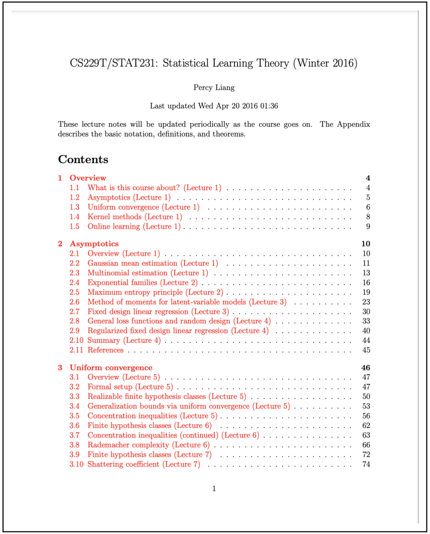
Trustworthy Machine Learning

Online Learning

Sangdon Park

POSTECH

Contents from



CS229T/STAT231: Statistical Learning Theory (Winter 2016)

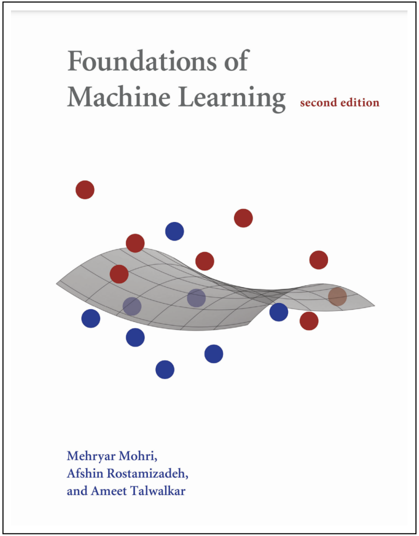
Percy Liang

Last updated Wed Apr 20 2016 01:36

These lecture notes will be updated periodically as the course goes on. The Appendix describes the basic notation, definitions, and theorems.

Contents

1



and various papers.

Motivation

- We have considered *statistical learning* (*i.e.*, learning under the i.i.d. assumption)

Motivation

- We have considered *statistical learning* (*i.e.*, learning under the i.i.d. assumption)
- However, this assumption can be broken, *e.g.*, distribution shift, price data

Motivation

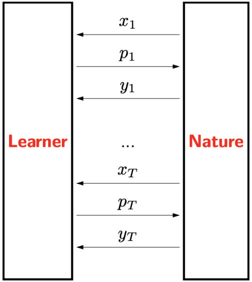
- We have considered *statistical learning* (*i.e.*, learning under the i.i.d. assumption)
- However, this assumption can be broken, *e.g.*, distribution shift, price data
- Here, we will weaken this assumption.
    - batch to online: “how data arrives”
    - statistical to adversarial: “how data are generated”

Setup

- Prediction task: learn to map an example $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$

Setup

- Prediction task: learn to map an example $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$
- Online learning game between a learner and nature



Protocol:

for $t = 1, \ldots, T$ **do**

Learner receives an example $x_t \in \mathcal{X}$

Learner outputs a prediction $p_t \in \mathcal{Y}$

Learner receives a true label $y_t \in \mathcal{Y}$

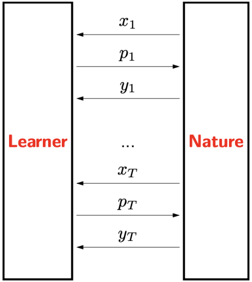
Learner suffers loss $\ell(y_t, p_t)$

Learner update model parameters

end for

Setup

- Prediction task: learn to map an example $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$
- Online learning game between a learner and nature



Protocol:

for $t = 1, \ldots, T$ **do**

Learner receives an example $x_t \in \mathcal{X}$

Learner outputs a prediction $p_t \in \mathcal{Y}$

Learner receives a true label $y_t \in \mathcal{Y}$

Learner suffers loss $\ell(y_t, p_t)$

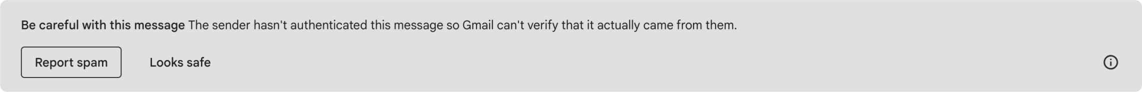
Learner update model parameters

end for

- The learner is a function $\mathcal{A}$ that returns the current prediction given the full history, *i.e.*,

$$p_{t+1} = \mathcal{A}(x_{1:t}, p_{1:t}, y_{1:t}, x_{t+1})$$

Example: Online Binary Classification for Spam Filtering



- examples: $\mathcal{X} := \{0,1\}^d$ are boolean feature vectors (presence or absence of a word)
- labels: $\mathcal{Y} := \{+1,-1\}$ are whether a document is spam or not
- zero-one loss: $\ell(y_t, p_t) = \mathbb{1}\,(y_t \neq p_t)$ is whether the prediction was incorrect

Remarks

- In batch learning, we have a training phase and test phase; but in online learning, they are interleaved.

Remarks

- In batch learning, we have a training phase and test phase; but in online learning, they are interleaved.
- The online learning setup allows to use the full history.
  - The history grows in time by $O(T)$ and in space by $O(T)$.
  - This means that we can use any batch learning algorithm on the history at each time.
  - However, online algorithms tend to be lightweight, *i.e.*, the amount of work by an algorithm should not grow with t.

Remarks

- In batch learning, we have a training phase and test phase; but in online learning, they are interleaved.
- The online learning setup allows to use the full history.
  - The history grows in time by $O(T)$ and in space by $O(T)$.
  - This means that we can use any batch learning algorithm on the history at each time.
  - However, online algorithms tend to be lightweight, *i.e.*, the amount of work by an algorithm should not grow with t.
- Online learning algorithms have the potential to adapt.
  - *e.g.*, we have labels on adversarial examples!

Remarks

- In batch learning, we have a training phase and test phase; but in online learning, they are interleaved.
- The online learning setup allows to use the full history.
  - The history grows in time by $O(T)$ and in space by $O(T)$.
  - This means that we can use any batch learning algorithm on the history at each time.
  - However, online algorithms tend to be lightweight, *i.e.*, the amount of work by an algorithm should not grow with t.
- Online learning algorithms have the potential to adapt.
  - *e.g.*, we have labels on adversarial examples!
- For some applications (*e.g.*, spam filtering), examples are generated by an adversary.

"Goodness" Metric

How to measure the performance of an online learning algorithm?

“Goodness” Metric

How to measure the performance of an online learning algorithm?

- The cumulative loss of the learner, *i.e.*,

$$\sum_{t=1}^{T} \ell(y_t, p_t)$$

“Goodness” Metric

How to measure the performance of an online learning algorithm?

- The cumulative loss of the learner, *i.e.*,

$$\sum_{t=1}^{T} \ell(y_t, p_t)$$

- No! In the adversarial setting, the adversary can manipulate data to make the learner trivially bad loss (*e.g.*, Nature can simulate Learner).

"Goodness" Metric

How to measure the performance of an online learning algorithm?

- The cumulative loss of the learner, *i.e.*,

$$\sum_{t=1}^{T} \ell(y_t, p_t)$$

- No! In the adversarial setting, the adversary can manipulate data to make the learner trivially bad loss (*e.g.*, Nature can simulate Learner).
- What do you do when your grade is awful? Compare to the best grade in your class!

Regret

Definition

$$\text{Regret} := \underbrace{\sum_{t=1}^{T} \ell(y_t, p_t)}_{\text{learner}} - \underbrace{\min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))}_{\text{best expert}}$$

- $\mathcal{H}$ is a class of experts.
- The best export is a role model of the learner.
- We will consider the worst case regret (*i.e.*, labeled examples are generated by an adversary)

Negative Result

claim

For any deterministic learner $\mathcal{A}$, there exists an $\mathcal{H}$ and the sequence of labeled examples such that

$$\textit{Regret} \geq \frac{T}{2}.$$

- Too bad...
- Why? Prove under the following setup.
  - binary classification, *i.e.*, $y \in \{-1, +1\}$
  - zero-one loss, *i.e.*, $\ell(y_t, p_t) := \mathbb{1}(p_t \neq y_t)$
  - the learner is fully **deterministic**.

Negative Result: Why? Intuition

- An adversary (who has full knowledge of the learner) can choose y_t to make it different to the learner's choice p_t.
- Thus, the learner's cumulative loss is T!
- Not yet; how about the best expert's loss?
- Consider two experts, *i.e.*, $\mathcal{H} := \{h_{-1}, h_{+1}\}$ (where h_y always predict y).
- Thus, we have

$$\ell(y_t, h_{-1}(x_t)) + \ell(y_t, h_{+1}(x_t)) = 1 \quad \Rightarrow \quad \sum_{t=1}^{T} \ell(y_t, h_{-1}(x_t)) + \sum_{t=1}^{T} \ell(y_t, h_{+1}(x_t)) = T$$

$$\Rightarrow \sum_{t=1}^{T} \ell(y_t, h_{-1}(x_t)) \le \frac{T}{2} \quad \text{or} \quad \sum_{t=1}^{T} \ell(y_t, h_{+1}(x_t)) \le \frac{T}{2}$$

$$\Rightarrow \text{Regret} := \underbrace{\sum_{t=1}^{T} \ell(y_t, p_t)}_{=T} - \underbrace{\min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))}_{\le \frac{T}{2}} \ge \frac{T}{2}.$$

Outline

- Halving Algorithm
  - Deterministic
  - Separable assumption
  - Finite $\mathcal{H}$
- Exponential Weighting Algorithm
  - Randomized
  - No separable assumption
  - Finite $\mathcal{H}$
- Perceptron Algorithm
  - Deterministic
  - Separable assumption
  - Infinite $\mathcal{H}$

Add an Assumption without Randomization

Assumption (separable)

Assume that the best expert $h^* \in \mathcal{H}$ *obtains zero cumulative loss (i.e.,* $\ell(y_t, h^*(x_t)) = 0$ *for all* $t \in \{1, \dots, T\}$*).*

Add an Assumption without Randomization

Assumption (separable)

Assume that the best expert $h^* \in \mathcal{H}$ *obtains zero cumulative loss (i.e.,* $\ell(y_t, h^*(x_t)) = 0$ *for all* $t \in \{1, \ldots, T\}$*).*

- This impose restrictions on adversaries.

Add an Assumption without Randomization

Assumption (separable)

Assume that the best expert $h^* \in \mathcal{H}$ *obtains zero cumulative loss (i.e.,* $\ell(y_t, h^*(x_t)) = 0$ *for all* $t \in \{1, \ldots, T\}$*).*

- This impose restrictions on adversaries.
- We saw a similar assumption in PAC learning.

Add an Assumption without Randomization

Assumption (separable)

Assume that the best expert $h^* \in \mathcal{H}$ *obtains zero cumulative loss (i.e.,* $\ell(y_t, h^*(x_t)) = 0$ *for all* $t \in \{1, \ldots, T\}$*).*

- This impose restrictions on adversaries.
- We saw a similar assumption in PAC learning.
- Practical setup? adaptive conformal prediction

Halving Algorithm

Algorithm 1 Halving Algorithm

```
H_1 ← H
for t = 1, ..., T do
    Observe x_t
    Predict ŷ_t = MajorityVote(H_t, x_t)
    Observe y_t
    if ŷ_t ≠ y_t then
        H_{t+1} ← {h ∈ H_t | h(x_t) = y_t}
    else
        H_{t+1} ← H_t
    end if
end for
```

- $\mathcal{Y} := \{-1, +1\}$
- $\mathcal{H}_t$: a set of correct experts.
- Under the separable assumption, keep only correct experts.
- Due to the separable assumption, we can discard at least half of experts at some iterations!

Halving Algorithm: A Regret Bound

Theorem

Under the realizable assumption, for any $(x_t, y_t)_{t=1}^T$*, we have*

$$\textit{Regret} \leq \log_2 |\mathcal{H}|.$$

Halving Algorithm: A Regret Bound

Theorem

Under the realizable assumption, for any $(x_t, y_t)_{t=1}^T$*, we have*

$$\textit{Regret} \le \log_2 |\mathcal{H}|.$$

- Very strong results due to the separable assumption.
  - after a finite number of iterations, the predictor never makes mistakes.

Halving Algorithm: A Regret Bound

Proof Sketch

- Let M be the number of mistakes.
- For each mistake, at least half of the experts are eliminated, *i.e.*, if $\hat{y}_i$ made a mistake,

$$\frac{|\mathcal{H}_{i+1}|}{|\mathcal{H}_i|} \leq \frac{1}{2} \Rightarrow \frac{|\mathcal{H}_{T+1}|}{|\mathcal{H}|} \leq \frac{1}{2^M}.$$

- Due to the realizable assumption, we have

$$1 \leq |\mathcal{H}_{T+1}|.$$

- M = Regret.

Remove the Separable Assumption

- The separable assumption is too strong
- Let's remove this.
- Then, we need a randomization algorithm.
- One example: Exponential weighting algorithm.

Exponential Weighting Algorithm

Algorithm 2 Exponential Weighting Algorithm

$w_1 \leftarrow (1/|\mathcal{H}|, \ldots, 1/|\mathcal{H}|)$
for $t = 1, \ldots, T$ **do**
    Observe x_t
    Predict $\hat{y}_t = h^{i_t}(x_t)$, where $i_t \sim w_t$
    Observe y_t
    Update $w_{t+1}(i) \propto w_t(i) \exp\left\{-\eta \ell(h^i(x_t), y_t)\right\}$ for all $i \in \{1, \ldots, |\mathcal{H}|\}$
end for

- $\mathcal{H}$: a set of experts
- $\ell(\cdot) \in [0, 1]$

Exponential Weighting Algorithm

Algorithm 3 Exponential Weighting Algorithm

1: $w_1 \leftarrow (1/|\mathcal{H}|, \ldots, 1/|\mathcal{H}|)$
2: **for** $t = 1, \ldots, T$ **do**
3: Observe x_t
4: Predict $\hat{y}_t = h^{i_t}(x_t)$, where $i_t \sim w_t$
5: Observe y_t
6: Update $w_{t+1}(i) \propto w_t(i) \exp\left\{-\eta \ell(h^i(x_t), y_t)\right\}$ for all $i \in \{1, \ldots, |\mathcal{H}|\}$
7: **end for**

- $\mathcal{H}$: a set of experts
- $\ell(\cdot) \in [0, 1]$
- Due to the randomization in (4), an adversary cannot completely fool the learner.

Exponential Weighting Algorithm: A Regret Bound

Theorem

For any loss function ℓ with the range of $[0,1]$, we have

$$\textit{Regret} := \sum_{t=1}^{T} \mathbb{E}_{h\sim p_t} \ell(y_t, h(x_t)) - \min_{h\in\mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t)) \leq \sqrt{T \ln |\mathcal{H}|}$$

if $\eta = \sqrt{\frac{8 \ln |\mathcal{H}|}{T}}$.

- No separable assumption.
- "learnable", *i.e.,* $\frac{\text{Regret}}{T} = \sqrt{\frac{\ln |\mathcal{H}|}{T}}$ with a mild assumption on loss.
- Still we assume a finite set of experts.

Exponential Weighting Algorithm I

Proof sketch

Definitions:

- $L_t^i := \sum_{s=1}^t \ell(h_i(x_s), y_s)$: the cumulative loss of h_i up to t
- $W_t := \sum_{i=1}^{|\mathcal{H}|} \exp\{-\eta L_t^i\}$: a "potential value" at time t
- $W_0 := |\mathcal{H}|$: a "potential value" at time 0

Steps:

1. The lower bound of the "log-potential difference":

$$\ln \frac{W_T}{W_0} = \ln \sum_{i=1}^{|\mathcal{H}|} \exp\{-\eta L_T^i\} - \ln |\mathcal{H}| \geq \ln \left(\max_{i \in \{1,\ldots,|\mathcal{H}|\}} \exp\{-\eta L_T^i\} \right) - \ln |\mathcal{H}| = -\eta \min_{i \in \{1,\ldots,|\mathcal{H}|\}} L_T^i - \ln |\mathcal{H}|.$$

Exponential Weighting Algorithm II

Proof sketch

2. The upper bound of the "log-potential difference":

$$\ln \frac{W_t}{W_{t-1}} = \ln \frac{\sum_{i=1}^{|\mathcal{H}|} \exp\{-\eta L_t^i\}}{\sum_{i=1}^{|\mathcal{H}|} \exp\{-\eta L_{t-1}^i\}} = \ln \frac{\sum_{i=1}^{|\mathcal{H}|} \exp\{-\eta \ell(h_t^i(x_t), y_t)\} \exp\{-\eta L_{t-1}^i\}}{\sum_{i=1}^{|\mathcal{H}|} \exp\{-\eta L_{t-1}^i\}}$$

$$= \ln \mathbb{E}_{i_t \sim w_t} \exp\left\{-\eta \ell(h^{i_t}(x_t), y_t)\right\} \le -\eta \mathbb{E}_{i_t \sim w_t} \ell(h^{i_t}(x_t), y_t) + \frac{\eta^2}{8}$$

$$\Rightarrow \ln \frac{W_T}{W_0} \le -\eta \sum_{t=1}^{T} \mathbb{E}_{i_t \sim w_t} \ell(h^{i_t}(x_t), y_t) + \frac{\eta^2 T}{8}$$

- For any $s \in \mathbb{R}$ and a random variable $X \in [a, b]$, $\ln \mathbb{E} e^{sX} \le s\mathbb{E}X + \frac{s^2(b-a)^2}{8}$.

Exponential Weighting Algorithm III

Proof sketch

3. Combine the lower and upper bounds:

$$-\eta \min_{i\in\{1,\ldots,|\mathcal{H}|\}} L_T^i - \ln|\mathcal{H}| \leq -\eta \sum_{t=1}^{T} \mathbb{E}_{i_t\sim w_t}\ell(h^{i_t}(x_t), y_t) + \frac{\eta^2 T}{8} \Rightarrow$$

$$\sum_{t=1}^{T} \mathbb{E}_{i_t\sim w_t}\ell(h^{i_t}(x_t), y_t) - \min_{i\in\{1,\ldots,|\mathcal{H}|\}} L_T^i \leq \frac{\eta T}{8} + \frac{\ln|\mathcal{H}|}{\eta}$$

Algorithms So Far

- Halving Algorithm
  - Deterministic
  - Separable assumption
  - Finite $\mathcal{H}$

Algorithms So Far

- Halving Algorithm
  - Deterministic
  - Separable assumption
  - Finite $\mathcal{H}$
- Exponential Weighting Algorithm
  - Randomized
  - No separable assumption
  - Finite $\mathcal{H}$

Algorithms So Far

- Halving Algorithm
  - Deterministic
  - Separable assumption
  - Finite $\mathcal{H}$
- Exponential Weighting Algorithm
  - Randomized
  - No separable assumption
  - Finite $\mathcal{H}$
- What's the next?

Algorithms So Far

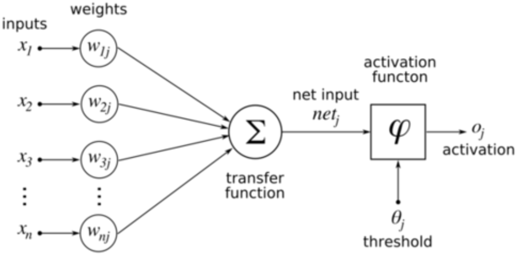
- Halving Algorithm
  - Deterministic
  - Separable assumption
  - Finite $\mathcal{H}$
- Exponential Weighting Algorithm
  - Randomized
  - No separable assumption
  - Finite $\mathcal{H}$
- What's the next?
  - Remove the assumption on the finiteness of $\mathcal{H}$ (under some assumptions)

Algorithms So Far

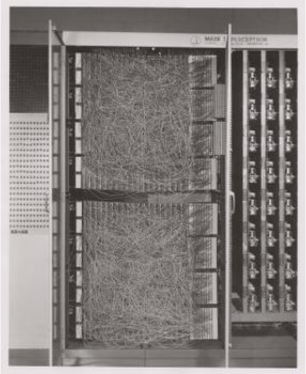
- Halving Algorithm
  - Deterministic
  - Separable assumption
  - Finite $\mathcal{H}$
- Exponential Weighting Algorithm
  - Randomized
  - No separable assumption
  - Finite $\mathcal{H}$
- What's the next?
  - Remove the assumption on the finiteness of $\mathcal{H}$ (under some assumptions)
  - Deterministic
  - Separable assumption (with some margin)
  - Infinite $\mathcal{H}$

Perceptron: History

TLDR: Father of Neural Networks!



(a) Perceptron



(b) Mark I Perceptron machine

- Invented in 1943 by Warren McCulloch and Walter Pitts.
- Firstly implemented in 1958 by Frank Rosenblatt(!)

Perceptron Algorithm: Setup

- $\mathcal{D}$: change over time but require the separable assumption.
- $\mathcal{H}$: linear functions without bias terms – additional assumption
- ℓ: 0-1 loss for classification

Perceptron Algorithm

Algorithm 4 Perceptron Algorithm

$w_1 \leftarrow w_0 := 0$
for $t = 1, \ldots, T$ **do**
  Receives an example $x_t \in \mathcal{X}$
  $\hat{y}_t \leftarrow \mathrm{sign}(w_t \cdot x_t)$
  Receives a true label $y_t \in \mathcal{Y}$
  if $\hat{y}_t \neq y_t$ **then**
    $w_{t+1} \leftarrow w_t + y_t x_t$
  else
    $w_{t+1} \leftarrow w_t$
  end if
end for

Perceptron Algorithm: A Regret Bound

Theorem

Suppose $\|x_t\|_2 \leq r$ *for all* t *and for some* r, *and there exists* $\gamma > 0$ *and* $v \in \mathbb{R}^d$ *such that for all* t *we have*

$$\gamma \leq \frac{y_t(v \cdot x_t)}{\|v\|_2}.$$

Then, we have

$$\textit{Regret} \leq \frac{r^2}{\gamma^2}.$$

Perceptron Algorithm: A Regret Bound

Theorem

Suppose $\|x_t\|_2 \leq r$ *for all* t *and for some* r*, and there exists* $\gamma > 0$ *and* $v \in \mathbb{R}^d$ *such that for all* t *we have*

$$\gamma \leq \frac{y_t(v \cdot x_t)}{\|v\|_2}.$$

Then, we have

$$\textit{Regret} \leq \frac{r^2}{\gamma^2}.$$

- Assumption: a sequence is separable by a perfect classifier v with some margin

Perceptron Algorithm: A Regret Bound

Theorem

Suppose $\|x_t\|_2 \le r$ *for all* t *and for some* r, *and there exists* $\gamma > 0$ *and* $v \in \mathbb{R}^d$ *such that for all* t *we have*

$$\gamma \le \frac{y_t(v \cdot x_t)}{\|v\|_2}.$$

Then, we have

$$\textit{Regret} \le \frac{r^2}{\gamma^2}.$$

- Assumption: a sequence is separable by a perfect classifier v with some margin
- The bound does not depend on T

Perceptron Algorithm: A Proof Sketch

- Let $\mathcal{J} \subseteq \{1, \dots, T\}$ be the set of time indices when updated. Thus, Regret $= |\mathcal{J}|$.
- From the "margin" assumption, there exists γ and v for any $\mathcal{J}$ such that a margin of v from any mis-classified sample is larger than γ.

$$
\begin{aligned}
\overbrace{\gamma|\mathcal{J}|}^{\text{sum of margins}} &\le \frac{\sum_{t\in\mathcal{J}} y_t(v \cdot x_t)}{\|v\|} = \frac{v}{\|v\|} \cdot \sum_{t\in\mathcal{J}} y_t x_t \\
&\le \left\| \sum_{t\in\mathcal{J}} y_t x_t \right\| \\
&= \left\| \sum_{t\in\mathcal{J}} w_{t+1} - w_t \right\| = \|w_{T+1}\| = \sqrt{\|w_{T+1}\|^2} = \sqrt{\|w_{T+1}\|^2 - \|w_0\|^2} \\
&= \sqrt{\sum_{t\in\mathcal{J}} \|w_{t+1}\|^2 - \|w_t\|^2} = \sqrt{\sum_{t\in\mathcal{J}} \|w_t + y_t x_t\|^2 - \|w_t\|^2} = \sqrt{\sum_{t\in\mathcal{J}} 2y_t w_t \cdot x_t + \|x_t\|^2} \\
&\le \sqrt{\sum_{t\in\mathcal{J}} \|x_t\|^2} \le \sqrt{\sum_{t\in\mathcal{J}} r^2} = r\sqrt{|\mathcal{J}|}.
\end{aligned}
\tag{1}
$$

- (1): Cauchy-Schwarz inequality, *i.e.*, $u \cdot v \le \|u\|\|v\|$

Conclusion

- What we learned
  - Halving Algorithm
    - Deterministic
    - Separable assumption
    - Finite $\mathcal{H}$
  - Exponential Weighting Algorithm
    - Randomized
    - No separable assumption
    - Finite $\mathcal{H}$
  - Perceptron Algorithm
    - Deterministic
    - Separable assumption
    - Infinite $\mathcal{H}$
- Interesting materials
  - Online convex optimization
  - Stochastic bandits
  - Adversarial bandits