




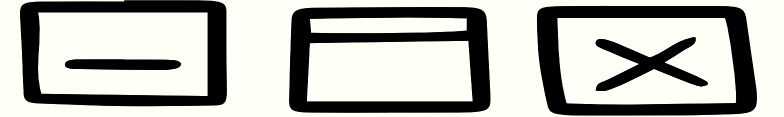
Reinforcement Learning and Autonomous Systems (AI 4102)



Lecture 6 (31/08/2023)
Lecture 7 (01/09/2023)
Lecture 8 (04/09/2023)

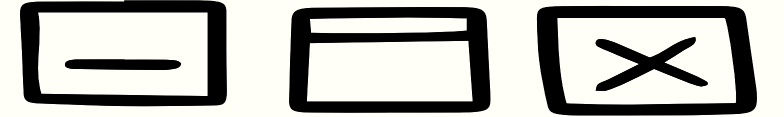
Instructor: Gourav Saha

Lecture Content



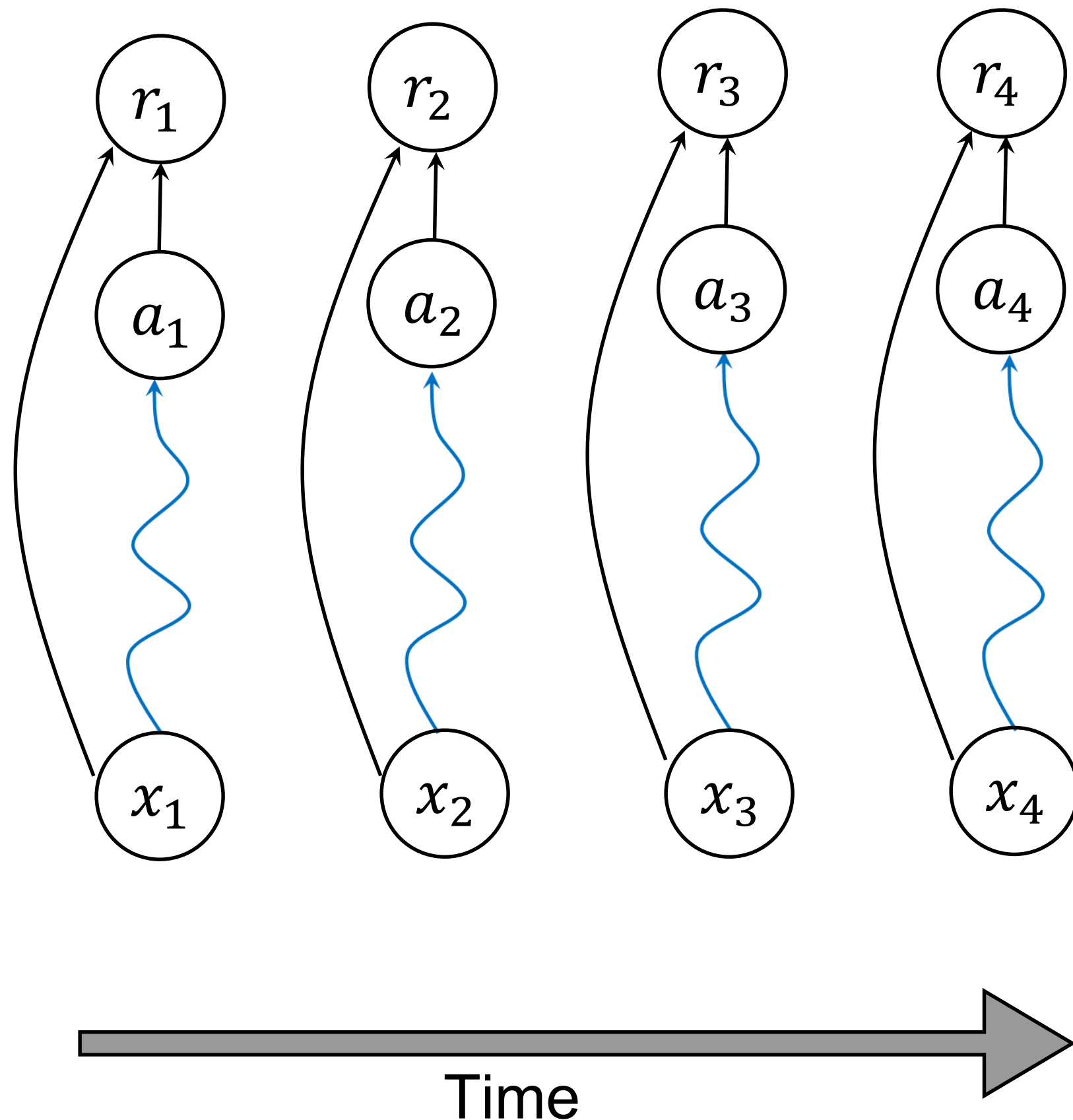
- Contextual Bandit setup.
- Contextual Bandit vs Supervised Learning.
- Mathematical Notations and Concepts.
- A Naïve Approach of Solving Contextual Bandits.
- Linear Bandits.
- Policy Gradient.

Lecture Content



- Contextual Bandit setup.
 - Understanding Context and Context Space.
 - Applications of Contextual Bandits.
- Contextual Bandit vs Supervised Learning.
- Mathematical Notations and Concepts.
- A Naïve Approach of Solving Contextual Bandits.
- Linear Bandits.
- Policy Gradient.

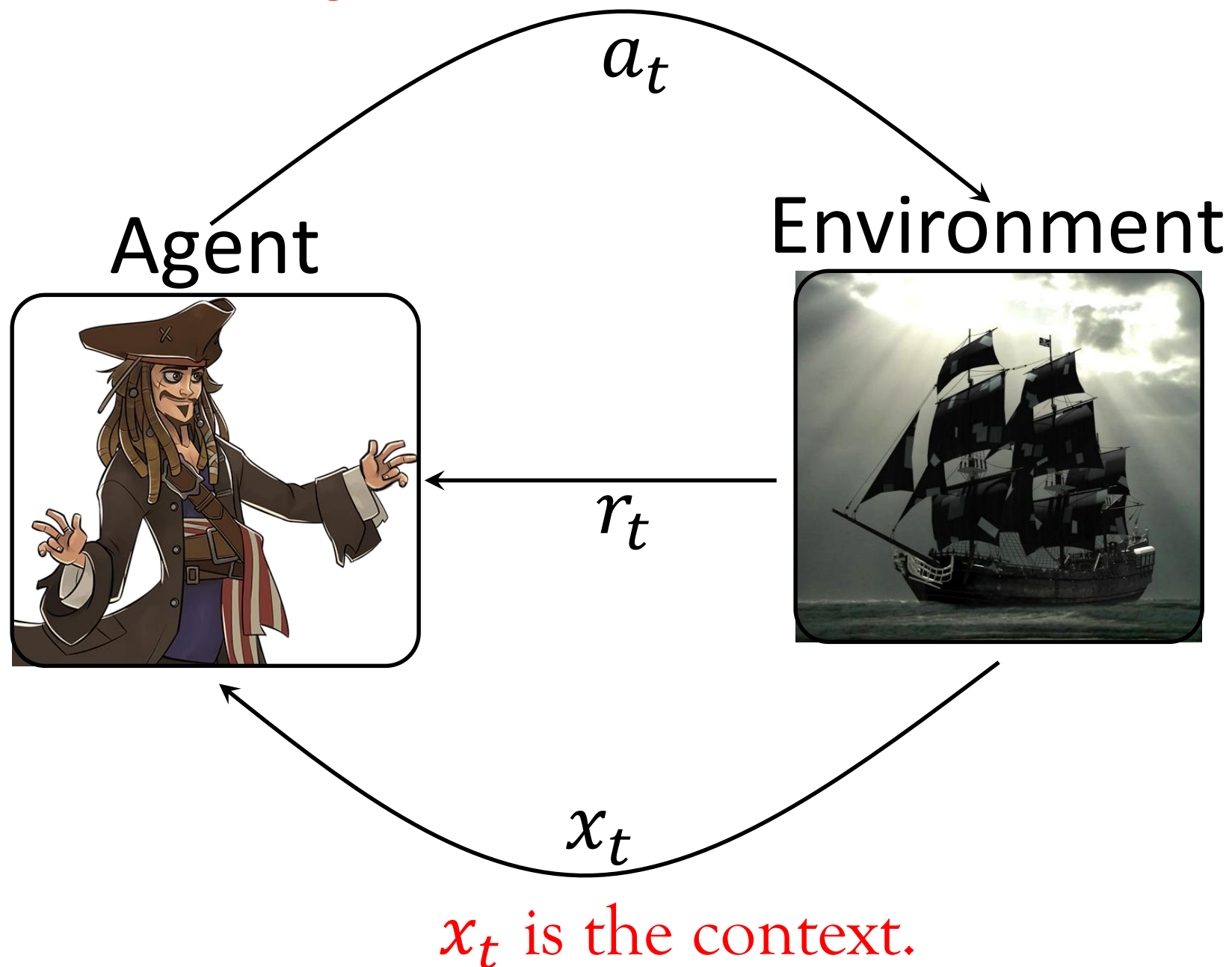
Contextual Bandit Setup



- Recall the Probabilistic Graphical Model (shown in the left) of Contextual Bandit from Lecture 2 and 3 slides.
- The difference between Contextual Bandit and Multi-Armed Bandit is:
 - In Contextual Bandit the states \mathbf{x}_t could be different at different time.
 - In Multi-Armed Bandit the states \mathbf{x}_t is same at all time.
- NOTE: “States” and “context” are analogous in Contextual Bandits. Will be using the word “context” because it is more conventional for Contextual Bandits.

Contextual Bandit Setup

IMPORTANT: Action a_t does not change the environment.

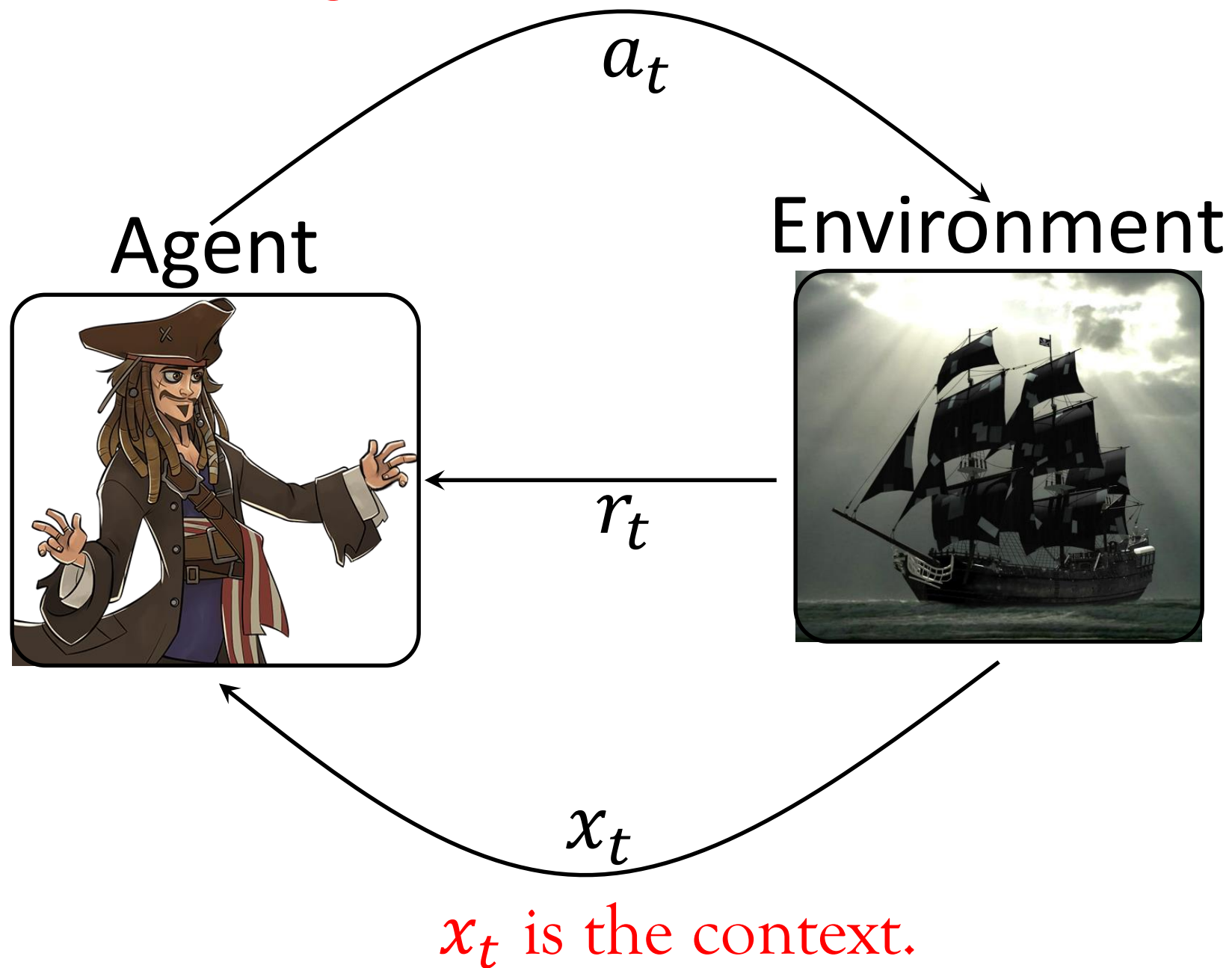


Contextual Bandit setup is as follows:

- The **set of actions** is $\mathcal{A} = \{1, 2, \dots, A\}$. We have $a_t \in \mathcal{A}$ for all t .
- The **set of contexts** is \mathcal{C} . We have $x_t \in \mathcal{C}$ for all t .
 - Set of contexts is also called **context space**.
- The context x_t is generated by the environment by sampling for a distribution $F_{\mathcal{C}}$. The **agent does not know $F_{\mathcal{C}}$** .
- **Context-Action pair (x, a)** has the reward distribution $P_{x,a}$. The agent **does not know $P_{x,a}$** for any $(x, a) \in \mathcal{C} \times \mathcal{A}$.

Contextual Bandit Setup

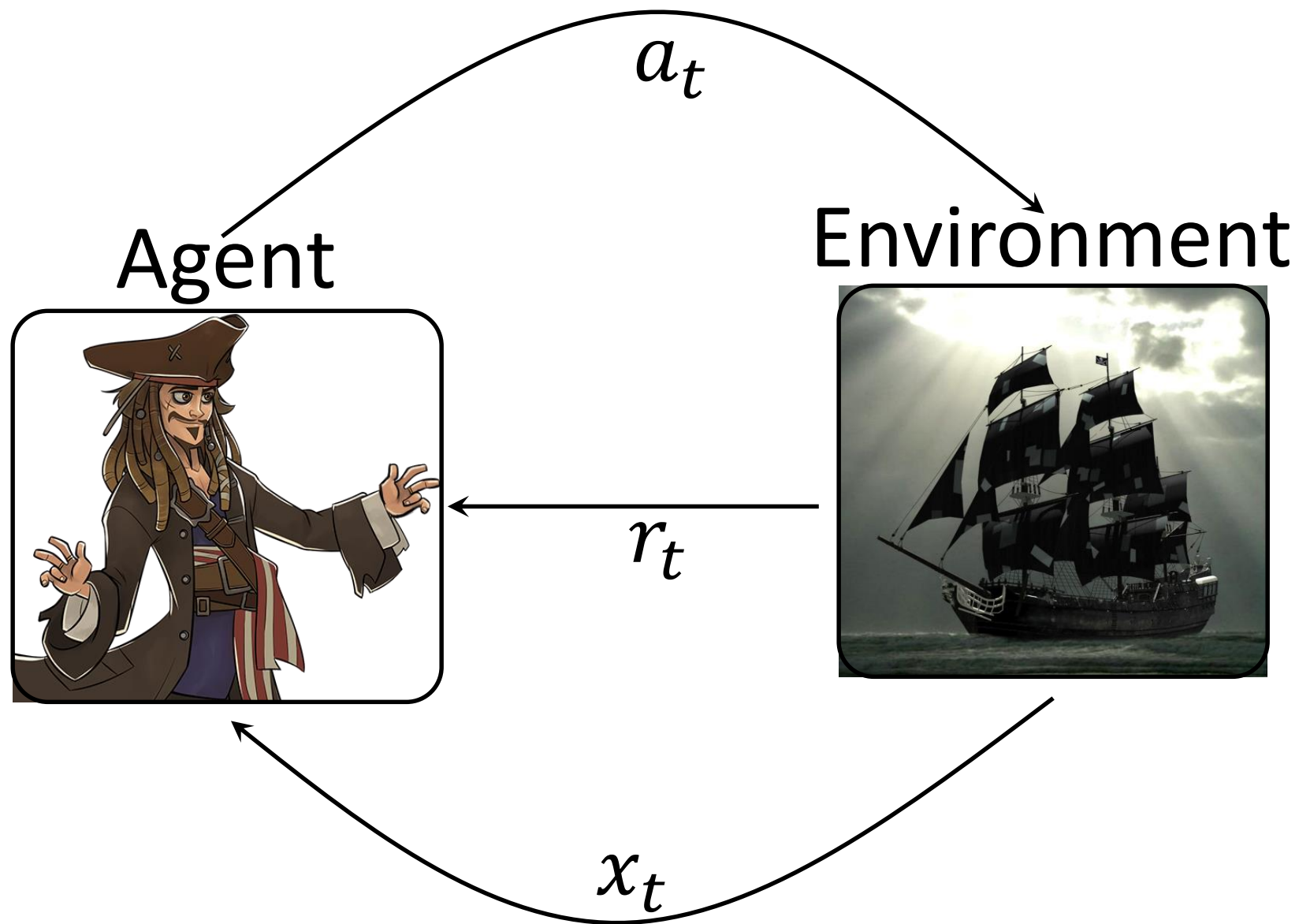
IMPORTANT: Action a_t does not change the environment.



Contextual Bandit setup is as follows:

- In time slot t :
 - Step 1: The environment samples a context $x_t \sim F_C$.
 - Step 2: The agent observes the context x_t .
 - Step 3: The agent decides its action a_t .
 - Step 4: The agent receives a reward $r_t \sim P_{x_t, a_t}$ that is generated by the environment.

Contextual Bandit Setup

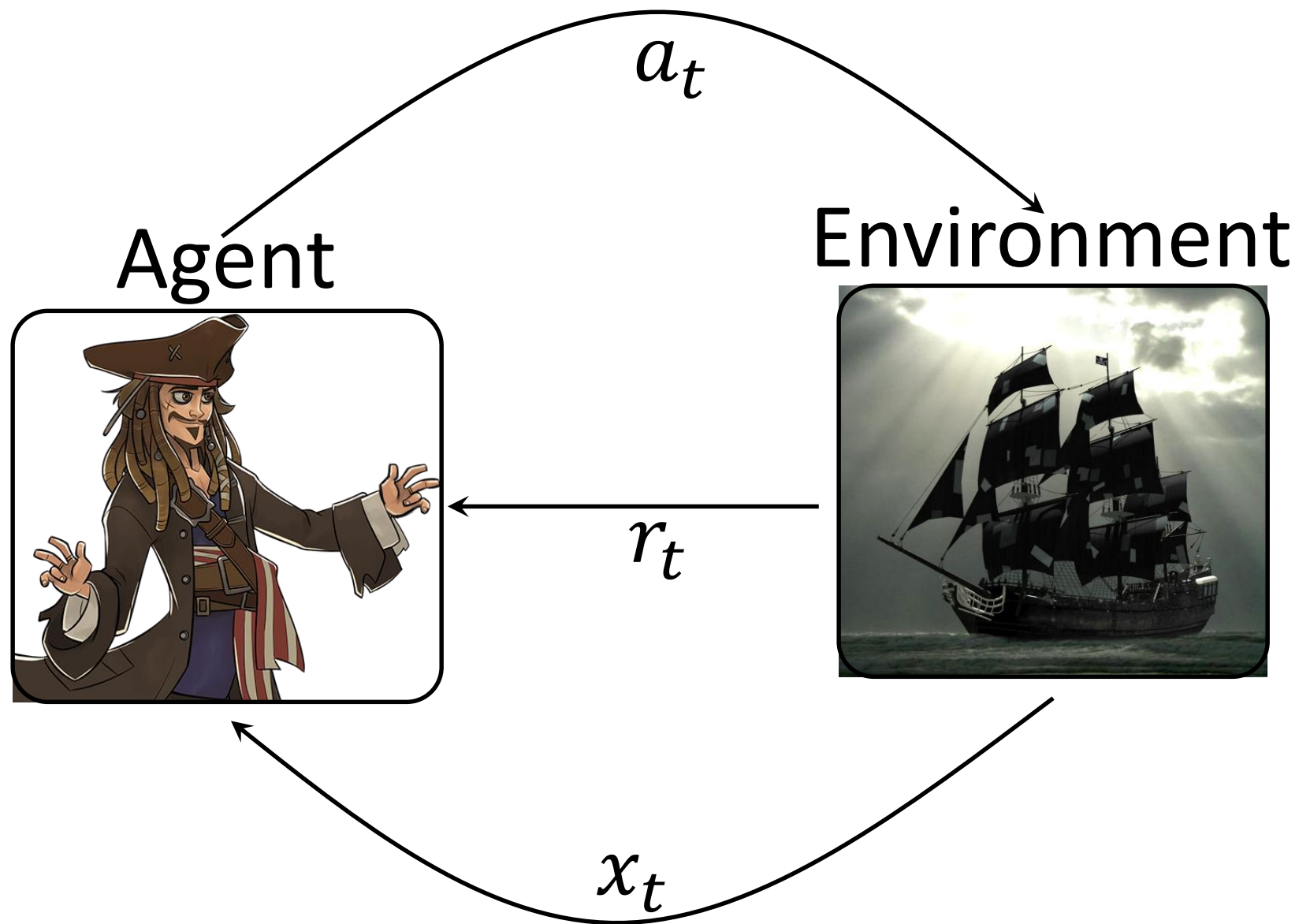


IMPORTANT: The agent **does not** get to see the reward of those actions that it did not take.

Contextual Bandit setup is as follows:

- In time slot t :
 - Step 1: The environment samples a context $x_t \sim F_C$.
 - Step 2: The agent observes the context x_t .
 - Step 3: The agent decides its action a_t .
 - Step 4: The agent receives a reward $r_t \sim P_{x_t, a_t}$ that is generated by the environment.

Contextual Bandit Setup



Contextual Bandit setup is as follows:

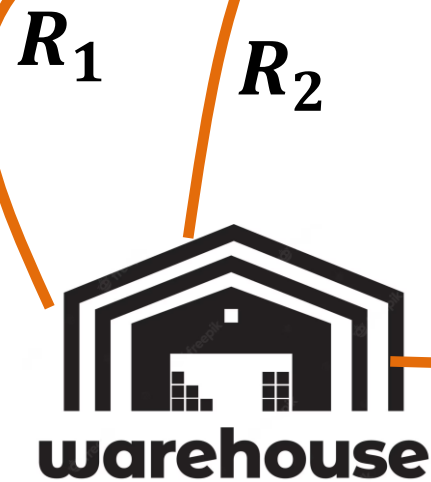
- In time slot t :
 - Step 1: The environment samples a context $x_t \sim F_{\mathcal{C}}$.
 - Step 2: The agent observes the context x_t .
 - Step 3: The agent decides its action a_t .
 - Step 4: The agent receives a reward $r_t \sim P_{x_t, a_t}$ that is generated by the environment.
- The goal of the agent is to maximize the expected total reward,

$$\mathbb{E} \left[\sum_{\tau=1}^t r_{\tau} \right]$$

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



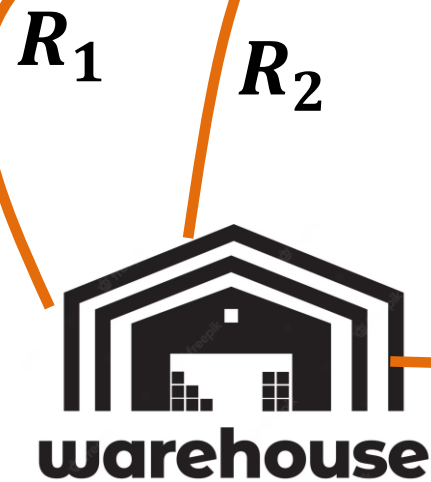
Understanding Context and Context Space

- Let's take an example to understand the difference between context and context space.
 - We use the warehouse routing example.
- Recall that in the contextual bandit version of warehouse routing, there were two contexts:
 1. Average travel time of vehicles dispatched in the **previous hour**.
 2. Average travel time of vehicles dispatched in the **current hour of the previous day**.

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



R_3

Understanding Context and Context Space

- For time slot t :
 - Let $x_{t,1,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **previous hour**.
 - Let $x_{t,2,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **current hour of the previous day**.
- The context x_t at time t is a **vector**. What is x_t for this example?

Answer:

$$x_t = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ x_{t,1,2} \\ x_{t,2,2} \\ x_{t,1,3} \\ x_{t,2,3} \end{bmatrix}$$

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



R_1

R_2

R_3



warehouse

Understanding Context and Context Space

- For time slot t :
 - Let $x_{t,1,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **previous hour**.
 - Let $x_{t,2,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **current hour of the previous day**.

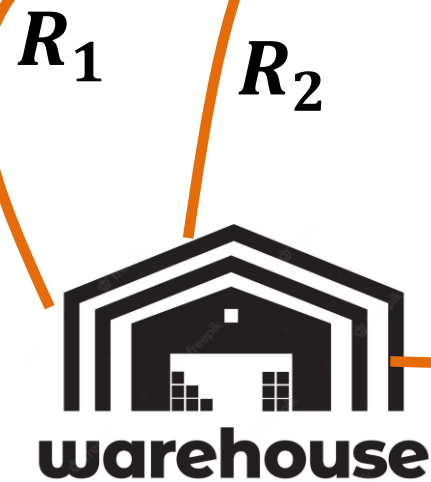
Discrete Context space:

- Let $x_{t,i,a} \in \{L, M, H\}$, where $a \in \{1,2,3\}$ and $i \in \{1,2\}$. L , M , and H implied that the travel time is low, medium, or high.
- What is the context space?

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



Understanding Context and Context Space

- For time slot t :
- Let $x_{t,1,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **previous hour**.
 - Let $x_{t,2,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **current hour of the previous day**.

Discrete Context space:

$$x_t = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ x_{t,1,2} \\ x_{t,2,2} \\ x_{t,1,3} \\ x_{t,2,3} \end{bmatrix}$$

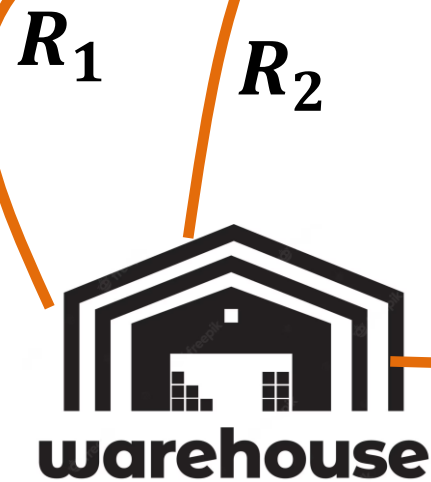
Each an every element of x_t can assume values in the set $\{L, M, H\}$. There are six elements. Hence, the context space of x_t is the cartesian product:

$$\underbrace{\{L, M, H\} \times \{L, M, H\} \times \cdots \times \{L, M, H\}}_{6 \text{ times}}$$

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



Understanding Context and Context Space

- For time slot t :
- Let $x_{t,1,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **previous hour**.
 - Let $x_{t,2,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **current hour of the previous day**.

Discrete Context space:

$$x_t = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ x_{t,1,2} \\ x_{t,2,2} \\ x_{t,1,3} \\ x_{t,2,3} \end{bmatrix}$$

Each and every element of x_t can assume values in the set $\{L, M, H\}$. There are six elements. Hence, the context space of x_t is the cartesian product: $\{L, M, H\}^6$ **Equivalent representation of the cartesian product.**

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



R_1

R_2

R_3



warehouse

Understanding Context and Context Space

- For time slot t :
 - Let $x_{t,1,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **previous hour**.
 - Let $x_{t,2,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **current hour of the previous day**.

Continuous Context space:

- Let $x_{t,i,a} \in [25,75]$ minutes, where $a \in \{1,2,3\}$ and $i \in \{1,2\}$.
- What is the context space?

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



R_1

R_2

R_3



warehouse

Understanding Context and Context Space

- For time slot t :
- Let $x_{t,1,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **previous hour**.
 - Let $x_{t,2,a}$, where $a \in \{1,2,3\}$, denote the average travel time of vehicles dispatched through **route a** in the **current hour of the previous day**.

Continuous Context space:

Similar to the discrete context space, the context space here is

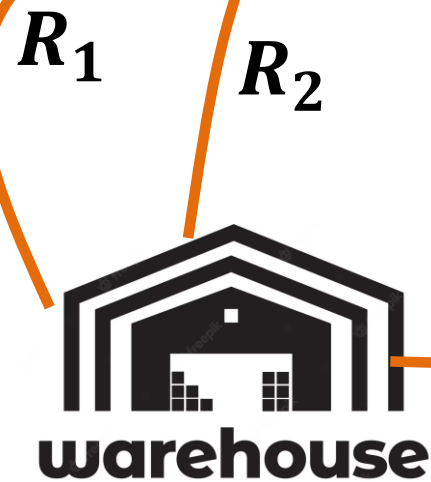
$$[25, 75] \times [25, 75] \times \cdots \times [25, 75]$$

which can be equivalently written as $[25, 75]^6$.

Contextual Bandit Setup

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



This part has nothing to do with the heading “Understanding Context and Context Space”. Just some extra information about this example.

Understanding Context and Context Space

- Lets use the warehouse routing example to understand context space.
- Recall that in the contextual bandit version of warehouse routing, there were two contexts:
 1. Average travel time of vehicles dispatched in the **previous hour**.
 2. Average travel time of vehicles dispatched in the **current hour of the previous day**.
- What other contexts are possible for this example?
 - **Image** of google maps for all three routes with color coding for various level of traffic.
 - **Review/tweets** about the road condition in the past few hours.

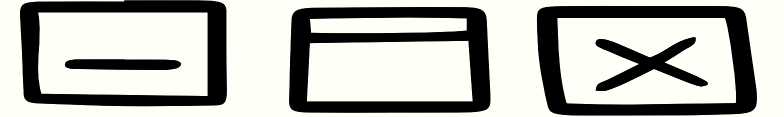
Contextual Bandit Setup

Applications of Contextual Bandit:

- **Online advertising:** Which advertisement to show to get more click through rate?
- **Recommendation systems (Netflix):** Which movies to recommend to maximize user satisfaction?
- **Clinical trials:** Which drug or what dose of drug to administer to cure a patient?
- **Network routing:** In wireless/road network, which path to take in order to minimize the time to go from source to destination?
- **Dynamic Pricing:** What price to set in online market place in order to maximize revenue?

Discuss about: 1) Actions, 2) Contexts, 3) Rewards, and 4) Source of uncertainty.

Lecture Content



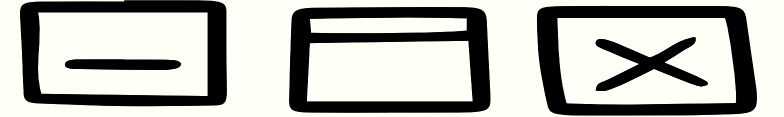
- Contextual Bandit setup.
- Contextual Bandit vs Supervised Learning.
- Mathematical Notations and Concepts.
- A Naïve Approach of Solving Contextual Bandits.
- Linear Bandits.
- Policy Gradient.

Contextual Bandit vs Supervised Learning



- Different reward scenario:
 - Binary.
 - Continuous (noise environment).
- The idea of getting rewards for only the action taken.

Lecture Content



- Contextual Bandit setup.
- Contextual Bandit vs Supervised Learning.
- Mathematical Notations and Concepts.
- A Naïve Approach of Solving Contextual Bandits.
- Linear Bandits.
- Policy Gradient.

Mathematical Notations and Concepts

- $q_*(x, a)$ denote the **context-action-value**. It is the **true** expected value of reward from context-action pair (x, a) ,

$$q_*(x, a) = \mathbb{E}[r_t \mid x_t = x, a_t = a]$$

- The optimal action $a^*(x)$ is,

$$a^*(x) = \operatorname{argmax}_{a \in \mathcal{A}} q_*(x, a)$$

NOTE: Unlike MAB, the **optimal action is a function of the context**.

- To measure the performance of a policy are: **1) the total expected reward**

$$\mathbb{E} \left[\sum_{\tau=1}^t r_{\tau} \right]$$

Mathematical Notations and Concepts

- $q_*(x, a)$ denote the **context-action-value**. It is the **true** expected value of reward from context-action pair (x, a) ,

$$q_*(x, a) = \mathbb{E}[r_t | x_t = x, a_t = a]$$

- The optimal action $a^*(x)$ is,

$$a^*(x) = \operatorname{argmax}_{a \in \mathcal{A}} q_*(x, a)$$

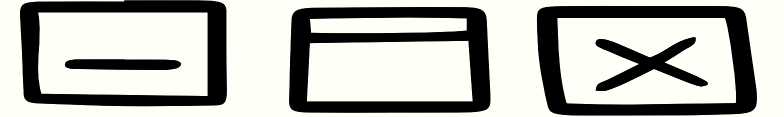
NOTE: Unlike MAB, the **optimal action is a function of the context**.

- To measure the performance of a policy are: **2) regret**

$$L_t = \mathbb{E} \left[\sum_{\tau=1}^t r_{\tau}^{a^*(x_{\tau})} - \sum_{\tau=1}^t r_{\tau} \right]$$

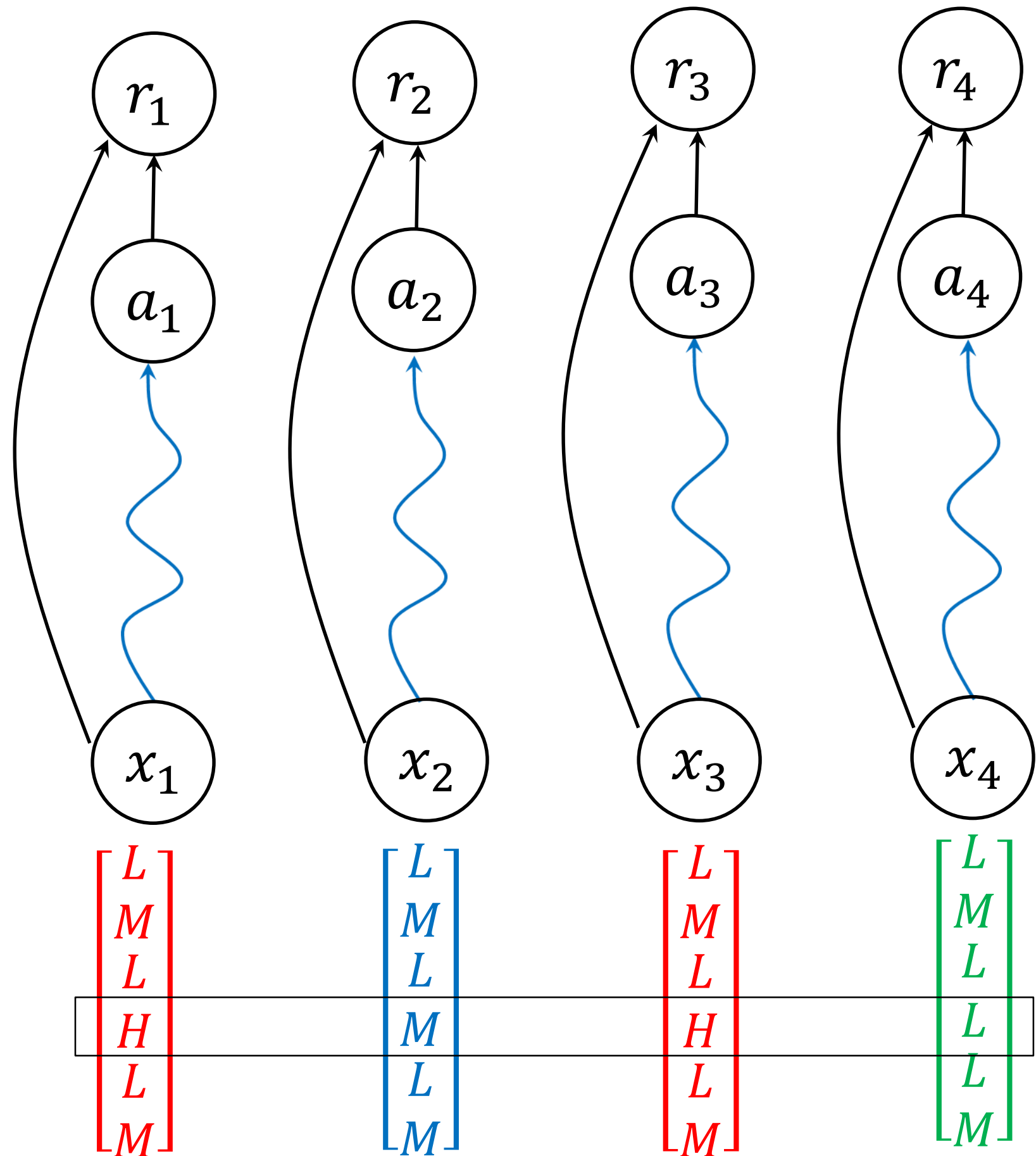
where $r_{\tau}^{a^*(x_{\tau})}$ is the reward earned by the policy that knows the optimal action $a^*(x_{\tau})$ at time τ .

Lecture Content



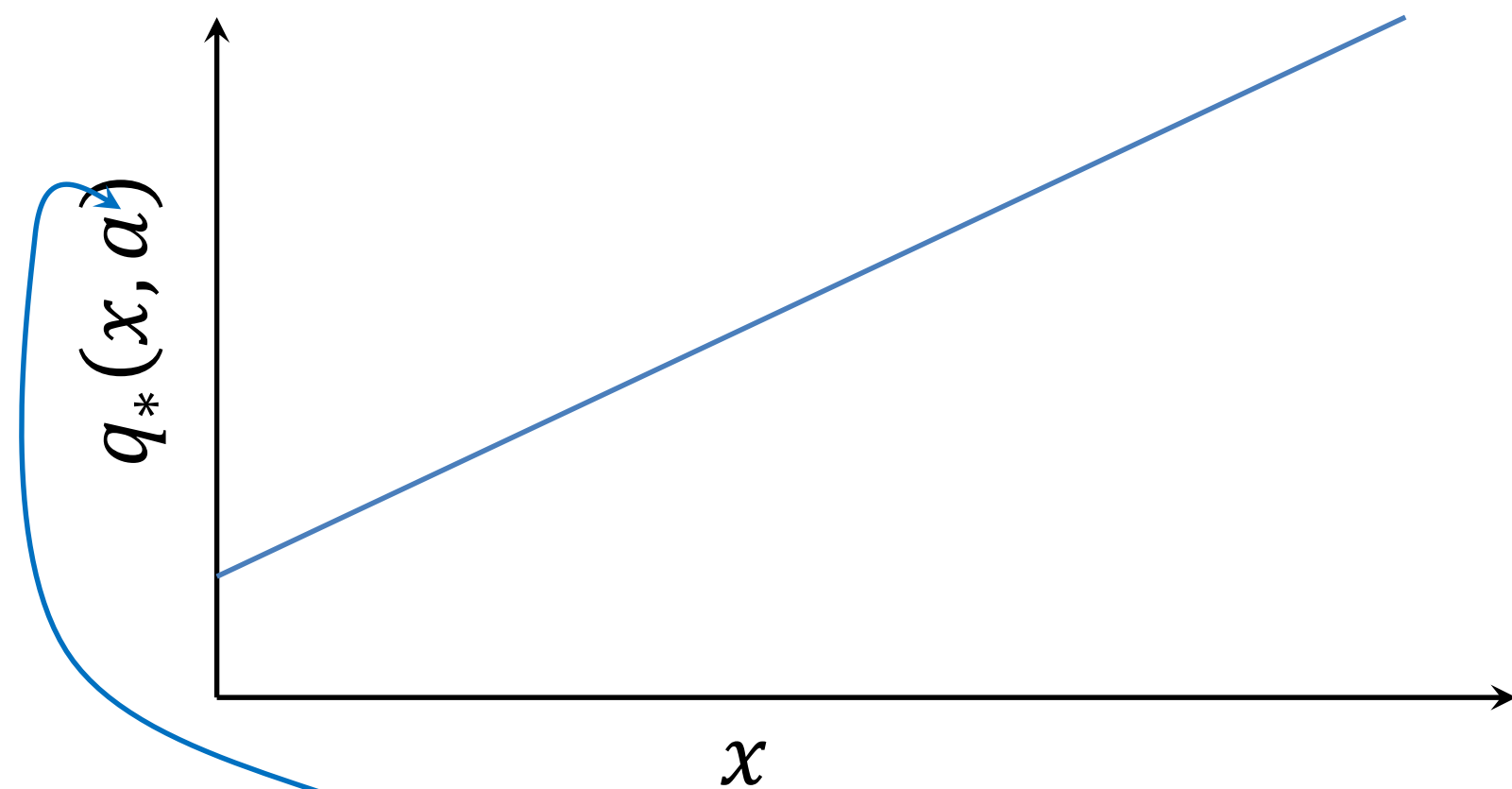
- Contextual Bandit setup.
- Contextual Bandit vs Supervised Learning.
- Mathematical Notations and Concepts.
- A Naïve Approach of Solving Contextual Bandits.
- Linear Bandits.
- Policy Gradient.

A Naïve Approach of Solving Contextual Bandits



- A naïve approach is that **for each context $x \in \mathcal{C}$, we solve a MAB problem.**
- Why is this Naïve?
 - Consider the warehouse routing (in the left).
- Because it does not consider the **correlation of rewards across the contexts.**
- **Some contexts may not have enough samples...
Some similarity with transfer learning.**

A Naïve Approach of Solving Contextual Bandits



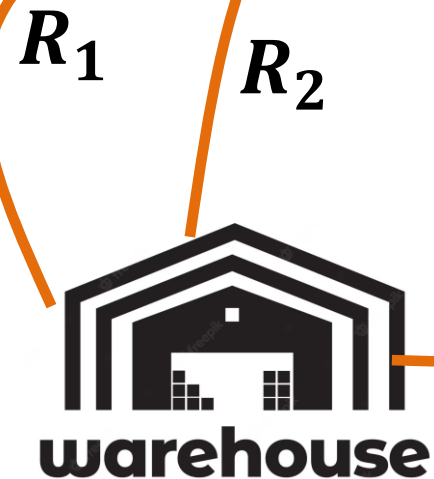
- Considering a scalar context x for better visualization.
- We are fixing action a .

- A naïve approach is that for each context $x \in \mathcal{C}$, we solve a MAB problem.
- Why is this Naïve?
 - Consider the warehouse routing (in the left).
- Because it does not consider the correlation of rewards across the actions.

A Naïve Approach of Solving Contextual Bandits

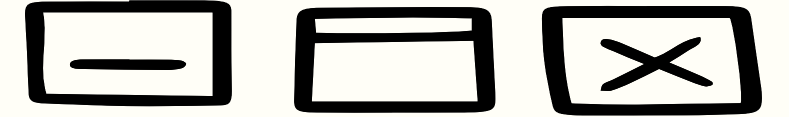
NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



- A naïve approach is that **for each context $x \in \mathcal{C}$, we solve a MAB problem.**
 - Why is this Naïve?
 - Consider the warehouse routing (in the left).
 - SIDE NOTE: In many situations, there are **correlation of rewards across the actions.**
 - Example 1: Travel time of route 1 and route 2 are correlated because they merge.
 - Example 2: When designing ads, an **action is a vector**, e.x. [image placement, font size]. Two ads with similar image placement but different font size may have some correlation in there **click-through rate.**
- We will not talk a lot about this case.**

Lecture Content



- Contextual Bandit setup.
- Contextual Bandit vs Supervised Learning.
- Mathematical Notations and Concepts.
- A Naïve Approach of Solving Contextual Bandits.
- Linear Bandits.
 - What are linear bandits?
 - ϵ -greedy Policy.
 - Upper Confidence Bound Policy.
- Policy Gradient.

Linear Bandits

Linear Bandit is a special case of Contextual Bandit where we **assume** that,

$$q_*(x, a) = x^T \theta_*^a + b_*^a \quad (\text{L.1})$$

where if $x \in \mathbb{R}^d$ then $\theta_*^a \in \mathbb{R}^d$. Also, $b_*^a \in \mathbb{R}$. For the warehouse routing example, $d = 6$.

Consequently, the reward at time t is,

$$r_t = x_t^T \theta_*^{a_t} + b_*^{a_t} + \eta_t \quad (\text{L.2})$$

where, x_t and a_t are the context, and the chosen action at time t respectively.

η_t is a **zero-mean** random variable sampled from distribution F_η .

And, as usual, x_t is sampled from distribution $F_{\mathcal{C}}$.

Linear Bandits

Linear Bandit is a special case of Contextual Bandit where we **assume** that,

$$q_*(x, a) = x^T \theta_*^a + b_*^a \quad (\text{L.1})$$

where if $x \in \mathbb{R}^d$ then $\theta_*^a \in \mathbb{R}^d$. Also, $b_*^a \in \mathbb{R}$. For the warehouse routing example, $d = 6$.

Consequently, the reward at time t is,

$$r_t = x_t^T \theta_*^{a_t} + b_*^{a_t} + \eta_t \quad (\text{L.2})$$

where, x_t and a_t are the context, and the chosen action at time t respectively.

η_t is a **zero-mean** random variable sampled from distribution F_η .

And, as usual, x_t is sampled from distribution $F_{\mathcal{C}}$.

The agent does not know θ_*^a , b_*^a , F_η , and $F_{\mathcal{C}}$.

Linear Bandits

Linear Bandit is a special case of Contextual Bandit where we **assume** that,

$$q_*(x, a) = x^T \theta_*^a + b_*^a \quad (\text{L.1})$$

- How do we know that $q_*(x, a)$ is of the above form?

Answer: We don't! It is an assumption just like in any ML problem.

Linear Bandits

Linear Bandit is a special case of Contextual Bandit where we **assume** that,

$$q_*(x, a) = x^T \theta_*^a + b_*^a \quad (\text{L.1})$$

- Suppose we find out after trying that $q_*(x, a)$ does not follow the above form. Can we still use linear bandits?

Answer: To some extent... We may be able to find a map $z_a = \psi_a(x)$ such that,

$$q_*(x, a) = z_a^T \theta_*^a \quad (\text{L.3})$$

- Finding ψ_a depends on domain expertise. This is similar to “**feature engineering**”. We can in fact call z_a as the **features** of action a .
- Just to stress, the mapping ψ_a can be different for different actions. Hence, the length of the vector z_a can be different for different actions.
- Equation (L.3) **assumes that the last element of z_a is 1**. By doing so we don't have to separately account for the bias parameter b_*^a . This is done for notational simplicity.

Linear Bandits

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



R_1

R_2

R_3



warehouse

- Each route (routes are the actions) is associated with two average travel time (refer [this slide](#) to refresh your memory). These two travel times form the feature for the corresponding route.

$$z_a = \psi_a(x)$$

$$q_*(x, a) = z_a^T \theta_*$$

Example (Finding ψ_a for the warehouse example):

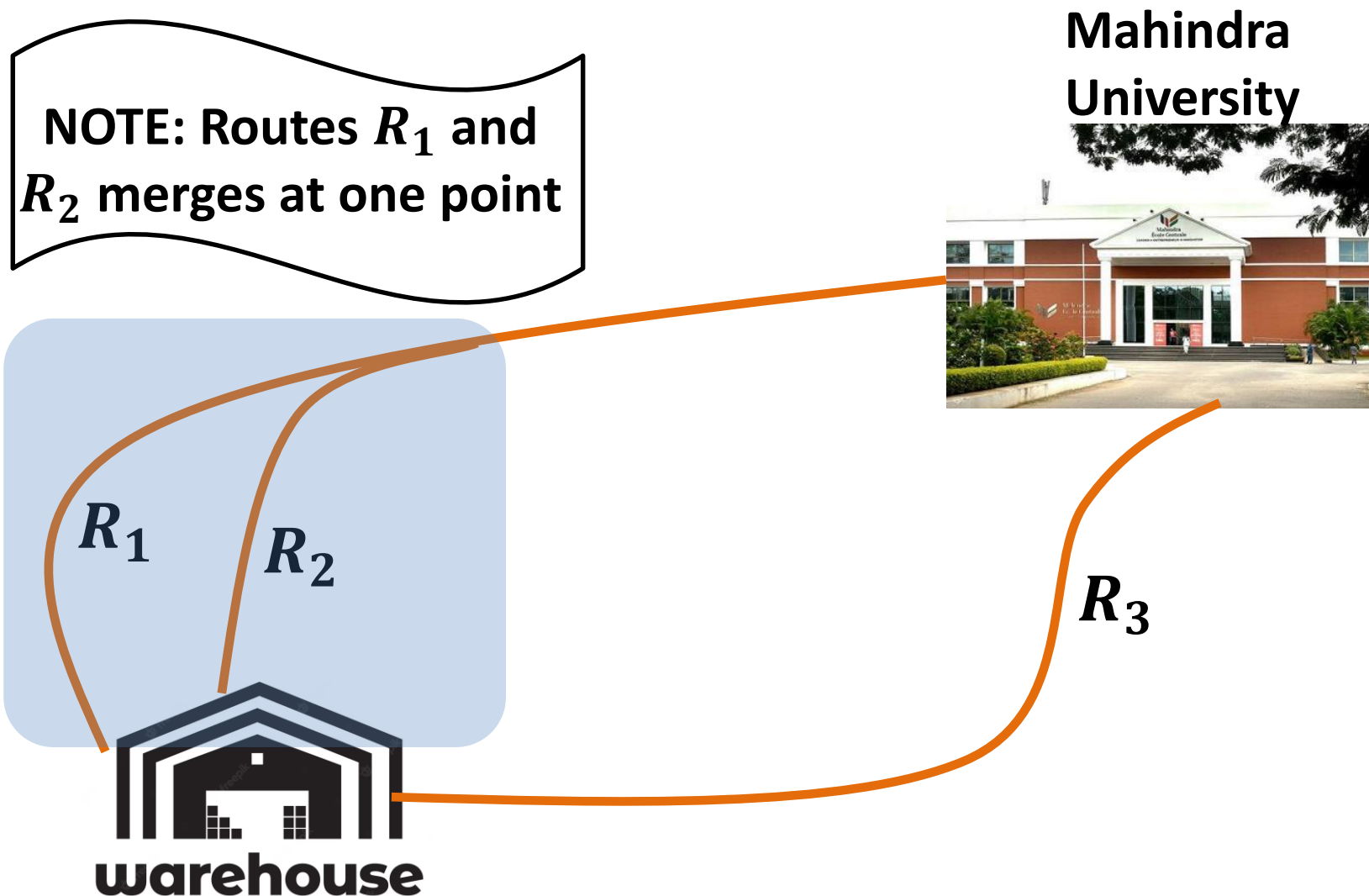
- The context is,

$$x_t = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ x_{t,1,2} \\ x_{t,2,2} \\ x_{t,1,3} \\ x_{t,2,3} \end{bmatrix}$$

- What are $z_{t,1}$, $z_{t,2}$, and $z_{t,3}$? (1st possibility)

$$z_{t,1} = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ 1 \end{bmatrix} \quad z_{t,2} = \begin{bmatrix} x_{t,1,2} \\ x_{t,2,2} \\ 1 \end{bmatrix} \quad z_{t,3} = \begin{bmatrix} x_{t,1,3} \\ x_{t,2,3} \\ 1 \end{bmatrix}$$

Linear Bandits



- Since routes R_1 and R_2 are connected (check the highlighted part of the figure above), it is possible that the average travel times of one route can be a feature for the other route.

$$z_a = \psi_a(x)$$

$$q_*(x, a) = z_a^T \theta_*^a$$

Example (Finding ψ_a for the warehouse example):

- The context is,

$$x_t = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ x_{t,1,2} \\ x_{t,2,2} \\ x_{t,1,3} \\ x_{t,2,3} \end{bmatrix}$$

- What are $z_{t,1}$, $z_{t,2}$, and $z_{t,3}$? (2nd possibility)

$$z_{t,1} = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ x_{t,1,2} \\ x_{t,2,2} \\ 1 \end{bmatrix} \quad z_{t,2} = \begin{bmatrix} x_{t,1,1} \\ x_{t,2,1} \\ x_{t,1,2} \\ x_{t,2,2} \\ 1 \end{bmatrix} \quad z_{t,3} = \begin{bmatrix} x_{t,1,3} \\ x_{t,2,3} \\ 1 \end{bmatrix}$$

Linear Bandits

- Moving forward we assume that for linear bandits, $q_*(x, a)$ is given by (L.3) and not (L.1) because (L.3) is a generalization of (L.1)*. Consequently (L.2) gets modified as,

$$r_t = z_{t,a_t}^T \theta_*^{a_t} + \eta_t \quad (\text{L.4})$$

where $z_{t,a_t} = \psi_{a_t}(x_t)$.

- Recall that for MAB, we had to estimate $q_*(a)$ using sample average which we denoted using $Q_t(a)$. We then used $Q_t(a)$ to design ϵ -greedy and UCB policy.
- Similarly to extend ϵ -greedy and UCB policy to linear bandit case, we need to get an estimate of $q_*(x, a)$. **For linear bandits, estimating $q_*(x, a)$ is equivalent to estimating θ_*^a of (L.3).**

* This is because by setting $z_a = [x \quad 1]^T$ in (L.3), we get back (L.1).

Linear Bandits

- It is obvious from (L.4) that θ_*^a can be estimated using **linear regression** with $z_{t,a}$ as the features and r_t as the target. However, there are two differences compared to supervised learning:
- Unlike supervised learning, the data needs to be collected online and the agent will get to sample $(z_{t,a}, r_t)$ pair only when it selects action a . This point has been repeated in multiple forms throughout all the lectures.
 - (IMPORTANT) We need to keep updating our estimate of θ_*^a . Hence, we need a **recursive implementation of linear regression** (similar to recursive estimate of $Q_t(a)$ for MAB). This will be the topic of discussion of the next **five** slides.

Linear Bandits

Recursive Implementation of Linear Regression (Standalone Topic)

NOTE: For the next **five** slides, we forget about linear bandits and focus on the simple linear regression. Also, the notations used in the next few slides are standalone to these slides only.

➤ Consider the feature matrix $X \in \mathbb{R}^{N \times M}$, the target vector $Y \in \mathbb{R}^N$ for linear regression. Here, N is the number of samples and M is the number of features.

➤ The objective is to find a parameters $\theta \in \mathbb{R}^M$ that minimizes the mean square error,

$$\frac{1}{N} (Y - X\theta)^T (Y - X\theta)$$

➤ The θ that minimizes the mean square error is,

$$\theta = (X^T X)^{-1} (X^T Y)$$

(L.5)

Linear Bandits

Recursive Implementation of Linear Regression (Standalone Topic)

- Now suppose, we got an additional sample (x, y) where $x \in \mathbb{R}^M$ and $y \in \mathbb{R}$. Hence, our modified feature matrix and target vector are,

$$\bar{X} = \begin{bmatrix} X \\ x^T \end{bmatrix} \quad \bar{Y} = \begin{bmatrix} Y \\ y \end{bmatrix}$$

- Therefore, the updated parameter that minimizes the mean square error is,

$$\bar{\theta} = (\bar{X}^T \bar{X})^{-1} (\bar{X}^T \bar{Y}) \tag{L.6}$$

- If we have to update the parameters using (L.6) we have to remember the history of features and target. This leads to space complexity of $\mathcal{O}(NM)$. Also, the time complexity to get $\bar{X}^T \bar{X}$ and $\bar{X}^T \bar{Y}$ is $\mathcal{O}(N^2 M^2)$. Both time and space complexity is huge when N is large.
- We will now find a more efficient way to compute (L.6).

Linear Bandits

Recursive Implementation of Linear Regression (Standalone Topic)

- Now suppose, we got an additional sample (x, y) where $x \in \mathbb{R}^M$ and $y \in \mathbb{R}$. Hence, our modified feature matrix and target vector are,

$$\bar{X} = \begin{bmatrix} X \\ x^T \end{bmatrix} \quad \bar{Y} = \begin{bmatrix} Y \\ y \end{bmatrix}$$

- Therefore, the updated parameter that minimizes the mean square error is,

$$\bar{\theta} = (\bar{X}^T \bar{X})^{-1} (\bar{X}^T \bar{Y}) \tag{L.6}$$

- In (L.6),

$$\bar{X}^T \bar{X} = \begin{bmatrix} X \\ x^T \end{bmatrix}^T \begin{bmatrix} X \\ x^T \end{bmatrix} = \begin{bmatrix} X^T & x \end{bmatrix} \begin{bmatrix} X \\ x^T \end{bmatrix} = X^T X + x x^T \tag{L.7}$$

$$\bar{X}^T \bar{Y} = \begin{bmatrix} X \\ x^T \end{bmatrix}^T \begin{bmatrix} Y \\ y \end{bmatrix} = \begin{bmatrix} X^T & x \end{bmatrix} \begin{bmatrix} Y \\ y \end{bmatrix} = X^T Y + y x \tag{L.8}$$

Linear Bandits

Recursive Implementation of Linear Regression (Standalone Topic)

- We calculated $X^T X$ and $X^T Y$ in (L.5). If we saved, our calculation as,

$$A = X^T X$$

$$B = X^T Y$$

Then, $\bar{\theta}$ can be computed as follows,

$$\bar{A} = A + x x^T \tag{L.9}$$

$$\bar{B} = B + y x \tag{L.10}$$

$$\bar{\theta} = \bar{A}^{-1} \bar{B} \tag{L.11}$$

- We just have to save matrix A and B . Hence, the space complexity is $\mathcal{O}(M^2)$. Computing \bar{A} and \bar{B} using (L.9) and (L.10) has a time complexity of $\mathcal{O}(M^2)$. Note that both **time and space complexity is independent of the number of samples**. Hence, a massive upgrade on (L.6).

Linear Bandits

Recursive Implementation of Linear Regression (Standalone Topic)

- To summarize: Initialize matrix $A \in \mathbb{R}^{M \times M}$ and $B \in \mathbb{R}^M$ such that all their elements are **zero**. Then, when a new sample (x, y) comes, upgrade the parameter θ as follows,

$$A \leftarrow A + xx^T$$

$$B \leftarrow B + yx$$

$$\theta \leftarrow A^{-1}B$$

ϵ – Greedy Policy

➤ At time t :

1. Observe the context x_t . Compute $z_{t,a} = \psi_a(x_t)$ for all $a \in \mathcal{A}$.

2. Sample a random variable v between 0 to 1 from a uniform distribution.

3. Set $\hat{\theta}_a = A_a^{-1} b_a$ for all $a \in \mathcal{A}$.

4. If $v \leq \epsilon$:

a_t is chosen uniformly at random from \mathcal{A} .

Else:

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} z_{t,a}^T \hat{\theta}_a$$

3. Based on the action chosen, the agent will receive a reward $r_t \sim P_{x_t, a_t}$ from the environment.

4. Update $A_{a_t} \leftarrow A_{a_t} + z_{t,a_t} z_{t,a_t}^T$
 $b_{a_t} \leftarrow b_{a_t} + r_t z_{t,a_t}$

This matrix may sometime be non-invertible (especially when the number of rows is less than number of columns). In that case, don't update $\hat{\theta}_a$.

This is a “value function” based policy.

Upper Confidence Bound Policy

➤ At time t :

1. Observe the context x_t . Compute $z_{t,a} = \psi_a(x_t)$ for all $a \in \mathcal{A}$.

2. Sample a random variable v between 0 to 1 from a uniform distribution.

3. Set $\hat{\theta}_a = A_a^{-1} b_a$ for all $a \in \mathcal{A}$.

4. Set $a_t = \operatorname{argmax}_{a \in \mathcal{A}} z_{t,a}^T \hat{\theta}_a + \epsilon \sqrt{z_{t,a_t}^T A_a^{-1} z_{t,a_t}}$

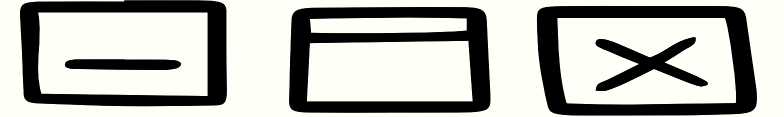
This matrix may sometime be non-invertible (especially when the number of rows is less than number of columns). In that case, don't update $\hat{\theta}_a$.

3. Based on the action chosen, the agent will receive a reward $r_t \sim P_{x_t, a_t}$ from the environment.

4. Update $A_{a_t} \leftarrow A_{a_t} + z_{t,a_t} z_{t,a_t}^T$
 $b_{a_t} \leftarrow b_{a_t} + r_t z_{t,a_t}$

This is a “value function” based policy.

Lecture Content



- Contextual Bandit setup.
- Contextual Bandit vs Supervised Learning.
- Mathematical Notations and Concepts.
- A Naïve Approach of Solving Contextual Bandits.
- Linear Bandits.
- Policy Gradient.

Policy Gradient

What is policy gradient?

- In policy gradient, we **fix the structure** of the policy and then **tune the parameters θ** of the policy using **gradient ascent**.
- Why gradient ascent and NOT gradient descent?
 - Answer: In gradient descent, we try to **minimize** a **loss**. To minimize, we have to go against the gradient. However, in gradient ascent, we try to **maximize** a **reward**. To maximize, we have to go along the gradient. Since in reinforcement learning we are trying to maximize reward (conventionally), we use gradient ascent.

Policy Gradient

What is policy gradient?

- In policy gradient, we **fix the structure** of the policy and then **tune the parameters θ** of the policy using **gradient ascent**.
- We denote the policy as **$\pi(a|\theta, x)$** . Here, **θ** is the parameter, **x** is the context, and **a** is the action. So basically, the policy is a mapping from context to action and the mapping is parameterized by θ .
- Example 1: Select action a with probability $\pi(a|\theta, x)$ where,

$$\pi(a|\theta, x) = \frac{e^{x^T w_a + b_a}}{\sum_{k=1}^A e^{x^T w_k + b_k}} \quad \left. \vphantom{\frac{e^{x^T w_a + b_a}}{\sum_{k=1}^A e^{x^T w_k + b_k}}} \right\} \text{Softmax function}$$

where if $x_k \in \mathbb{R}^d$ then $w_k \in \mathbb{R}^d$ for all k , and b_k is a scalar for all k . The parameter θ is a vector containing w_k and b_k for all k . This is a **randomized policy**.

Policy Gradient

What is policy gradient?

- In policy gradient, we **fix the structure** of the policy and then **tune the parameters θ** of the policy using **gradient ascent**.
- We denote the policy as **$\pi(a|\theta, x)$** . Here, **θ** is the parameter, **x** is the context, and **a** is the action. So basically, the policy is a mapping from context to action and the mapping is parameterized by θ .
- Example 2: Another policy?



HINT: Any function with x as input and probability of an action a as output.

Policy Gradient

What is policy gradient?

- In policy gradient, we **fix the structure** of the policy and then **tune the parameters θ** of the policy using **gradient ascent**.
- We denote the policy as **$\pi(a|\theta, x)$** . Here, **θ** is the parameter, **x** is the context, and **a** is the action. So basically, the policy is a mapping from context to action and the mapping is parameterized by **θ** .
- Example 2: Any **deep learning model** with **x** as input and a softmax as the last layer; basically a **classification model**. Rather than softmax we can use any other function whose output is “probability like”. **The policy is to select actions with probability given by the last layer**. In this case, **θ** are the parameters of the deep learning model. The benefits of such deep learning based policies are:
 - Such a policy can handle **unstructured** context **x** like image, audio, video, text etc.
 - Such policies are capable of **automated feature engineering**.

Policy Gradient

What is policy gradient?

- In policy gradient, we **fix the structure** of the policy and then **tune the parameters θ** of the policy using **gradient ascent**.
- We denote the policy as **$\pi(a|\theta, x)$** . Here, **θ** is the parameter, **x** is the context, and **a** is the action. So basically, the policy is a mapping from context to action and the mapping is parameterized by **θ** .
- Example 3: Another policy like softmax whose output is “probability like”?



Policy Gradient

What is policy gradient?

- In policy gradient, we **fix the structure** of the policy and then **tune the parameters θ** of the policy using **gradient ascent**.
- We denote the policy as **$\pi(a|\theta, x)$** . Here, **θ** is the parameter, **x** is the context, and **a** is the action. So basically, the policy is a mapping from context to action and the mapping is parameterized by θ .
- Example 3: Select action a with probability $\pi(a|\theta, x)$ where,

$$\pi(a|\theta, x) = \frac{(x^T w_a + b_a)^2}{\sum_{k=1}^A (x^T w_k + b_k)^2}$$

The reason to square is to make the output “probability like”; probability can’t be negative.

Policy Gradient

How is policy gradient different from “value function” based policy?

- In “value function” approach, we keep an estimate of the context-action value (or action value for MAB) and use this estimate to design the policy.
- In policy gradient approach, we directly design the policy.

Policy Gradient

What are the benefits of policy gradient compared to “value function” based policy?

- Policy gradient are **good for large action space**. This is because value function based approach has to keep an estimate of the context-action value (or action value for MAB) for each action. This becomes **computationally costly** for large action space.
- Policy gradient has surprising **similarity with supervised learning**.
- Policy gradient is good for unstructured data.
Side note: Even value function based policies like ϵ -greedy policy and UCB policy can be used for unstructured data by using neural networks instead of linear regression but those works are more at “research level” than being conventional.

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

➤ We are interested in maximizing the expected reward over a horizon T ,

$$\mathbb{E} \left[\sum_{\tau=1}^T r_{\tau} \right] \tag{P.1}$$

➤ Now, the above equation can be written as,

$$\mathbb{E} \left[\sum_{\tau=1}^T r_{\tau} \right] = \sum_{\tau=1}^T \mathbb{E}[r_{\tau}] = T \mathbb{E}[r_{\tau}] \tag{P.2}$$

Using linearity of expectation.

This step is possible because the involved probability distributions that decides $\mathbb{E}[r_{\tau}]$ are stationary. Hence, $\mathbb{E}[r_{\tau}]$ is same for all τ .

➤ Hence, maximizing (P.1) is same as maximizing $\mathbb{E}[r_{\tau}]$ of (P.2).

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

- Now $\mathbb{E}[r_\tau]$ is a function of θ because $\mathbb{E}[r_\tau]$ is decided by the policy and the policy depends on θ . So we have,

$$J(\theta) = \mathbb{E}[r_\tau]$$

- The gradient ascent equation is as follows,

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} J(\theta) \Big|_{\theta=\theta_t}$$

Step size.
May be time
variant.

Gradient $\nabla_{\theta} J(\theta)$
evaluated at $\theta = \theta_t$.

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

- Now $\mathbb{E}[r_\tau]$ is a function of θ because $\mathbb{E}[r_\tau]$ is decided by the policy and the policy depends on θ . So we have,

$$J(\theta) = \mathbb{E}[r_\tau]$$

- The gradient ascent equation is as follows,

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} J(\theta) \Big|_{\theta=\theta_t}$$

- Now all we have to do is compute this gradient and we are done. How to compute this gradient?



Give me one approach. Does not have to be an efficient one.

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

- Now $\mathbb{E}[r_\tau]$ is a function of θ because $\mathbb{E}[r_\tau]$ is decided by the policy and the policy depends on θ . So we have,

$$J(\theta) = \mathbb{E}[r_\tau]$$

- The gradient ascent equation is as follows,

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} J(\theta) \Big|_{\theta=\theta_t}$$

- How to compute this gradient?

- Remember that $\nabla_{\theta} J(\theta)$ is a vector whose i^{th} element is $\nabla_{\theta_i} J(\theta)$; the partial derivative of $J(\theta)$ with respect to the i^{th} element of θ . So we can simply calculate $\nabla_{\theta_i} J(\theta)$ for all i in order to calculate $\nabla_{\theta} J(\theta)$.

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

- Now $\mathbb{E}[r_\tau]$ is a function of θ because $\mathbb{E}[r_\tau]$ is decided by the policy and the policy depends on θ . So we have,

$$J(\theta) = \mathbb{E}[r_\tau]$$

- The gradient ascent equation is as follows,

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} J(\theta) \Big|_{\theta=\theta_t}$$

- How to compute this gradient?

- In order to calculate $\nabla_{\theta_i} J(\theta) \Big|_{\theta=\theta_t}$, we can create a new parameter $\tilde{\theta}_t^i$ as follows,

$$\begin{aligned}\theta_{i,t} &= \theta_{i,t} + \delta_i \\ \theta_{j,t} &= \theta_{j,t} \quad \text{if } j \neq i\end{aligned}$$

$\theta_{i,t}$ is the i^{th} element of $\theta_{i,t}$ (similarly $\theta_{j,t}$). Basically we are perturbing just the i^{th} element of $\theta_{i,t}$. δ_i should be small.

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

- Now $\mathbb{E}[r_\tau]$ is a function of θ because $\mathbb{E}[r_\tau]$ is decided by the policy and the policy depends on θ . So we have,

$$J(\theta) = \mathbb{E}[r_\tau]$$

- The gradient ascent equation is as follows,

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} J(\theta) \Big|_{\theta=\theta_t}$$

- How to compute this gradient?

- Then,

$$\nabla_{\theta_i} J(\theta) \Big|_{\theta=\theta_t} = \frac{J(\tilde{\theta}_t^i) - J(\theta_t)}{\delta_i}$$

Can be computed using empirical estimate, i.e. sampling.

- The problem with this approach of calculating gradient is that if the number of parameters are high (usually the case with deep learning), this will be computationally costly

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

- Now $\mathbb{E}[r_\tau]$ is a function of θ because $\mathbb{E}[r_\tau]$ is decided by the policy and the policy depends on θ . So we have,

$$J(\theta) = \mathbb{E}[r_\tau]$$

- The gradient ascent equation is as follows,

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} J(\theta) \Big|_{\theta=\theta_t}$$

- How to compute this gradient?
 - The problem with this approach of calculating gradient is that if the **number of parameters are high** (usually the case with deep learning), this will be **computationally costly**.
 - Also, the **frequency of updates** of θ is going to be low because we have to get $J(\tilde{\theta}_t^i)$ for each element of θ and to compute each $J(\tilde{\theta}_t^i)$ we need a certain number of time slots (in order to get empirical estimates).

Policy Gradient

How to tune the parameters using gradient ascent (basically the training procedure)?

- Now $\mathbb{E}[r_\tau]$ is a function of θ because $\mathbb{E}[r_\tau]$ is decided by the policy and the policy depends on θ . So we have,

$$J(\theta) = \mathbb{E}[r_\tau]$$

- The gradient ascent equation is as follows,

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} J(\theta) \Big|_{\theta=\theta_t}$$

- **CAUTION:** The policy should be such that $J(\theta)$ should be differentiable with respect to θ .
 - In supervised learning for classification problem, we choose the class with the maximum probability. What if we choose the action with the maximum probability? In such a case, is $J(\theta)$ differentiable with respect to θ . (Mandatory homework problem; will not be giving the solution).

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$J(\theta) = \mathbb{E}[r_{\tau}]$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \mathbb{E}[r_{\tau} | x_{\tau}] P[x_{\tau}] \quad \left. \vphantom{\sum_{x_{\tau} \in \mathcal{C}}} \right\} \begin{array}{l} \text{Using Law of Total} \\ \text{Expectation ([link here](#))} \end{array}.$$

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$J(\theta) = \mathbb{E}[r_{\tau}]$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \mathbb{E}[r_{\tau} | x_{\tau}] P[x_{\tau}]$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \mathbb{E}[r_{\tau} | x_{\tau}] f_{\mathcal{C}}(x_{\tau})$$

Same. $f_{\mathcal{C}}(x_{\tau})$ is the probability mass function corresponding to the context distribution $F_{\mathcal{C}}$ (refer slide 5).

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$J(\theta) = \mathbb{E}[r_{\tau}]$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \mathbb{E}[r_{\tau} | x_{\tau}] P[x_{\tau}]$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \mathbb{E}[r_{\tau} | x_{\tau}] f_{\mathcal{C}}(x_{\tau})$$

Same. This again uses the law of total expectation but conditional to the universe where the context is x_{τ} .

$$= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \mathbb{E}[r_{\tau} | x_{\tau}, a_{\tau}] P[a_{\tau} | x_{\tau}] f_{\mathcal{C}}(x_{\tau})$$

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$J(\theta) = \mathbb{E}[r_{\tau}]$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \mathbb{E}[r_{\tau} | x_{\tau}] P[x_{\tau}]$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \mathbb{E}[r_{\tau} | x_{\tau}] f_{\mathcal{C}}(x_{\tau})$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \mathbb{E}[r_{\tau} | x_{\tau}, a_{\tau}] P[a_{\tau} | x_{\tau}] f_{\mathcal{C}}(x_{\tau})$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \mathbb{E}[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

Same. $P[a_{\tau} | x_{\tau}]$ is basically the policy $\pi(a_{\tau} | \theta, x_{\tau})$.

To simplify, we will simply keep the last equation in the next slide.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned} J(\theta) &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \mathbb{E}[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} P[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \end{aligned}$$

\mathcal{R} is the set of rewards.
Rewards can be discrete or continuous. If continuous, the summation will be replaced by integral but the overall derivation remains the same.

Same. Just writing expected value in terms of the corresponding probability distribution.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$J(\theta) = \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \mathbb{E}[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} P[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

Same. $f_R(r_{\tau} | x_{\tau}, a_{\tau})$ is the probability mass function corresponding to the reward distribution $P_{x,a}$ (refer slide 5).

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned} J(\theta) &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \mathbb{E}[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} P[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \end{aligned}$$

➤ So now we differentiate the last equation with respect to θ to get $\nabla_{\theta} J(\theta)$:

$$\nabla_{\theta} J(\theta) = \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

Only this term is
a function of θ .

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned} J(\theta) &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \mathbb{E}[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} P[r_{\tau} | x_{\tau}, a_{\tau}] \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \end{aligned}$$

➤ So now we differentiate the last equation with respect to θ to get $\nabla_{\theta} J(\theta)$:

$$\nabla_{\theta} J(\theta) = \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

To simplify, we will simply keep the last equation in the next slide.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\nabla_{\theta} J(\theta) = \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \quad (\text{P.3})$$

- We definitely know $\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})$ because we know our policy and can hence we can differentiate (partial differentiation) that policy with respect to θ .
- However, we don't know $f_R(r_{\tau} | x_{\tau}, a_{\tau})$ and $f_{\mathcal{C}}(x_{\tau})$ (in fact that is where the “learning” comes into picture). Hence, we can't compute $\nabla_{\theta} J(\theta)$ using (P.3).
- In the next few slides we rearrange (P.3) so as to get empirical estimate of $\nabla_{\theta} J(\theta)$.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})\end{aligned}$$

Same.



Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\ &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})\end{aligned}$$

Just rearranged the
previous equation.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\nabla_{\theta} J(\theta) = \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

$$= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau})$$

$$= \mathbb{E}_{f_R, \pi_{\theta}, f_{\mathcal{C}}} \left[r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \right]$$

f_R , π_{θ} , and $f_{\mathcal{C}}$ are the short-hand version of $f_R[r_{\tau} | x_{\tau}, a_{\tau}]$, $\pi(a_{\tau} | \theta, x_{\tau})$, and $f_{\mathcal{C}}(x_{\tau})$ resp.

We are basically taking expectation of **this term** with respect to this **joint probability distribution**.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\&= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\&= \sum_{x_{\tau} \in \mathcal{C}} \sum_{a_{\tau} \in \mathcal{A}} \sum_{r_{\tau} \in \mathcal{R}} r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} f_R(r_{\tau} | x_{\tau}, a_{\tau}) \pi(a_{\tau} | \theta, x_{\tau}) f_{\mathcal{C}}(x_{\tau}) \\&= \mathbb{E}_{f_R, \pi_{\theta}, f_{\mathcal{C}}} \left[r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \right]\end{aligned}$$

To simplify, we will simply keep the last equation in the next slide.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{f_R, \pi_{\theta}, f_c} \left[r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \right]$$

log is the natural logarithm.

$$= \mathbb{E}_{f_R, \pi_{\theta}, f_c} [r_{\tau} \nabla_{\theta} \log(\pi(a_{\tau} | \theta, x_{\tau}))]$$

Because $\frac{d}{dx} \log(f(x)) = \frac{1}{f(x)} \frac{d}{dx} f(x)$;

simple application of chain rule of differentiation. The more involved derivation for partial differentiation is your homework.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{f_R, \pi_{\theta}, f_C} \left[r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \right]$$

$$= \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_{\tau} \nabla_{\theta} \log(\pi(a_{\tau} | \theta, x_{\tau}))]$$

$$= \nabla_{\theta} \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_{\tau} \log(\pi(a_{\tau} | \theta, x_{\tau}))]$$

Just moved the
gradient outside.

(P.4)

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{f_R, \pi_{\theta}, f_C} \left[r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \right] \\ &= \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_{\tau} \nabla_{\theta} \log(\pi(a_{\tau} | \theta, x_{\tau}))] \\ &= \nabla_{\theta} \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_{\tau} \log(\pi(a_{\tau} | \theta, x_{\tau}))]\end{aligned}$$

(P.4)

- We can't directly calculate the RHS of (P.4) because we don't know the probability mass functions $f_R[r_{\tau} | x_{\tau}, a_{\tau}]$ and $f_C(x_{\tau})$.
- However we can get an empirical estimate of the RHS of (P.4) by **sampling** from the distributions $P_{x,a}$ (distribution corresponding to $f_R[r_{\tau} | x_{\tau}, a_{\tau}]$), F_C (distribution corresponding to $f_C(x_{\tau})$), and finally the policy $\pi(a_{\tau} | \theta, x_{\tau})$.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{f_R, \pi_{\theta}, f_C} \left[r_{\tau} \frac{\nabla_{\theta} \pi(a_{\tau} | \theta, x_{\tau})}{\pi(a_{\tau} | \theta, x_{\tau})} \right] \\ &= \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_{\tau} \nabla_{\theta} \log(\pi(a_{\tau} | \theta, x_{\tau}))] \\ &= \nabla_{\theta} \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_{\tau} \log(\pi(a_{\tau} | \theta, x_{\tau}))]\end{aligned}$$

(P.4)

- There are two steps:
- Step 1: Generating the samples.
 - Step 2: Using the samples to estimate $\nabla_{\theta} J(\theta)$.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Generating the samples

We do the following three steps for every time t ,

Step 1: Generate context x_t by sampling from $F_{\mathcal{C}}$.

Step 2: For the given x_t , sample from $\pi(a|\theta, x_t)$ to get the action a_t .

Step 3: For the given x_t and a_t , sample from P_{x_t, a_t} to generate the reward r_t .

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Generating the samples

We do the following three steps for every time t ,

Step 1: Generate context x_t by sampling from $F_{\mathcal{C}}$.

Step 2: For the given x_t , sample from $\pi(a|\theta, x_t)$ to get the action a_t .

Step 3: For the given x_t and a_t , sample from P_{x_t, a_t} to generate the reward r_t .

One might ask that in order to sample from the distributions in Steps 1 and 3 we have to know $P_{x,a}$ and $F_{\mathcal{C}}$. But, in learning, $F_{\mathcal{C}}$ and $P_{x,a}$ are not known. So how are we sampling?

➤ If the agent is interacting with a **real environment**, there is NO sampling. x_t and r_t are generated by the environment and probability distribution that governs the generation of x_t and r_t are $F_{\mathcal{C}}$ and $P_{x,a}$ respectively. The agent does not know $F_{\mathcal{C}}$ and $P_{x,a}$.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Generating the samples

We do the following three steps for every time t ,

Step 1: Generate context x_t by sampling from $F_{\mathcal{C}}$.

Step 2: For the given x_t , sample from $\pi(a|\theta, x_t)$ to get the action a_t .

Step 3: For the given x_t and a_t , sample from P_{x_t, a_t} to generate the reward r_t .

One might ask that in order to sample from the distributions in Steps 1 and 3 we have to know $P_{x,a}$ and $F_{\mathcal{C}}$. But, in learning, $F_{\mathcal{C}}$ and $P_{x,a}$ are not known. So how are we sampling?

- If the agent is interacting with a **simulated environment**, $F_{\mathcal{C}}$ and $P_{x,a}$ are assumed to be known in order to generate the samples. However, **we are NOT using $F_{\mathcal{C}}$ and $P_{x,a}$ in order to design our policy**. So, basically the simulated environment is a **proxy** for the real one.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_t \log(\pi(a_t | \theta, x_t))] \quad (\text{P.4})$$

$$\approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

➤ x_t , a_t , and r_t are obtained by sampling.

➤ We are sampling over N samples.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_t \log(\pi(a_t | \theta, x_t))] \quad (\text{P.4})$$

$$\approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

➤ Recall from the discussion in the beginning of policy gradient. The policy $\pi(a_t | \theta, x_t)$ can be the output of a **Deep Learning model** designed for **classification problem**. Let,

$$p_{a_t} = \pi(a_t | \theta, x_t)$$

- p_{a_t} means probability of action a_t .
- p_{a_t} is also dependent on θ and x_t but I am not writing that for simplicity.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{f_R, \pi_{\theta}, f_C} [r_t \log(\pi(a_t | \theta, x_t))] \quad (\text{P.4})$$

$$\approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

➤ Recall from the discussion in the beginning of policy gradient. The policy $\pi(a_t | \theta, x_t)$ can be the output of a **Deep Learning model** designed for **classification problem**. Now the summation of (P.5) can be re-written as,

$$R_t = \frac{1}{N} \sum_{t=1}^N r_t \log(p_{a_t})$$

In the next slide we will substitute R_t in (P.5).

where $p_{a_t} = \pi(a_t | \theta, x_t)$.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

$$\approx \nabla_{\theta} (R_t) \quad \text{where,} \quad (\text{P.6})$$

$$R_t = \frac{1}{N} \sum_{t=1}^N r_t \log(p_{a_t}) \quad (\text{P.7})$$

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

$$\approx \nabla_{\theta} (R_t) \quad \text{where,} \quad (\text{P.6})$$

$$R_t = \frac{1}{N} \sum_{t=1}^N r_t \log(p_{a_t}) \quad (\text{P.7})$$

➤ Recall the **cross-entropy** loss function used for classification problem,

$$L_t = -\frac{1}{N} \sum_{t=1}^N \log(p_{\hat{y}_t})$$

The conventional cross entropy function has summation over all class. But it melts down this expression.

where y_t is the true class and \hat{y}_t is the predicted class.

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

$$\approx \nabla_{\theta} (R_t) \quad \text{where,} \quad (\text{P.6})$$

$$R_t = \frac{1}{N} \sum_{t=1}^N r_t \log(p_{a_t}) \quad (\text{P.7})$$

Cross Entropy

$$L_t = -\frac{1}{N} \sum_{t=1}^N \log(p_{\hat{y}_t})$$

➤ The similarity between R_t and L_t is striking.

➤ The difference is the additional term r_t of R_t .

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

$$\approx \nabla_{\theta} (R_t) \quad \text{where,} \quad (\text{P.6})$$

$$R_t = \frac{1}{N} \sum_{t=1}^N r_t \log(p_{a_t}) \quad (\text{P.7})$$

HURRAY MOMENT!!!

Cross Entropy

$$L_t = -\frac{1}{N} \sum_{t=1}^N \log(p_{\hat{y}_t})$$

➤ This observation suggests that we can treat R_t as the average reward of a “batch” and then do Stochastic Gradient Ascent (not descent) to update θ just like we do for supervised learning.

$$\theta_{t+1} = \theta_t + \epsilon \cdot \nabla_{\theta} (R_t)$$

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

$$\approx \nabla_{\theta} (R_t) \quad \text{where,} \quad (\text{P.6})$$

$$R_t = \frac{1}{N} \sum_{t=1}^N r_t \log(p_{a_t}) \quad (\text{P.7})$$

HURRAY MOMENT!!!

We have to be careful of the following if we are using the usual libraries for supervised learning:

- Cross Entropy
- $$L_t = -\frac{1}{N} \sum_{t=1}^N \log(p_{\hat{y}_t})$$
- We can use $-R_t$ in order to maximize R_t using gradient DESCENT .
 - We have to implement a custom loss function in order to account for r_t .

Policy Gradient

Efficient estimation of $\nabla_{\theta} J(\theta)$:

Using the samples to estimate $\nabla_{\theta} J(\theta)$

$$\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \left(\frac{1}{N} \sum_{t=1}^N r_t \log(\pi(a_t | \theta, x_t)) \right) \quad (\text{P.5})$$

$$\approx \nabla_{\theta} (R_t) \quad \text{where,} \quad (\text{P.6})$$

$$R_t = \frac{1}{N} \sum_{t=1}^N r_t \log(p_{a_t}) \quad (\text{P.7})$$

I may not allow you to use the usual libraries for supervised learning for Programming Assignment 1. So, it is crucial that you know how to do gradient descent for a simple Deep Learning model, specifically when $\pi(a_t | \theta, x_t)$ is the softmax function (logistic regression). In order to do this, you will need the closed form expression for the **partial derivative of the softmax function with respect to its parameters**. This is also important for Minor 1.

Homework: Derive partial derivative of the softmax function with respect to its parameters.

Policy Gradient

Pseudocode

- For $batch = 1, \dots, B$:
- Create an empty dataset $batch_training_data$.
- For $t = (batch - 1) * N + 1 \dots batch * N$:
- Using the current policy π_θ , sample $\{x_t, a_t, r_t\}$ using the steps in [this slide](#).
- Append $\{x_t, a_t, r_t\}$ onto $batch_training_data$.
- Use $batch_training_data$ to update θ using stochastic gradient ascent (refer [this slide](#) and the slide after this slide).

Policy Gradient

Pseudocode

- For $batch = 1, \dots, B$:
 - We are treating time in batches.
Each batch consist of multiple times slot.
- Create an empty dataset $batch_training_data$.
- For $t = (batch - 1) * N + 1 \dots batch * N$:
- Using the current policy π_θ , sample $\{x_t, a_t, r_t\}$ using the steps in [this slide](#).
- Append $\{x_t, a_t, r_t\}$ onto $batch_training_data$.
- Use $batch_training_data$ to update θ using stochastic gradient ascent (refer [this slide](#) and the slide after this slide).

Policy Gradient

Pseudocode

- For $batch = 1, \dots, B$:
- Create an empty dataset $batch_training_data$.
- For $t = (batch - 1) * N + 1 \dots batch * N$:
- Using the current policy π_θ , sample $\{x_t, a_t, r_t\}$ using the steps in [this slide](#).
- Append $\{x_t, a_t, r_t\}$ onto $batch_training_data$.
- Use $batch_training_data$ to update θ using stochastic gradient ascent (refer [this slide](#) and the slide after this slide).

When $N = 1$, i.e. batch size is 1, the algorithm has a special name:
REINFORCE
(this is not an acronym for reinforcement learning).

Policy Gradient

Pseudocode

- For $batch = 1, \dots, B$:
- Create an empty dataset $batch_training_data$.
- For $t = (batch - 1) * N + 1 \dots batch * N$:
- Using the current policy π_{θ} , sample $\{x_t, a_t, r_t\}$ using the steps in [this slide](#).
- Append $\{x_t, a_t, r_t\}$ onto $batch_training_data$.
- Use $batch_training_data$ to update θ using stochastic gradient ascent (refer [this slide](#) and the slide after this slide).

To do this we can use the `train_on_batch()` method of Keras/Tensorflow. If we do so, we have to follow the points mentioned in [this slide](#). We can also choose to write our own code for backpropagation if the model is simple (like a logistic regression).

