Online Optimization

Axel Böhm

November 8, 2021

What is online Learning

Consider the repeated game: In each round $t = 1, \dots, T$

- an adversary chooses a (secret) number in $y_t \in [0, 1]$
- you guess the real number, choosing $x_t \in [0, 1]$;
- you pay the squared difference $(x_t - y_t)^2$.

Task: guess a sequence of numbers as precisely as possible. To be a game, we now have to decide what is the "winning condition". Let's see what makes sense to consider as winning condition.

Question: How to measure success?

Adversary plays i.i.d.

Consider: Adversary number are drawn from a fixed distribution (with mean μ and Variance σ^2). If we knew the distribution, we could pick the mean and pay in expectation $\sigma^2 T$ (optimal).

$$\mathbb{E}_Y\left[\sum_{t=1}^{T}(x_t - Y)^2\right] - \sigma^2 T,$$

or equivalently considering the average

$$\frac{1}{T}\mathbb{E}_Y\left[\sum_{t=1}^{T}(x_t - Y)^2\right] - \sigma^2 .$$

Minimizing Regret

Let's rewrite a bit more general

$$\mathbb{E}\left[\sum_{t=1}^{T}(x_t - Y)^2\right] - \min_{x\in[0,1]} \mathbb{E}\left[\sum_{t=1}^{T}(x - Y)^2\right] .$$

($\sigma^2 T$ was just the payoff of the best possible strategy)
Finally:

- remove the assumption on how the data is generated,
- consider any arbitrary sequence of y_t (can remove the expectation).

$$R_T := \sum_{t=1}^{T}(x_t - y_t)^2 - \min_{x\in[0,1]} \sum_{t=1}^{T}(x - y_t)^2$$

Called **Regret**, because it measures how much the algorithm regrets for not sticking on all the rounds to the optimal choice in hindsight.

General loss functions

Online Learning is the study of algorithms to minimize the *regret* over a sequence of loss functions with respect to an arbitrary competitor $u \in V \subseteq \mathbb{R}^d$:

$$R_T(u) := \sum_{t=1}^{T} \ell_t(x_t) - \sum_{t=1}^{T} \ell_t(u) \ .$$

Regret framework allows to

- reformulate problems in ML and optimization as similar games.
- analyze situations in which the data are not i.i.d. yet want to guarantee that the algorithm is "learning" something.

For example, online learning can be used to analyze

- Click prediction problems;
- Routing on a network;
- Convergence to equilibrium of *repeated* games.

It can *also* be used to analyze stochastic algorithms, e.g., Stochastic Gradient Descent, but the adversarial nature of the

Back to the numbers game

Consider: the **best strategy in hindsight**, that is argmin of the second term of the regret. Clearly

$$x_T^\star := \arg\min_{x\in[0,1]} \sum_{t=1}^{T}(x-y_t)^2 = \frac{1}{T}\sum_{t=1}^{T} y_t \ .$$

- Don't know the future: x_T^* is not an option in each round
- But do know the past. in each round: best number over the past.
- not because we expect the future to be like the past (not true)
- optimal guess should not change too much between rounds (so we can try to "track" it over time)

Hence, on each round t our strategy is to guess $x_t = x_{t-1}^\star = \frac{1}{t-1}\sum_{i=1}^{t-1} y_i$, called **Follow-the-Leader** (FTL).

Follow the leader

Now we show that this strategy will allow us to win the game.

Lemma

Let $V \subseteq \mathbb{R}^d$ and $\ell_t : V \to \mathbb{R}$ an arbitrary sequence of loss functions. Denote by $x_t^\star$ a minimizer of the cumulative losses over the previous t rounds in V. Then, we have

$$\sum_{t=1}^{T} \ell_t(x_t^\star) \leq \sum_{t=1}^{T} \ell_t(x_T^\star) .$$

Proof.

We prove it by induction over T. The base case is

$$\ell_1(x_1^\star) \leq \ell_1(x_1^\star),$$

that is trivially true. Now, for $T \geq 2$, we assume that

Follow the leader II

Theorem

Let $y_t \in [0,1]$ for $t = 1, \dots, T$ an arbitrary sequence of numbers. Let the algorithm's output $x_t = x^\star_{t-1} := \frac{1}{t-1}\sum_{i=1}^{t-1} y_i$. Then, we have

$$R_T = \sum_{t=1}^{T}(x_t - y_t)^2 - \min_{x\in[0,1]} \sum_{t=1}^{T}(x - y_t)^2 \leq 4 + 4 \ln T \ .$$

Proof.

Exercise. □

Failure of FTL

Let $V = [-1, 1]$ and consider the sequence of losses $\ell_t(x) = z_t x + i_V(x)$, where $z_1 = -0.5$

$$z_t = \begin{cases} 1, & t \text{ even} \\ -1, & t \text{ odd} \end{cases}$$

Predictions of FTL will be $x_t = 1$ for t even and $x_t = -1$ for t odd. Cumulative loss of the FTL algorithm will be T while the cumulative loss of the fixed solution $u = 0$ is 0. Thus, the regret of FTL is T.

Outlook:

- Follow the *regularized* leader
- Online gradient descent

Weighted majority algorithm

Consider the *learning from experts* scenario. Experts $= 1, \ldots, n$.
Decision: "Yes" or "No".

$$f_t(x_t) = \begin{cases} 1 & \text{if wrong} \\ 0 & \textit{otherwise} \end{cases}$$

(i) $w_1(i) = 1$ for all $i = 1, \ldots, n$

(ii) for $t = 1, \ldots, T$

1. compare weights $\sum_{i \in YES} w_t(i)$ vs. $\sum_{i \in NO} w_t(i)$
2. choose Yes or No depending on above comparison
3. observe feedback
4. update weights:

$$w_{t+1}(i) = \begin{cases} w_t(i) & \text{if Expert } i \text{ was right} \\ (1-\alpha)w_t(i) & \text{if Expert } i \text{ made a mistake} \end{cases}$$

Weighted majority algorithm II

Theorem

Let M_t be the number mistakes we make after t attempts and $m_t(i) = \#$ the number of mistakes expert i made... Then,

$$M_T \leq 2(1+\alpha)m_T(i) + 2\frac{\log(n)}{\alpha}$$

Also

$$M_T - m_T(i^*) = R_T$$

Proof of the Theorem

We always have $\|w_{t+1}\|_1 \leq \|w_t\|_1$. Also, if we made a mistake, then

$$\begin{aligned}\|w_{t+1}\|_1 &\leq \frac{1}{2}\|w_t\|_1 + \frac{1}{2}\|_1 w_t\|(1-\alpha)\\ &= \|w_t\|_1(1-\alpha/2)\\ &\leq \|w_1\|_1(1-\alpha/2)^{M_t}\\ &= n(1-\alpha/2)^{M_t}\end{aligned}$$

Next

$$w_{t+1}(i) = (1-\alpha)^{m_t(i)} \leq \|w_{t+1}\|_1$$

Combining the above two yields

$$(1-\alpha)^{m_t(i)} \leq n(1-\alpha/2)^{M_t}$$

and

$$m_t(i)\log(1-\alpha) \leq \log(n) + M_T \log(1-\alpha/2).$$

remainder of the proof

use the fact that for $x \in (0, \frac{1}{2})$

$$-x - x^2 \leq \log(1-x) \leq -x$$

to deduce that

$$-m_t(i)(\alpha + \alpha^2) \leq \log(n) - M_T \frac{\alpha}{2} - 2m_t(i)(1+\alpha) \quad \leq \frac{2}{\alpha} \log(n) - M_T$$

which yields

$$M_T - \leq 2m_t(i)(1+\alpha) + \frac{2}{\alpha} \log(n) -$$

Randomized Weighted Majority

Instead of picking the optinion of the (weighted) majority, we only do so with a **probability**.

(i) $w_1(i) = 1$ for all $i = 1, \dots, n$ and $\alpha \in (0, 1)$

(ii) for $t = 1, \dots, T$

1. compute $p_t(i) = w_t(i) / \|w_t\|_1$
2. choose expert i with probability $p_t(i)$
3. observe feedback
4. update weights:

$$w_{t+1}(i) = \begin{cases} w_t(i) & \text{if expert } i \text{ was right} \\ (1-\alpha) w_t(i) & \text{if expert } i \text{ made a mistake} \end{cases}$$

Comment: Randomizing algorithms typically improves the (worst case) analysis.

Randomized Weighted Majority

Instead of picking the optinion of the (weighted) majority, we only do so with a **probability**.

(i) $w_1(i) = 1$ for all $i = 1, \ldots, n$ and $\alpha \in (0,1)$
(ii) for $t = 1, \ldots, T$
  1. compute $p_t(i) = w_t(i)/\|w_t\|_1$
  2. choose expert i with probability $p_t(i)$
  3. observe feedback
  4. update weights:

$$w_{t+1}(i) = \begin{cases} w_t(i) & \text{if expert } i \text{ was right} \\ (1-\alpha)w_t(i) & \text{if expert } i \text{ made a mistake} \end{cases}$$

Comment: Randomizing algorithms typically improves the (worst case) analysis.

Randomized Weighted Majority contd.

As before:
$M_t = \#$ of mistakes we make after t attempts and $m_t(i) = \#$ of mistakes expert i made.

Theorem

$$\mathbb{E}[M_T] \leq (1+\alpha)m_T(i) + \frac{\log(n)}{\alpha}$$

Improved constants!

proof of randomized WMA

Multiplicative Weights Algorithm

Before: Loss l_t was 0 or 1 Now: General loss functions $\boldsymbol{\ell}_t = (\ell_t(1), \ldots, \ell_t(n))$ with $\ell_t(i) \in [-1, 1]$

(i) $w_1(i) = 1$ for all $i = 1, \ldots, n$ and $\alpha \in (0, 1)$

(ii) for $t = 1, \ldots, T$

1. compute $p_t(i) = w_t(i)/\|w_t\|_1$
2. choose expert i with probability $p_t(i)$
3. observe loss $\boldsymbol{\ell}_t = (\ell_t(1), \ldots, \ell_t(n))$
4. update weights:

$$w_{t+1}(i) = (1 - \alpha\ell_t(i))w_t(i)$$

Note that

$$\langle \boldsymbol{p}_t, \boldsymbol{\ell}_t \rangle = p_t(1)\ell_t(1) + \cdots + p_t(n)\ell_t(n) = \mathbb{E}_i[\ell_t(i)]$$

gives expected loss of round t.

Multiplicative Weights Algorithm [contd]

Theorem

if $\ell_t(i) \in [-1, 1]$ *and* $\alpha < \frac{1}{2}$, *then* ***MWA*** *guarantees*

$$\sum_{t=1}^{T} \langle \boldsymbol{p}_t, \boldsymbol{\ell}_t \rangle - \sum_{t=1}^{T} \ell_t(i) \leq \alpha \sum_{t=1}^{T} |\ell_t(i)| + \frac{\log(n)}{\alpha} \quad \forall i$$

Hedge Algorithm

(i) $w_1(i) = 1$ for all $i = 1, \ldots, n$ and $\alpha \in (0,1)$

(ii) for $t = 1, \ldots, T$

1. compute $p_t(i) = w_t(i)/\|w_t\|_1$
2. choose expert i with probability $p_t(i)$
3. observe loss $\ell_t = (\ell_t(1), \ldots, \ell_t(n))$
4. update weights:

$$w_{t+1}(i) = w_t(i) e^{-\alpha \ell_t(i)}$$

Note:

$$e^{-x} \approx 1 - x$$

Hedge Algorithm [contd]

Theorem

*If $\ell_t(i) \in [-1, 1]$ and $\alpha < \frac{1}{2}$, then **Hedge Alg.** guarantees*

$$\sum_{t=1}^{T} \langle \boldsymbol{p}_t, \boldsymbol{\ell}_t \rangle - \sum_{t=1}^{T} \ell_t(i) \le \alpha \sum_{t=1}^{T} \langle p_t, \ell_t \rangle^2 + \frac{\log(n)}{\alpha} \quad \forall i.$$

Observe: Iteration t is just

$$w_{t+1}(i) = w_t(i) e^{-\alpha \ell_t(i)}$$

$$p_{t+1}(i) = \frac{w_{t+1}(i)}{\|w_{t+1}\|_1}$$

Online mirror descent! (KL-divergence setting:)

$$h(x) = \sum_i x(i) \log(x(i))$$

Hedge Algorithm [contd]

Theorem

If $\ell_t(i) \in [-1, 1]$ and $\alpha < \frac{1}{2}$, then ***Hedge Alg.*** *guarantees*

$$\sum_{t=1}^{T} \langle \boldsymbol{p}_t, \boldsymbol{\ell}_t \rangle - \sum_{t=1}^{T} \ell_t(i) \le \alpha \sum_{t=1}^{T} \langle p_t, \ell_t \rangle^2 + \frac{\log(n)}{\alpha} \quad \forall i.$$

Observe: Iteration t is just

$$w_{t+1}(i) = w_t(i) e^{-\alpha \ell_t(i)}$$

$$p_{t+1}(i) = \frac{w_{t+1}(i)}{\|w_{t+1}\|_1}$$

Online mirror descent! (KL-divergence setting:)

$$h(x) = \sum_i x(i) \log(x(i))$$