Exploration in RL: Contextual Bandit
Recap: MAB

Interactive learning process:

For $t = 0 \rightarrow T - 1$ (# based on historical information)

1. Learner pulls arm $I_t \in \{1, \ldots, K\}$

2. Learner observes an i.i.d reward $r_t \sim \nu_{I_t}$ of arm $I_t$
Recap: MAB

Interactive learning process:

For $t = 0 \rightarrow T - 1$

1. Learner pulls arm $I_t \in \{1, \ldots, K\}$

2. Learner observes an i.i.d reward $r_t \sim \nu_{I_t}$ of arm $I_t$

Learning metric:

$$\text{Regret}_T = T\mu^* - \sum_{t=0}^{T-1} \mu_{I_t}$$
Recap: MAB

Interactive learning process:

For $t = 0 \to T - 1$

1. Learner pulls arm $I_t \in \{1, \ldots, K\}$

2. Learner observes an i.i.d reward $r_t \sim \nu_{I_t}$ of arm $I_t$

Learning metric:

$$\text{Regret}_T = T\mu^* - \sum_{t=0}^{T-1} \mu_{I_t}$$

Arm distributions are fixed across learning.
Question for Today:

Incorporate contexts into the interactive learning framework
Outline for today:

1. Introduction of the model

2. Algorithm

3. Theory and some practical considerations
Make the framework Context Dependent:

Interactive learning process:

For $t = 0 \rightarrow T - 1$

1. A new context $x_t \in \mathcal{X}$ appears
Make the framework Context Dependent:

Interactive learning process:

For $t = 0 \rightarrow T - 1$

1. A new context $x_t \in \mathcal{X}$ appears
2. Learner picks action $a_t \in \mathcal{A}$ (# based on context $x_t$ and historical information)
Make the framework Context Dependent:

Interactive learning process:

For $t = 0 \rightarrow T - 1$

1. A new context $x_t \in \mathcal{X}$ appears
2. Learner picks action $a_t \in \mathcal{A}$ (# based on context $x_t$ and historical information)
3. Learner observes an reward $r_t := r(x_t, a_t)$

Reward is context and arm dependent now!
Make the framework Context Dependent:

Interactive learning process:

- Environment: $x_t \sim \mu$
- Learning algorithm: $a_t$
- Reward: $r_t := r(x_t, a_t)$
Examples:

Personalize recommendation system
Examples:

Personalize recommendation system

Context: user’s information (e.g., history health conditions, age, height, weight, job type, etc)
Examples:

Personalize recommendation system

**Context**: user’s information (e.g., history health conditions, age, height, weight, job type, etc)

**Decisions** (arms): news articles
Examples:

**Personalize recommendation system**

**Context:** user’s information (e.g., history health conditions, age, height, weight, job type, etc)

**Decisions (arms):** news articles

**Goal:** learn to maximizes user click rate
Examples:

**Personalize recommendation system**

**Context:** user’s information (e.g., history, health conditions, age, height, weight, job type, etc)

**Decisions** (arms): news articles

**Goal:** learn to maximize user click rate

Different users have different preferences on news, so need to personalize
Equivalently, it is an MDP with $H = 1$

Finite horizon MDP with $H = 1$

$\mathcal{M} = \{\mathcal{X}, \mathcal{A}, r, H = 1, \mu\}$
Equivalently, it is an MDP with $H = 1$

Finite horizon MDP with $H = 1$

$$\mathcal{M} = \{\mathcal{X}, \mathcal{A}, r, H = 1, \mu\}$$

Objective function:

$$\max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} \left[ r(x, \pi(x)) \right]$$
Equivalently, it is an MDP with $H = 1$

Finite horizon MDP with $H = 1$

$\mathcal{M} = \{ \mathcal{X}, \mathcal{A}, r, H = 1, \mu \}$

Objective function:

$$\max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} \left[ r(x, \pi(x)) \right]$$

For simplicity, we assume reward is deterministic; The challenge is really from randomness in contexts
The Regret Metric

Fix a policy class $\Pi$ (think about $\pi$ as a classifier from $x \rightarrow a$)

Denote optimal policy $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} r(x, \pi(x))$
The Regret Metric

Fix a policy class $\Pi$ (think about $\pi$ as a classifier from $x \to a$)

Denote optimal policy $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} r(x, \pi(x))$

Every iteration, learner has a policy $\pi^t \in \Pi$
(Recommends $a_t = \pi^t(x_t)$, receives reward $r_t := r(x_t, \pi(x_t))$)
The Regret Metric

Fix a policy class $\Pi$ (think about $\pi$ as a classifier from $x \rightarrow a$)

Denote optimal policy $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} r(x, \pi(x))$

Every iteration, learner has a policy $\pi^t \in \Pi$
(Recommends $a_t = \pi^t(x_t)$, receives reward $r_t := r(x_t, \pi(x_t))$)

$$\text{Regret}_T = T\mathbb{E}_{x \sim \mu} [r(x, \pi^*(x))] - \sum_{t=0}^{T-1} \mathbb{E}_{x \sim \mu} [r(x, \pi^t(x))]$$
The Regret Metric

Fix a policy class $\Pi$ (think about $\pi$ as a classifier from $x \rightarrow a$)

Denote optimal policy $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} r(x, \pi(x))$

Every iteration, learner has a policy $\pi^t \in \Pi$
(Recommends $a_t = \pi^t(x_t)$, receives reward $r_t := r(x_t, \pi(x_t))$)

$$\text{Regret}_T = T\mathbb{E}_{x \sim \mu}[r(x, \pi^*(x))] - \sum_{t=0}^{T-1} \mathbb{E}_{x \sim \mu}[r(x, \pi^t(x))]$$

Total expected reward if we always uses $\pi^*$ to recommend
The Regret Metric

Fix a policy class $\Pi$ (think about $\pi$ as a classifier from $x \rightarrow a$)

Denote optimal policy $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} r(x, \pi(x))$

Every iteration, learner has a policy $\pi^t \in \Pi$
(Recommends $a_t = \pi^t(x_t)$, receives reward $r_t := r(x_t, \pi(x_t))$)

$$\text{Regret}_T = T\mathbb{E}_{x \sim \mu}[r(x, \pi^*(x))] - \sum_{t=0}^{T-1} \mathbb{E}_{x \sim \mu}[r(x, \pi^t(x))]$$

Total expected reward if we always uses $\pi^*$ to recommend

Total expected reward of our learned sequence of policies
Outline for today:

1. Introduction of the model

2. Algorithm

3. Theory and some practical considerations
Ingredient 1: Importance Weighting

The key challenging here is that we observe $r_t := r(x_t, a_t)$, but we do not know $r(x_t, a)$ for $a \neq a_t$.
Ingredient 1: Importance Weighting

The key challenging here is that we observe $r_t := r(x_t, a_t)$, but we do not know $r(x_t, a)$ for $a \neq a_t$.

Importance weighting actually allows us to get unbiased estimate for ALL actions!
Ingredient 1: Importance Weighting

The key challenging here is that we observe \( r_t := r(x_t, a_t) \),
but we do not know \( r(x_t, a) \) for \( a \neq a_t \)

Importance weighting actually allows us to get unbiased estimate for ALL actions!

Assume \( a_t \sim p \ (p \in \Delta(\mathcal{A})) \), and we log \( p(a_t) \), receive \( r_t = r(x_t, a_t) \),
**Ingredient 1: Importance Weighting**

The key challenging here is that we observe \( r_t := r(x_t, a_t) \),
but we do not know \( r(x_t, a) \) for \( a \neq a_t \)

Importance weighting actually allows us to get
unbiased estimate for ALL actions!

Assume \( a_t \sim p \ (p \in \Delta(\mathcal{A})) \), and we log \( p(a_t) \), receive \( r_t = r(x_t, a_t) \),

For all \( a \in \mathcal{A} \), define \( \hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)} \),
Ingredient 1: Importance Weighting

The key challenging here is that we observe \( r_t := r(x_t, a_t) \),
but we do not know \( r(x_t, a) \) for \( a \neq a_t \).

Importance weighting actually allows us to get
unbiased estimate for ALL actions!

Assume \( a_t \sim p \) (\( p \in \Delta(\mathcal{A}) \)), and we log \( p(a_t) \), receive \( r_t = r(x_t, a_t) \),

For all \( a \in \mathcal{A} \), define \( \hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)} \),

\[
\hat{r} := \begin{bmatrix} 0 \\ 0 \\ \vdots \\ r_t/p(a_t) \\ 0, \\ \vdots \\ 0 \end{bmatrix}
\]
Ingredient 1: Importance Weighting

The key challenging here is that we observe $r_t := r(x_t, a_t)$, but we do not know $r(x_t, a)$ for $a \neq a_t$.

Importance weighting actually allows us to get unbiased estimate for ALL actions!

Assume $a_t \sim p$ ($p \in \Delta(\mathcal{A})$), and we log $p(a_t)$, receive $r_t = r(x_t, a_t)$,

For all $a \in \mathcal{A}$, define $\hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$, $\hat{r} := \begin{bmatrix} 0 \\ 0 \\ \vdots \\ r_t/p(a_t) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$,

$\mathbb{E}_{a_t \sim p}[\hat{r}[a]] = r(x_t, a), \forall a \in \mathcal{A}$
Proving Importance Weighting

Assume $a_t \sim p \,(p \in \Delta(\mathcal{A}))$, and we log $p_t(a)$, receive $r_t = r(x_t, a_t)$.

For all $a \in \mathcal{A}$, define $\hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$, we have: $\mathbb{E}_{a_t \sim p} \hat{r}[a] = r(x_t, a), \forall a \in \mathcal{A}$
Proving Importance Weighting

Assume $a_t \sim p$ ($p \in \Delta(\mathcal{A})$), and we log $p_t(a)$, receive $r_t = r(x_t, a_t)$,

For all $a \in \mathcal{A}$, define $\hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$, we have: $\mathbb{E}_{a_t \sim p} \hat{r}[a] = r(x_t, a), \forall a \in \mathcal{A}$

Consider any $a \in \mathcal{A}$:
Proving Importance Weighting

Assume $a_t \sim p (p \in \Delta(\mathcal{A}))$, and we log $p_t(a)$, receive $r_t = r(x_t, a_t)$.

For all $a \in \mathcal{A}$, define $\hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$, we have: $\mathbb{E}_{a_t \sim p} \hat{r}[a] = r(x_t, a), \forall a \in \mathcal{A}$

Consider any $a \in \mathcal{A}$:

$$\mathbb{E}_{a_t \sim p} \frac{r(x_t, a)1[a = a_t]}{p(a_t)} = \sum_{a_t \in \mathcal{A}} p(a_t) \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$$
Proving Importance Weighting

Assume $a_t \sim p \ (p \in \Delta(\mathcal{A}))$, and we log $p_t(a)$, receive $r_t = r(x_t, a_t)$.

For all $a \in \mathcal{A}$, define $\hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$, we have: $\mathbb{E}_{a_t \sim p} \hat{r}[a] = r(x_t, a), \forall a \in \mathcal{A}$

Consider any $a \in \mathcal{A}$:

$$\mathbb{E}_{a_t \sim p} \frac{r(x_t, a)1[a = a_t]}{p(a_t)} = \sum_{a_t \in \mathcal{A}} \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$$

$$= p(a) \frac{r(x_t, a)}{p(a)} = r(x_t, a)$$
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:

\[
\{x_i, y_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, y_i \in \{1, \ldots, |\mathcal{A}|\}
\]
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:

\[ \{x_i, y_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, y_i \in \{1, \ldots, |\mathcal{A}|\} \]

\[ \arg \max_{\pi \in \Pi} \sum_{i=1}^N 1\{\pi(x_i) = y_i\} \]
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:

\[
\{x_i, y_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, y_i \in \{1, \ldots, |\mathcal{A}|\}
\]

\[
\arg \max_{\pi \in \Pi} \sum_{i=1}^{N} 1\{\pi(x_i) = y_i\}
\]

Let’s generalize it to Reward-Sensitive Classification:
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:

\[
\{x_i, y_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, y_i \in \{1, \ldots, |\mathcal{A}|\}
\]

\[
\arg \max_{\pi \in \Pi} \sum_{i=1}^{N} \mathbf{1}\{\pi(x_i) = y_i\}
\]

Let’s generalize it to Reward-Sensitive Classification:

\[
\{x_i, r_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, r_i \in [0,1]^{|\mathcal{A}|}
\]
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:

\[
\{x_i, y_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, y_i \in \{1, \ldots, |\mathcal{A}|\}
\]

\[
\arg \max_{\pi \in \Pi} \sum_{i=1}^N 1\{\pi(x_i) = y_i\}
\]

Let's generalize it to **Reward-Sensitive Classification**:

\[
\{x_i, r_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, r_i \in [0,1]^{|\mathcal{A}|}
\]

\[
\begin{pmatrix}
  r[1] \\
r[2] \\
  \vdots \\
r[|\mathcal{A}|]
\end{pmatrix}
\]
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:

\[
\{x_i, y_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, y_i \in \{1, \ldots, |\mathcal{A}|\}
\]

\[
\arg \max_{\pi \in \Pi} \sum_{i=1}^N 1\{\pi(x_i) = y_i\}
\]

Let’s generalize it to Reward-Sensitive Classification:

\[
\{x_i, r_i\}_{i=1}^N, \quad x_i \in \mathcal{X}, r_i \in [0,1]^{|\mathcal{A}|}
\]

\[
\arg \max_{\pi \in \Pi} \sum_{i=1}^N r_i[\pi(x_i)]
\]

\[
\begin{pmatrix}
    x, \\
    \begin{bmatrix}
        r[1] \\
        r[2] \\
        \vdots \\
        r[|\mathcal{A}|]
    \end{bmatrix}
\end{pmatrix}
\]
Ingredient 2: Reward-sensitive Classification Oracle

Recall classic classification:

\[ \{x_i, y_i\}_{i=1}^{N}, \quad x_i \in \mathcal{X}, y_i \in \{1, \ldots, |\mathcal{A}|\} \]

\[
\arg \max_{\pi \in \Pi} \sum_{i=1}^{N} 1\{\pi(x_i) = y_i\}
\]

Let’s generalize it to **Reward-Sensitive Classification**:

\[ \{x_i, r_i\}_{i=1}^{N}, \quad x_i \in \mathcal{X}, r_i \in [0,1]^{|\mathcal{A}|} \]

\[
\arg \max_{\pi \in \Pi} \sum_{i=1}^{N} r_i[\pi(x_i)]
\]

We will do reduction to RSC
Summary so far:

1. Importance Weighting: we “magically” get unbiased estimate for all actions!

Assume \( a_t \sim p \ (p \in \Delta(\mathcal{A})) \), For all \( a \in \mathcal{A} \), define \( \hat{r}[a] = \frac{r(x_t, a)\mathbf{1}[a = a_t]}{p(a_t)} \), we have:

\[
\mathbb{E}_{a_t \sim p} \hat{r}[a] = r(x_t, a), \ \forall a \in \mathcal{A}
\]
Summary so far:

1. Importance Weighting: we “magically” get unbiased estimate for all actions!

Assume $a_t \sim p (p \in \Delta(\mathcal{A}))$, For all $a \in \mathcal{A}$, define $\hat{r}[a] = \frac{r(x_t, a)1[a = a_t]}{p(a_t)}$, we have:

$$E_{a_t \sim p} \hat{r}[a] = r(x_t, a), \forall a \in \mathcal{A}$$

$\hat{r} := \begin{bmatrix} 0 \\ 0 \\ \vdots \\ r_t/p(a_t) \\ 0, \ldots, 0 \end{bmatrix}$

2. Reward-Sensitive Classification:

$$\{x_i, r_i\}_{i=1}^N, \ x_i \in \mathcal{X}, r_i \in [0,1]^{\mathcal{A}}$$

$$\arg \max_{\pi \in \Pi} \sum_{i=1}^N r_i[\pi(x_i)]$$
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1$ : (# exploration phase)
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1$ :  (# exploration phase)

1. Observe $x_t \sim \mu$
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1$ : \hspace{1em} (# exploration phase)

1. Observe $x_t \sim \mu$

2. **Uniform-randomly** sample $a_t \sim \text{Unif}(\mathcal{A})$, receive reward $r_t = r(x_t, a_t)$
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1 : \quad (#$ exploration phase$)$

1. Observe $x_t \sim \mu$

2. **Uniform-randomly** sample $a_t \sim \text{Unif}(\mathcal{A})$, receive reward $r_t = r(x_t, a_t)$

3. Use IW, form unbiased estimate $\hat{r}_t[a] = \begin{cases} 0 & a \neq a_t \\ \frac{r_t}{1/|\mathcal{A}|} & a = a_t \end{cases}$
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1$ : (# exploration phase)

1. Observe $x_t \sim \mu$

2. **Uniform-randomly** sample $a_t \sim \text{Unif}(\mathcal{A})$, receive reward $r_t = r(x_t, a_t)$

3. Use IW, form unbiased estimate $\hat{r}_t[a] = \begin{cases} 0 & a \neq a_t \\ \frac{r_t}{1/|\mathcal{A}|} & a = a_t \end{cases}$

\[ \hat{r} := \begin{bmatrix} 0 \\ 0 \\ \vdots \\ r_t/p(a_t) \\ 0, \quad \ldots \quad 0 \end{bmatrix} \]
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1$:

1. Observe $x_t \sim \mu$

2. **Uniform-randomly** sample $a_t \sim \text{Unif}({\mathcal{A}})$, receive reward $r_t = r(x_t, a_t)$

3. Use IW, form unbiased estimate $\hat{r}_{t}[a] = \begin{cases} 0 & a \neq a_t \\ \frac{r_{t}}{1/|\mathcal{A}|} & a = a_t \end{cases}$

Call RSC oracle: $\hat{\pi} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{N-1} \hat{r}_{i}[\pi(x_{i})]$
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1$ :  (# exploration phase)

1. Observe $x_t \sim \mu$

2. **Uniform-randomly** sample $a_t \sim \text{Unif}(\mathcal{A})$, receive reward $r_t = r(x_t, a_t)$

3. Use $\text{IW}$, form unbiased estimate $\hat{r}_t[a] = \begin{cases} 0 & a \neq a_t \\ \frac{r_i}{1/|\mathcal{A}|} & a = a_t \end{cases}$

Call RSC oracle: $\hat{\pi} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{N-1} \hat{r}_i[\pi(x_i)]$

For $t = N \rightarrow T - 1$ :  (# exploitation phase)
Put things together: Explore and Commit

For $t = 0 \rightarrow N - 1$ : (# exploration phase)

1. Observe $x_t \sim \mu$

2. **Uniform-randomly** sample $a_t \sim \text{Unif}(\mathcal{A})$, receive reward $r_t = r(x_t, a_t)$

3. Use **IW**, form unbiased estimate \( \hat{r}_t[a] = \begin{cases} 0 & a \neq a_t \\ \frac{r_t}{1/|\mathcal{A}|} & a = a_t \end{cases} \)

**Call RSC oracle:** \( \hat{\pi} = \arg \max_{\pi \in \Pi} \sum_{i=0}^{N-1} \hat{r}_i[\pi(x_i)] \)

For $t = N \rightarrow T - 1$ : (# exploitation phase)

1. Observe $x_t \sim \mu$, and play $a_t = \hat{\pi}(x_t)$
Outline for today:

1. Introduction of the model

2. Algorithm

3. Theory and some practical considerations
Theory of the Explore and Commit Algorithm

For simplicity, assume $\Pi$ is discrete (but could be exponential large)
Theory of the Explore and Commit Algorithm

For simplicity, assume $\Pi$ is discrete (but could be exponential large)

[Theorem—informal] W/ high probability, properly setting the hyper-parameter $N$, Explore-and-Commit has the following regret:

$$\text{Regret}_T = T \mathbb{E}_{x \sim \mu}[r(x, \pi^*(x))] - \sum_{t=0}^{T-1} \mathbb{E}_{x \sim \mu}[r(x, \pi^t(x))] = O \left( T^{2/3} K^{1/3} \cdot \ln(|\Pi|)^{1/3} \right)$$
Practical Consideration

Instead of setting a hard threshold for explore and commit, we often interleave explore and exploitation.
Practical Consideration

Instead of setting a hard threshold for explore and commit, we often interleave explore and exploitation

\( \epsilon \)-greedy:

Every iteration \( t \):
With probability \( 1 - \epsilon \), we play \( a_t = \pi^t(x_t) \),
and with probability \( \epsilon \), we play \( a_t \sim \text{Unif}(\mathcal{A}) \)
Practical Consideration

Instead of setting a hard threshold for explore and commit, we often interleave exploration and exploitation

**ε-greedy:**

Every iteration $t$:
- With probability $1 - \epsilon$, we play $a_t = \pi^t(x_t)$,
- and with probability $\epsilon$, we play $a_t \sim \text{Unif}(\mathcal{A})$

Q: What’s the action distribution induced by ε-greedy at iteration $t$?
Practical Consideration

Instead of setting a hard threshold for explore and commit, we often interleave explore and exploitation.

ε-greedy:

Every iteration $t$:
With probability $1 - \epsilon$, we play $a_t = \pi^t(x_t)$,
and w/ probability $\epsilon$, we play $a_t \sim \text{Unif}(\mathcal{A})$.

Q: What’s the action distribution induced by ε-greedy at iteration $t$?

$$a \sim p_t, \quad p_t = (1 - \epsilon)\delta(\pi^t(x_t)) + \epsilon\text{Unif}(\mathcal{A})$$
Put things together: \( \varepsilon \)-greedy

For \( t = 0 \rightarrow \infty \) (# interleave exploration & exploitation)

1. Observe \( x_t \sim \mu \)
Put things together: $\epsilon-$greedy

For $t = 0 \rightarrow \infty$ (# interleave exploration & exploitation)

1. Observe $x_t \sim \mu$

2. Use $\epsilon$-greedy to form action distribution $p_t = (1 - \epsilon)\delta(\pi^t(x_t)) + \epsilon \text{Unif}(\mathcal{A})$
Put things together: $\epsilon$-greedy

For $t = 0 \rightarrow \infty$ (# interleave exploration & exploitation)

1. Observe $x_t \sim \mu$

2. Use $\epsilon$-greedy to form action distribution $p_t = (1 - \epsilon)\delta(\pi_t(x_t)) + \epsilon\text{Unif}(\mathcal{A})$

3. Play $a_t \sim p_t$, and observe $r_t := r(x_t, a_t)$
Put things together: $\epsilon$—greedy

For $t = 0 \to \infty$ (# interleve exploration & exploitation)

1. Observe $x_t \sim \mu$

2. Use $\epsilon$-greedy to form action distribution $p_t = (1 - \epsilon)\delta(\pi^t(x_t)) + \epsilon \text{Unif}(\mathcal{A})$

3. Play $a_t \sim p_t$, and observe $r_t := r(x_t, a_t)$

4. Via IW, form unbiased estimate $\hat{r}_t$
Put things together: $\epsilon-$greedy

For $t = 0 \to \infty$  (# interleave exploration & exploitation)

1. Observe $x_t \sim \mu$

2. Use $\epsilon$-greedy to form action distribution $p_t = (1 - \epsilon)\delta(\pi^t(x_t)) + \epsilon\text{Unif}(A)$

3. Play $a_t \sim p_t$, and observe $r_t := r(x_t, a_t)$

4. Via IW, form unbiased estimate $\hat{r}_t$

5. Update via RSC oracle: $\pi^{t+1} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{t} \hat{r}_i[\pi(x_i)]$
Put things together: $\epsilon-$greedy

For $t = 0 \to \infty$ (# interleave exploration & exploitation)

1. Observe $x_t \sim \mu$

2. Use $\epsilon$-greedy to form action distribution $p_t = (1 - \epsilon)\delta(\pi^t(x_t)) + \epsilon\text{Unif}(\mathcal{A})$

3. Play $a_t \sim p_t$, and observe $r_t := r(x_t, a_t)$

4. Via IW, form unbiased estimate $\hat{r}_t$

5. Update via RSC oracle: $\pi^{t+1} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{t} \hat{r}_i[\pi(x_i)]$

(Additionally 6. Gradually decay $\epsilon$...)
CB algorithm is being used in real world application at Microsoft:

Framework

1. Rank API
2. Rank Response
3. Reward API