Online Optimization

Oran Danon

§1. Introduction

We play a game against an adversary while being aided by n experts giving us advice: For $t = 1, 2, 3, \dots, T$ do:

1. Each expert $i \in \{1, 2, \dots, n\}$ advises a value in $[-1, 1]$.
2. We pick a distribution vector $\vec{x_t} \coloneqq (x_t(1), x_t(2), \dots, x_t(n))$, where $x_t(i)$ denotes the probability to take the i^{th} expert advice at time t.
3. The adversary knowing the advices we got and $\vec{x_t}$ reveals a cost/lost vector $\vec{\ell_t} \coloneqq (\ell_t(1), \dots, \ell_t(n))$, where $\ell_t(i)$ denotes the price to pay if following expert's i advice at time t.
4. we pay $\vec{x_t} \cdot \vec{\ell_t}$.

Goal: Devise an algorithm to determine $\vec{x_t}$ for all t as to minimize

$$\sum_{i=1}^{T} \vec{x_t} \cdot \vec{\ell_t}$$

§ 1.1 Definitions.

Online convex optimization deals with the following setup:

- $K \coloneqq$ convex set of feasible solutions.
- One seeks an algorithm such that for all discrete time step $t = 1, 2, \dots$ the algorithm comes up with solution $\vec{x_t} \in K$.
- After $\vec{x_t}$ is fixed a convex cost function $f_t : K \to \mathbb{R}$ taken from a collection of admissible set of cost functions, namely R, is revealed and the algorithm pays the loss $f_t(x_t)$.
  - In Particular note that f_t is tied to t and $f_t \neq f_{t'}$ is possible.

The algorithm has to come up with a solution without knowing what are the functions it is supposed to optimize.

- The cost functions $f_1, f_2, \ldots$ are not set in advance. Instead, those are generated dynamically influenced by the selection of the algorithm.

Definition 1. *The **offline optimum** after T steps is given by:*

$$\min_{x \in K} \sum_{t=1}^{T} f_t(x)$$

Definition 2. *The **regret** after T steps is given by*

$$\mathbf{REGERT_T} \coloneqq \underbrace{\sum_{t=1}^{T} f_t(x_t)}_{\substack{\text{what the algorithm paid} \\ \text{(note that } f_t \text{ is applied to } x_t\text{)}}} - \underbrace{\min_{x \in K} \sum_{t=1}^{T} f_t(x)}_{\text{offline optimum}}$$

§2. Multiplicative Weights Algorithm

§ 2.1 The setting:.

- n - experts giving advice, each giving advice at discrete times $t = 1, 2, \ldots$
- We seek an algorithm that at each time t outputs a probability vector: $\vec{x_t} \coloneqq (x_t(1), \ldots, x_t(n))$, where $x_t(i)$ is the probability that the advice of expert i should be taken at time t.
  - In particular:

$$\sum_{i=1}^{n} x_t(i) = 1 \qquad \forall_t$$
$$x_t(i) \geq 0 \qquad \forall i \in \{1, 2 \ldots, n\}, \forall t$$

- After the algorithm makes its choice $\vec{x_t}$ a loss vector $\vec{\ell_t}$ is revealed where $\vec{\ell_t} \coloneqq (\ell_t(1), \ldots, \ell_t(n))$, where $\ell_t(i)$ denotes the loss one has to pay at time t if the advice of expert i is taken.
- The excepted loss at time t by the algorithm is then given by

$$< \vec{x_t}, \vec{\ell_t} > \coloneqq \sum_{i=1}^{n} x_t(i)\ell_t(i).$$

- The algorithm's regret after T steps is then:

$$\mathbf{REGERT_T} := \sum_{\mathbf{i=1}}^{\mathbf{T}} < \tilde{\mathbf{x}}_\mathbf{t}, \tilde{\ell}_\mathbf{t} > - \min_{\mathbf{1 \leq i \leq n}} \sum_{\mathbf{i=1}}^{\mathbf{T}} \ell_\mathbf{t}(\mathbf{i}).$$

- How does all this fit to the initial framework we described for online optimization?
  - K is the set $\Delta \subseteq \mathbb{R}^n$ of probability distributions over the sample space $\{1, 2, \ldots, n\}$
  - The $\overbrace{\cancel{\text{loss}}}^{\text{cost}}$ function $f_t(\vec{x}),\ \vec{x} \in \Delta$, has the form $\sum_i x_i(t)\ell_i(t)$ (so it is linear).
- It is clear that in order to bound the regret one also has to assume a bound on the magnitude of the loss function, so throughout we shall assume $|\ell_t(i)| \leq 1$ for all $1 \leq i \leq n$ and for all t. (Otherwise, we can scale everything by $\max_{t,i} |\ell_t(i)|$).

§ 2.2 The algorithm.

- The algorithm maintains a vector $\vec{w_t} := (w_t(1), w_t(2), \ldots, w_t(n))$ of "weights" per time step t.
- The vector $\vec{w_t}$ is initialized as $\vec{w_1} := (1, 1, \ldots, 1)$ and $\vec{x_1} := (1/n, \ldots, 1/n)$.
- At time t:
  - set $w_t(i) := w_{t-1}(i)e^{-\varepsilon \ell_{t-1}(i)}$, $\varepsilon > 0$ is hardwired into the algorithm and will be optimized later on.
  - set $x_t(i) := \frac{w_t(i)}{\sum_{j=1}^n w_t(j)}$
- **Heuristic:** the higher the loss expert i brought at previous step the smaller is its weight.

$$w_{t-1}(i)e^{-\varepsilon \ell_{t-1}(i)}$$

Theorem 3. *Assuming $|\ell_t(i)| \leq 1$ for all i and for all t then for all $0 < \varepsilon := \varepsilon(n) < 1/2$ after T steps the algorithm above fetches:*

$$\begin{aligned} \mathbf{REGERT_T} &\leq \varepsilon \sum_{t=1}^{T} \sum_{i=1}^{n} x_t(i)\ell_t^2(i) + \frac{\ln n}{\varepsilon} \\ &\leq \varepsilon T + \frac{\ln n}{\varepsilon} \end{aligned}$$

Remark 4. If $T > 4 \ln n$ and $\varepsilon = \sqrt{\frac{\ln n}{T}}$ then:

$$\mathbf{REGERT_T} \leq \mathbf{2}\sqrt{\ln \mathbf{n}} \cdot \sqrt{\mathbf{T}}$$

$\mathbf{REGERT_T}$ is independent of the loss vectors chosen by the adversary !!!

For time t set $w_t := \sum_{i=1}^{n} w_t(i)$ – total expert weight assigned.
Define

$$L^\star := \min_{1\leq i\leq n} \sum_{t=1}^{T} \ell_t(i) \text{– loss of the best expert.}$$

Lemma 5. *(If w_{T+1} is small then $L^\star$ is large)*

$$W_{T+1} \geq e^{-\varepsilon L^\star}$$

The messages from the lemma:

- The more losses the algorithm accumulates throughout the lower the overall weight assigned will be.

  > The more losses the less weight.

- The lemma then says that if the algorithm incurs a lot of overall loss then so will the expert which incurs the least loss over all experts.

Proof.

- Let $j \in \{1, 2, \ldots, n\}$ satisfy $L^\star = \sum_{t=1}^{T} \ell_t(j)$.
- Then

$$W_{T+1} = \sum_{i=1}^{n} w_{T+1}(i) = \sum_{i=1}^{n} w_T(i)\, e^{-\varepsilon \ell_T(i)} = \sum_{i=1}^{n} w_{T-1}(i) e^{-\varepsilon \ell_{T-1}(i) - \varepsilon \ell_{T-1}(i)}$$
$$= \sum_{i=1}^{n} e^{-\varepsilon \sum_{t=1}^{T} \ell_t(i)} \geq e^{-\varepsilon \sum_{t=1}^{T} \ell_t(j)} = e^{-\varepsilon L^\star}$$

∎

Previously we stated that the more loss the algorithm incure the smallest is W_{T+1}. We now make this precise:

- Consider the term:

$$\sum_{i=1}^{n} x_t(i) e^{-\varepsilon \ell_t(i)}$$

then:

$$\boxed{\sum_{i=1}^{n} x_t(i)e^{-\varepsilon \ell_t(i)}} = \sum_{i=1}^{n} \frac{w_t(i)}{W_t} e^{-\varepsilon \ell_t(i)} \tag{6}$$

$$= \frac{1}{W_t} \sum_{i=1}^{n} w_t(i) e^{-\varepsilon \ell_t(i)} \tag{7}$$

$$= \boxed{\frac{W_{t+1}}{W_t}} \tag{8}$$

- Applying the Taylor series of $e^{-x} = \sum_{k=0}^{\infty} \frac{x^n}{k!}$ for e^{-z} we can obtain $e^{-z} \leq 1 - z + z^2$ for $|z| \leq 1/2$. Hence:

$$e^{\overbrace{-\varepsilon \ell_t(i)}^{\substack{\text{Recall that } 0 < \varepsilon < 1/2 \\ \text{and that } |\ell_t(i)| \leq 1,\ \forall_{i,t} \\ \text{so } |\varepsilon \ell_t(i)| \leq 1/2}}} \leq 1 - \varepsilon \ell_t(i) + \varepsilon^2 \ell_t^2(i)$$

Then returning back to (6) and substitute what we just got implies:

$$\begin{aligned} \frac{W_{t+1}}{W_t} &= \sum_{i=1}^{n} x_t(i) e^{-\varepsilon \ell_t(i)} \\ &\leq \sum_{i=1}^{n} x_t(i) \left(1 - \varepsilon \ell_t(i) + \varepsilon^2 \ell_t^2(i)\right) \\ &= \underbrace{\sum_{i=1}^{n} x_t(i)}_{=1} - \varepsilon < \vec{x_t}, \vec{\ell_t} > + \varepsilon^2 < \vec{x_t}, \vec{\ell_t^2} > \end{aligned}$$

Put another way, we've got the following recursion relation:

$$W_{t+1} \leq W_t \left(\right) 1 - \varepsilon < \vec{x_t}, \vec{\ell_t} > + \varepsilon^2 < \vec{x_t}, \vec{\ell_t^2} >$$

Solving the above recursion and submitting the values from initialization (i.e. $W_1 = n$) yields:

Lemma 9.

$$W_{T+1} \leq n \cdot \prod_{t=1}^{T} (1 - \varepsilon < \vec{x_t}, \vec{\ell_t} > + \varepsilon^2 < \vec{x_t}, \vec{\ell_t^2} >$$

Note that if the algorithm incurs large losses then the accumulative weight is small.

Taking ln on both sides gives:

$$\ln(W_{T+1}) \leq \ln n + \sum_{t=1}^{T} \ln\left(1 - \varepsilon < \vec{x}_t, \vec{\ell}_t > + \varepsilon^2 < \vec{x}_t, \vec{\ell}_t^2 >\right) \tag{10}$$

Recall that $1 - z \leq e^{-z}$ when $|z| \leq 1$, consider,

$$z = \varepsilon \underbrace{< \vec{x}_t, \vec{\ell}_t >}_{\substack{\text{excepted loss} \\ \text{at time } t}} - \varepsilon^2 \underbrace{< \vec{x}_t, \vec{\ell}_t^2 >}_{\substack{\text{excepted squared} \\ \text{loss at time } t}}$$

Note that:

$$\left\|\vec{\ell}_t\right\|_\infty \leq 1 \implies \begin{cases} < \vec{x}_t, \vec{\ell}_t > & \leq \sum_{i=1}^n x_t(i) & = 1 \\ < \vec{x}_t, \vec{\ell}_t^2 > & \leq \sum_{i=1}^n x_t(i) & = 1 \end{cases}$$

Then

$$\begin{aligned} \ln\left(1 - \varepsilon < \vec{x}_t, \vec{\ell}_t > + \varepsilon^2 < \vec{x}_t, \vec{\ell}_t^2 >\right) &\leq \left(\exp\left(-\varepsilon < \vec{x}_t, \vec{\ell}_t > + \varepsilon^2 < \vec{x}_t, \vec{\ell}_t^2 >\right)\right) \\ &= \varepsilon^2 < \vec{x}_t, \vec{\ell}_t^2 > - \varepsilon < \vec{x}_t, \vec{\ell}_t > \end{aligned}$$

Returning to (10) yields:

$$\ln(W_{T+1}) \leq \ln n + \sum_{t=1}^{T} \left(\varepsilon^2 < \vec{x}_t, \vec{\ell}_t^2 > - \varepsilon < \vec{x}_t, \vec{\ell}_t >\right) \tag{11}$$

Proof of Theorem 3. From lemma 5 we have:

$$\ln W_{T+1} \geq -\varepsilon L^\star$$

Together with equation 11 one obtains the following:

$$\begin{aligned} &-\varepsilon L^\star \leq \ln(W_{T+1}) \leq \ln n + \sum_{t=1}^{T} \left(\varepsilon^2 < \vec{x}_t, \vec{\ell}_t^2 > - \varepsilon < \vec{x}_t, \vec{\ell}_t >\right) \\ &\varepsilon \left(\sum_{t=1}^{T} < \vec{x}_t, \vec{\ell}_t > - L^\star\right) \leq \ln n + \varepsilon^2 \sum_{t=1}^{T} < \vec{x}_t, \vec{\ell}_t^2 > \end{aligned}$$

$$\underbrace{\sum_{t=1}^{T} < \vec{x}_t, \vec{\ell}_t > -L^\star}_{\textbf{REGERT}_\textbf{T}} \leq \frac{\ln n}{\varepsilon} + \varepsilon \sum_{t=1}^{T} < \vec{x}_t, \vec{\ell}_t^2 > .$$

■

§3. Pseudorandom Sets

Random bits are hard to come by (so I am told). The following is then interest:

Definition 12. *Let $\mathcal{C} := \{C_1, \ldots, C_N\}$ be a collection of boolean functions*

$$C_i : \{0,1\}^n \to \{0,1\}, \quad \forall_{1 \leq i \leq N}$$

*A multiset $\mathcal{S} \subseteq \{0,1\}^n$ is said to be ε-**pseudo-random** for $\mathcal{C}$ if for every $C_i \in \mathcal{C}$ it holds:*

$$\left| \mathop{\mathbf{Pr}}_{u \sim \mathcal{U}ni\{0,1\}^n} [C_i(u) = 1] - \mathop{\mathbf{Pr}}_{s \sim \mathcal{S}} [C_i(s) = 1] \right| \leq \varepsilon$$

Note the following:

- Drawing uniformly from $\{0,1\}^n$ requires n bits while drawing from $\mathcal{S}$ requires $\log_2 |\mathcal{S}|$ bits. Ideally one would have $|\mathcal{S}|$ small.
- Standard probabilistic techniques applied to a random set of size $O\left(\frac{\log N}{\varepsilon^2}\right)$ would produce an ε-pseudorandom set of that size so that $\log \log N + 2 \log {}^1\!/\!_\varepsilon + O(1)$ random bits are enough. The problem with this argument is that it does not yield a "construction" for the desired set.
- In what follows the multiplicative weights algorithm is used in order to construct such a set.
- If $\mathcal{C}$ is symmetric in the sense that

$$C \in \mathcal{C} \implies 1 - C \in \mathcal{C}$$

then the pseudo-randomness condition is equivalent to

$$\forall_{C \in \mathcal{C}} : \qquad \mathop{\mathbf{Pr}}_{u \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} [C_i(u) = 1] - \mathop{\mathbf{Pr}}_{s \sim \mathcal{S}} [C_i(s) = 1] \geq -\varepsilon$$

§ 3.1 Experts setup.

- There is an expert i for function C_i.
- At time t the algorithm produces a probabilistic vector $\vec{x_t}$.
- At time t the adversary chooses a string $s_t \in \{0,1\}^n$ and defines the cost function

$$f_t(\vec{x}) := \sum_{i=1}^{N} x(i) \cdot \left(\mathop{\mathbf{Pr}}_{u \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} [C_i(u) = 1] - C_i(s_t) \right)$$

  Where s_t is chosen such that $f_t(\vec{x_t}) \geq 0$.

Claim 13. *For all t a choice s_t is possible.*

Proof. For a random set $s \sim \{0,1\}^n$:

$$\mathop{\mathbb{E}}_{s \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} \sum_{i=1}^{N} x_t(i) \cdot \left(\mathop{\mathbf{Pr}}_{u \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} [C_i(u) = 1] - \overbrace{C_i(s)}^{\text{This is the only R.V.}} \right)$$
$$= \sum_{i=1}^{N} x_t(i) \mathop{\mathbf{Pr}}_{u \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} [C_i(u) = 1] - \sum_{i=1}^{N} x_t(i) \mathop{\mathbb{E}}_{s \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} C_i(s) = 0$$

Thus there exists s_t such that

$$\sum_{i=1}^{N} x(i) \cdot \left(\mathop{\mathbf{Pr}}_{u \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} [C_i(u) = 1] - C_i(s_t) \right) \geq 0$$

■

Claim 14. *The cost function $f_t(x)$ is of the form $f_t(\vec{x}) =< \vec{\ell_t}, \vec{x_t} >$ with $\left\| \vec{\ell_t} \right\|_\infty \leq 1$.*

Proof. We are led to having

$$\ell_t(i) = \mathop{\mathbf{Pr}}_{u \sim \mathcal{U}\mathrm{ni}\{0,1\}^n} [C_i(u) = 1] - C_i(s_t)$$

and this clearly satisfies $|\ell_t(i)| \leq 1$ ■

Applying the multiplicative weights algorithm to this setting we got that after T steps $\mathbf{REGERT_T} \leq \mathbf{2\sqrt{T \ln n}}$, and throughout this algorithm a set $S_1, S_2, \ldots, S_T$ of strings have chosen by the adversary. This sequence (which need not be a set) defines a $\underbrace{2\sqrt{\frac{\ln n}{T}}}_{\text{This is the } \varepsilon}$ - pseudorandom set of size T.

$$\text{If } \varepsilon = 2\sqrt{\frac{\ln n}{T}} \text{ then } T = \frac{4 \ln n}{\varepsilon^2}$$

so we get an ε-pseudo-random set of size $\frac{4\ln n}{\varepsilon^2}$. To prove this let us note first by construction:

$$\sum_{t=1}^{T} f_t(x_t) \geq 0.$$

Moreover, recalling that:

$$\mathbf{REGERT_T} = \underbrace{\sum_{t=1}^{T} f_t(\vec{x_t})}_{\geq \mathbf{0}} - \min_{\mathbf{x}\in\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}} \sum_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{f_t}(\mathbf{x})$$

so that

$$-\mathbf{REGERT_T} = \underbrace{-\sum_{t=1}^{T} f_t(\vec{x_t})}_{\leq \mathbf{0}} + \min_{\mathbf{x}\in\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}} \sum_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{f_t}(\mathbf{x})$$

Then

$$-\mathbf{REGERT_T} \leq \min_{\mathbf{x}\in\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}} \sum_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{f_t}(\mathbf{x})$$

Let $\vec{e_i} \coloneqq (0 \dots \underset{i}{1} \dots 0)$ be the distribution vector trusting expert i completely. Then:

$$-\mathbf{REGERT_T} \leq \min_{\mathbf{x}\in\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}} \sum_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{f_t}(\mathbf{x}) \leq \sum_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{f_t}(\tilde{\mathbf{e}}_{\mathbf{i}}) \qquad \forall_{\mathbf{i}}$$

In particular this yields:

$$\forall_{j\in[n]} : \sum_{t=1}^{T} \left(\sum_{i=1}^{N} e_j(i) \left(\underset{u\sim\mathcal{U}\mathrm{ni}\{0,1\}^n}{\mathbf{Pr}} \left[C_i\left(u\right) = 1\right] - C_i(s_t) \right) \right) \geq -2\sqrt{T \ln n}$$

as $e_j(t) = 1$ only for $t = j$ it follows that:

$$\forall_{j\in[n]} : \sum_{t=1}^{T} \left(\underset{u\sim\mathcal{U}\mathrm{ni}\{0,1\}^n}{\mathbf{Pr}} \left[C_j\left(u\right) = 1\right] - C_j(s_t) \right) \geq -2\sqrt{T \ln n}$$

Division by T yields:

$$\forall_{j\in[n]} : \underbrace{\cancel{\frac{1}{T}} \cancel{\sum_{t=1}^{T}} \underset{u\sim\mathcal{U}\mathrm{ni}\{0,1\}^n}{\mathbf{Pr}} \left[C_j\left(u\right) = 1\right]}_{\underset{u\sim\mathcal{U}\mathrm{ni}\{0,1\}^n}{\mathbf{Pr}}[C_j(u)=1]} - \underbrace{\frac{1}{T} \sum_{t=1}^{T} C_j(s_t)}_{\underset{s\sim\mathcal{U}\mathrm{ni}\{S_1,\dots,S_T\}}{\mathbf{Pr}}[C_j(s)=1]} \geq -2\sqrt{T \ln n}$$

The claim follows. $\square$