Reinforcement Learning

CS 59300: RL1

September 2, 2025

Joseph Campbell
Department of Computer Science

PURDUE UNIVERSITY®

Today’s lecture

1. Multi-armed Bandits

2. Markov Decision Processes

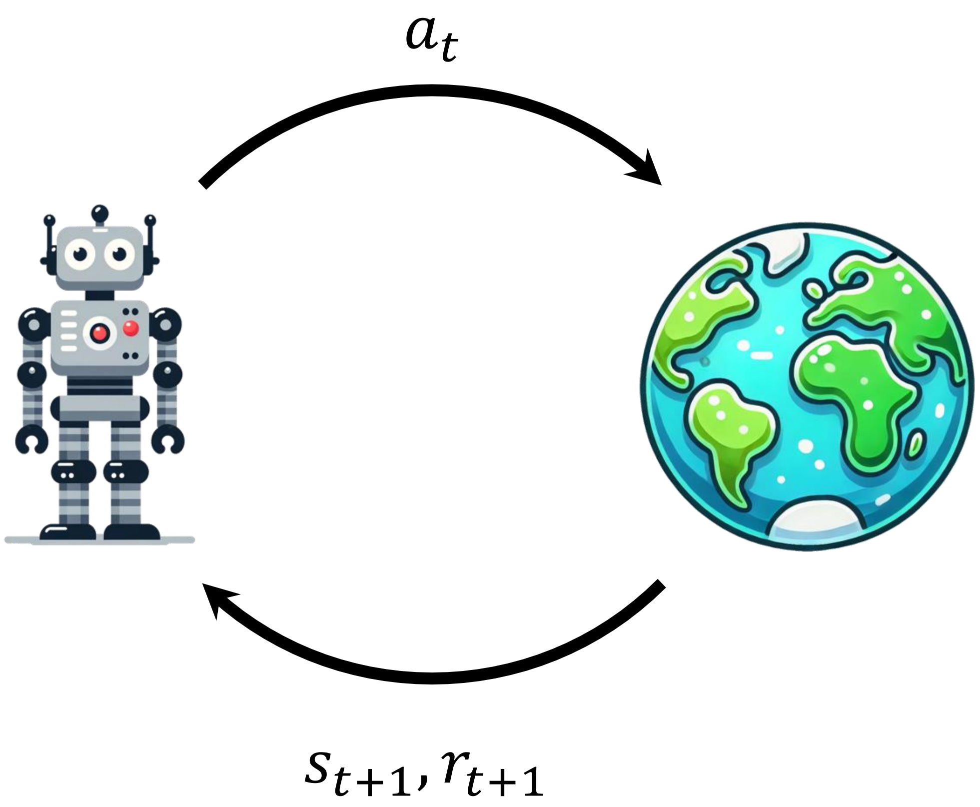
Some content inspired by Katerina Fragkiadaki’s CMU 10-403, Sergey Levine’s Berkeley CS285, and Cathy Wu’s MIT 6.7950 courses

Multi-armed Bandits

Recap

The agent and environment operate at discrete timesteps $t = 0,1,2, \ldots$

- The agent observes state s_t at time t
- The agent takes action a_t
- The agent gets the resulting reward r_{t+1} and the subsequent state s_{t+1}

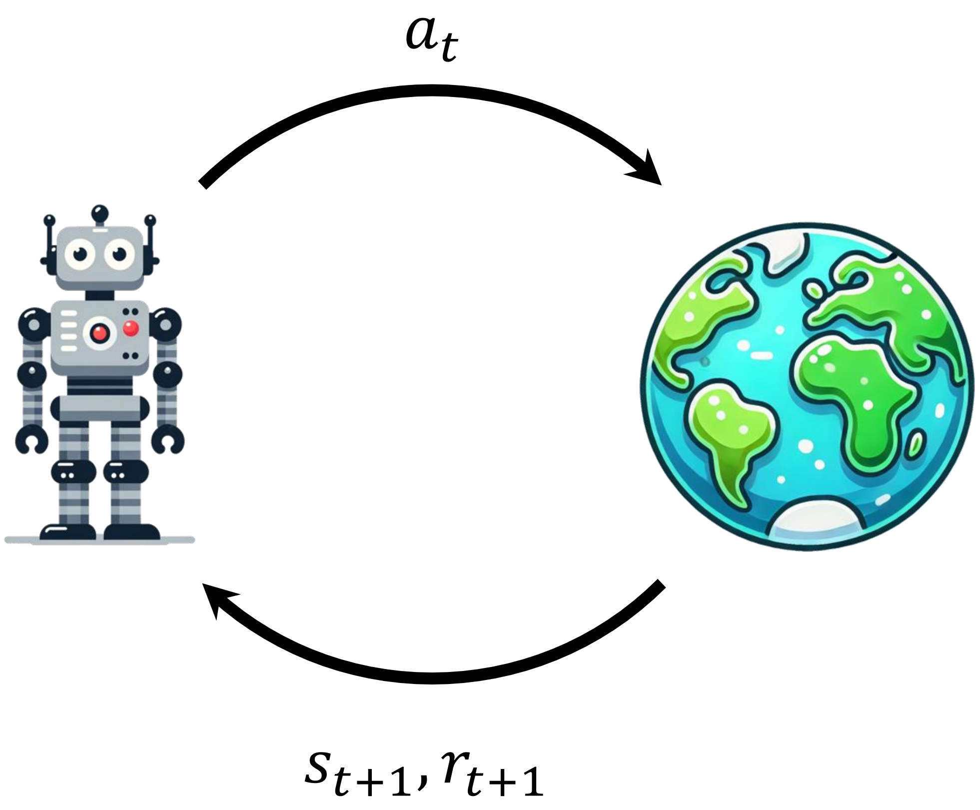


Recap

Action a_t is chosen by sampling actions from a probability distribution

$$a_t \sim \pi(a|s)$$

The probability distribution π is referred to as a policy.



Let’s start with a simple problem formulation

Goal: learning to act in a non-sequential manner

Each action generates an immediate reward

- Choose actions that maximize immediate expected reward

What is an expectation in probability theory?

Let’s start with a simple problem formulation

Goal: learning to act in a non-sequential manner

Each action generates an immediate reward

- Choose actions that maximize immediate expected reward

State-less scenario!

- Our actions do not change our state and do not change the underlying reward distribution

Let’s start with a simple problem formulation

Goal: learning to act in a non-sequential manner

Each action gen

- Choose action d reward

Side note: what is an **expectation** in probability theory?

State-less scenario!

- Our actions do not change our state and do not change the underlying reward distribution

One-armed bandit

Blame any image weirdness on ChatGPT

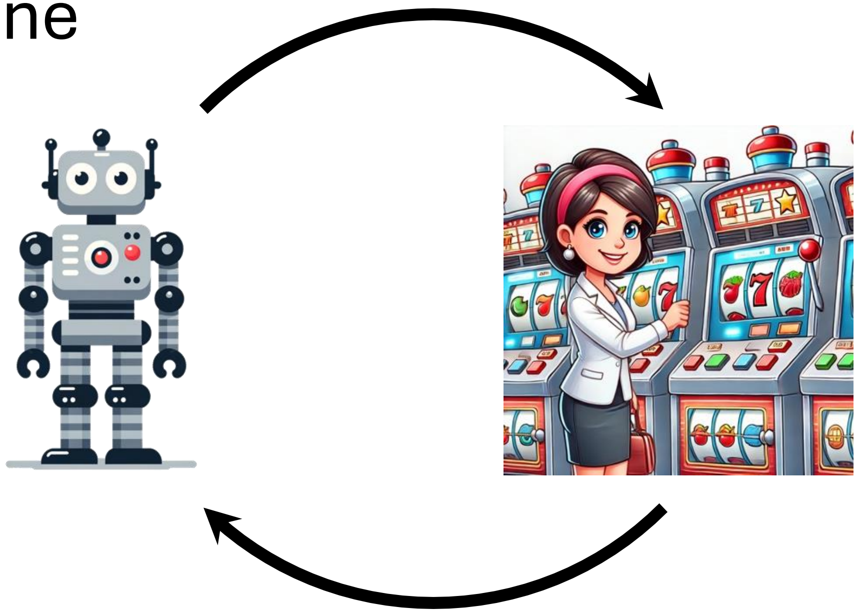
Multi-armed bandit

Blame any image weirdness on ChatGPT

Multi-armed bandit

There are K slot machines

- At each time step the agent plays one machine
- Reward $r_{k,t}$ is drawn from probability distribution $\mathcal{P}_k$ with μ_k
- Agent does not know $\mathcal{P}_k$ or μ_k

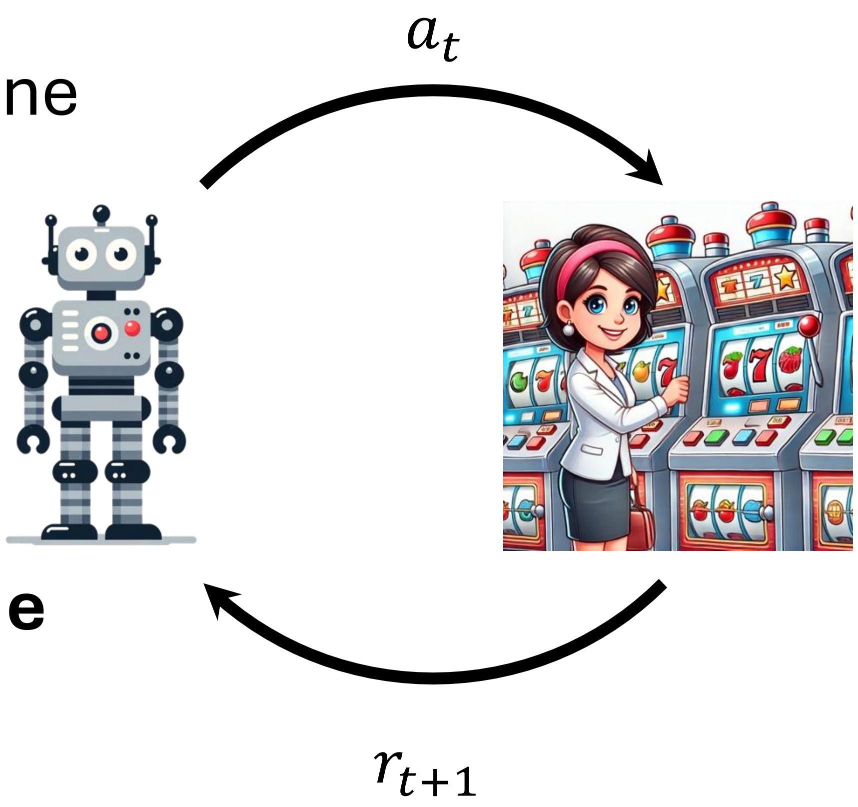


Notice the state doesn't change!

Multi-armed bandit

There are K slot machines

- At each time step the agent plays one machine
- Reward $r_{k,t}$ is drawn from probability distribution $\mathcal{P}_k$ with μ_k
- Agent does not know $\mathcal{P}_k$ or μ_k



Goal: maximize cumulative reward over time

- In practice: over a finite or infinite horizon

Multi-armed bandit

Goal: maximize cumulative reward over time

Simple approach: find arm with the highest mean reward μ_k and play it forever.

Problem: ???

Multi-armed bandit

Goal: maximize cumulative reward over time

Simple approach: find arm with the highest mean reward μ_k and play it forever.

Problem: which arm has the highest mean reward!?

Exploration-exploitation trade-off

Definition: the action-value for action a is its mean reward:

$$Q(a) = \mathbb{E}[r_t | a_t = a]$$

Exploration-exploitation trade-off

Definition: the action-value for action a is its mean reward:

What is a ”value”?

Measures the ”goodness” of a particular action.

How good is it for the agent to take this action?

Exploration-exploitation trade-off

Definition: the action-value for action a is its mean reward:

$$Q(a) = \mathbb{E}[r_t | a_t = a]$$

Suppose we start pulling arms at random to observe the resulting rewards. We now make an estimate of Q at time t:

$$\hat{Q}_t(a) \approx Q(a) \text{ for } \forall a$$

Estimating action-values

What is the easiest way to compute $\hat{Q}_t(a)$?

- Assume at every time step we tried the same action so far…

Estimating action-values

What is the easiest way to compute $\hat{Q}_t(a)$?

- Assume at every time step we tried the same action so far...

$$\hat{Q}_t(a) = \frac{r_1 + r_2 + \cdots + r_t}{t}$$

Although in practice we will be choosing different actions so we need to make sure to sum the correct rewards

Or equivalently…

$$\hat{Q}_t(a) = Q_{t-1} + \frac{1}{t}[r_t - Q_{t-1}(a)]$$

Which is a form that you will see a lot of this semester!

$$New = Old + Step[Target - Old]$$

Or equivalently…

$$\hat{Q}_t(a) = Q_{t-1} + \frac{1}{t}[r_t - Q_{t-1}(a)]$$

Which is a for

Does this work if the reward distribution changes (is non-stationary)?

Exploration-exploitation trade-off

Definition: the greedy action $\hat{a}_t^*$ at time t is:

$$\hat{a}_t^* = \arg\max_a \hat{Q}_t(a)$$

Exploration-exploitation trade-off

Definition: the greedy action $\hat{a}_t^*$ at time t is:

$$\hat{a}_t^* = \arg\max_a \hat{Q}_t(a)$$

Two cases:

1. If $a_t = \hat{a}_t^*$ then you are exploiting
2. If $a_t \neq \hat{a}_t^*$ then you are exploring

Exploration-exploitation intuition

1. If $a_t = \hat{a}_t^*$ then you are exploiting

This means are greedily taking the action that we think leads to the best mean reward.

Recall: $\hat{Q}_t(a)$ is our **estimate** of the expected reward!

- What if it is not accurate and we do not get the best reward?
- Should we still be greedily taking this action?

Exploration-exploitation intuition

2. If $a_t \neq \hat{a}_t^*$ then you are exploring

This means we are *not* taking the action that we think leads to the best reward and instead we are trying other things. We are **exploring!**

Exploration-exploitation intuition

1. If $a_t = \hat{a}_t^*$ then you are exploiting
2. If $a_t \neq \hat{a}_t^*$ then you are exploring

We cannot do both at the same time and yet we need to do both

- How do we decide when to exploit and when to explore?

Exploration-exploitation intuition

1. If $a_t = \hat{a}_t^*$ then you are exploiting
2. If $a_t \neq \hat{a}_t^*$ then you are exploring

We cannot do both at the same time and yet we need to do both

- How do we decide when to exploit and when to explore?

There are many solutions, and yet this is very much an open problem!

Exploration-exploitation intuition

Online decision-making involves this fundamental choice:

- Exploit: make the best decision given current information
- Explore: gather more information

The best long-term strategy may require short-term sacrifices

- We need to gather enough information to make the best decision

Exploration-exploitation intuition

Online decision-making involves this fundamental choice:

- Exploit: make the best decision given current information
- Explore: gather more information

The best long-term strategy may require short-term sacrifices

- We need to gather enough information to make the best decision

This is a fundamental dilemma in any decision-making problem!

We deal with this everyday!

Where do we go to lunch?

- Exploit: go to your favorite restaurant that you **know about**
- Explore: go to a new restaurant; maybe it is your new favorite?

We deal with this everyday!

Where do we go to lunch?

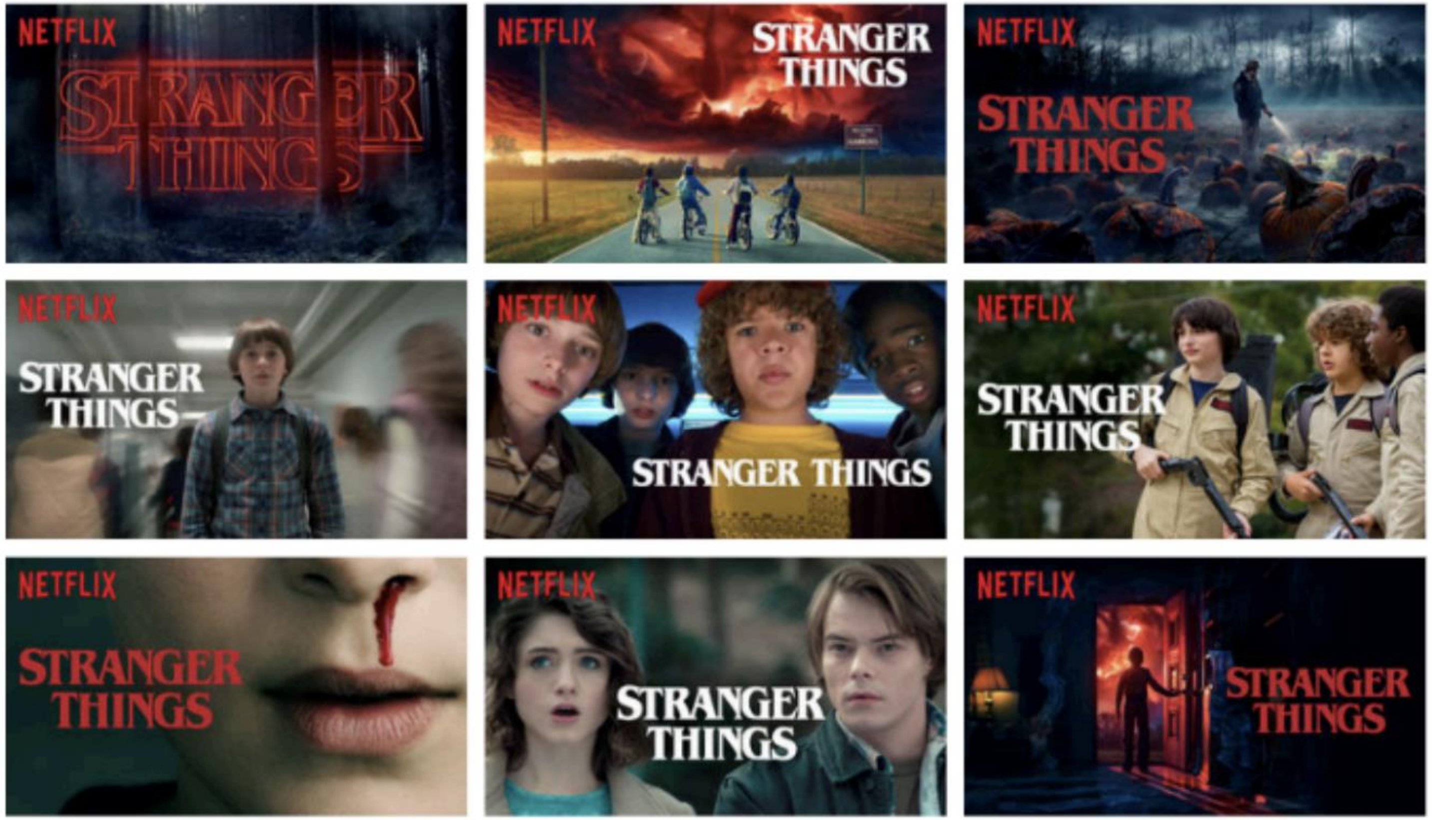
- Exploit: go to your favorite restaurant that you **know about**
- Explore: go to a new restaurant; maybe it is your new favorite?

What do we watch on Netflix?

- Exploit: the next episode in your comfort series
- Explore: watch a new show

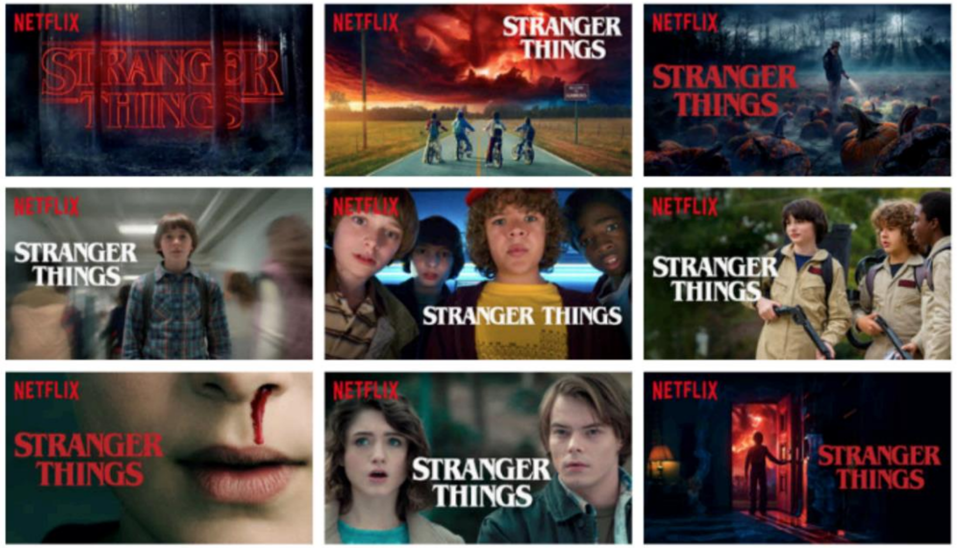
Real-world example

What artwork is shown to a user in order to maximize watch-rate?



Real-world example

For a particular movie, we want to pick an image to show to users



Actions:

- Display one of K images

Reward:

- 1 if the user watches, 0 otherwise
- Mean reward: percentage of users that watched the show

Simplest solution: ϵ-greedy

Intuition: take the (estimated) optimal action most of the time but occasionally take a random action.

Simplest solution: ϵ-greedy

Intuition: take the (estimated) optimal action most of the time but occasionally take a random action.

This is ϵ-greedy!

With greedy action selection you always exploit.

With ϵ-greedy action selection you are usually greedy, but with some probability ϵ you take a non-greedy random action

- Simplest way to balance exploration-exploitation!

Simplest solution: ϵ-greedy

Intuition: take the (estimated) optimal action most of the time but occasionally take a random action.

This is ϵ-greed

If we are clever we can start with a large ϵ and decay it over time…

With greedy action selection you always exploit.

With ϵ-greedy action selection you are usually greedy, but with some probability ϵ you take a non-greedy random action

- Simplest way to balance exploration-exploitation!

Regret

The action-value for action a is its mean reward:

$$Q(a) = \mathbb{E}[r_t | a_t = a]$$

The optimal action is then:

$$a^* = \arg\max_a Q(a)$$

Unlike before this is not an estimate ($\hat{a}_t^*$), this is optimal...

- Notice there's no timestep since the optimal does not depend on time

Regret

The regret is the expected opportunity loss for one step

$$I_t = \mathbb{E}[Q(a^*) - Q(a_t)]$$

It is the difference between the best reward we *could* have received if we chose the best arm, and the reward that we *actually* received.

Total regret over T steps:

$$L_t = \mathbb{E}[\sum_{t=1}^{T} Q(a^*) - Q(a_t)]$$

Regret

The regret is the expected opportunity loss for one step

$$l_t = \mathbb{E}[Q(a^*) - Q(a_t)]$$

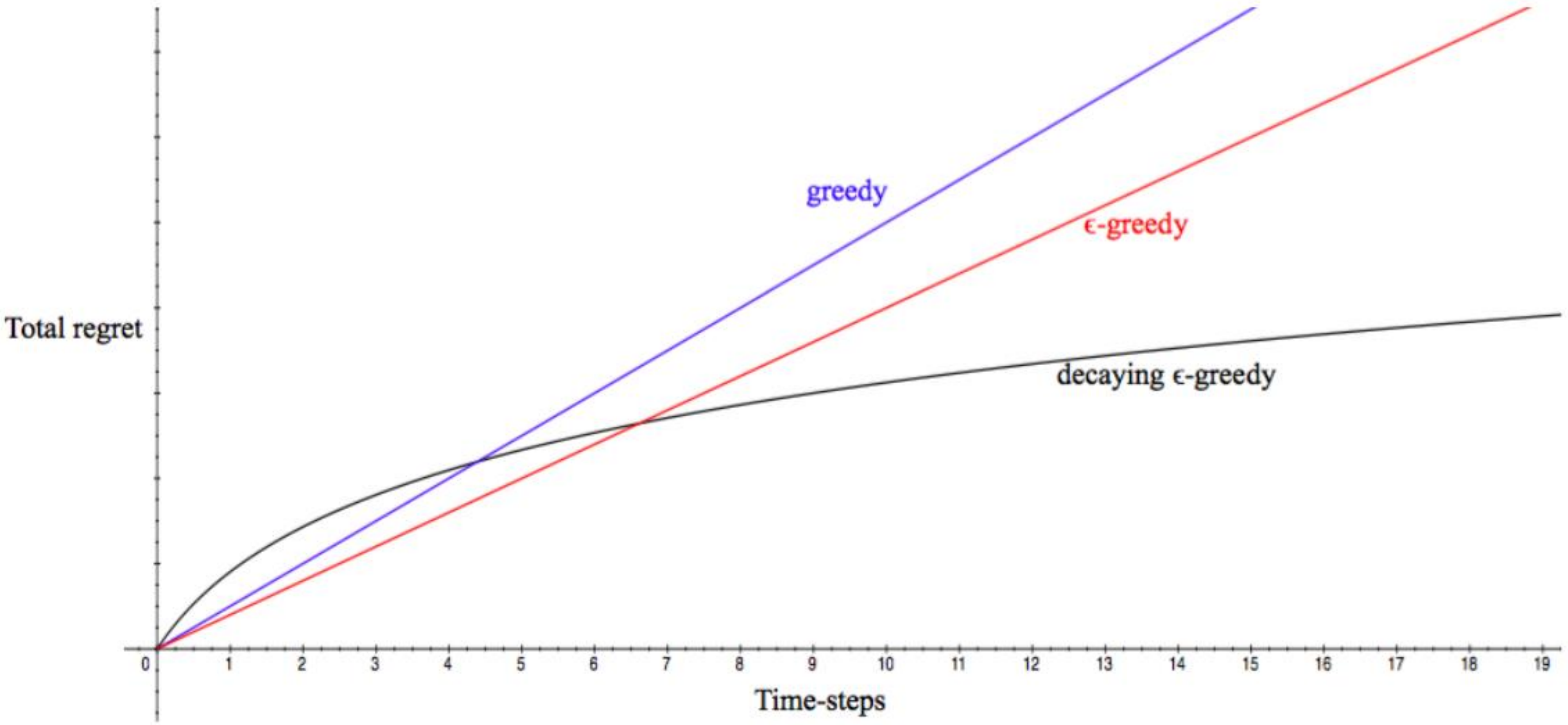
It is the diff … eived if we chose the b …

Maximizing the cumulative expected reward is equivalent to minimizing the total regret!

Total regret over T steps:

$$L_t = \mathbb{E}[\sum_{t=1}^{T} Q(a^*) - Q(a_t)]$$

Regret in greedy algorithms



Other approaches: Upper Confidence Bounds

Estimate an upper confidence $\hat{U}_t(a)$ for each action-value such that with high probability:

$$Q(a) \leq \hat{Q}_t + \hat{U}_t(a)$$

The confidence depends on the number of times that action a has been selected...

- Small number of times → a large uncertainty
- Large number of times → a small uncertainty

Other approaches: Upper Confidence Bounds

In UCB, we select the action that maximizes this value

$$a_t = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t + \hat{U}_t(a)$$

$$\hat{U}_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

Comes from Hoeffding's inequality

Other approaches: Upper Confidence Bounds

$$a_t = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t + \hat{U}_t(a)$$

Intuition: the less we have selected a certain action, the bigger the bonus that we provide, meaning the more likely we are to select it.

Why do we want this?

Other approaches: Upper Confidence Bounds

$$a_t = \mathrm{argmax}_{a \in \mathcal{A}} \hat{Q}_t + \hat{U}_t(a)$$

Intuition: the less we have selected a certain action, the bigger the bonus that we provide, meaning the more likely we are to select it.

Why do we want this?

To encourage exploration!

Other approaches: Upper Confidence Bounds

$$a_t = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t + \hat{U}_t(a)$$

For further reading see *Reinforcement Learning: An Introduction*

- Section 2.7, page 35 in Second Edition
- Seminal textbook by Richard Sutton and Andrew Barto
- Free PDF available at: http://incompleteideas.net/book/the-book-2nd.html

Other approaches: Thompson Sampling

Model a distribution over the mean reward for each bandit rather than just a point-estimate of the mean reward

1. Sample from the mean reward distributions
$$\theta_1 \sim p_1, \dots, \theta_K \sim p_K$$
2. Choose action
$$a = \text{argmax}_{a \in \mathcal{A}} \mathbb{E}_\theta[R(a)]$$
3. Observe reward
4. Update mean reward distributions

Other approaches: Thompson Sampling

For further reading see *A Tutorial on Thompson Sampling*

- By Daniel Russo et al.
- PDF available at https://web.stanford.edu/~bvr/pubs/TS_Tutorial.pdf

Markov Decision Processes

Sequential decision-making

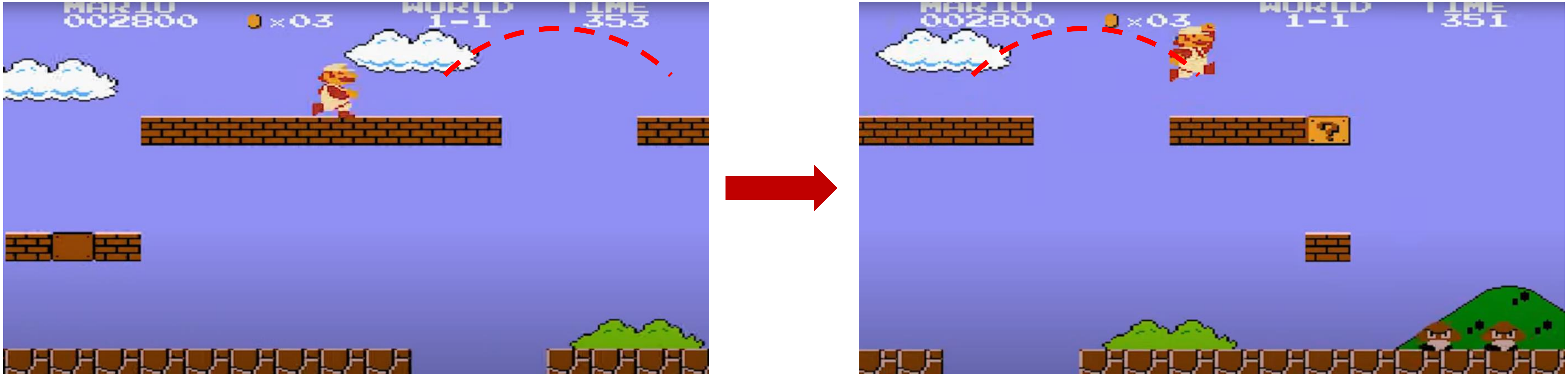
Recall: Bandits are non-sequential and stateless

- Each action is independent of previous actions
- Rewards depend only on the arm we choose

But what if actions **aren't** independent?

For example, a game-playing agent? Or robotics?


MARIO
002800
×03
WORLD
1-1
TIME
344



Whether Mario makes this jump depends on its previous actions!

Mario must first run to the right to build appropriate speed before jumping. This is sequential decision-making.

Markov Decision Process

We formulate sequential decision-making problems as MDPs

Definition: a tuple consisting of $(\mathcal{S}, \mathcal{A}, R, T, \gamma)$:

- $\mathcal{S}$ is the set of states in our environment
- $\mathcal{A}$ is a set of actions that can be taken by an agent
- R is the reward function. $\mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- T is the state transition probability. $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0,1]$
- γ is a discount factor. $\gamma \in [0,1]$

Markov Decision Process

A state captures all information that is available to the agent about its environment at time t

Example: in Mario…

- The position of Mario
- The position of all blocks
- The position of all enemies
- The velocity/acceleration of Mario
- The score and time remaining
- …

What do we mean by “Markov”?

Definition: the Markov property means that future states depend only on the present state (and not on any preceding states).

What do we mean by "Markov"?

Definition: the Markov property means that future states depend only on the present state (and not on any preceding states).

If I know Mario's current position, velocity, acceleration, and the location of all blocks and enemies…

…do I care about what happened previously? Does this change my next decision?

What do we mean by “Markov”?

Definition: the Markov property means that future states depend only on the present state (and not on any preceding states).

If I know Mario’s current position, velocity, acceleration, and the location of all blocks and enemies…

…do I care about what happened previously? Does this change my next decision? **No!**

Markov property

A little more formally…

$$p(r_{t+1}, s_{t+1} | s_0, a_0, r_1, \ldots, a_{t-1}, r_t, s_t, a_t)$$

$$=$$

$$p(r_{t+1}, s_{t+1} | s_t, a_t)$$

Markov property

A little more formally…

$$p(r_{t+1}, s_{t+1} | \cancel{s_0, a_0, r_1, \ldots, a_{t-1}, r_t}, s_t, a_t)$$

$$=$$

$$p(r_{t+1}, s_{t+1} | s_t, a_t)$$

Rewards in sequential decision-making

In sequential decision-making problems, rewards reflect goals

- They specify what the agent needs to achieve, but not how
- Example: Mario gets +1 for increasing score and -1 for dying

How do we evaluate the expected quality of a policy?

Rewards in sequential decision-making

Simple way: cumulative reward over T steps (much like Bandits)

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_T$$

However, this treats all rewards as equal!

- What if we want to favor more immediate results?

Rewards in sequential decision-making

Simple way: cumulative reward over T steps

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_T$$

However, this treats all rewards as equal!

- What if we want to favor more immediate results?

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t} r_T$$

Rewards in sequential decision-making

Simple

However

- What i

The discount factor γ controls how "short-sighted" our policy is.

A smaller gamma means short-term rewards are favored over long-term rewards.

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t} r_T$$

Value functions

Previously we discussed action-values...

$$Q(a) = \mathbb{E}[r_t | a_t = a]$$

But in MDPs the value depends on the state and action and policy.

- In the Bandit problems our policy was greedy or ϵ-greedy, but that is not the case here...

Value functions

Definition: the action-value function of an MDP is the expected return starting from state s, taking action a, and following policy π for all subsequent states

$$Q_\pi(s, a) = \mathbb{E}[G_t | s_t = s, a_t = a]$$

Definition: the state-value function of an MDP is the expected return starting from state s and following policy π

$$V_\pi(s) = \mathbb{E}[G_t | s_t = s]$$

Why are value functions useful?

Recall: value functions measure the “goodness” of taking a particular action from a particular state

- With respect to a given policy…

An optimal policy can be found if we know either the *optimal* action-value or state-value functions.

Finding an optimal policy

If we know the optimal action-value function $Q^*(s, a)$…

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s, a) \\ 0, & \text{otherwise} \end{cases}$$

…we immediately have the optimal policy. We simply follow it.

Finding an optimal policy

If we know the optimal state-value function $V^*(s)$...

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \text{argmax}_{a\in\mathcal{A}}[\sum_{s',r} p(s',r|s,a)(r + \gamma V^*(s'))] \\ 0, & \text{otherwise} \end{cases}$$

...we need access to the state transition function (defined in the MDP earlier as T) to do one-step ahead lookup

- Take the action which leads us to the state with highest value

Next time...

- How do we find optimal policies when both the state transition function and reward function are known?
  - This is planning!

- How do we find optimal policies when neither the state transition function nor the reward function are known?
  - This is learning!

Take-aways:

Multi-Armed Bandit

- Non-sequential / stateless
  - Each action is independent of previous actions
- Reward distribution is fixed
  - Reward depends only on arm
- Finding policies…
  - ϵ-greedy, UCB, Thompson, …

Markov Decision Process

- Sequential / stateful
  - Action depends on the current state (Markovian)
- Reward distribution changes
  - Reward depends on state/action
- Finding policies…
  - Next time!