

Announcements

➤ HW1 due **next Tue, before class**

CMSC 3540 I: The Interplay of Learning and Game Theory (Autumn 2022)

Intro to Online Learning

Instructor: Haifeng Xu

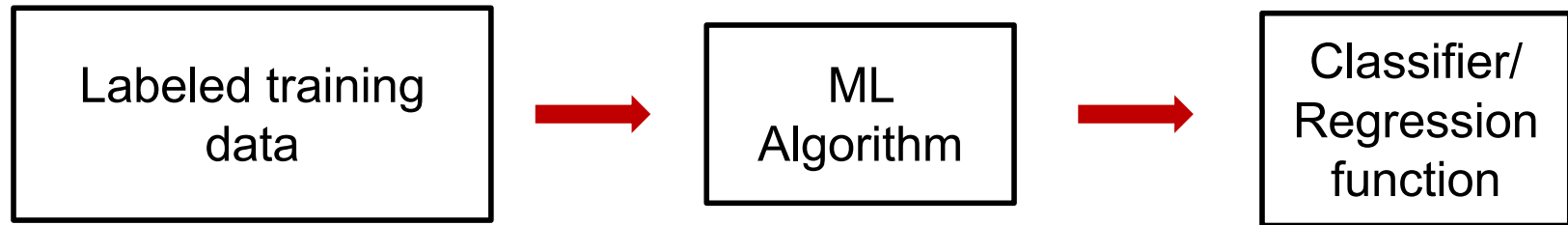


Outline

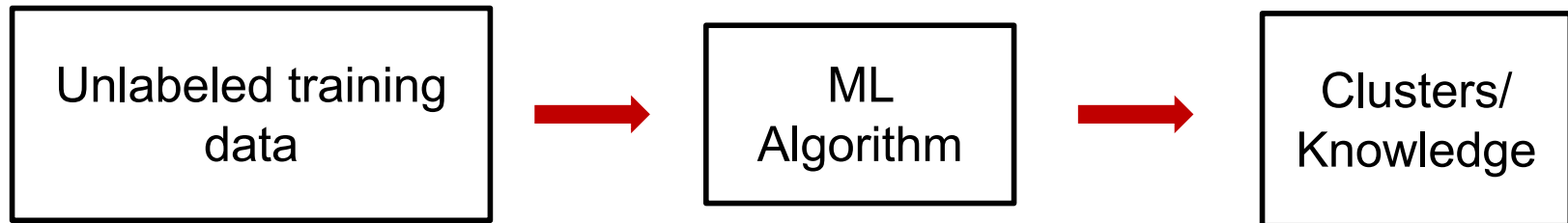
- Online Learning/Optimization
- Measure Algorithm Performance via Regret
- Warm-up: A Simple Example

Overview of Machine Learning

➤ Supervised learning



➤ Unsupervised learning



➤ Semi-supervised learning (a combination of the two)

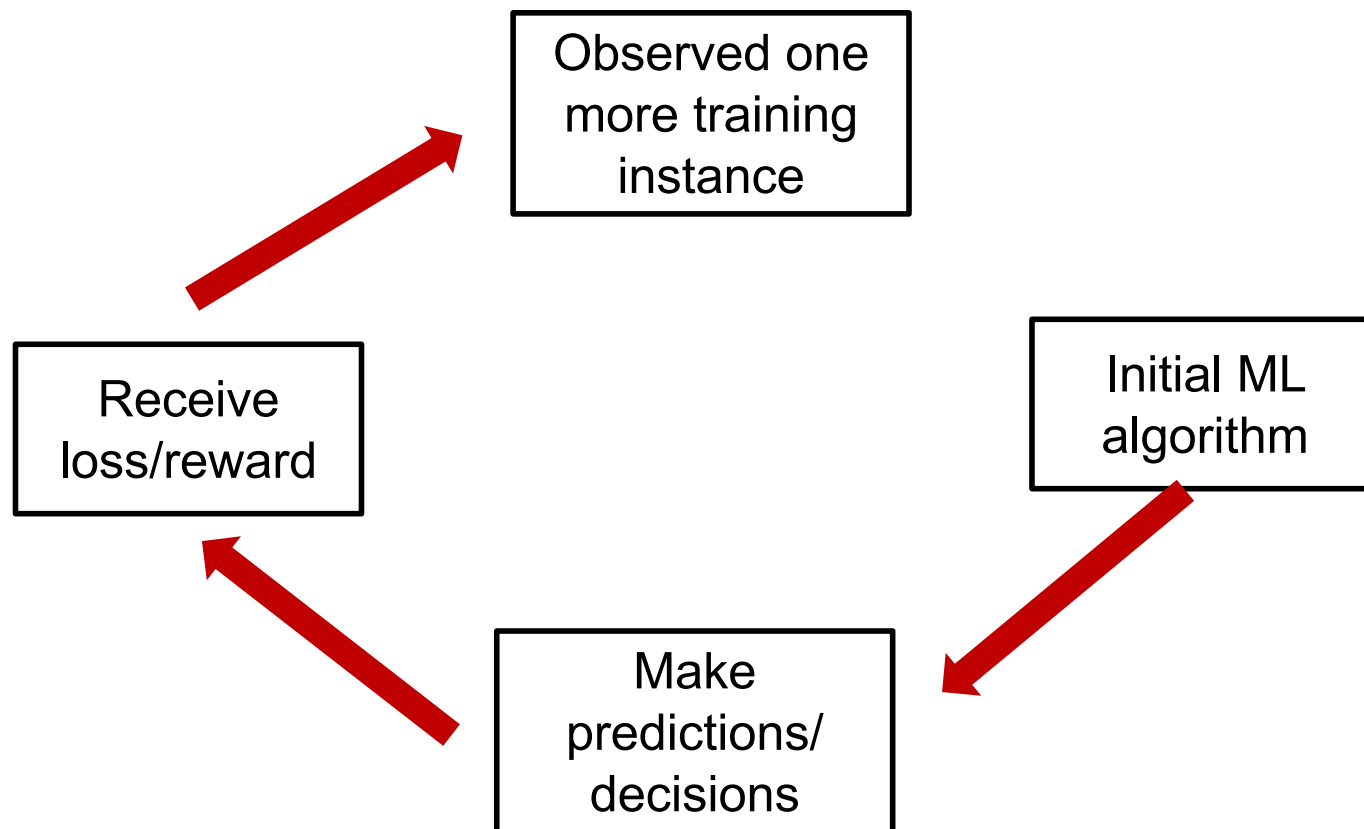
What else are there?

Overview of Machine Learning

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Online learning
- Reinforcement learning
- Active learning
-

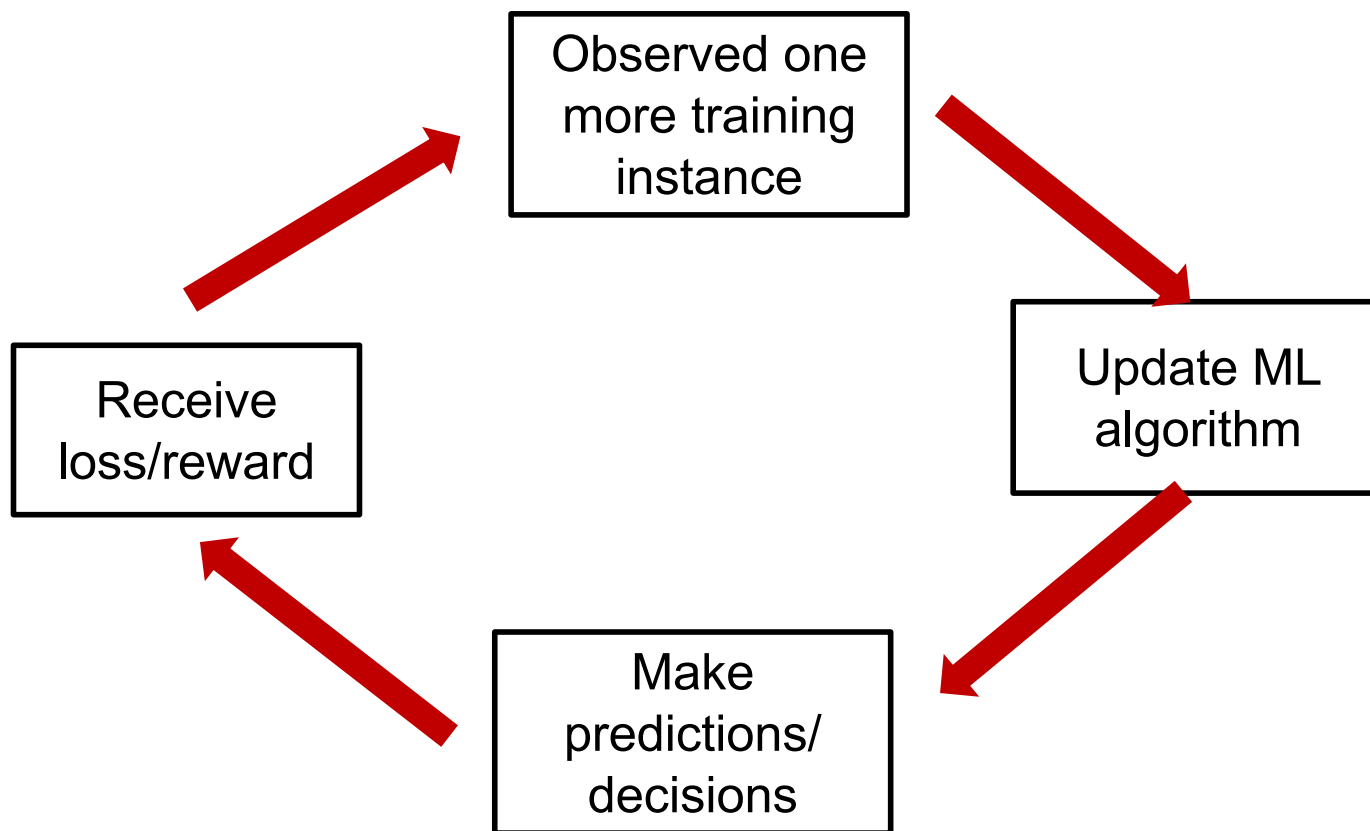
Online Learning: When Data Come Online

The online learning pipeline



Online Learning: When Data Come Online

The online learning pipeline

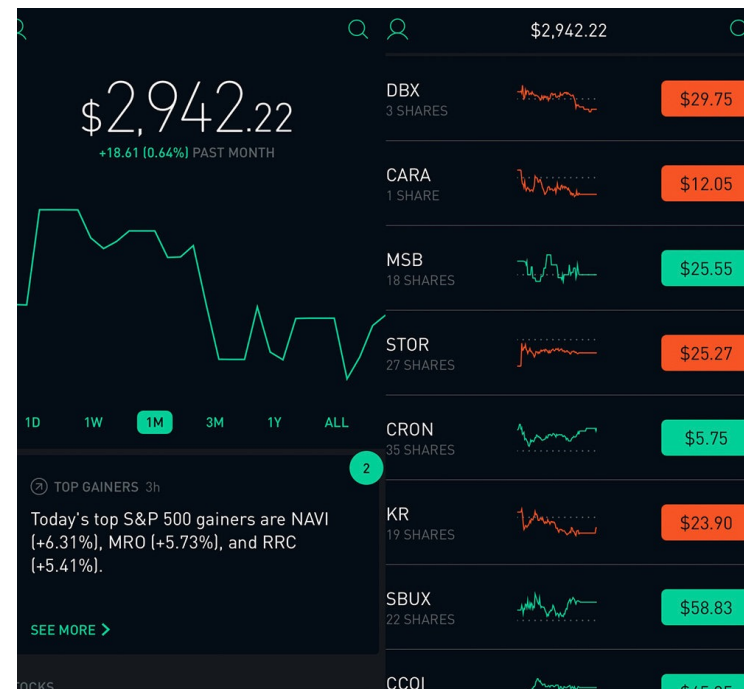


Typical Assumptions on Data

- Statistical feedback: instances drawn from a fixed distribution
 - Image classification, predict stock prices, choose restaurants, gambling machine (a.k.a., bandits)
- Adversarial feedback: instances are drawn adversarially
 - Spam detection, anomaly detection, game playing
- Markovian feedback: instances drawn from a distribution which is dynamically changing
 - Interventions, treatments

Online learning for Decision Making

- Learn to commute to school
 - Bus, walking, or driving? Which route? Uncertainty on the way?
- Learn to gamble or buy stocks



Online learning for Decision Making

- Learn to commute to school
 - Bus, walking, or driving? Which route? Uncertainty on the way?
- Learn to gamble or buy stocks
- Advertisers learn to bid for keywords

The image shows a Google search interface for the query "seo audit services". The search results page displays several entries. A red arrow labeled "ADWORDS" points to the first search result, which is an advertisement for "Certified, Proven SEO Company - Use an Agency You Can Trust" from www.thesearchmarketingshop.com.cy. Below the main text of the ad, there are two columns of smaller text: "Adverts on Search Engines" and "Shopping Ads" on the left, and "SEO Services" and "PPC Management" on the right. A second red arrow labeled "SEO" points to the second search result, which is a link to "SEO Audits | SEO Services | SEO Company | PushFire®" from https://pushfire.com/seo/.

Google

seo audit services

All Images News Maps Videos More Settings Tools

About 4,490,000 results (0.45 seconds)

Certified, Proven SEO Company - Use an Agency You Can Trust
Ad www.thesearchmarketingshop.com.cy/SEO/Agency +44 330 122 4960
Get a free **SEO audit** of your website from a certified **SEO** agency. Contact Now!
Ask a Question · Improve Your Campaign Now · Arrange a Meeting · Book a Proposal
Services: SEO, Shopping Feed Management, Search Engine Marketing

Adverts on Search Engines SEO Services
Shopping Ads PPC Management

SEO Audits | SEO Services | SEO Company | PushFire®
<https://pushfire.com/seo/>
Tired of **SEO** services that take shortcuts or attempt to game the latest algorithm? We offer defensible strategies, audits & ongoing services that work.

SEO Audit Service for your Website. Delivered by Real Person ...
<https://www.v9seo.com/seo-audit/>
Don't trust your **SEO** to an online audit tool! Let our team of experts deliver a **SEO Audit** that will bring you results in 60 days!

Online learning for Decision Making

- Learn to commute to school
 - Bus, walking, or driving? Which route? Uncertainty on the way?
- Learn to gamble or buy stocks
- Advertisers learn to bid for keywords
- Recommendation systems learn to make recommendations

The screenshot shows the Yelp website interface for Lexington, MA. At the top, there is a search bar with the text "Search for (e.g. taco, cheap dinner, Max's)" and a "Near" field with "Lexington, MA 02420". The main navigation bar includes links for "Welcome", "About Me", "Write a Review", "Find Reviews", "Find Friends", "Messaging", "Talk", "Events", and "Member Search". Below the navigation bar, the page is titled "Yelp Lexington" with a list of other cities: "Boston", "San Francisco", "New York", "San Jose", "Los Angeles", "Chicago", and "More Cities". A prominent yellow banner reads "Yelp is the best way to find great local businesses" with a "Create Your Free Account" button. To the right, there is a "Review of the Day" section featuring a review by Sarah D. for Beantown Taqueria, which has a 5-star rating. Below this, there is a "Yelp on the Go" section promoting the mobile app. The main content area is titled "The Best of Lexington" and lists various categories: "Restaurants" (5,575 reviewed), "Nightlife" (940 reviewed), "Food" (2,960 reviewed), "Shopping" (4,337 reviewed), "Bars" (684 reviewed), and "American (New)" (424 reviewed). The "Restaurants" section is expanded to show a list of top-rated businesses, including "1. Royal India Bistro" (5 stars, 61 reviews) and "2. Wagon Wheel Nursery and Farm Stand" (5 stars, 29 reviews).

Online learning for Decision Making

- Learn to commute to school
 - Bus, walking, or driving? Which route? Uncertainty on the way?
- Learn to gamble or buy stocks
- Advertisers learn to bid for keywords
- Recommendation systems learn to make recommendations
- Clinical trials
- Robotics learn to react
- Learn to play games (video games and strategic games)
- Even how you learn to make decisions in your life
- . . .

Model Sketch

- A learner acts in an uncertain world for T time steps
- Each step $t = 1, \dots, T$, learner takes action $i_t \in [n] = \{1, \dots, n\}$
- Learner observes **cost vector** c_t where $c_t(i) \in [0,1]$ is the cost of action $i \in [n]$
 - Learner suffers cost $c_t(i_t)$ at step t
 - Can be similarly defined as reward instead of cost, not much difference
 - There are also “partial feedback” models (will not cover here)
- Adversarial feedbacks: c_t is chosen by an adversary
 - The powerful adversary has access to all the history (learner actions, past costs, etc.) until $t - 1$ and also the learner’s algorithm
 - There are models of stochastic feedbacks (will not cover in this course)
- Learner’s goal: minimize $\sum_{t \in [T]} c_t(i_t)$

Formal Procedure of the Model

At each time step $t = 1, \dots, T$, the following occurs in order:

1. Learner picks a distribution p_t over actions $[n]$
2. Adversary picks cost vector $c_t \in [0,1]^n$ (he knows p_t)
3. Action $i_t \sim p_t$ is chosen and learner incurs cost $c_t(i_t)$
4. Learner observes c_t (for use in future time steps)

- Learner tries to pick distribution sequence p_1, \dots, p_T to minimize expected cost $\mathbb{E} [\sum_{t \in T} c_t(i_t)]$
 - Expectation over randomness of action
- The adversary does not have to really exist – it is assumed mainly for the purpose of **worst-case analysis**

Well, Adversary Seems Too Powerful?

- Adversary can choose $c_t \equiv 1, \forall t$; learner suffers cost T regardless
 - Cannot guarantee anything non-trivial? Are we done?
- If $c_t \equiv 1 \forall t$, if you look back at the end, you do not **regret** anything
 - had you known such costs **in hindsight**, you cannot do better
 - From this perspective, cost T in this case is not bad

So what is a good measure for the performance of an online learning algorithm?

Outline

- Online Learning/Optimization
- Measure Algorithm Performance via Regret
- Warm-up: A Simple Example

Regret

- Measures how much the learner regrets, had he known the cost vector c_1, \dots, c_T in hindsight
- Formally,

$$R_T = \mathbb{E}_{i_t \sim p_t} \sum_{t \in [T]} c_t(i_t) - \min_{i \in [n]} \sum_{t \in [T]} c_t(i)$$

- Benchmark $\min_{i \in [n]} \sum_t c_t(i)$ is the learner utility had he known c_1, \dots, c_T and is allowed to take the best **single action across all rounds**
 - There are other concepts of regret, e.g., swap regret (coming later)
 - But, $\min_{i \in [n]} \sum_t c_t(i)$ is mostly used

Regret is an appropriate performance measure of online algorithms

- It measures exactly the loss due to not knowing the data in advance

Average Regret

$$\bar{R}_T = \frac{R_T}{T} = \mathbb{E}_{i_t \sim p_t} \frac{1}{T} \sum_{t \in [T]} c_t(i_t) - \min_{i \in [n]} \frac{1}{T} \sum_{t \in [T]} c_t(i)$$

- When $\bar{R}_T \rightarrow 0$ as $T \rightarrow \infty$, we say the algorithm has **vanishing regret** or **no-regret**; the algorithm is called a no-regret online learning algorithm
- Equivalently, R_T is **sublinear** in T
 - Both are used, depending on your habits

Our goal: design no-regret algorithms by minimizing regret

A Naive Strategy: Follow the Leader (FTL)

- That is, pick the action with the smallest accumulated cost so far

What is the worst-case regret of FTL?

A Naive Strategy: Follow the Leader (FTL)

- That is, pick the action with the smallest accumulated cost so far

What is the worst-case regret of FTL?

Answer: worst (largest) regret $T/2$

- Consider following instance with 2 actions

t	1	2	3	4	5	...	T
$c_t(1)$	1	0	1	0	1	...	*
$c_t(2)$	0	1	0	1	0	...	*

- FTL always pick the action with cost 1 \rightarrow total cost T
- Best action in hindsight has cost at most $T/2$

Randomization is Necessary

In fact, any deterministic algorithm suffers (linear) regret $(n - 1)T/n$

- Recall, adversary knows history and learner's algorithm
 - So he can infer our p_t at time t (but do **not** know our sampled $i_t \sim p_t$)
- But if p_t is deterministic, action i_t can also be inferred
- Adversary simply sets $c_t(i_t) = 1$ and $c_t(i) = 0$ for all $i \neq i_t$
- Learner suffers total cost T
- Best action in hindsight has cost at most T/n

Can randomized algorithm achieve sublinear regret?

Outline

- Online Learning/Optimization
- Measure Algorithm Performance via Regret
- Warm-up: A Simple Example

Consider a Simpler (Special) Setting

- Binary costs for all actions, i.e., $c_t(i) \in \{0,1\}$
- One of the actions is perfect – it always has cost 0
 - Minimum cost in hindsight is thus 0
 - Learner does not know which action is perfect

Is it possible to achieve sublinear regret in this simpler setting?

A Natural Algorithm

Observations:

1. If an action ever had non-zero costs, it is not perfect
2. Actions with all zero costs so far, we do not really know how to distinguish them currently

These motivate to the following natural algorithm

For $t = 1, \dots, T$

- Identify the set of actions with zero total cost so far, and pick one action from the set uniformly at random.

Note: there is always at least one action to pick since the perfect action is always a candidate

Analysis of the Algorithm

- Fix a round t , we examine the expected loss from this round
- Let $S_{good} = \{\text{actions with zero total cost before } t\}$ and $k = |S_{good}|$
 - So each action in S_{good} is picked with probability $1/k$

Analysis of the Algorithm

- Fix a round t , we examine the expected loss from this round
- Let $S_{good} = \{\text{actions with zero total cost before } t\}$ and $k = |S_{good}|$
 - So each action in S_{good} is picked with probability $1/k$
- For any parameter $\epsilon \in [0,1]$, one of the following two happens
 - 😊 • **Case 1:** **at most** ϵk actions from S_{good} have cost 1, in which case we suffer expected cost **at most** ϵ
 - ☹️ • **Case 2:** **at least** ϵk actions from S_{good} have cost 1, in which case we suffer expected cost **at most** 1

Analysis of the Algorithm

- Fix a round t , we examine the expected loss from this round
- Let $S_{good} = \{\text{actions with zero total cost before } t\}$ and $k = |S_{good}|$
 - So each action in S_{good} is picked with probability $1/k$
- For any parameter $\epsilon \in [0,1]$, one of the following two happens
 - 😊 • **Case 1:** **at most** ϵk actions from S_{good} have cost 1, in which case we suffer expected cost **at most** ϵ
 - ☹ • **Case 2:** **at least** ϵk actions from S_{good} have cost 1, in which case we suffer expected cost **at most** 1
- How many times can **Case 2** happen?
 - Each time it happens, size of S_{good} shrinks from k to at most $(1 - \epsilon)k$
 - At most $\log_{1-\epsilon} n^{-1}$ times
- The total cost of the algorithm is at most $T \times \epsilon + \log_{1-\epsilon} n^{-1} \times 1$

Analysis of the Algorithm

➤ The cost upper bound can be further bounded as follows

$$\text{Total Cost} \leq T \times \epsilon + \log_{1-\epsilon} n^{-1} \times 1$$

$$= T\epsilon + \frac{\ln n}{-\ln(1-\epsilon)}$$

$$\text{Since } \log_a b = \frac{\ln b}{\ln a}$$

$$\leq T\epsilon + \frac{\ln n}{\epsilon}$$

$$\text{Since } -\ln(1-\epsilon) \geq \epsilon, \forall \epsilon \in (0,1)$$

➤ The above upper bound holds for any ϵ , so picking $\epsilon = \sqrt{\ln n / T}$ we have

$$R_T = \text{Total Cost} \leq 2\sqrt{T \ln n}$$

Sublinear in T



What about the General Case?

- $c_t \in [0,1]^n$
- No perfect action
- Previous algorithm can be re-written in a more “mathematically beautiful” way, which turns out to generalize

For $t = 1, \dots, T$

- Identify the set of actions with zero total cost so far, and pick one action from the set uniformly at random.

What about the General Case?

- $c_t \in [0,1]^n$
- No perfect action
- Previous algorithm can be re-written in a more “mathematically beautiful” way, which turns out to generalize

Initialize weight $w_1(i) = 1, \forall i = 1, \dots, n$

For $t = 1, \dots, T$

1. Let $W_t = \sum_{i \in [n]} w_t(i)$, pick action i with probability $w_t(i)/W_t$
2. Observe cost vector $c_t \in \{0,1\}^n$
3. For any $i \in [n]$, update $w_{t+1}(i) = w_t(i) \cdot (1 - c_t(i))$

What about the General Case?

- $c_t \in [0,1]^n$ → the weight update process is still okay
- No perfect action → more conservative when eliminating actions
- Previous algorithm can be re-written in a more “mathematically beautiful” way, which turns out to generalize

Initialize weight $w_1(i) = 1, \forall i = 1, \dots, n$

For $t = 1, \dots, T$

1. Let $W_t = \sum_{i \in [n]} w_t(i)$, pick action i with probability $w_t(i)/W_t$
2. Observe cost vector $c_t \in [0,1]^n$
3. For any $i \in [n]$, update $w_{t+1}(i) = w_t(i) \cdot (1 - \epsilon \cdot c_t(i))$

Multiplicative Weight Update (MWU)

Theorem. Multiplicative Weight Update (MWU) achieves regret at most $O(\sqrt{T \ln n})$ for the previously described general setting.

- Proof of the theorem is left to the next lecture
- Note: we really care about theoretical bound for online algorithms
 - The environment is uncertain and difficult to simulate, there is no easy way to experimentally evaluate the algorithm

Is $O(\sqrt{T \ln n})$ is best possible regret?

Next, we show $\sqrt{T \ln n}$ is tight

Lower Bound I

$(\ln n)$ term is necessary

- Consider any $T \approx \ln(n - 1)$
- Will construct a series of random costs such that there is a perfect action yet any algorithm will have **expected** cost $T/2$
 - At $t = 1$, randomly pick half actions to have cost 1 and remaining actions have cost 0
 - At $t = 2, 3, \dots, T$: among perfect actions so far, randomly pick half of them to have cost 1 and remaining actions have cost 0
- Since $T < \ln(n)$, at least one action remains perfect at the end
- But any algorithm suffers expected cost $1/2$ at each round (why?); The total cost will be $T/2$
- Costs are stochastic, not adversarial? → Will be provably worse when costs become adversarial
 - Just FYI: A formal proof is by Yao's minimax principle

Lower Bound 2

(\sqrt{T}) term is necessary

- Consider 2 actions only, still stochastic costs
- For $t = 1, \dots, T$, cost vector $c_t = (0,1)$ or $(1,0)$ uniformly at random
 - c_t 's are independent across t 's
- Any algorithm has 50% chance of getting cost 1 at each round, and thus suffers total expected cost $T/2$
- What about the best action in hindsight?
 - From action 1's perspective, its costs form a 0 – 1 bit sequence, each bit drawn independently and uniformly at random
 - $c[1] = \sum_{t \in T} c_t(1)$ is $Binomial(T, \frac{1}{2})$ and $c(2) = T - c[1]$
 - The cost of best action in hindsight is $\min(c[1], T - c[1])$
 - $\mathbb{E} \min(c[1], T - c[1]) = \frac{T}{2} - \Theta(\sqrt{T})$

Thank You

Haifeng Xu

University of Chicago

haifengxu@uchicago.edu