Expert Learning

Yingkai Li

EC4501/EC4501HM

Expert Learning

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

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

Expert Learning

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

- 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 expert i_t^*;
- the designer receives a payoff of $v_{i_t^*,t}$;
- the designer observes the realized payoffs for all experts.

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 without any knowledge about the future reward realizations?

Follow the Leader

No need for exploration: observes the payoffs of all experts.

Follow the Leader

No need for exploration: observes the payoffs of all experts.

Follow-the-Leader Algorithm:

- at any time $t \leq T$, select the expert (with random tie breaking)

$$i_t^* = \operatorname*{argmax}_{i \in [n]} \sum_{s<t} v_{i,s}.$$

Follow the Leader

No need for exploration: observes the payoffs of all experts.

Follow-the-Leader Algorithm:

- at any time $t \leq T$, select the expert (with random tie breaking)

$$i_t^* = \operatorname*{argmax}_{i \in [n]} \sum_{s<t} v_{i,s}.$$

Follow-the-Leader algorithm has regret $\mathrm{R}_T = \Theta(T)$.

Follow the Leader

No need for exploration: observes the payoffs of all experts.

Follow-the-Leader Algorithm:

- at any time $t \leq T$, select the expert (with random tie breaking)

$$i_t^* = \operatorname*{argmax}_{i \in [n]} \sum_{s<t} v_{i,s}.$$

Follow-the-Leader algorithm has regret $\mathrm{R}_T = \Theta(T)$.
Consider an example with two experts:

- expert 1 has reward sequence $1, 0, 0, 1, 1, 0, 0, \ldots;$
- expert 2 has reward sequence $0, 1, 1, 0, 0, 1, 1, \ldots;$
- each expert gets $\frac{T}{2}$, the algorithm gets $\frac{T}{4}$. Regret is $\frac{T}{4}$.

Follow the Leader

No need for exploration: observes the payoffs of all experts.

Follow-the-Leader Algorithm:

- at any time $t \leq T$, select the expert (with random tie breaking)

$$i_t^* = \operatorname*{argmax}_{i \in [n]} \sum_{s<t} v_{i,s}.$$

Follow-the-Leader algorithm has regret $\mathrm{R}_T = \Theta(T)$.

Consider an example with two experts:

- expert 1 has reward sequence $1, 0, 0, 1, 1, 0, 0, \ldots;$
- expert 2 has reward sequence $0, 1, 1, 0, 0, 1, 1, \ldots;$
- each expert gets $\frac{T}{2}$, the algorithm gets $\frac{T}{4}$. Regret is $\frac{T}{4}$.

Need randomization in algorithms: hedge against adversarial rewards.

- Any deterministic algorithm (e.g., Explore-then-Exploit, UCB) has linear regret.

Hedge Algorithm

Hedge algorithm with learning rate η:
the probability 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 $\hat{\mu}_{i,t} = \sum_{s<t} v_{i,s}$ is the historical rewards for expert i.

Hedge Algorithm

Hedge algorithm with learning rate η:
the probability 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 $\hat{\mu}_{i,t} = \sum_{s<t} v_{i,s}$ is the historical rewards for expert i.

- $\eta = 0$: uniform random selection;
- $\eta \to \infty$: selecting the arm with maximum average history reward.

Hedge Algorithm

Hedge algorithm with learning rate η:
the probability 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 $\hat{\mu}_{i,t} = \sum_{s<t} v_{i,s}$ is the historical rewards for expert i.

- $\eta = 0$: uniform random selection;
- $\eta \to \infty$: selecting the arm with maximum average history reward.

Theorem

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

Hedge Algorithm

Lemma

The worst-case regret of Hedge is $\mathrm{R}_T \le \frac{\log n}{\eta} + \frac{\eta T}{2}$.

By setting $\eta = \sqrt{\frac{2\log n}{T}}$, we have $\mathrm{R}_T \le \sqrt{2T \cdot \log n}$.

Hedge Algorithm

Lemma

The worst-case regret of Hedge is $\mathrm{R}_T \leq \frac{\log n}{\eta} + \frac{\eta T}{2}$.

By setting $\eta = \sqrt{\frac{2\log n}{T}}$, we have $\mathrm{R}_T \leq \sqrt{2T \cdot \log n}$.

Proof: Define the potential function as the sum of weights:

$$W_t = \sum_{i=1}^{n} e^{\eta \hat{\mu}_{i,t}}.$$

Initially, $W_1 = n$. After one step:

$$W_{t+1} = \sum_{i=1}^{n} e^{\eta \hat{\mu}_{i,t+1}} = \sum_{i=1}^{n} e^{\eta \hat{\mu}_{i,t}} \cdot e^{\eta v_{i,t}}.$$

By convexity and Jensen's inequality:

$$W_{t+1} = W_t \cdot \sum_{i=1}^{n} p_t(i) e^{\eta v_{i,t}} \leq W_t \cdot e^{\eta \sum_{i=1}^{n} p_t(i) v_{i,t} + \frac{\eta^2}{2}}.$$

Hedge Algorithm

Unrolling the recursion:

$$W_{T+1} \leq n \cdot e^{\eta \sum_{t=1}^{T} \sum_{i=1}^{n} p_t(i) v_{i,t} + \frac{\eta^2 T}{2}}.$$

Thus, for any expert i, we have:

$$e^{\eta \hat{\mu}_{i,T+1}} \leq n \cdot e^{\eta \sum_{t=1}^{T} \sum_{i=1}^{n} p_t(i) v_{i,t} + \frac{\eta^2 T}{2}}.$$

Taking logs and rearranging:

$$\mathrm{R}_T = \hat{\mu}_{i,T+1} - \sum_{t=1}^{T} \sum_{i=1}^{n} p_t(i) v_{i,t} \leq \frac{\log n}{\eta} + \frac{\eta T}{2}.$$

Follow the Regularized Leader (FTRL)

The FTRL algorithm is parametrized by a (strongly convex) regularization function $l(p)$.

Follow the Regularized Leader (FTRL)

The FTRL algorithm is parametrized by a (strongly convex) regularization function $l(p)$.

For each round $t \leq T$: choose a distribution p_t

$$p_t = \arg\max_p \left(\mathbf{E}_p[\hat{\mu}_{i,t}] - l(p) \right).$$

Follow the Regularized Leader (FTRL)

The FTRL algorithm is parametrized by a (strongly convex) regularization function $l(p)$.

For each round $t \leq T$: choose a distribution p_t

$$p_t = \arg\max_p \left(\mathbf{E}_p[\hat{\mu}_{i,t}] - l(p) \right).$$

Example Regularization:

- *L2 regularization:* $l(p) = \frac{\lambda}{2}\|p\|^2$.
- *Entropy regularization (logarithmic barrier):* $l(p) = \eta \sum_i p_i \log(p_i)$ for probability distributions.

Follow the Regularized Leader (FTRL)

The FTRL algorithm is parametrized by a (strongly convex) regularization function $l(p)$.

For each round $t \leq T$: choose a distribution p_t

$$p_t = \arg\max_p \left(\mathbf{E}_p[\hat{\mu}_{i,t}] - l(p) \right).$$

Example Regularization:

- *L2 regularization:* $l(p) = \frac{\lambda}{2}\|p\|^2$.
- *Entropy regularization (logarithmic barrier):* $l(p) = \eta \sum_i p_i \log(p_i)$ for probability distributions.

Remark: The regularization term $R(x)$ controls the trade-off between fitting past observations and encouraging exploration or stability in the decision sequence.

Follow the Regularized Leader (FTRL)

The FTRL algorithm is parametrized by a (strongly convex) regularization function $l(p)$.

For each round $t \leq T$: choose a distribution p_t

$$p_t = \arg\max_p \left(\mathbf{E}_p[\hat{\mu}_{i,t}] - l(p) \right).$$

Example Regularization:

- *L2 regularization:* $l(p) = \frac{\lambda}{2}\|p\|^2$.
- *Entropy regularization (logarithmic barrier):* $l(p) = \eta \sum_i p_i \log(p_i)$ for probability distributions.

Remark: The regularization term $R(x)$ controls the trade-off between fitting past observations and encouraging exploration or stability in the decision sequence.

Hedge is FTRL with entropy regularization.

Calibration

We want the prediction of the forecast to be credible and trustworthy:

- If a weather forecaster predicts the probability of raining, we want the frequency of raining to match the prediction; e.g., if the forecaster predicts the probability of raining is 50% for some days, the prediction is calibrated if half of those days are raining.
- If a financial manager predicts the probability of a positive return for an investment option, we want the frequency of positive return to match the prediction.

Calibration

prediction	50%	50%	33.3%	50%	33.3%	33.3%	50%
outcome	rain	sunny	sunny	rain	rain	sunny	sunny

Table: Calibrated Forecast

prediction	42.9%	42.9%	42.9%	42.9%	42.9%	42.9%	42.9%
outcome	rain	sunny	sunny	rain	rain	sunny	sunny

Table: Calibrated Forecast

prediction	50%	25%	25%	50%	25%	25%	50%
outcome	rain	sunny	sunny	rain	rain	sunny	sunny

Table: Non-calibrated Forecast

Swap Regret

Swap Regret (Internal 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 (Internal 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'.

Swap Regret

Swap Regret (Internal 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'.

Lemma

For any bandit instance and any learning algorithm, $\mathrm{SR}_T \geq \mathrm{R}_T$.

Intuitive Connections

Calibration: probabilistic forecasts; no improvement by changing any forecast.

No-swap-regret: utility maximization; no improvement by switching actions.

Intuitive Connections

Calibration: probabilistic forecasts; no improvement by changing any forecast.

No-swap-regret: utility maximization; no improvement by switching actions.

Connecting probabilistic forecasts with utility maximization: proper scoring rule $S(p, \omega)$

$$\mathbf{E}_{\omega \sim p}[S(p, \omega)] \geq \mathbf{E}_{\omega \sim p}\left[S(p', \omega)\right], \forall p, p'.$$

- Quadratic scoring rule: $S(p, \omega) = 1 - (p - \omega)^2$.
- Log scoring rule: $S(p, \omega) = \log p(\omega)$.

Reduction

A calibrated forecast based on any no-swap-regret algorithm $\mathcal{A}$:

- construct a proper scoring rule for converting probabilistic forecasts to realized payoffs;
- apply no-swap-regret algorithm $\mathcal{A}$ with the realized payoffs to determine the action.

Reduction

A calibrated forecast based on any no-swap-regret algorithm $\mathcal{A}$:

- construct a proper scoring rule for converting probabilistic forecasts to realized payoffs;
- apply no-swap-regret algorithm $\mathcal{A}$ with the realized payoffs to determine the action.

By the definition of proper scoring rules, the following are equivalent:

- the forecast is calibrated, i.e., for any forecast p, the empirical distribution in periods predicting p is also p;
- the algorithm has no swap regret, i.e., for any action i, the utility of swapping i to another action i' is lower.

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 $O(n\sqrt{T\log n})$.

Intuition:

- build a no (external) regret algorithm for each expert to ensure the regret of swapping that expert with others is small;
- find a smart way of aggregating the recommendations of different algorithm to ensure no swap regret.

No Swap Regret

1. Initialize an algorithm $\mathcal{A}_i$ for each expert i;
2. Let $q_{i,t}$ be the recommended distribution over experts from algorithm $\mathcal{A}_i$ at time t. Aggregate them into a distribution p_t.
3. Select an expert according to p_t. The designer observes rewards $v_{i,t}$ for all i.
4. For each algorithm $\mathcal{A}_i$, scale the rewards by $p_t(i)$ as feedback. I.e, $\mathcal{A}_i$ sees reward vector $p_t(i) \cdot v_t$.

No Swap Regret

1. Initialize an algorithm $\mathcal{A}_i$ for each expert i;
2. Let $q_{i,t}$ be the recommended distribution over experts from algorithm $\mathcal{A}_i$ at time t. Aggregate them into a distribution p_t.
3. Select an expert according to p_t. The designer observes rewards $v_{i,t}$ for all i.
4. For each algorithm $\mathcal{A}_i$, scale the rewards by $p_t(i)$ as feedback. I.e, $\mathcal{A}_i$ sees reward vector $p_t(i) \cdot v_t$.

In step 2, the aggregate distribution p_t satisfies

$$p_t(i) = \sum_{j \in [n]} p_t(j) \cdot q_{j,t}(i), \forall i \in [n].$$

That is, $p_t = p_t \times q_t$.

No Swap Regret

For algorithm $\mathcal{A}_i$ and any expert $\pi(i) \in [n]$ its regret is

$$\mathrm{R}_{i,T} \geq \sum_{t \leq T} p_t(i) \cdot v_{\pi(i),t} - \sum_{t \leq T} p_t(i) \cdot \sum_{i \in [n]} q_{i,t} v_t.$$

No Swap Regret

For algorithm $\mathcal{A}_i$ and any expert $\pi(i) \in [n]$ its regret is

$$\mathrm{R}_{i,T} \geq \sum_{t \leq T} p_t(i) \cdot v_{\pi(i),t} - \sum_{t \leq T} p_t(i) \cdot \sum_{i \in [n]} q_{i,t} v_t.$$

Summing over $i \in [n]$, we have

$$\sum_i \mathrm{R}_{i,T} \geq \sum_{i \in [n]} \sum_{t \leq T} p_t(i) \cdot v_{\pi(i),t} - \sum_{i \in [n]} \sum_{t \leq T} p_t(i) \cdot \sum_{i \in [n]} q_{i,t} v_t$$
$$= \mathbf{E}\left[\sum_{t \leq T} v_{\pi(i_t^*),t}\right] - \sum_{t \leq T} p_t v_t = \mathrm{SR}_T.$$

No Swap Regret

For algorithm $\mathcal{A}_i$ and any expert $\pi(i) \in [n]$ its regret is

$$\mathrm{R}_{i,T} \geq \sum_{t \leq T} p_t(i) \cdot v_{\pi(i),t} - \sum_{t \leq T} p_t(i) \cdot \sum_{i \in [n]} q_{i,t} v_t.$$

Summing over $i \in [n]$, we have

$$\begin{aligned}\sum_i \mathrm{R}_{i,T} &\geq \sum_{i \in [n]} \sum_{t \leq T} p_t(i) \cdot v_{\pi(i),t} - \sum_{i \in [n]} \sum_{t \leq T} p_t(i) \cdot \sum_{i \in [n]} q_{i,t} v_t \\ &= \mathbf{E}\left[\sum_{t \leq T} v_{\pi(i_t^*),t}\right] - \sum_{t \leq T} p_t v_t = \mathrm{SR}_T.\end{aligned}$$

Since we have algorithms such that $\mathrm{R}_{i,T} \leq \sqrt{2T \log n}$ for all $i \in [n]$, we have $\mathrm{SR}_T \leq n\sqrt{2T \log n}$.