Adversarial Bandits

Yingkai Li

EC4501/EC4501HM Semester 2, AY2024/25

Adversarial Bandits

Consider an online decision process with T periods and n arms.

- the sequence of payoffs $\{v_{i,t}\}_{i\in[n],t\in[T]}$ are determined by an adversary, where $v_{i,t} \in [0, 1]$.

Adversarial Bandits

Consider an online decision process with T periods and n arms.

- the sequence of payoffs $\{v_{i,t}\}_{i\in[n],t\in[T]}$ are determined by an adversary, where $v_{i,t} \in [0,1]$.

At any time $t \leq T$:

- designer selects an arm i_t^*;
- the designer receives a payoff of $v_{i_t^*,t}$.
- the designer only observes the payoffs for the selected arm.

Regret Minimization

Optimal-in-hindsight Benchmark:

$$B_T = \max_{i \in [n]} \sum_{t \in T} v_{i,t}.$$

Regret Minimization

Optimal-in-hindsight Benchmark:

$$B_T = \max_{i \in [n]} \sum_{t \in T} v_{i,t}.$$

(External) Regret:

$$\mathrm{R}_T = B_T - \sum_{t \in T} v_{i_t^*,t}.$$

Regret Minimization

Optimal-in-hindsight Benchmark:

$$B_T = \max_{i \in [n]} \sum_{t \in T} v_{i,t}.$$

(External) Regret:

$$\mathrm{R}_T = B_T - \sum_{t \in T} v_{i_t^*,t}.$$

An algorithm has no-regret if $\mathrm{R}_T = o(T)$.

- Is it possible to design no-regret algorithms with adversarial rewards under bandit feedback?
- The designer cannot predict future rewards based on historical observation.

Intuitions

Algorithms for stochastic environments fail for adversarial bandits:

- E.g., for Explore-then-Exploit, the adversary may generate low rewards in the exploitation phase for arms that performs the best in the exploration phase.

Intuitions

Algorithms for stochastic environments fail for adversarial bandits:

- E.g., for Explore-then-Exploit, the adversary may generate low rewards in the exploitation phase for arms that performs the best in the exploration phase.

Algorithms for expert learning algorithms:

- they require payoffs for all arms, which are not observable in bandit settings.

Intuitions

Algorithms for stochastic environments fail for adversarial bandits:

- E.g., for Explore-then-Exploit, the adversary may generate low rewards in the exploitation phase for arms that performs the best in the exploration phase.

Algorithms for expert learning algorithms:

- they require payoffs for all arms, which are not observable in bandit settings.

Idea: adopt expert learning algorithms with counterfactual estimations.

Inverse Propensity Score (IPS) Estimator

How to estimate the reward for each arm in adversarial settings?

- the designer can only observe the realized reward for the chosen action.

Inverse Propensity Score (IPS) Estimator

How to estimate the reward for each arm in adversarial settings?

- the designer can only observe the realized reward for the chosen action.

Inverse Propensity Score (IPS) Estimator:

$$\hat{v}_{i,t} = \frac{v_{i,t} \cdot \mathbf{1}\left(i_t^* = i \right)}{p_t(i)}$$

where $p_t(i)$ is the probability of choosing arm i in period t.

Lemma

For any arm i and any sequence of rewards, the IPS estimator is unbiased, i.e.,

$$\mathbf{E}[\hat{v}_{i,t}] = v_{i,t}.$$

Other Estimators

Alternative estimator:

$$\hat{v}_{i,t} = 1 - \frac{(1 - v_{i,t}) \cdot \mathbf{1}\left(i_t^* = i \right)}{p_{i,t}}$$

Intuitively, this is the IPS estimator for the loss of $y_{i,t} = 1 - v_{i,t}$.

- this is also an unbiased estimator;
- $\hat{v}_{i,t} \leq 1$.

Exponential-weight Algorithm

Exponential-weight algorithm for Exploration and Exploitation (EXP3) with learning rate η: the probability of choosing action i at time t is

$$p_t(i) = \frac{\exp(\eta \cdot \hat{\mu}_{i,t})}{\sum_{j=1}^{n} \exp(\eta \cdot \hat{\mu}_{i,t})}.$$

where $\hat{\mu}_{i,t} = \sum_{s<t} \hat{v}_{s,t}$.

Exponential-weight Algorithm

Exponential-weight algorithm for Exploration and Exploitation (EXP3) with learning rate η: the probability of choosing action i at time t is

$$p_t(i) = \frac{\exp(\eta \cdot \hat{\mu}_{i,t})}{\sum_{j=1}^{n} \exp(\eta \cdot \hat{\mu}_{i,t})}.$$

where $\hat{\mu}_{i,t} = \sum_{s<t} \hat{v}_{s,t}$.

Remark: EXP3 is similar to the Hedge algorithm, by replacing the empirical reward for each arm with its estimation.

Exponential-weight Algorithm

Exponential-weight algorithm for Exploration and Exploitation (EXP3) with learning rate η: the probability of choosing action i at time t is

$$p_t(i) = \frac{\exp(\eta \cdot \hat{\mu}_{i,t})}{\sum_{j=1}^{n} \exp(\eta \cdot \hat{\mu}_{i,t})}.$$

where $\hat{\mu}_{i,t} = \sum_{s<t} \hat{v}_{s,t}$.

Remark: EXP3 is similar to the Hedge algorithm, by replacing the empirical reward for each arm with its estimation.

Theorem

The worst-case regret of EXP3 is $O(\sqrt{nT \cdot \log n})$.

Exponential-weight Algorithm

Recall the proof for Hedge in expert learning setting, we have

$$\mathrm{R}_T \leq \frac{\log n}{\eta} + \frac{\eta}{2} \sum_{t \in [T]} \sum_{i \in [n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2.$$

Exponential-weight Algorithm

Recall the proof for Hedge in expert learning setting, we have

$$\mathrm{R}_T \leq \frac{\log n}{\eta} + \frac{\eta}{2} \sum_{t \in [T]} \sum_{i \in [n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2.$$

In expert learning setting without reward estimations, $\sum_{t \in [T]} \sum_{i \in [n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2 \leq T$.

Exponential-weight Algorithm

Recall the proof for Hedge in expert learning setting, we have

$$\mathrm{R}_T \leq \frac{\log n}{\eta} + \frac{\eta}{2} \sum_{t\in[T]} \sum_{i\in[n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2.$$

In expert learning setting without reward estimations, $\sum_{t\in[T]} \sum_{i\in[n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2 \leq T$.

In adversarial bandits, with reward estimations, we can show that

$$\mathbf{E}\left[\sum_{t\in[T]} \sum_{i\in[n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2\right] \leq nT.$$

Exponential-weight Algorithm

Recall the proof for Hedge in expert learning setting, we have

$$\mathrm{R}_T \le \frac{\log n}{\eta} + \frac{\eta}{2} \sum_{t\in[T]} \sum_{i\in[n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2.$$

In expert learning setting without reward estimations, $\sum_{t\in[T]} \sum_{i\in[n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2 \le T$.

In adversarial bandits, with reward estimations, we can show that

$$\mathbf{E}\left[\sum_{t\in[T]} \sum_{i\in[n]} p_t(i) \cdot (\hat{v}_{i,t} - 1)^2\right] \le nT.$$

Combining inequalities, we have $\mathrm{R}_T \le \frac{\log n}{\eta} + \frac{\eta n T}{2}$.
When $\eta = \sqrt{2nT \cdot \log n}$, we have $\mathrm{R}_T \le \sqrt{2nT \cdot \log n}$.

Variation of Loss

Let $\hat{y}_{i,t} = 1 - \hat{v}_{i,t}$. We have

$$p_t(i) \cdot \hat{y}_{i,t} = p_t(i) \cdot \frac{(1 - v_{i,t}) \cdot \mathbf{1}\left(i_t^* = i \right)}{p_t(i)} = (1 - v_{i,t}) \cdot \mathbf{1}\left(i_t^* = i \right) \leq 1.$$

Therefore, since $\hat{y}_{i,t}$ is unbiased,

$$\mathbf{E}\left[\sum_{t \in [T]} \sum_{i \in [n]} p_t(i) \cdot \hat{y}_{i,t}^2 \right] \leq \mathbf{E}\left[\sum_{t \in [T]} \sum_{i \in [n]} \hat{y}_{i,t} \right] = \sum_{t \in [T]} \sum_{i \in [n]} y_{i,t} \leq nT.$$

Swap Regret

Swap Regret:

$$\mathrm{SR}_T = \max_{\pi:A\to A} \sum_{t\in T} v_{\pi(i_t^*),t} - \sum_{t\in T} v_{i_t^*,t}.$$

Swap Regret

Swap Regret:

$$\mathrm{SR}_T = \max_{\pi : A \to A} \sum_{t \in T} v_{\pi(i_t^*), t} - \sum_{t \in T} v_{i_t^*, t}.$$

Note that the (external) regret can be viewed as swap regret under the restriction that $\pi(i) = \pi(i')$ for any i, i'.

No Swap Regret

Theorem (Blum and Mansour '07)

When there are n actions and T periods, there is an algorithm that achieves swap regret at most $O(n\sqrt{nT\log n})$.

Idea is similar to the reduction for expert learning.

No Swap Regret

Theorem (Blum and Mansour '07)

When there are n actions and T periods, there is an algorithm that achieves swap regret at most $O(n\sqrt{nT\log n})$.

Idea is similar to the reduction for expert learning.

Subtle difference: the aggregate distribution over arms is not the same as the individual algorithm generated

- from the perspective of each algorithm $\mathcal{A}_i$, the observed arm does not follow the distribution recommended by the algorithm.

No Swap Regret

Theorem (Blum and Mansour '07)

When there are n actions and T periods, there is an algorithm that achieves swap regret at most $O(n\sqrt{nT\log n})$.

Idea is similar to the reduction for expert learning.

Subtle difference: the aggregate distribution over arms is not the same as the individual algorithm generated

- from the perspective of each algorithm $\mathcal{A}_i$, the observed arm does not follow the distribution recommended by the algorithm.

Further adjust the feedback reward according to the aggregate distribution over arms for unbiased estimations within each algorithm $\mathcal{A}_i$.