".

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

**"Brevet Sans Garantie Du Gouvernement"**, translated to English: "**Patented without government warranty**".

---

*Table 1.* Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

# Zero-shot LLMs

- GPT-2 (1.5B parameters) for unsupervised prediction on various tasks
- GPT-2 models  $p(\text{output} \mid \text{input}, \text{task})$ 
  - translation: (translate to french, english text, french text)
  - reading comprehension: (answer the question, document, question, answer)
- Why does this work?

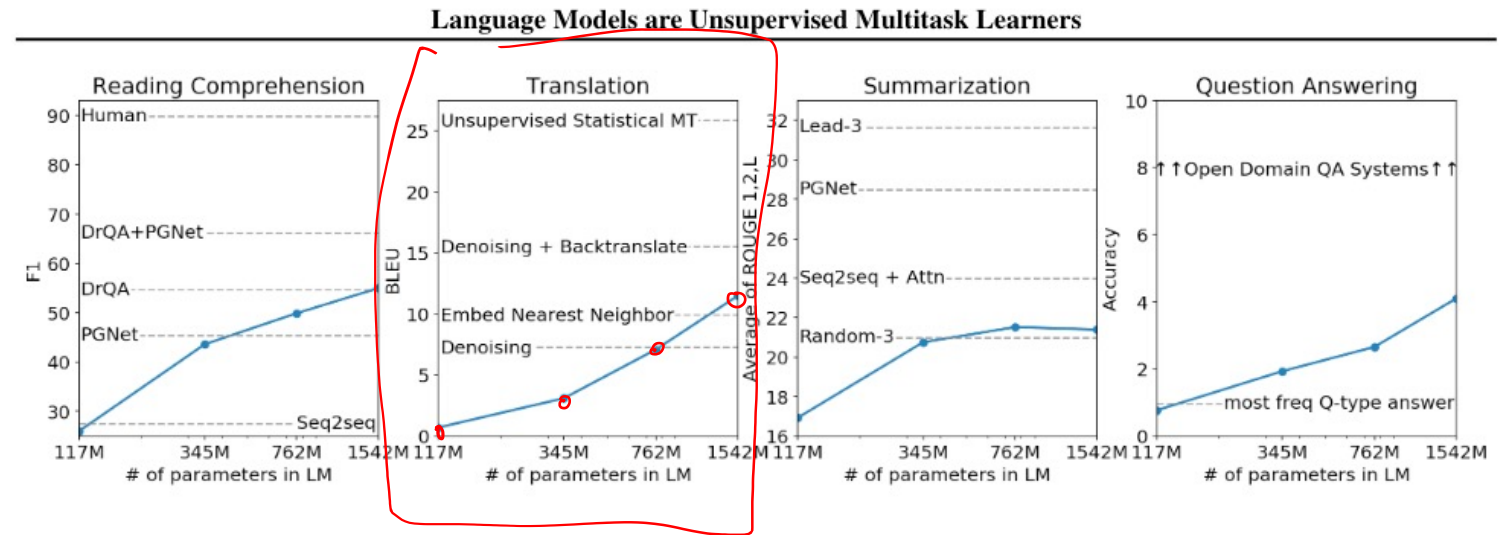


Figure 1. Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	<b>21.8</b>
117M	<b>35.13</b>	45.99	<b>87.65</b>	<b>83.4</b>	<b>29.41</b>	65.85	1.16	1.17	37.50	75.20
345M	<b>15.60</b>	55.48	<b>92.35</b>	<b>87.1</b>	<b>22.76</b>	47.33	1.01	<b>1.06</b>	26.37	55.72
762M	<b>10.87</b>	<b>60.12</b>	<b>93.45</b>	<b>88.0</b>	<b>19.93</b>	<b>40.31</b>	<b>0.97</b>	<b>1.02</b>	22.05	44.575
1542M	<b>8.63</b>	<b>63.24</b>	<b>93.30</b>	<b>89.05</b>	<b>18.34</b>	<b>35.76</b>	<b>0.93</b>	<b>0.98</b>	<b>17.48</b>	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

# **IN-CONTEXT LEARNING**

# Few-shot Learning with LLMs

Suppose you have...

- a dataset  $D = \{(x_i, y_i)\}_{i=1}^N$  and  $N$  is rather small (i.e. few-shot setting)
- a very large (billions of parameters) pre-trained language model

There are two ways to “learn”

This section!



## Option A: Supervised fine-tuning

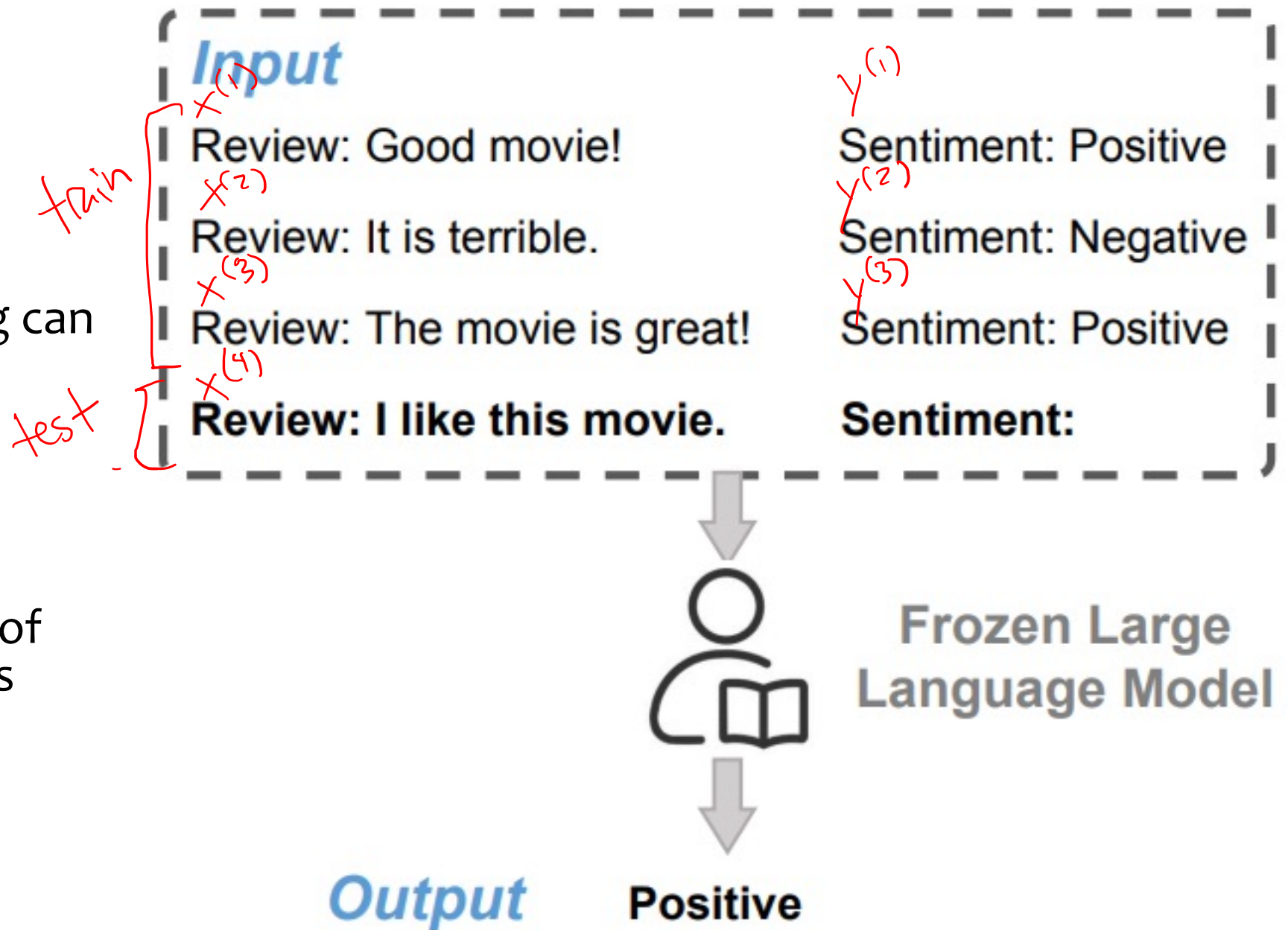
- **Definition:** fine-tune the LLM on the training data using...
  - a standard supervised objective
  - backpropagation to compute gradients
  - your favorite optimizer (e.g. Adam)
- **Pro:** fits into the standard ML recipe
- **Pro:** still works if  $N$  is large
- **Con:** backpropagation requires  $\sim 3\times$  the memory and computation time as the forward computation
- **Con:** you might not have access to the model weights at all (e.g. because the model is proprietary)

## Option B: In-context learning

- **Definition:**
  1. feed training examples to the LLM as a prompt
  2. allow the LLM to infer patterns in the training examples during inference (i.e. decoding)
  3. take the output of the LLM following the prompt as its prediction
- **Con:** the prompt may be very long and Transformer LMs require  $O(N^2)$  time/space where  $N$  = length of context
- **Pro:** no backpropagation required and only one pass through the training data
- **Pro:** does not require model weights, only API access

# Few-shot In-context Learning

- Few-shot learning can be done via in-context learning
- Typically, a task description is presented first
- Then a sequence of input/output pairs from a training dataset are presented in sequence



# Few-shot In-context Learning

- Few-shot learning can be done via in-context learning
- Typically, a task description is presented first
- Then a sequence of input/output pairs from a training dataset are presented in sequence

The three settings we explore for in-context learning

## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



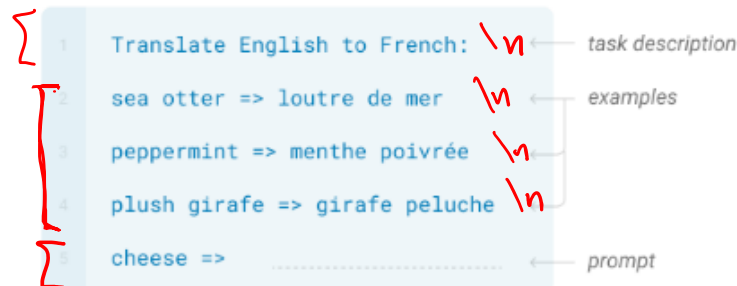
## One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

## Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



# **LEARNING PARADIGMS**

# Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ $\mathbf{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \dots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector

# Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \dots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$

# Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \dots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$

# Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \dots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$

# Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \dots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost

# Learning Paradigms

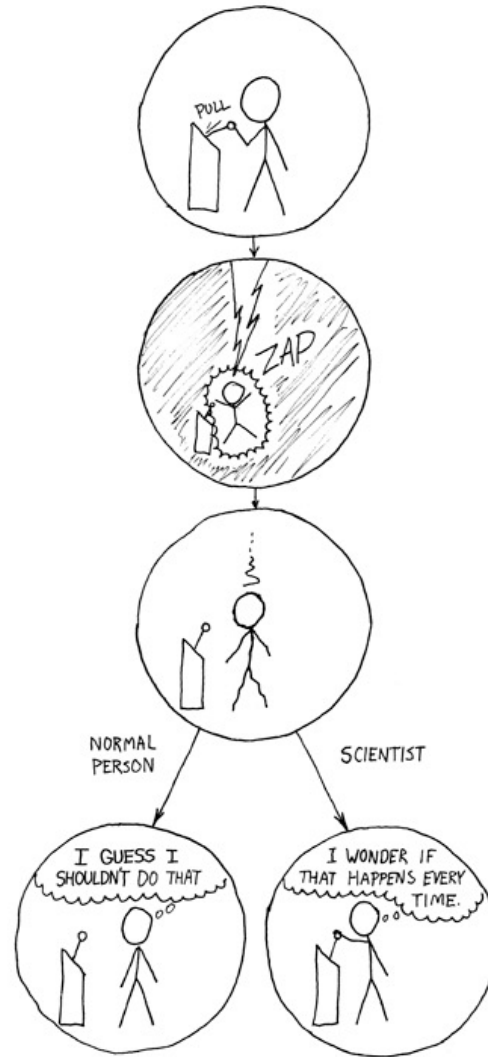
Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ $\mathbf{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \dots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ $\mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \dots\}$

# Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \dots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \dots\}$
Reinforcement Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \dots\}$

# **REINFORCEMENT LEARNING**

# Reinforcement Learning



# RL: Examples



Source: <https://www.cnet.com/news/boston-dynamics-robot-dog-spot-finally-goes-on-sale-for-74500/>

Source: <https://techobserver.net/2019/06/argo-ai-self-driving-car-research-center/>

Source: <https://twitter.com/alphagomovie>

Source: <https://www.wired.com/2012/02/high-speed-trading/>



# AlphaGo

Source: [https://www.youtube.com/watch?v=WXuK6gekU1Y&ab\\_channel=DeepMind](https://www.youtube.com/watch?v=WXuK6gekU1Y&ab_channel=DeepMind)

# History of Reinforcement Learning

- Roots in the **psychology of animal learning** (**Thorndike, 1911**).
- Another independent thread was the problem of **optimal control**, and its solution using **dynamic programming** (**Bellman, 1957**).
- Idea of **temporal difference** learning (on-line method), e.g., playing board games (**Samuel, 1959**).
- A major breakthrough was the discovery of **Q-learning** (**Watkins, 1989**).

# Reinforcement Learning Overview

## Overview

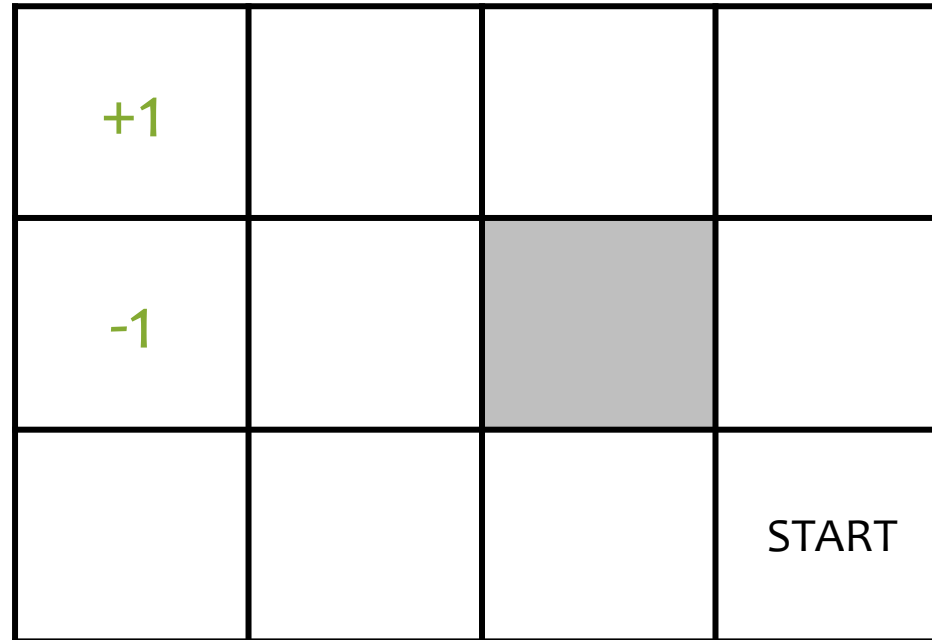
- *What sets RL apart:*
  - RL must learn how to take many actions in sequence over many steps
  - An RL agent learns by trial and error (not exactly supervised)
  - Training signal comes from rewards, which can be very delayed
  - Actions may influence future reward, not just the immediate reward
- *The goal:*
  - To learn a policy that maximizes some value function

## Outline

- *Problem formulation*
  - Time discounted cumulative reward
  - Markov decision processes (MDPs)
- *Key components:*
  - **policy:** map from states to actions
  - **reward function:** maps a state (or state-action pair) to a numerical reward
  - **value function:** the value of a state (or state-action pair) is the expected future reward from following a policy from there
- *Algorithms:*
  - Value & policy iteration (dynamic programming)
  - (Deep) Q-learning (temporal difference learning)
  - Policy Gradient

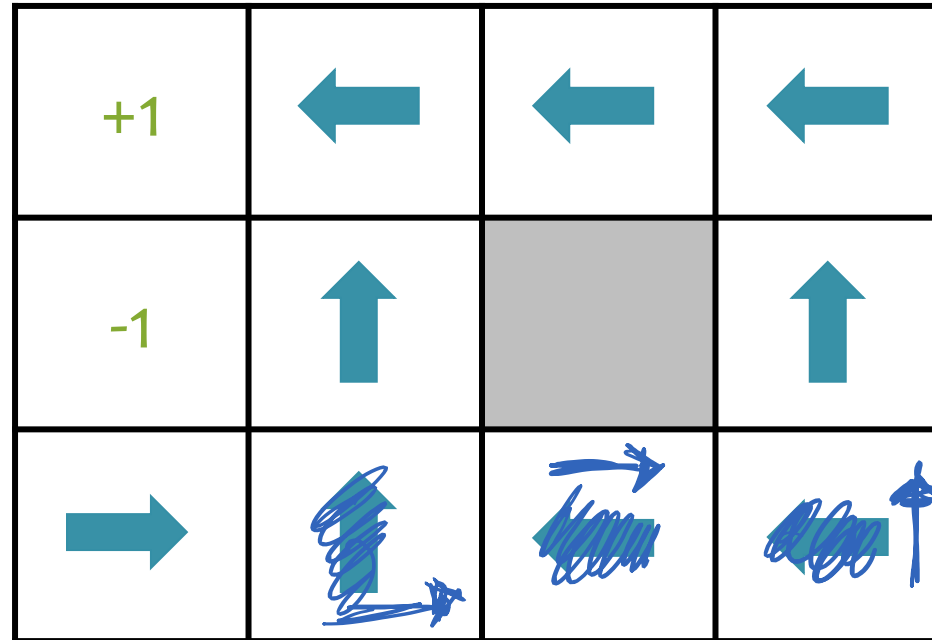
# Example: Robot in a Room

- $\mathcal{S}$  = all empty squares in the grid
- $\mathcal{A} = \{\text{up, down, left, right}\}$
- Deterministic transitions
- Rewards of +1 and -1 for entering the labelled squares
- Terminate after receiving either reward



# Example: Robot in a Room

- $\mathcal{S}$  = all empty squares in the grid
- $\mathcal{A}$  = {up, down, left, right}
- Deterministic transitions
- Rewards of +1 and -1 for entering the labelled squares
- Terminate after receiving either reward



A: Yes 55%  
B: Toxic  
C: No 45%

Q1

## Poll Question:

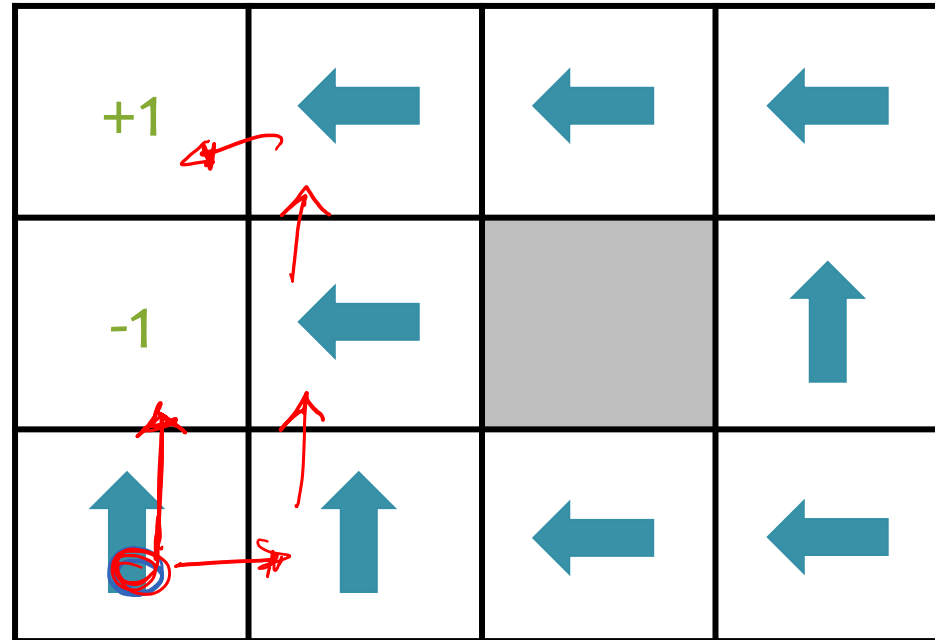
Is this policy optimal: yes or no? Briefly justify your answer.

Q2

**Answer:** (Hint: both yes and no are acceptable answers, I'm interested in your justification.)

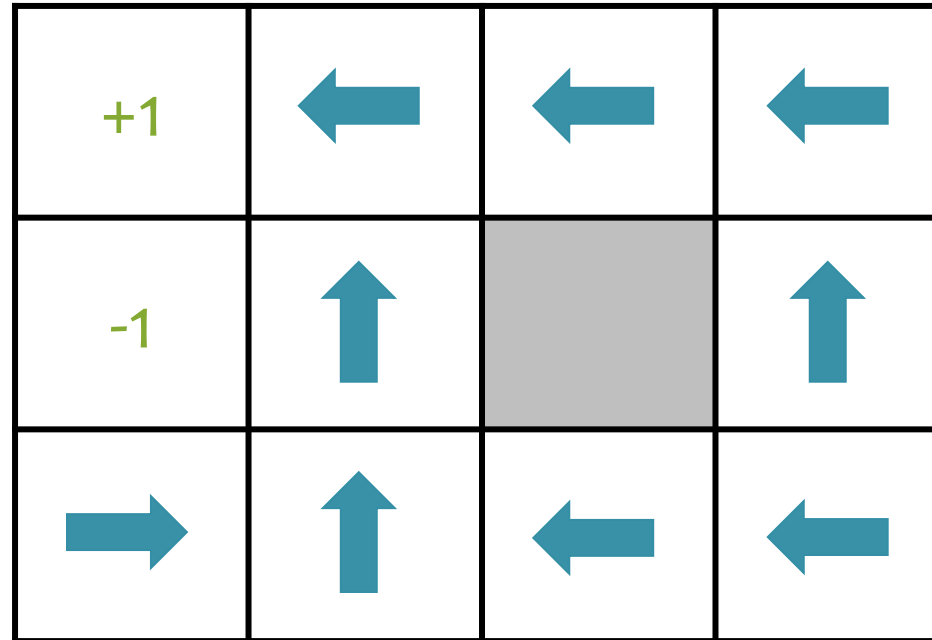
# Example: Robot in a Room

- $\mathcal{S}$  = all empty squares in the grid
- $\mathcal{A} = \{\text{up, down, left, right}\}$
- Deterministic transitions
- Rewards of +1 and -1 for entering the labelled squares
- **Reward for each step is -2**
- Terminate after receiving either reward



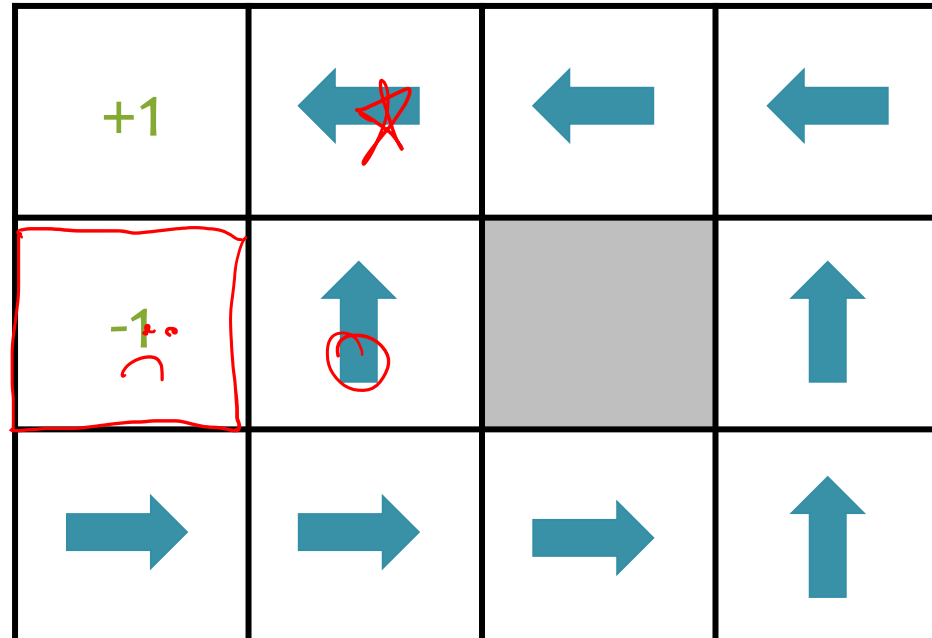
# Example: Robot in a Room

- $\mathcal{S}$  = all empty squares in the grid
- $\mathcal{A} = \{\text{up, down, left, right}\}$
- Deterministic transitions
- Rewards of +1 and -1 for entering the labelled squares
- **Reward for each step is 0.1**
- Terminate after receiving either reward



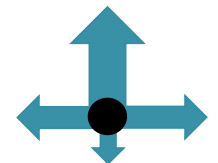
# Example: Robot in a Room

- $\mathcal{S}$  = all empty squares in the grid
- $\mathcal{A} = \{\text{up, down, left, right}\}$
- ~~Deterministic transitions~~
- Rewards of +1 and -1 for entering the labelled squares
- **Reward for each step is 0.01**
- Terminate after receiving either reward



- Stochastic transitions:
  - If we take the action “up”
  - Then our actual movement follows a probability distribution:

75% up  
10% left  
10% right  
5% down



- This implies stochastic rewards as well (i.e. reward is determined by where we end up)

# **MARKOV DECISION PROCESSES**

# RL: Components

## From the Environment (i.e. the MDP)

- State space,  $\mathcal{S}$
- Action space,  $\mathcal{A}$
- Reward function,  $R(s, a)$ ,  $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
- Transition probabilities,  $p(s' | s, a)$ 
  - Deterministic transitions:

$$p(s' | s, a) = \begin{cases} 1 & \text{if } \delta(s, a) = s' \\ 0 & \text{otherwise} \end{cases}$$

where  $\delta(s, a)$  is a transition function

### Markov Assumption

$$p(s_{t+1} | s_t, a_t, \dots, s_1, a_1) \\ = p(s_{t+1} | s_t, a_t)$$

## From the Model

- Policy,  $\pi : \mathcal{S} \rightarrow \mathcal{A}$
- Value function,  $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ 
  - Measures the expected total payoff of starting in some state  $s$  and executing policy  $\pi$
  - Def.: we **execute** a policy  $\pi$  by taking action  $a = \pi(s)$  when in state  $s$

# Markov Decision Process (MDP)

- For **supervised learning** the **PAC learning framework** provided assumptions about where our data came from:

$$\mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$$

- For **reinforcement learning** we assume our data comes from a **Markov decision process (MDP)**

# Markov Decision Processes (MDP)

In RL, the source of our data is an MDP:

1. Start in some initial state  $s_0 \in \mathcal{S}$
  2. For time step  $t$ :
    1. Agent observes state  $s_t \in \mathcal{S}$
    2. Agent takes action  $a_t \in \mathcal{A}$  where  $a_t = \pi(s_t)$
    3. Agent receives reward  $r_t \in \mathbb{R}$  where  $r_t = R(s_t, a_t)$
    4. Agent transitions to state  $s_{t+1} \in \mathcal{S}$  where  $s_{t+1} \sim p(s' | s_t, a_t)$
  3. Total reward is  $\sum_{t=0}^{\infty} \gamma^t r_t$ 
    - The value  $\gamma$  is the “discount factor”, a hyperparameter  $0 < \gamma < 1$
- Makes a Markov assumption: the next state only depends on the current state and action.

# RL: Components

(Stochastic Rewards)

## From the Environment (i.e. the MDP)

- State space,  $\mathcal{S}$
- Action space,  $\mathcal{A}$
- Reward function,  $R(s, a, s')$ ,  $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$
- Transition probabilities,  $p(s' | s, a)$ 
  - Deterministic transitions:

$$p(s' | s, a) = \begin{cases} 1 & \text{if } \delta(s, a) = s' \\ 0 & \text{otherwise} \end{cases}$$

where  $\delta(s, a)$  is a transition function

Markov Assumption

$$p(s_{t+1} | s_t, a_t, \dots, s_1, a_1) \\ = p(s_{t+1} | s_t, a_t)$$

## From the Model

- Policy,  $\pi : \mathcal{S} \rightarrow \mathcal{A}$
- Value function,  $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ 
  - Measures the expected total payoff of starting in some state  $s$  and executing policy  $\pi$

# Markov Decision Processes (MDP)

(Stochastic Rewards)

In RL, the source of our data is an MDP:

1. Start in some initial state  $s_0 \in \mathcal{S}$
2. For time step  $t$ :
  1. Agent observes state  $s_t \in \mathcal{S}$
  2. Agent takes action  $a_t \in \mathcal{A}$  where  $a_t = \pi(s_t)$
  3. Agent receives reward  $r_t \in \mathbb{R}$  where  $r_t = R(s_t, a_t, s_{t+1})$
  4. Agent transitions to state  $s_{t+1} \in \mathcal{S}$  where  $s_{t+1} \sim p(s' | s_t, a_t)$
3. Total reward is  $\sum_{t=0}^{\infty} \gamma^t r_t$ 
  - The value  $\gamma$  is the “discount factor”, a hyperparameter  $0 < \gamma < 1$

- Makes the same Markov assumption we used for HMMs! The next state only depends on the current state and action.
- Def.: we **execute** a policy  $\pi$  by taking action  $a = \pi(s)$  when in state  $s$

# **EXPLORATION VS. EXPLOITATION**

# Exploration vs. Exploitation Tradeoff

- In RL, there is a **tension** between two strategies an agent can follow when interacting with its environment:
  - **Exploration**: the agent takes actions to visit (state, action) pairs it has not seen before, with the hope of uncovering previously unseen high reward states
  - **Exploitation**: the agent takes actions to visit (state, action) pairs it knows to have high reward, with the goal of maximizing reward given its current (possibly limited) knowledge of the environment
- Balancing these two is critical to success in RL!
  - If the agent **only explores**, it performs no better than a random policy
  - If the agent **only exploits**, it will likely never discover an optimal policy
- One approach for trading off between these:  
**the  $\epsilon$ -greedy policy**

# MDP Example: Multi-armed bandit

Single state:  $|\mathcal{S}| = 1$

Three actions:  $\mathcal{A} = \{1, 2, 3\}$

Deterministic transitions

Rewards are stochastic

$$r_t \sim p(\cdot)$$



Bandit 1	Bandit 2	Bandit 3
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???

# **RL OBJECTIVE FUNCTION**

# RL: Objective Function

- Goal: Find a policy  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  for choosing “good” actions that maximize:

$$\mathbb{E}[\text{total reward}] = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$$

$$0 < \gamma < 1$$

- The above is called the  
“finite horizon expected future discounted reward”

↑  
in

- Can we define other notions of optimality?

no-discounting

finite horizon

other-discounting

$$\mathbb{E} \left[ \sum_{t=0}^{\infty} r_t \right]$$

$$\mathbb{E} \left[ \sum_{t=0}^h r_t \right]$$

$$\sum_{t=0}^{\infty} (r_t - c)$$