Lecture 17: Multi-armed Bandits

1) Interactive coding demo - jupyter notebook

2) Formal Setting

Simplified RL setting with no state and no transitions

\( \mathcal{A} : 1, 2, \ldots, k \) k discrete actions ("arms")

\( \mathcal{R} : \mathcal{A} \xrightarrow{} \Delta(\mathbb{R}) \) noisy reward \( r_t \sim r(\mathcal{A}_t) \)

denote \( \mathbf{E}[r(\mathcal{A}_t)] = \mu_a \)

\( T : \mathbb{Z}_+ \) integer time horizon

Goal: maximize cumulative expected reward

\[
\mathbf{E} \left[ \sum_{t=1}^{T} r(\mathcal{A}_t) \right] = \sum_{t=1}^{T} \mu_{a_t}
\]

What is the optimal action?

\( a^* = \arg \max_{a=1, \ldots, k} \mu_a \)

This very simple MDP is easy to solve if rewards are known. When rewards are unknown, we must devise a strategy for balancing exploration (trying out different actions) against exploitation (selecting actions that perform well).

We measure the performance of a strategy, or algorithm, by comparing it against the optimal action.

Definition (Regret):

The regret of an algorithm which chooses actions \( a_1, \ldots, a_T \) is

\[
R(t) = \mathbf{E} \left[ \sum_{t=1}^{T} r(\mathcal{A}_t) - r(a_t) \right] = \sum_{t=1}^{T} \mu_{a^*} - \mu_{a_t}
\]
our goal is to find algorithms with sublinear regret. That way, the average suboptimality converges to 0:
\[
\lim_{T \to \infty} \frac{R(T)}{T} \to 0 \quad \text{if } R(T) \text{ sublinear e.g. } R(T) \leq p^T \quad \text{for } p < 1.
\]

3) Balancing exploration & exploitation

Consider the following two algorithms:

**Alg 1: Random**

\[
\text{for } t = 1, \ldots, T
\]
\[
a_t \sim \text{unif}(1, \ldots, K)
\]

pure explore

**Alg 2: Greedy**

\[
\text{for } t = 1, \ldots, K
\]
\[
a_t = t
\]
\[
R_t \sim r(a_t)
\]
\[
\text{for } t = K+1, \ldots, T
\]
\[
a_t = \arg\max_{a_t \in 1, \ldots, K} r(a_t)
\]

pure exploit

Both of these suffer from linear regret.

Why? \( R(T) = \sum_{t=1}^{T} E[r(a^*) - r(a_t)] = \)
\[
= \sum_{t=1}^{T} E[\sum_{a_t \neq a^*} (r(a^*) - r(a_t))]
\]
\[
\geq \sum_{t=1}^{T} \mathbb{P}[a_t \neq a^*] \min_{a_t \neq a^*} (y^* - y_{a_t}) = C \cdot T
\]

probability of not pulling \( a^* \) (constant for Alg 1 & 2)

Exercise: what is \( \mathbb{P}[a_t \neq a^*] \) for Alg 1 & 2?
Alg 3: Explore-then-Commit:
For $t = 1, \ldots, N+k$ pull each arm $N$ times
\[ \hat{y}_a = \frac{1}{N} \sum_{i=1}^{N} r_{a,i} \] compute average reward
For $t = N+k+1, \ldots, T$
\[ a_t = \arg \max_a \hat{y}_a = \hat{a}^* \]

This algorithm balances exploration & exploitation.
How to set $N$?
Let's do some analysis.

Lemma (Hoeffding's):
Suppose $r_{i} \in [0, 1]$ and $\mathbb{E}[r_{i}] = \mu$.
Then for $r_1, \ldots, r_N$ iid, with probability $1-\delta$,
\[ \left| \frac{1}{N} \sum_{i=1}^{N} r_i - \mu \right| \leq \sqrt{\frac{\log(1/\delta)}{N}} \]
Proof is out of scope.

Lemma (Explore): After exploration phase, for all arms $a=1, \ldots, K$,
\[ |\hat{y}_a - y_a| \leq \sqrt{\frac{\log(K/\delta)}{N}} \] with probability $1-\delta$.
Proof: Hoeffding & Union Bound $P(A \cup B) \leq P(A) + P(B)$.

This gives us $1-\delta$ confidence intervals:
\[ y_a \in \left[ \hat{y}_a \pm \sqrt{\frac{\log(K/\delta)}{N}} \right] \]

The regret decomposes:
\[ R(T) = \sum_{t=1}^{T} y^* - y_{a_t} = \sum_{t=1}^{N} y^* - y_{a_t} + \sum_{t=N+k+1}^{T} y^* - y_{a_t} \]
\[ R_1 \quad R_2 \]
for rewards bounded $[0, 1]$, $R_1 \leq NK$

we use confidence intervals to bound $R_2$.

$$R_2 = (T-NK)(\hat{y}^* - \hat{y}_{\hat{a}^*}) \leq (T-NK)\left[\hat{y}_{\hat{a}^*} + \sqrt{\frac{\log(\delta/\alpha)}{N}} - (\hat{y}_{\hat{a}^*} - \sqrt{\frac{\log(\delta/\alpha)}{N}})\right]$$

= $(T-NK)(\hat{y}_{\hat{a}^*} - \hat{y}_{\hat{a}^*} + 2\sqrt{\frac{\log(\delta/\alpha)}{N}})$

$\leq 0$ by definition of $\hat{a}^*$

Combining everything, we have

$$R(T) = R_1 + R_2 \leq NK + 2T\sqrt{\frac{\log(\delta/\alpha)}{N}} \quad \text{w.p. 1-}\delta$$

explore cost

$$\text{exploit cost (if wrong)}$$

Minimizing this upper bound with respect to $N$,

$$N = \left(\frac{T}{2K}\right)\frac{\log(\delta/\alpha)}{1/3}$$

and w.p. 1-\delta,

$$R(T) \leq T^{2/3} K^{1/3} \log^{1/3}(\frac{K}{\delta})$$

for explore-then-commit sublinear!

Next Lecture: consider confidence intervals directly in our algorithm.