CPSC 532D — 7. ONLINE LEARNING

Bingshan Hu with Danica J. Sutherland
University of British Columbia, Vancouver
Fall 2024

In batch (offline) learning, usually, we have a learning phase and a prediction phase. Learner first receives a batch of i.i.d. training samples and then uses these data to learn a hypothesis, e.g., an ERM (learning phase). The learned hypothesis will be used for predicting the labels of future samples (prediction phase). In contrast, in online learning, there is no separation between the learning phase and the prediction phase. We blend these two phases together by specifying a learning protocol that regulates all parties participating in the learning.

7.1 ONLINE BINARY CLASSIFICATION IN REALIZABLE SETTING

Chapter 21.1 of [SSBD14].

Still, we have an instance space $\mathcal{X}$, a label space $\mathcal{Y} = \{-1, 1\}$, a hypothesis class $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$, and the $0-1$ loss function $\ell_{0-1}(y, \hat{y}) = \mathbf{1}\{y \neq \hat{y}\}$. We play this game sequentially with the following learning protocol.

This is a picture for this.

In each round $t = 1, 2, \ldots, \mathrm{T}$,
1. Nature (Adversary/Environment) selects $x_t \in \mathcal{X}$ and reveals it to Learner ;
2. Learner chooses a hypothesis $f_t \in \mathcal{H}$ and predicts $\hat{y}_t = f_t(x_t) \in \{-1, 1\}$;
3. Nature plays label y_t and reveals it to Learner ;
4. Learner obtains a data sample (x_t, y_t) and suffers loss $\ell(y_t, \hat{y}_t) = \mathbf{1}\{y_t \neq \hat{y}_t\}$.

If $\ell(y_t, \hat{y}_t) = \mathbf{1}\{y_t \neq \hat{y}_t\} = 1$, we say Learner makes a mistake in round t, as its predicted label is not correct. The goal of Learner is to make as few mistakes as possible. Intuitively, to choose a good predictor f_t in round t, Learner should use all the data samples attained in previous rounds $\mathrm{S}_{t-1} := ((x_1, y_1), (x_2, y_2), \ldots, (x_{t-1}, y_{t-1}))$ and even the input x_t in round t.

The data sequence $((x_1, y_1), (x_2, y_2), \ldots, (x_{\mathrm{T}}, y_{\mathrm{T}}))$ does not need to be iid, which is quite different from the assumption usually batch learning needs.

No *statistical assumption.*

Since Adversary can decide y_t based on $\hat{y}_t$, it is hopeless to learn for some some hypothesis class $\mathcal{H}$, e.g., two constant functions $x \mapsto +1$ and $x \mapsto -1$ are in $\mathcal{H}$. Because Adversary can make Learner unhappy in each round by declaring $y_t = -\hat{y}_t$.

So, we need to put some constraints for Adversary.

With the assumption of realizability, setting $y_t = -\hat{y}_t$ for all $t \in [\mathrm{T}]$ may not always be possible.

We get started with a simple learning problem setup. It is about the aforementioned binary classification problem under the assumption of *realizability*, i.e., the true label function $f^* \in \mathcal{H}$, i.e., $y_t = f^*(x_t)$ for all $t \in [\mathrm{T}]$, and Learner knows $\mathcal{H}$ and the fact that $f^* \in \mathcal{H}$. [1] With this restriction on how Adversary generates the data sequence,

For more, visit https://cs.ubc.ca/~dsuth/532D/24w1/.

[1] Nature is regulated to play a label function that is consistent with the history data S_{t-1}.

Learner should make as few mistakes as possible. We are interested in how many mistakes Learner makes after playing this sequential game after T rounds.

Hypothesis class $\mathcal{H}$ is finite. Since we know the true label function $f^* \in \mathcal{H}$, i.e., there is a perfect hypothesis incurring zero loss in $\mathcal{H}$, it is quite natural to eliminate any hypothesis in $\mathcal{H}$ that has made a mistake after observing (x_t, y_t) at the end of each round t. This intuition gives us an idea, called *version space*, to develop learning algorithms. The version space at the end of round t is defined as $\mathcal{H}_t = \{f \in \mathcal{H} : f(x_s) = y_s, \forall s \in [t]\}$. Then, we can maintain a sequence of version spaces $\mathcal{H} = \mathcal{H}_0 \supseteq \mathcal{H}_1 \supseteq \ldots \supseteq \mathcal{H}_{t-1} \supseteq \mathcal{H}_t \supseteq \ldots \supseteq \{f^*\}$. Hypotheses that have made mistakes will be kicked out from the version space at the end of each round t. We eliminate hypotheses from $\mathcal{H}$ during the learning once we are confident they are not good almost surely!

Let's see **Consistent Algorithm** first.

> Initially, we set $\mathcal{H}_0 = \mathcal{H}$.
> In each round $t = 1, 2, \ldots, \mathrm{T}$,
> 1. Learner receives x_t ;
> 2. Learner chooses $f_t \in \mathcal{H}_{t-1}$ *arbitrarily* and predicts $\hat{y}_t = f_t(x_t)$;
> 3. Nature reveals true label $y_t = f^*(x_t)$ and Learner updates $\mathcal{H}_t = \{f \in \mathcal{H}_{t-1} : f(x_t) = y_t\}$.

Note that the number of mistakes does not depend on the learning horizon T. *For any length of* T, *the number of mistakes is at most* $|\mathcal{H}| - 1$.

Now, we exam how many mistakes (the total loss $\sum_{t=1}^{\mathrm{T}} \ell(y_t, \hat{y}_t) = \sum_{t=1}^{\mathrm{T}} \ell(y_t, f_t(x_t))$) that **Consistent Algorithm** makes by the end of round T. It is not hard to see that the number of mistakes is at most $|\mathcal{H}| - 1$, as each mistake forces Learner to eliminate at least one hypothesis from the version space at the end of that round.

$f(x) = o(g(x))$ is equivalent to $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$ if $g(x) > 0$.

One may ask is it possible to reduce the number of mistakes from $|\mathcal{H}|$ to $o(|\mathcal{H}|)$?

The answer is yes!

Actually, we can do better if we do not pick f_t in an arbitrary way! We introduce another idea, called *majority vote*. Since some hypotheses in $\mathcal{H}_{t-1}$ predict +1 while some predict −1, we simply count which side has more supporters. The side of more supporters wins! Combining majority vote with version space, we have a much nicer algorithm, called **Halving Algorithm**, which reduces the number of mistakes from $\mathcal{O}(|\mathcal{H}|)$ to $\log |\mathcal{H}|$.

> Initially, we set $\mathcal{H}_0 = \mathcal{H}$.
> In round $t = 1, 2, \ldots, \mathrm{T}$,
> 1. Learner receives x_t ;
> 2. Learner predicts $\hat{y}_t \in \arg\max_{y \in \{-1,1\}} \left|\{f \in \mathcal{H}_{t-1} : f(x_t) = y\}\right|$;
> 3. Nature reveals true label $y_t = f^*(x_t)$ and Learner updates $\mathcal{H}_t = \{f \in \mathcal{H}_{t-1} : f(x_t) = y_t\}$.

Theorem 7.1. ***Halving Algorithm*** *makes at most* $\log_2(|\mathcal{H}|)$ *mistakes.*

Proof. According to the majority vote, the version space is halved on each mistake. If Learner makes a mistake in round t, we have $|\mathcal{H}_t| \leq \frac{1}{2} |\mathcal{H}_{t-1}|$. Let M be the total

number of mistakes. We have

$$1 \leq |\mathcal{H}_T| \leq |\mathcal{H}_0|\, 2^{-M} = |\mathcal{H}|\, 2^{-M} \quad ,$$

which yields $M \leq \log_2(|\mathcal{H}|)$. Note that f^* is always in the version spaces. So, we have $1 \leq |\mathcal{H}_T|$. □

General Hypothesis Class including $\mathcal{H}$ is infinite. Still under the assumption of realizability, now, let us consider a hypothesis class $\mathcal{H}$ that may be infinite in size. For some hypothesis class $\mathcal{H}$, Learner is able to have a strategy that guarantees it makes a finite number of mistakes. For other hypothesis classes, Nature can force Learner to make infinitely many mistakes. To gain an understanding of which hypothesis class falls in the former category and which falls in the latter, we need to have new concepts, something like VC dimension to measure the richness of a hypothesis class. We aim at characterizing online learnability. In particular, we target the following question: What is the optimal online classification learning algorithm for a given hypothesis class $\mathcal{H}$?

Littlestone dimension characterizes online learnability. A hypothesis $\mathcal{H}$ is online learnable if it has a finite Littlestone dimension denoted as Ldim($\mathcal{H}$). The idea of Littlestone dimension is to view online learning as a 2-player sequential game between Learner and Adversary. The job of Adversary is to force Learner to make mistakes while preserving realizability.

How does Adversary choose x_t to force Learner to make the maximum number of mistakes, while ensuring realizability? It is easy as Adversary can always choose $y_t = -\hat{y}_t$ for the first Ldim($\mathcal{H}$) rounds in the sequential game.

The strategy for Adversary can be formally described by using a complete binary tree. Each node of the tree is associated with an instance $x_t \in \mathcal{X}$. If Learner predicts $\hat{y}_t = +1$, Adversary will declare the prediction is wrong, i.e., $y_t = -\hat{y}_t = -1$ and will traverse to the right child of the current node. If Learner predicts $\hat{y}_t = -1$, Adversary will set $y_t = -\hat{y}_t = +1$ and will traverse to the left child of the current node.

Add a picture of the binary tree.

To introduce the definition of Litterstone dimension, let us give the definition of an $\mathcal{H}$ shattered tree first.

The depth of the tree is defined as the number of edges in a path from the root to a leaf, i.e., the number of layers in the tree.

Definition 7.2. ($\mathcal{H}$ shattered tree.) A shattered tree of depth d is a sequence of inputs $v_1, v_2, \ldots, v_{2^d-1} \in \mathcal{X}$ such that for every root-to-leaf path, $\exists f^* \in \mathcal{H}$ such that all labels along the path are achieved.

Definition 7.3. (Littlestone dimension.) The Littlestone dimension of hypothesis class $\mathcal{H}$, Ldim($\mathcal{H}$), is the maximal integer d such that there exists a shattered tree of depth d that $\mathcal{H}$ shatters. If there is no such largest d, then the Litterstone dimension is infinite.

Theorem 7.4. *Any algorithm makes* at least *Ldim($\mathcal{H}$) mistakes.*

Proof. Let $d_* = \text{Ldim}(\mathcal{H})$. Since $\mathcal{H}$ has Littlestone dimension d_*, we know there exists an $\mathcal{H}$ shattered tree with depth d_* that is shattered by $\mathcal{H}$. Adversary will "walk" on the tree for the first d_* rounds. In each round $t \in [d_*]$, Adversary sets $y_t = -\hat{y}_t$. Since the walk continues for d_* rounds, Learner makes d_* mistakes. Now, we check the assumption of realizability. Since $\mathcal{H}$ shatters the tree, there $\exists f_* \in \mathcal{H}$ such that all the labels along the root-to-leaf path selected by Adversary can be realized. □

Now, we show a learning algorithm, **Standard Optimal Algorithm (SOA)**, makes at most $\text{Ldim}(\mathcal{H})$ mistakes. The algorithm is similar to **Halving Algorithm**. We partition the version space by the end of round $t-1$ into two sub-version spaces. Let $\mathcal{H}_{t-1}^{(-1)} := \{f \in \mathcal{H}_{t-1} : f(x_t) = -1\}$ and $\mathcal{H}_{t-1}^{(+1)} := \{f \in \mathcal{H}_{t-1} : f(x_t) = +1\}$.

Initially, we set $\mathcal{H}_0 = \mathcal{H}$.
In round $t = 1, 2, \ldots, \text{T}$,

1. Learner receives x_t ;
2. Learner predicts $\hat{y}_t \in \arg\max_{y \in \{-1,1\}} \text{Ldim}\left(\mathcal{H}_{t-1}^{(y)}\right)$;
3. Nature reveals true label $y_t = f^*(x_t)$ and Learner updates $\mathcal{H}_t = \{f \in \mathcal{H}_{t-1} : f(x_t) = y_t\}$.

Theorem 7.5. *SOA makes* at most *$\text{Ldim}(\mathcal{H})$ mistakes.*

Proof. □

Corollary 7.6. *Let $\mathcal{H}$ be any hypothesis class. Then, SOA makes exactly $\text{Ldim}(\mathcal{H})$ mistakes.*

Proof. From Theorem 7.4, we know SOA makes at least $\text{Ldim}(\mathcal{H})$ mistakes. From Theorem 7.5, we know SOA makes at most $\text{Ldim}(\mathcal{H})$ mistakes. Combining these two results concludes the proof. □

VC dimension vs Littlestone Dimension.

Example 1. If $\mathcal{H}$ is a finite hypothesis class, we have $\text{Ldim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$. why?

Example 2. Let $\mathcal{X} = [0,1]$ and $\mathcal{H} = \{x \mapsto \mathbf{1}\{x < a\} : a \in [0,1]\}$ be the class of thresholds on the interval $[0,1]$. Then, we have $\text{VCdim}(\mathcal{H}) = 1$, but $\text{Ldim}(\mathcal{H}) = \infty$.

There is a picture for this.

Theorem 7.7. *For any hypothesis class $\mathcal{H}$, we have $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$. Further, the gap can be arbitrarily large.*

Proof. We first prove that $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$. Suppose $\text{VCdim}(\mathcal{H}) = d$ and let $\{x_1, x_2, \ldots, x_d\}$ be a shattered set by $\mathcal{H}$. Now, we construct a complete binary tree of inputs $v_1, v_2, \ldots, v_{2^d-1}$, where all nodes at depth i are set to be x_i.

There is a picture for this.

Now, the definition of a shattered set clearly implies that we constructed a valid shattered tree of depth d and conclude that $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$.

The class of threshold functions on the unit interval has VC dimension of 1, whereas its Littlestone dimension is infinite. □

7.2 DECISION-THEORETICAL ONLINE LEARNING AND EXPONENTIAL WEIGHTS (HEDGE)

Practically, we should not assume realizability always holds, i.e., the true labels $y_t = f^*(x_t), \forall t \in [\mathrm{T}]$, are generated by using f^*. Similar to agnostic setting in supervised learning, now, we can compare with the best predictor in $\mathcal{H}$. In online learning, we use the notion of regret, defined as

$$\mathcal{R}(\mathrm{T}) := \sup_{((x_1,y_1),\ldots,(x_\mathrm{T},y_\mathrm{T}))} \left\{ \sum_{t=1}^{\mathrm{T}} \ell(y_t, f_t(x_t)) - \min_{f\in\mathcal{H}} \sum_{t=1}^{\mathrm{T}} \ell(y_t, f(x_t)) \right\} , \tag{7.1}$$

to measure the performance gap between $\sum_{t=1}^{\mathrm{T}} \ell(y_t, f_t(x_t))$, the total loss Learner has made, and $\min_{f\in\mathcal{H}} \sum_{t=1}^{\mathrm{T}} \ell(y_t, f(x_t))$, the total loss of the best predictor in hindsight. If a learning algorithm achieves an $o(\mathrm{T})$ regret bound, we say it is a *no-regret* algorithm. The intuition is if you play this game long enough, you can compete with the best predictor in hindsight. Note that $o(\mathrm{T})$ regret implies $\lim_{\mathrm{T}\to\infty} \frac{o(\mathrm{T})}{\mathrm{T}} = 0$.

To control the regret at Learner's side, generally, we have two key principles.

1. *Randomization.* Deterministic Learner fails for some learning problems. Let hypothesis class $\mathcal{H} = \{f_+ : x \to +1, f_- : x \to -1\}$ only contain two constant functions. Adversary can always give Learner $y_t = -\hat{y}_t$ to force Learner to suffer loss. So, Learner suffers in total T loss. Now, we investigate the total loss in hindsight of the best predictor in $\mathcal{H}$, that is, $\min_{f\in\mathcal{H}} \sum_{t=1}^{\mathrm{T}} \ell(y_t, f(x_t)) \le \frac{1}{2} \cdot \left(\sum_{t=1}^{\mathrm{T}} \ell(y_t, f_{-1}(x_t)) + \sum_{t=1}^{\mathrm{T}} \ell(y_t, f_+(x_t)) \right) = 0.5\mathrm{T}$, which yields the regret is at least $0.5\mathrm{T}$.

   Since Learner always fails if revealing the predicted label $\hat{y}_t$, to make the problem interesting, we need to give some power to Learner. Now, we change the learning protocol a bit by allowing Learner to reveal only a probability distribution over $\{-1, 1\}$ instead of revealing the predicted label $\hat{y}_t$ itself.

2. *Exploitation.* We want to track the empirical performance of each predictor, but we do not eliminate any of them during the learning, as some of them may not perform well at the beginning of the learning, but later turns out to the best one.

   Here, exploitation refers to utilizing the already learned information, i.e., the total loss of each predictor observed so far.

   Since we do not plan to eliminate predictors during the learning, we can maintain a *(data-dependent) distribution* over all the predictors in $\mathcal{H}$ taking account of each predictor's total loss suffered so far. For some predictor performing poorly, we put a small mass on it. We eliminate some predictor in a soft way!

   If the data sequence are i.i.d. according to some fixed but unknown distribution, we can eliminate predictors.

Decision-Theoretical Online Learning (DTOL). Since Learner is allowed to reveal a probability distribution over the labels and revealing a distribution over $\{-1, 1\}$ can be translated to revealing a distribution over $\mathcal{H}$, now, we can re-formulate the online learning problem slightly and the modification will be, generally, useful for bandit problems (and even reinforcement learning problems). We do not use training samples at all. We get rid of the input space, label space, and hypothesis class. Instead, we have a fixed set of K actions, denoted by $[\mathrm{K}]$. The learning protocol

You can view each action in $[\mathrm{K}]$ *as a hypothesis in* $\mathcal{H}$.

is modified as follows.

Sometimes, Learner chooses $p_t \in \{e_1, e_2, \ldots, e_K\}$.

In each round $t = 1, 2, \ldots, T$,
1. Learner plays a probability distribution $p_t \in [0,1]^K$ over all actions ;
2. Nature plays loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \ldots, \ell_{K,t})$;
3. Learner observes ℓ_t and suffer a loss $\langle p_t, \ell_t \rangle = \sum_j p_{j,t}\ell_{j,t}$.

Remarks. (1) A probability distribution $p_t = (p_{1,t}, p_{2,t}, \ldots, p_{K,t})$ is a vector with all $p_{j,t} \geq 0$ and $\sum_j p_{j,t} = 1$. (2) In Step 2, Nature can give ℓ_t based on all the past information and even p_t.

For DTOL, we adapt the notion of regret shown in (7.1) to

$\ell_{j,t} = \langle e_j, \ell_t \rangle$.

$$\mathcal{R}(T) := \sup_{\ell_1, \ell_2, \ldots, \ell_T} \left\{ \sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t} \right\} \quad . \tag{7.2}$$

Exponential Weights (Hedge). There is a no-regret algorithm for DTOL. It has many names such as "Exponential Weights" and "Hedge". The idea of Hedge is to maintain weights over actions, and the weight of an action decays exponentially in the total loss incurred by that action over all previous rounds.

Input: learning rate $\eta \in (0, 1]$.
Initialize: $w_{j,0} = 1$ for all $j \in [K]$.
For $t = 1, 2, \ldots, T$,
1. Set $p_{j,t} = \frac{w_{j,t-1}}{\sum_{j'} w_{j',t-1}}$ for all $j \in [K]$;
2. Observe loss vector ℓ_t for Nature ;
3. Suffer loss $\langle p_t, \ell_t \rangle = \sum_j p_{j,t}\ell_{j,t}$;
4. Update $w_{j,t} = w_{j,t-1} \cdot e^{-\eta \ell_{j,t}}$ for all $j \in [K]$.

Theorem 7.8. *For any sequence of loss vectors $(\ell_1, \ell_2, \ldots, \ell_T) \in \left([0,1]^K\right)^T$, for any $\eta \in (0, 1]$, we have*

$$\sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t} \quad \leq \quad \frac{\eta}{2} \sum_{t=1}^{T} \left\langle p_t, \ell_t^2 \right\rangle + \frac{\log K}{\eta} \quad .$$

With $\eta = \sqrt{\frac{2 \log K}{T}}$, we have $\mathcal{R}(T) \leq \sqrt{2T \log K}$.

Proof. of Theorem 7.8: Fix a sequence of loss vectors $(\ell_1, \ell_2, \ldots, \ell_T)$. Let $Z_t = \sum_{j=1}^{K} w_{j,t} = \sum_{j=1}^{K} w_{j,t-1} \cdot e^{-\eta \ell_{j,t}} = \sum_{j=1}^{K} e^{-\eta \sum_{s=1}^{t} \ell_{j,s}}$ be the total weight at the end of round t. Note that $Z_0 = K$. We have $\log Z_T = \sum_{t=1}^{T} \log \frac{Z_t}{Z_{t-1}} + \log Z_0$.

Now, we construct the following upper bound to upper bound $\log Z_T$. We have

$$\begin{aligned}
\log \frac{Z_t}{Z_{t-1}} &= \log \frac{\sum_{j=1}^{K} w_{j,t-1} \cdot e^{-\eta \ell_{j,t}}}{\sum_{j'=1}^{K} w_{j',t-1}} \\
&= \log \left(\sum_{j=1}^{K} \frac{w_{j,t-1}}{\sum_{j'=1}^{K} w_{j',t-1}} \cdot e^{-\eta \ell_{j,t}} \right) \\
&= \log \left(\sum_{j=1}^{K} p_{j,t} \cdot e^{-\eta \ell_{j,t}} \right) \\
&\leq^{(a)} \log \left(\sum_{j=1}^{K} p_{j,t} \cdot \left(1 - \eta \cdot \ell_{j,t} + \frac{\eta^2 \cdot \ell_{j,t}^2}{2} \right) \right) \\
&= \log \left(1 - \eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \left\langle p_t, \ell_t^2 \right\rangle \right) \\
&\leq^{(b)} -\eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \left\langle p_t, \ell_t^2 \right\rangle,
\end{aligned}$$

where step (a) uses $e^{-x} \leq 1 - x + x^2/2$ and step (b) uses $\log(1 + x) \leq x$.

We also have a lower bound on $\log Z_T$, which is

$$\begin{aligned}
\log Z_T &= \log \left(\sum_{j=1}^{K} e^{-\eta \sum_{t=1}^{T} \ell_{j,t}} \right) \\
&\geq \log \left(\max_{j \in [K]} \left\{ e^{-\eta \sum_{t=1}^{T} \ell_{j,t}} \right\} \right) \\
&= \max_{j \in [K]} \left\{ -\eta \sum_{t=1}^{T} \ell_{j,t} \right\}.
\end{aligned} \tag{7.3}$$

Now, we have

$$\begin{aligned}
&\sum_{t=1}^{T} -\eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \left\langle p_t, \ell_t^2 \right\rangle \geq \sum_{t=1}^{T} \log \frac{Z_t}{Z_{t-1}} = \log Z_T - \log Z_0 \geq \max_{j \in [K]} \left\{ -\eta \sum_{t=1}^{T} \ell_{j,t} \right\} - \log K. \\
\Rightarrow \quad &\sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{j \in [K]} \left\{ \sum_{t=1}^{T} \ell_{j,t} \right\} \leq \frac{\eta}{2} \left\langle p_t, \ell_t^2 \right\rangle + \frac{\log K}{\eta} \quad .
\end{aligned} \tag{7.4}$$

□

Remark. Note that in order to achieve the $\sqrt{2T \log K}$ regret bound, **Hedge** needs to input the learning rate $\eta = \sqrt{\frac{2 \log K}{T}}$, depending on the learning horizon T. Later, we will show how to use doubling-trick to get rid of it!

Hedge with Doubling-Trick. The regret bound $\mathcal{R}(T) = \mathcal{O}\left(\sqrt{T \log K}\right)$ in Theorem 7.8 relies on inputting the learning rate $\eta = \sqrt{\frac{2 \log K}{T}}$ which relies on the knowledge of the time horizon T.

One may be curious to know is it possible to have a learning algorithm that does not need to know T in advance, but still preserving the same regret bound?

The answer is yes!!!

We introduce a useful idea, called *doubling-trick*, for solving online learning problems. Using *(Geometric) doubling-trick*, we can still achieve an $\mathcal{O}\left(\sqrt{T \log K}\right)$ regret *Geometric doubling vs exponential doubling can be found here.*

bound. The idea is to run the algorithm in epochs of lengths $2^0, 2^1, \ldots, 2^r, \ldots$ until stopping.

At the beginning of each epoch $r \geq 0$, we set the learning rate $\eta_r = \sqrt{\frac{2\log K}{2^r}}$ based on 2^r the length of the current epoch. At the end of epoch r, we reset the algorithm, that is, we forget all the stuff we have learned. Progressing in this way, we will run the algorithm for epochs $r = 0, 1, \ldots, d$, where $d = \lceil \log_2(T+1) - 1 \rceil = \mathcal{O}(\log T)$.

Theorem 7.9. *The regret of Hedge with doubling-trick is* $\mathcal{O}\left(\sqrt{T \log K}\right)$.

Proof. For any $r \geq 0$, from (7.8), we have

$$\sum_{t=2^r+1}^{2^{r+1}} \langle p_t, \ell_t \rangle - \min_{j_r \in [K]} \sum_{t=2^r+1}^{2^{r+1}} \ell_{j_r,t} \quad \leq \quad \frac{\eta_r}{2} \sum_{t=2^r+1}^{2^{r+1}} \left\langle p_t, \ell_t^2 \right\rangle + \frac{\log K}{\eta_r} \quad \leq \quad \mathcal{O}\left(\sqrt{2^r \log K}\right) \quad . \tag{7.5}$$

The regret is

$$\begin{aligned}
\mathcal{R}(T) &= \sup_{\ell_1, \ell_2, \ldots, \ell_T} \left\{ \sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t} \right\} \\
&= \sup_{\ell_1, \ell_2, \ldots, \ell_T} \left\{ \sum_{r \geq 0} \sum_{t=2^r+1}^{2^{r+1}} \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{r \geq 0} \sum_{t=2^r+1}^{2^{r+1}} \ell_{j,t} \right\} \\
&\leq \sup_{\ell_1, \ell_2, \ldots, \ell_T} \left\{ \sum_{r \geq 0} \sum_{t=2^r+1}^{2^{r+1}} \langle p_t, \ell_t \rangle - \sum_{r \geq 0} \min_{j_r \in [K]} \sum_{t=2^r+1}^{2^{r+1}} \ell_{j_r,t} \right\} \\
&= \sup_{\ell_1, \ell_2, \ldots, \ell_T} \left\{ \sum_{r \geq 0} \left(\sum_{t=2^r+1}^{2^{r+1}} \langle p_t, \ell_t \rangle - \min_{j_r \in [K]} \sum_{t=2^r+1}^{2^{r+1}} \ell_{j_r,t} \right) \right\} \\
&= \sup_{\ell_1, \ell_2, \ldots, \ell_T} \left\{ \sum_{r \geq 0} \mathcal{O}(\sqrt{2^r \log K}) \right\} \\
&= \sum_{r \geq 0} \mathcal{O}(\sqrt{2^r \log K}) \\
&= \mathcal{O}\left(\sqrt{T \log K}\right) \quad .
\end{aligned} \tag{7.6}$$

The last epoch may not have a full length, but we can add $\vec{0}$'s to make the last epoch have a full length

□

7.3 BANDITS

Last time we have talked about Hedge/DTOL learning. We usually say it is a full information game as Learner is able to observe each individual loss in $\ell_t = (\ell_{1,t}, \ell_{2,t}, \ldots, \ell_{K,t})$. Starting from this lecture, we will talk about multi-armed bandit (MAB) problems, where only some entry in the loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \ldots, \ell_{K,t})$ is revealed in each round.

Learning Protocol. We have a fixed arm set $[K]$.

Actions and arms are interchangeable. You can view an arm as a hypothesis.

In each round $t = 1, 2, \ldots, T$,

1. Adversary/Environment selects a loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \ldots, \ell_{K,t})$ that is hidden to Learner ;
2. Learner pulls an arm $J_t \in [K]$;
3. Learner suffers/observes loss $\ell_{J_t,t}$ associated with the pulled arm J_t.

You can view Step 1 and Step 2 occur simultaneously.

In Step 3, the remaining losses other than $\ell_{J_t,t}$ are still hidden to Learner, which is the key difference between DTOL and bandits.

The goal of Learner to pull a sequence of arms $(J_1, J_2, \ldots, J_T)$ to minimize the total loss by the end of round T.

Regret is defined as

$$\mathcal{R}(T) := \sum_{t=1}^{T} \ell_{J_t,t} - \min_{j\in[K]} \sum_{t=1}^{T} \ell_{j,t} \quad , \tag{7.7}$$

which is a random variable as $J_1, J_2, \ldots, J_T$ and all loss vectors are random.

Based on how the loss vectors $(\ell_1, \ell_2, \ldots, \ell_T)$ are generated, we have

1. Adversary bandits: no distributional assumption is made.
2. Stochastic bandits: all ℓ_t are i.i.d. over time according to a fixed but unknown probability distribution.

Learner needs to make a good balance between

1. *Exploitation*: Learner needs to pull arms that have smaller losses, as the goal is to minimize the total loss.
2. *Exploration*: Learner needs to pull arms that have not been observed too often to gain information.

7.3.1 *Adversarial bandits.*

In adversarial bandits, the loss vectors can be generated adversarially. We use *pseudo regret* to measure performance of the algorithm used by Learner, defined as

$$\overline{\mathcal{R}}(T) := \mathbb{E}\left[\sum_{t=1}^{T} \ell_{J_t,t}\right] - \min_{j\in[K]} \mathbb{E}\left[\sum_{t=1}^{T} \ell_{j,t}\right] \quad , \tag{7.8}$$

where the expectation is taken over $(J_1, J_2, \ldots, J_T)$ and all loss vectors over T rounds.

Sometimes people call it expected regret as it has an expectation. I am following the notation of Bubeck and Cesa-Bianchi [BC12].

EXP3 (Exponential weights for Exploration and Exploitation). It is quite similar to Hedge, but in EXP3, only the weight associated with the pulled arm will be updated at the end of each round t, as only the loss for that arm is revealed.

We construct an estimator $\tilde{\ell}_t = \left(\tilde{\ell}_{1,t}, \tilde{\ell}_{2,t}, \ldots, \tilde{\ell}_{K,t}\right)$ with each entry $\tilde{\ell}_{j,t} = \ell_{j,t} \frac{\mathbf{1}\{J_t = j\}}{p_{j,t}}$.

Importance sampling

It is not hard to see that for all arms not played in round t, we set $\tilde{\ell}_{j,t} = 0$. For the pulled arm in round t, we set $\tilde{\ell}_{\mathrm{J}_t,t} = \frac{\ell_{\mathrm{J}_t,t}}{p_{\mathrm{J}_t,t}}$ instead of $\ell_{\mathrm{J}_t,t}$ by making it more "important"! Actually, the constructed estimator $\tilde{\ell}_{j,t}$ is an unbiased estimator of $\ell_{j,t}$, as we have

$$\mathbb{E}_{\mathrm{J}_t \sim p_t}\left[\tilde{\ell}_{j,t}\right] \;=\; \mathbb{E}_{\mathrm{J}_t \sim p_t}\left[\ell_{j,t}\frac{\mathbf{1}\{\mathrm{J}_t = j\}}{p_{j,t}}\right] = \ell_{j,t} \quad . \tag{7.9}$$

Input: learning rate $\eta \in (0,1]$. Initialize: $w_{j,0} = 1$ for all $j \in [\mathrm{K}]$.
For $t = 1, 2, \ldots, \mathrm{T}$,

1. Adversary/Environment selects a loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \ldots, \ell_{\mathrm{K},t})$ that is hidden to Learner ;
2. Learner computes $p_{j,t} = \frac{w_{j,t-1}}{\sum_{j'} w_{j',t-1}}$ for all $j \in [\mathrm{K}]$;
3. Learner plays arm $\mathrm{J}_t \in [\mathrm{K}]$ according to $p_t = (p_{1,t}, p_{2,t}, \ldots, p_{\mathrm{K},t})$;
4. Learner computes loss estimates $\tilde{\ell}_{j,t} = \frac{\ell_{j,t}}{p_{j,t}}\mathbf{1}\{\mathrm{J}_t = j\}$ for all $j \in [\mathrm{K}]$;
5. Update $w_{j,t} = w_{j,t-1} \cdot e^{-\eta\tilde{\ell}_{j,t}}$ for all $j \in [\mathrm{K}]$.

Theorem 7.10. *Assume that all loss vectors are bounded with $[0,1]$ support. If **EXP3** is run with learning rate $\eta = \sqrt{\frac{\log \mathrm{K}}{\mathrm{KT}}}$, the pseudo regret is at most $\sqrt{2\mathrm{TK}\log \mathrm{K}}$.*

Proof.

$$\begin{aligned}
\overline{\mathcal{R}}(\mathrm{T}) &= \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \ell_{\mathrm{J}_t,t}\right] - \min_{j\in[\mathrm{K}]}\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \ell_{j,t}\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \left\langle p_t, \tilde{\ell}_t\right\rangle\right] - \min_{j\in[\mathrm{K}]}\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathrm{E}_{\mathrm{J}_t\sim p_t}\left[\tilde{\ell}_{j,t}\right]\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \left\langle p_t, \tilde{\ell}_t\right\rangle\right] - \min_{j\in[\mathrm{K}]}\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \tilde{\ell}_{j,t}\right] \\
&\le \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \left\langle p_t, \tilde{\ell}_t\right\rangle\right] - \mathbb{E}\left[\min_{j\in[\mathrm{K}]}\sum_{t=1}^{\mathrm{T}} \tilde{\ell}_{j,t}\right] \\
&= \mathbb{E}\left[\underbrace{\sum_{t=1}^{\mathrm{T}} \left\langle p_t, \tilde{\ell}_t\right\rangle - \min_{j\in[\mathrm{K}]}\sum_{t=1}^{\mathrm{T}} \tilde{\ell}_{j,t}}_{\textbf{regret of Hedge}}\right] \\
&\le^{(a)} \mathbb{E}\left[\frac{\eta}{2}\sum_{t=1}^{\mathrm{T}} \left\langle p_t, \tilde{\ell}_t^2\right\rangle + \frac{\log \mathrm{K}}{\eta}\right] \\
&=^{(b)} \mathbb{E}\left[\frac{\eta}{2}\sum_{t=1}^{\mathrm{T}}\sum_{j\in[\mathrm{K}]} p_{j,t}\frac{\ell_{j,t}^2}{p_{j,t}}\right] + \frac{\log \mathrm{K}}{\eta} \\
&\le \frac{\eta}{2}\mathrm{KT} + \frac{\log \mathrm{K}}{\eta}.
\end{aligned} \tag{7.10}$$

Tuning $\eta = \sqrt{\frac{\log \mathrm{K}}{\mathrm{KT}}}$ gives the stated regret bound.

Step (a) uses Theorem 7.8 in previous lecture. Note that from (7.9), we know $\mathbb{E}_{\mathrm{J}_t\sim p_t}\left[\tilde{\ell}_{j,t}\right] \in [0,1]$. So, it is safe to use Theorem 7.8 directly.

Step (b) uses $\mathbb{E}_{\mathrm{J}_t\sim p_t}\left[\tilde{\ell}_{j,t}^2\right] = \mathbb{E}_{\mathrm{J}_t\sim p_t}\left[\ell_{j,t}^2\frac{\mathbf{1}\{\mathrm{J}_t=j\}}{p_{j,t}^2}\right] = \frac{\ell_{j,t}^2}{p_{j,t}}$. □

Remark. Note that inputting $\eta = \sqrt{\frac{\log K}{KT}}$ into EXP3 means that it is not an anytime learning algorithm. To make it anytime, you can set the learning rate $\eta_t = \sqrt{\frac{\log K}{tK}}$ in each round t. The regret analysis is much more complicated (refer to Theorem 3.1 in [BC12]).

7.3.2 *Stochastic bandits*

In stochastic bandits, we have a fixed arm set $[K]$ and each arm $j \in [K]$ is associated with a reward distribution v_j. We can use $\Theta_K := (v_1, v_2, \ldots, v_K)$ to specify a K-armed stochastic bandit problem instance.

In each round $t = 1, 2, \ldots, T$,
1. Environment generate a reward vector $X_t = (X_{1,t}, X_{2,t}, \ldots, X_{K,t})$ with each $X_{j,t} \sim v_j$. This reward vector is hidden to Learner ;
2. Learner pulls an arm $J_t \in [K]$;
3. Learner obtains/observes $X_{J_t,t}$, the reward of the pulled arm J_t.

We still use pseudo regret to measure performance of **Alg** , defined as

$$\overline{\mathcal{R}}(\mathbf{Alg}; \Theta_K; T) := \max_{j\in[K]} \mathbb{E}\left[\sum_{t=1}^{T} X_{j,t}\right] - \mathbb{E}\left[\sum_{t=1}^{T} X_{J_t,t}\right] , \tag{7.11}$$

where the randomness is taken over $J_1, J_2, \ldots, J_T$ and all reward vectors. Note that since we have statistical assumptions on reward vectors, different Θ_K may give different regret. That is also to say, for a fixed algorithm, when working over different problem instances, the regret could be different.

The goal of Learner is to pull arms sequentially to minimize regret.

Let $\mu_j = \mathbb{E}_{X_j \sim v_j}[X_j]$ denote the mean reward of arm j. Without loss of generality, we assume the first arm is the optimal one, that is, $\mu_1 > \mu_j$ for all $j \neq 1$.

For any $j \neq 1$, let $\Delta_j := \mu_1 - \mu_j$ denote the mean reward gap. We also call it the sub-optimality gap between the optimal arm 1 and the sub-optimal arm j. Let $\Delta_1 = 0$.

Now, we can rewrite $\overline{\mathcal{R}}(T)$ as

$$\begin{aligned}
\overline{\mathcal{R}}(T) &= \max_{j\in[K]} \mathbb{E}\left[\sum_{t=1}^{T} X_{j,t}\right] - \mathbb{E}\left[\sum_{t=1}^{T} X_{J_t,t}\right] \\
&= T \cdot \mu_1 - \sum_{t=1}^{T} \mathbb{E}\left[\mu_{J_t}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\mu_1 - \mu_{J_t}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\Delta_{J_t}\right] \\
&= \sum_{t=1}^{T} \sum_{j\in[K]} \mathbb{E}\left[\mathbf{1}\left\{J_t = j\right\}\right] \cdot \Delta_j \\
&= \sum_{j\in[K]:\Delta_j>0} \mathbb{E}\left[\underbrace{\sum_{t=1}^{T} \mathbf{1}\left\{J_t = j\right\}}_{=:n_{j,T}}\right] \cdot \Delta_j \quad .
\end{aligned} \tag{7.12}$$

Let $n_{j,t-1} := \sum_{s=1}^{t-1} \mathbf{1}\{\mathrm{J}_s = j\}$ denote the total number of pulls for arm j by the end of round $t-1$. Then, $\mathbb{E}\left[n_{j,\mathrm{T}}\right]$ is the expected number of pulls of arm j by the end of learning and Δ_j is the singe round performance loss when pulling a sub-optimal arm j.

From the last step in (7.12), it is not hard to see, to minimize the regret, it is important to control the number of pulls of sub-optimal arms. But we have no idea which arms are sub-optimal. So, we have to pull each arm a certain amount of times in order to learn whether they are sub-optimal or not confidently. We need information!

Here, I should mention exploitation-vs-exploration.

Upper Confidence Bound (UCB). It is inspired by the principle of being optimistic in the face of uncertainty. Usually, all UCB-based algorithms are optimistic learning algorithms and follow a template to decompose regret. Also, they can be justified by concentration inequalities, e.g., Hoeffdin's inequality.

Recall $\Theta_{\mathrm{K}} := (v_1, v_2, \ldots, v_{\mathrm{K}})$ is the bandit instance we are interested in. Actually, for developing regret minimization algorithm, we are interested in $(\mu_1, \mu_2, \ldots, \mu_{\mathrm{K}})$. Note that only the mean reward gaps appear in the regret.

Recall $n_{j,t-1} = \sum_{s=1}^{t-1} \mathbf{1}\{\mathrm{J}_s = j\}$ is the number of pulls of arm j by the end of round $t-1$.

Now, let $\hat{\mu}_{j,n_{j,t-1}} := \frac{1}{n_{j,t-1}} \sum_{s=1}^{t-1} \mathrm{X}_{j,s} \mathbf{1}\{\mathrm{J}_s = j\}$ be the empirical mean of arm j by the end of round $t-1$, i.e., the average of $n_{j,t-1}$ iid random variables according to v_j.

exploitation-vs-exploration

Now, we can construct an empirical model $\hat{\Theta}_t = \left(\hat{\mu}_{1,n_{1,t-1}}, \hat{\mu}_{2,n_{2,t-1}}, \ldots, \hat{\mu}_{\mathrm{K},n_{\mathrm{K},t-1}}\right)$.

Hoeffding's inequality

upper confidence bound

w.t.p. $\bar{\mu}_{1,t} \geq \mu_1$

Assume all reward distributions have a $[0,1]$ support. The idea of UCB1[2], an algorithm in UCB family, is to construct an optimistic model $\overline{\Theta}_t = (\bar{\mu}_{1,t}, \bar{\mu}_{2,t}, \ldots, \bar{\mu}_{\mathrm{K},t})$ with each $j \in [\mathrm{K}]$

$$\bar{\mu}_{j,t} = \hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2\ln(t)}{n_{j,t-1}}} \quad . \tag{7.13}$$

In each round $t = 1, 2, \ldots, \mathrm{T}$,

1. Environment generate a reward vector $\mathrm{X}_t = (\mathrm{X}_{1,t}, \mathrm{X}_{2,t}, \ldots, \mathrm{X}_{\mathrm{K},t})$ with each $\mathrm{X}_{j,t} \sim v_j$. This reward vector is hidden to Learner ;
2. Learner constructs the upper confidence bound $\bar{\mu}_{j,t} = \hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2\ln(t)}{n_{j,t-1}}}$ for all $j \in [\mathrm{K}]$;
3. Learner pulls the arm with the highest upper confidence bound, i.e., $\mathrm{J}_t \in \arg\max_{j \in [\mathrm{K}]} \bar{\mu}_{j,t}$;
4. Learner observes $\mathrm{X}_{\mathrm{J}_t,t}$, the reward of the pulled arm J_t ;
5. Learner updates $n_{\mathrm{J}_t,t} = n_{\mathrm{J}_t,t-1} + 1$ and the empirical mean $\hat{\mu}_{\mathrm{J}_t,n_{\mathrm{J}_t,t}}$.

[2] UCB1 works for all Sub-Gaussian reward distributions.

Theorem 7.11. *If all reward distributions in Θ_{K} have a $[0,1]$ support, we have*

$$\overline{\mathcal{R}}(\textbf{\textit{UCB1}};\Theta_{\mathrm{K}};\mathrm{T}) \leq \sum_{j\in[\mathrm{K}]:\Delta_j>0} \frac{8\ln \mathrm{T}}{\Delta_j} + \textbf{\textit{Constant}} \quad .$$

Proof. Fix a sub-optimal arm j, we upper bound $\mathbb{E}[n_{j,\mathrm{T}}]$.

Let $\mathrm{L}_j := \frac{\square\cdot\ln \mathrm{T}}{\Delta_j^2}$, where $\square$ is a constant that will be tuned later.

You can view L_j as the amount of observations needed to conclude that this arm is not the optimal one.

We have

$$\begin{aligned}
\mathbb{E}\left[n_{j,\mathrm{T}}\right] &= \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j\right\}\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j, n_{j,t-1} \leq \mathrm{L}_j\right\}\right] + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j, n_{j,t} > n_{j,t-1}, n_{j,t-1} \leq \mathrm{L}_j\right\}\right] + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j, n_{j,t} > n_{j,t-1}, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
&\leq \mathrm{L}_j + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
&\leq \mathrm{L}_j + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j, \bar{\mu}_{j,t} \geq \bar{\mu}_{1,t}, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
&\leq \mathrm{L}_j + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\mathrm{J}_t = j, \bar{\mu}_{j,t} \geq \mu_1, n_{j,t-1} > \mathrm{L}_{j,t}\right\}\right] + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\bar{\mu}_{1,t} \leq \mu_1, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
&\leq \mathrm{L}_j + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\bar{\mu}_{j,t} \geq \mu_j + \Delta_j, n_{j,t-1} > \mathrm{L}_j\right\}\right] + \mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\bar{\mu}_{1,t} \leq \mu_1\right\}\right] \quad .
\end{aligned} \tag{7.14}$$

Now, we set $\mathrm{L}_j = \frac{8\ln(\mathrm{T})}{\Delta_j^2}$. Then, we have $\Delta_j = \sqrt{\frac{8\ln \mathrm{T}}{\mathrm{L}_j}}$.

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\bar{\mu}_{j,t} \geq \mu_j + \Delta_j, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
=\ &\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2\ln(t)}{n_{j,t-1}}} \geq \mu_j + \sqrt{\frac{8\ln \mathrm{T}}{\mathrm{L}_j}}, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
\leq\ &\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2\ln(\mathrm{T})}{n_{j,t-1}}} \geq \mu_j + \sqrt{\frac{8\ln \mathrm{T}}{n_{j,t-1}}}, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
\leq\ &\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\hat{\mu}_{j,n_{j,t-1}} \geq \mu_j + \sqrt{\frac{2\ln \mathrm{T}}{n_{j,t-1}}}, n_{j,t-1} > \mathrm{L}_j\right\}\right] \\
\leq\ &\sum_{t=1}^{\mathrm{T}} \sum_{h=\mathrm{L}_j}^{t-1} \mathbb{E}\left[\mathbf{1}\left\{\hat{\mu}_{j,h} \geq \mu_j + \sqrt{\frac{2\ln \mathrm{T}}{h}}\right\}\right] \\
\leq\ &\mathrm{T}^2 \cdot e^{-2\cdot 2\ln \mathrm{T}} \\
=\ &\mathcal{O}(1) \quad .
\end{aligned} \tag{7.15}$$

Similarly, we have $\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} \mathbf{1}\left\{\bar{\mu}_{1,t} \leq \mu_1\right\}\right] = \mathcal{O}(1)$.

Now, we have $\mathbb{E}\left[n_{j,\mathrm{T}}\right] \leq \mathrm{L}_j + \mathcal{O}(1) = \frac{8\ln \mathrm{T}}{\Delta_j^2} + \mathcal{O}(1)$, which gives

$$\begin{aligned}
\overline{\mathcal{R}}(\textbf{UCB1};\Theta_{\mathrm{K}};\mathrm{T}) &= \sum_{j\in[\mathrm{K}]:\Delta_j>0} \mathbb{E}\left[n_{j,\mathrm{T}}\right]\cdot\Delta_j \\
&\leq \sum_{j\in[\mathrm{K}]:\Delta_j>0} \frac{8\ln \mathrm{T}}{\Delta_j} + \textbf{Constant} \quad .
\end{aligned} \tag{7.16}$$

If problem-dependent parameters, e.g., all Δ_j, appear in the regret bound, we say it is a problem-dependent regret bound.

□

Worst-case Regret Bound for UCB1. Let us consider a 2-armed bandit problem, where the first arm is the optimal one and the second arm has a mean reward gap $\Delta = \frac{1}{T}$. Clearly, according to Theorem 7.11, we have $\overline{\mathcal{R}}(\mathbf{UCB1}; \Theta_K; T) = 8T \ln T$, which is even worse than T. Does it mean UCB1 fails this learning task?

To answer this question, we are motivated to study the worst-case regret bound, defined as

$$\sup_{\Theta_K \in \Pi^K} \overline{\mathcal{R}}(\mathbf{UCB1}; \Theta_K; T) \quad , \tag{7.17}$$

where Π is a set of distributions with a $[0, 1]$ support.

Theorem 7.12. *We have*

$$\sup_{\Theta_K \in \Pi^K} \overline{\mathcal{R}}(\boldsymbol{UCB1}; \Theta_K; T) \leq \mathcal{O}(\sqrt{KT \ln T}) \quad . \tag{7.18}$$

Proof. Fix Θ_K and set $\Delta := \sqrt{\frac{K \ln T}{T}}$. We have

$$\begin{aligned}
\overline{\mathcal{R}}(\mathbf{UCB1}; \Theta_K; T) &= \sum_{t=1}^{T} \sum_{j \in [K]} \mathbb{E}\left[\mathbf{1}\left\{J_t = j\right\}\right] \cdot \Delta_j \\
&= \sum_{t=1}^{T} \sum_{j \in [K]: \Delta_j \leq \Delta} \mathbb{E}\left[\mathbf{1}\left\{J_t = j\right\}\right] \cdot \Delta_j + \sum_{t=1}^{T} \sum_{j \in [K]: \Delta_j > \Delta} \mathbb{E}\left[\mathbf{1}\left\{J_t = j\right\}\right] \cdot \Delta_j \\
&\leq T \cdot \Delta + \sum_{t=1}^{T} \sum_{j \in [K]: \Delta_j > \Delta} \mathbb{E}\left[\mathbf{1}\left\{J_t = j\right\}\right] \cdot \Delta_j \\
&\leq T \cdot \Delta + \sum_{j \in [K]: \Delta_j > \Delta} \left(\frac{8 \ln T}{\Delta_j} + \mathcal{O}(1)\right) \\
&\leq T \cdot \Delta + \sum_{j \in [K]: \Delta_j > \Delta} \left(\frac{8 \ln T}{\Delta} + \mathcal{O}(1)\right) \\
&\leq T \cdot \Delta + K \frac{8 \ln T}{\Delta} + \mathcal{O}(K) \\
&= \mathcal{O}(\sqrt{KT \ln T}) \quad .
\end{aligned} \tag{7.19}$$

□

One may ask whether UCB1 is optimal in the worst case sense or not. The definition of minimax optimality, a joint of property between a family of algorithms and distributions, can answer this question. We skip the proof here and only show the conclusion: UCB1 is minimax optimal up to an extra $\sqrt{\ln T}$ factor.

Arm Elimination Algorithm. As UCB1 only has an $\mathcal{O}(\sqrt{\mathrm{KT}\ln \mathrm{T}})$ worst-case regret bound, now, we show an algorithm that enjoys an $\mathcal{O}(\sqrt{\mathrm{KT}\ln \mathrm{K}})$ worst-case regret bound, which is slightly better than UCB1.

Suppose we have a special K-armed bandit problem with one arm having a mean reward μ_* and all the remaining arms having the same mean reward $\mu_* - \Delta$, i.e., the mean reward gap for any sub-optimal arm is Δ. *Learner knows Δ and μ_* but does not know which arm is the optimal one.* To solve this special bandit problem, we can use the following **Arm Elimination** algorithm [AO10]:

Input: $[\mathrm{K}]$, μ_*, Δ, and T.

1. Pull each arm $n = \frac{2\ln(\mathrm{T}\Delta^2)}{\Delta^2}$ times and compute the empirical mean $\hat{\mu}_{j,n}$ of each arm $j \in [\mathrm{K}]$;
2. Commit to the arm with the highest empirical mean until the end of the learning, i.e., pull arm $\mathrm{J} = \arg\max_{j\in[\mathrm{K}]} \hat{\mu}_{j,n}$ for the remaining $\mathrm{T} - \mathrm{K}n$ rounds.

Theorem 7.13. ***Arm Elimination*** *enjoys an $\mathcal{O}\left(\frac{\mathrm{K}\ln(\mathrm{T}\Delta^2)}{\Delta} + \frac{\mathrm{K}}{\Delta}\right)$ problem-dependent regret bound. It also enjoys an $\mathcal{O}(\sqrt{\mathrm{KT}\ln \mathrm{K}})$ worst-case regret bound.*

Proof. Let i_* denote the index of the optimal arm. Without loss of generality, we assume it is unique. We first upper bound the probability that the committed arm is not the optimal one. We have

$$\mathbb{P}\left\{\mathrm{J} \neq i_*\right\} \leq \mathbb{P}\left\{\max_{j\in[\mathrm{K}]\setminus\{i_*\}} \hat{\mu}_{j,n} \geq \hat{\mu}_{i_*,n}\right\} \leq \sum_{j\in[\mathrm{K}]\setminus\{i_*\}} \mathbb{P}\left\{\hat{\mu}_{j,n} \geq \hat{\mu}_{i_*,n}\right\} \leq (\mathrm{K}-1)\cdot 2e^{-n\frac{\Delta^2}{2}} \quad . \tag{7.20}$$

The problem-dependent regret $\overline{\mathcal{R}}(\mathrm{T})$ is

$$\begin{aligned} & \underbrace{(\mathrm{K}-1)\cdot n\cdot\Delta}_{\text{regret in Step 1}} + \underbrace{\mathbb{P}\left\{\mathrm{J}\neq i_*\right\}\cdot(\mathrm{T}-\mathrm{K}n)\cdot\Delta}_{\text{regret in Step 2}} \\ \leq\ & \mathrm{K}\cdot\frac{2\ln(\mathrm{T}\Delta^2)}{\Delta^2}\cdot\Delta + \mathrm{K}\cdot 2e^{-n\frac{\Delta^2}{2}}\cdot\mathrm{T}\cdot\Delta \\ =\ & 2\mathrm{K}\frac{\ln(\mathrm{T}\Delta^2)}{\Delta} + \mathrm{K}\cdot\frac{2}{\mathrm{T}\Delta^2}\cdot\mathrm{T}\cdot\Delta \\ =\ & \mathcal{O}\left(\frac{\mathrm{K}\ln(\mathrm{T}\Delta^2)}{\Delta} + \frac{\mathrm{K}}{\Delta}\right) \quad . \end{aligned} \tag{7.21}$$

Let $\tilde{\Delta} := \frac{e\sqrt{\mathrm{K}}}{\sqrt{\mathrm{T}}}$. If $\Delta \leq \tilde{\Delta}$, we have the regret is at most $\mathrm{T}\cdot\tilde{\Delta} = e\sqrt{\mathrm{KT}}$. If $\Delta > \tilde{\Delta}$, we have $\mathcal{O}\left(\frac{\mathrm{K}\ln(\mathrm{T}\Delta^2)}{\Delta} + \frac{\mathrm{K}}{\Delta}\right) \leq \mathcal{O}\left(\frac{\mathrm{K}\ln(\mathrm{T}\tilde{\Delta}^2)}{\tilde{\Delta}} + \frac{\mathrm{K}}{\tilde{\Delta}}\right) = \mathcal{O}(\sqrt{\mathrm{KT}\ln \mathrm{K}})$, where the inequality uses the fact that $f(x) = \frac{\ln(\mathrm{T}x^2)}{x}$ is a decreasing function when $\mathrm{T}x^2 \geq e^2$. □

Arm Elimination Algorithm with Doubling-Trick. Since we cannot assume we know Δ in advance and all sub-optimal arms have the same Δ, we cannot use **Arm Elimination** directly for solving practical learning problems. A good thing is we can introduce **doubling-trick** into **Arm Elimination** to make it work by estimating Δ_j.

It is also fine to set $\hat{\Delta}_0 = 0$ and start from $r = 0$.

You can view B_r as a version space in batch learning.

This is a picture for this.

Input: [K] and T.
Initialization: Set $\hat{\Delta}_1 = 0.5$ and $B_1 = [K]$.
For epochs $r = 1, 2, \ldots$ up to $\log T$,

1. For each arm $j \in B_r$, pull it until the total number of pulls hits $n_r = \frac{2\ln(KT\hat{\Delta}_r^2)}{\hat{\Delta}_r^2}$;
2. All arms $i \in B_r$ such that

$$\underbrace{\hat{\mu}_{i,n_r} + \sqrt{\frac{\log(KT\hat{\Delta}_r^2)}{2 \cdot n_r}}}_{\text{upper confidence bound of arm } i} \geq \underbrace{\max_{j \in B_r} \hat{\mu}_{j,n_r} - \sqrt{\frac{\log(KT\hat{\Delta}_r^2)}{2 \cdot n_r}}}_{\text{lower confidence bound of } j_r^* \in \arg\max_{j \in B_r} \hat{\mu}_{j,n_r}} . \tag{7.22}$$

will be kept in B_{r+1}.

Set $\hat{\Delta}_{r+1} = \frac{\hat{\Delta}_r}{2} = 0.5^{r+1}$.

If $|B_{r+1}| = 1$, commit to that arm until the end of the learning.

Theorem 7.14. ***Arm Elimination with Doubling-Trick*** *has a* $\sum_{j \in [K]: \Delta_j > 0} \mathcal{O}\left(\frac{\ln(T\Delta_j^2)}{\Delta_j} + \frac{\ln K}{\Delta_j}\right)$ *problem-dependent regret bound and an* $\mathcal{O}(\sqrt{KT \ln K})$ *worst-case regret bound.*

Proof. Let i_* denote the index of the unique optimal arm. Fix a sub-optimal arm j. Let $r_j = \left\lceil \log\left(\frac{1}{\Delta_j}\right) \right\rceil$. Then, we have $0.5\Delta_j \leq 0.5^{r_j} = \hat{\Delta}_{r_j} \leq \Delta_j$.

We claim that the probability that this arm j is kept in B_{r_j+1} is very low. Formally, we have

$$\begin{aligned}
& \mathbb{P}\left\{j \in B_{r_j+1}\right\} \\
= \; & \mathbb{P}\left\{j \in B_{r_j}, \hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \max_{j \in B_{r_j}} \hat{\mu}_{j,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}\right\} \\
\leq \; & \mathbb{P}\left\{\hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \max_{j \in B_{r_j}} \hat{\mu}_{j,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}, i_* \in B_{r_j}\right\} \\
+ \; & \mathbb{P}\left\{\hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \max_{j \in B_{r_j}} \hat{\mu}_{j,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}, i_* \notin B_{r_j}\right\} \\
\leq \; & \underbrace{\mathbb{P}\left\{\hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \hat{\mu}_{i_*,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}\right\}}_{\text{UCB analysis, Hoeffding's inequality}} \\
+ \; & \mathbb{P}\{i^* \notin B_2\} + \mathbb{P}\{i^* \in B_2, i^* \notin B_3\} + \ldots + \mathbb{P}\left\{i^* \in B_2, \ldots, i^* \in B_{r_j-1}, i^* \notin B_{r_j}\right\} \\
\leq \; & \frac{2}{KT\hat{\Delta}_{r_j}^2} + \sum_{r=1}^{r_j-1} \frac{2}{T\hat{\Delta}_r^2} \\
\leq \; & \sum_{r=1}^{r_j} \frac{2}{T\hat{\Delta}_r^2} \\
= \; & \sum_{r=1}^{r_j} \frac{2}{T \cdot 0.5^{2r}} \\
= \; & \mathcal{O}\left(\frac{1}{T \cdot 0.5^{2r_j}}\right) \\
= \; & \mathcal{O}\left(\frac{1}{T \cdot \Delta_j^2}\right) .
\end{aligned} \tag{7.23}$$

The total regret from this sub-optimal arm j is at most

$$\begin{aligned}
&\underbrace{n_{r_j}\cdot\Delta_j}_{\text{regret until the end of epoch } r_j} + \underbrace{\mathrm{T}\cdot\mathbb{P}\left\{j\in \mathrm{B}_{r_j+1}\right\}\cdot\Delta_j}_{\text{regret for the remaining rounds}} \\
\leq\ & \frac{2\ln(\mathrm{KT}\hat{\Delta}_{r_j}^2)}{\hat{\Delta}_{r_j}^2}\cdot\Delta_j + \mathrm{T}\cdot\mathcal{O}\left(\frac{1}{\mathrm{T}\cdot\Delta_j^2}\right)\cdot\Delta_j \\
\leq\ & \frac{2\ln(\mathrm{KT}\Delta_j^2)}{0.25\Delta_j^2}\cdot\Delta_j + \mathrm{T}\cdot\mathcal{O}\left(\frac{1}{\mathrm{T}\cdot\Delta_j^2}\right)\cdot\Delta_j \\
=\ & \mathcal{O}\left(\frac{\ln(\mathrm{T}\Delta_j^2)}{\Delta_j}+\frac{\ln \mathrm{K}}{\Delta_j}\right) \quad .
\end{aligned} \tag{7.24}$$

Summing over all the sub-optimal arms, we have the problem-dependent regret is at most

$$\overline{\mathcal{R}}(\mathrm{T}) = \sum_{j\in[\mathrm{K}]:\Delta_j>0}\mathcal{O}\left(\frac{\ln(\mathrm{T}\Delta_j^2)}{\Delta_j}+\frac{\ln \mathrm{K}}{\Delta_j}\right) \quad . \tag{7.25}$$

Let $\tilde{\Delta} := \frac{e\sqrt{\mathrm{K}\ln \mathrm{K}}}{\sqrt{\mathrm{T}}}$. We have

$$\begin{aligned}
\overline{\mathcal{R}}(\mathrm{T}) &\leq \mathrm{T}\cdot\tilde{\Delta} + \sum_{j\in[\mathrm{K}]:\Delta_j>\tilde{\Delta}}\mathcal{O}\left(\frac{\ln(\mathrm{T}\Delta_j^2)}{\Delta_j}+\frac{\ln \mathrm{K}}{\Delta_j}\right) \\
&\leq e\sqrt{\mathrm{KT}\ln \mathrm{K}} + \sum_{j\in[\mathrm{K}]:\Delta_j>\tilde{\Delta}}\mathcal{O}\left(\frac{\ln(\mathrm{T}\tilde{\Delta}^2)}{\tilde{\Delta}}+\frac{\ln \mathrm{K}}{\tilde{\Delta}}\right) \\
&\leq e\sqrt{\mathrm{KT}\ln \mathrm{K}} + \mathcal{O}\left(\frac{\mathrm{K}\ln(\mathrm{T}\tilde{\Delta}^2)}{\tilde{\Delta}}+\frac{\mathrm{K}\ln \mathrm{K}}{\tilde{\Delta}}\right) \\
&\leq e\sqrt{\mathrm{KT}\ln \mathrm{K}} + \mathcal{O}\left(\frac{\mathrm{K}\ln(e^2\mathrm{K}\ln \mathrm{K})}{\frac{e\sqrt{\mathrm{K}\ln \mathrm{K}}}{\sqrt{\mathrm{T}}}}+\frac{\mathrm{K}\ln \mathrm{K}}{\frac{e\sqrt{\mathrm{K}\ln \mathrm{K}}}{\sqrt{\mathrm{T}}}}\right) \\
&= \mathcal{O}(\sqrt{\mathrm{KT}\ln \mathrm{K}}) \quad .
\end{aligned} \tag{7.26}$$

We have

$$\begin{aligned}
\mathbb{P}\{i^*\notin \mathrm{B}_2\} &= \mathbb{P}\left\{\hat{\mu}_{i_*,n_1}+\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}} < \max_{j\in\mathrm{B}_1\setminus\{i_*\}}\hat{\mu}_{j,n_1}-\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}}\right\} \\
&\leq \sum_{j\in\mathrm{B}_1\setminus\{i_*\}}\mathbb{P}\left\{\hat{\mu}_{i_*,n_1}+\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}} < \hat{\mu}_{j,n_1}-\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}}\right\} \\
&\leq \sum_{j\in\mathrm{B}_1\setminus\{i_*\}}\left(\mathbb{P}\left\{\hat{\mu}_{i_*,n_1}+\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}} \leq \mu_1\right\}+\mathbb{P}\left\{\hat{\mu}_{j,n_1}-\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}} \geq \mu_j\right\}\right) \\
&\leq \frac{2}{\mathrm{T}\hat{\Delta}_1^2} \quad .
\end{aligned} \tag{7.27}$$

$$\begin{aligned}
&\mathbb{P}\{i^*\in \mathrm{B}_2, i^*\notin \mathrm{B}_3\} \\
=\ & \mathbb{P}\left\{\hat{\mu}_{i_*,n_1}+\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}} \geq \max_{j\in\mathrm{B}_1}\hat{\mu}_{j,n_1}-\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_1^2)}{2\cdot n_1}}, \hat{\mu}_{i_*,n_2}+\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_2^2)}{2\cdot n_2}} < \max_{j\in\mathrm{B}_2\setminus\{i_*\}}\hat{\mu}_{j,n_2}-\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_2^2)}{2\cdot n_2}}\right\} \\
\leq\ & \mathbb{P}\left\{\hat{\mu}_{i_*,n_2}+\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_2^2)}{2\cdot n_2}} < \max_{j\in\mathrm{B}_2\setminus\{i_*\}}\hat{\mu}_{j,n_2}-\sqrt{\frac{\log(\mathrm{KT}\hat{\Delta}_2^2)}{2\cdot n_2}}\right\} \\
\leq\ & \frac{2}{\mathrm{T}\hat{\Delta}_2^2} \quad .
\end{aligned} \tag{7.28}$$

Similarly, we have

$$\mathbb{P}\left\{i^*\in \mathrm{B}_2,\ldots,i^*\in \mathrm{B}_{r_j-1}, i^*\notin \mathrm{B}_{r_j}\right\} \leq \frac{2}{\mathrm{T}\hat{\Delta}_{r_j-1}^2} \quad . \qquad \square$$

REFERENCES

[AO10] Peter Auer and Ronald Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica* 61.1-2 (2010), pages 55–65.

[BC12] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5.1 (2012), pages 1–122. arXiv: 1204.5721.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.