Online Convex Optimization and Its Surprising Applications

Francesco Orabona

KAUST

Learning Theory Summer School, Copenhagen, Denmark, 2026

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Parameter-free Online Algorithms

What is a Parameter-free Algorithm?

Definition

We call parameter-free any online algorithm that satisfies the optimal regret bound with respect to T and to all the comparators in the feasible set $\mathcal{V}$, up to poly-logarithmic multiplicative factors.

- Exponentiated Gradient: $Regret_T(\boldsymbol{u}) \leq \frac{KL(\boldsymbol{u};\boldsymbol{\pi})}{\eta} + \frac{T\eta}{2} \Rightarrow$
  NormalHedge: $Regret_T(\boldsymbol{u}) = \mathcal{O}(\sqrt{T(KL(\boldsymbol{u};\boldsymbol{\pi})+1)})$ [Chaudhuri et al., NeurIPS'09][Chernov&Vovk, UAI'10][Orabona&Pál, NeurIPS'16]
- OSD: $Regret_T(\boldsymbol{u}) \leq \frac{\|\boldsymbol{x}_1-\boldsymbol{u}\|_2^2}{2\eta} + \frac{\eta T}{2} \Rightarrow$
  KT (next slides): $Regret_T(\boldsymbol{u}) = \mathcal{O}(\|\boldsymbol{x}_1 - \boldsymbol{u}\|_2 \sqrt{T \ln(1 + T\|\boldsymbol{x}_1 - \boldsymbol{u}\|_2/\epsilon) + \epsilon})$
- How do I tune my learning rate/regularizer without knowing $\boldsymbol{u}$?

Parameter-Free through Duality on Guarantee

- We saw the online-to-batch conversion (deterministic case for simplicity):

$$F(\bar{\boldsymbol{x}}_T) - F(\boldsymbol{u}) \leq \frac{1}{T}\sum_{t=1}^{T}(F(\boldsymbol{x}_t) - F(\boldsymbol{u})) \leq \frac{1}{T}\sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle \leq \frac{\psi_T(\boldsymbol{u})}{T}$$

Theorem (McMahan&Orabona, COLT'14)

Consider two sequences $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ and $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_T$. Then

$$\begin{aligned} \sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u}\rangle &\leq \psi_T(\boldsymbol{u}), \quad \forall \boldsymbol{u} \in \mathbb{R}^d \\ &\Updownarrow \\ -\sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle &\geq \psi_T^\star\left(-\sum_{t=1}^{T}\boldsymbol{g}_t\right), \ \forall \boldsymbol{g}_1, \ldots, \boldsymbol{g}_T \end{aligned}$$

where $\psi_T^\star$ is the Fenchel conjugate of ψ_T defined as $\psi_T^\star(\boldsymbol{\theta}) = \sup_{\boldsymbol{x}} \langle \boldsymbol{\theta}, \boldsymbol{x}\rangle - \psi_T(\boldsymbol{x})$

The Main Tool

- Assume $\|\boldsymbol{g}_t\|_2 \leq 1$
- Set $\boldsymbol{x}_t = \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t} \left(1 - \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x}_i \rangle\right)$
- Claim: $\boldsymbol{x}_t$ guarantees

$$-\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle \geq \psi_T^\star \left(-\sum_{t=1}^{T} \boldsymbol{g}_t\right)$$

  where $\psi_T^\star(\boldsymbol{\theta}) \approx \frac{1}{\sqrt{T}} \exp\left(\frac{\|\boldsymbol{\theta}\|_2^2}{2T}\right) - 1$

- This implies $\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{u} \rangle \leq \|\boldsymbol{x}^\star\|_2 \sqrt{T \ln(\|\boldsymbol{u}\|_2 T + 1)} + 1$

- Where do the update rule and the inequality in orange come from?

Optimization through Optimal Gambling

Krichevsky&Trofimov (KT) betting strategy:

- Observe sequence of coins outcomes $c_t \in [-1, 1]$, start with \$1, bet x_t money, win/lose $x_t c_t$
- On round t bet a signed fraction of your money equal to $\frac{\sum_{i=1}^{t-1} c_i}{t}$
- Exponential amount of money

$$\text{Wealth of KT after } T \text{ rounds} = 1 + \sum_{t=1}^{T} x_t c_t \geq \frac{\exp\left(\frac{(\sum_{t=1}^{T} c_t)^2}{2T}\right)}{2\sqrt{T}}$$

- No assumptions on the coin!

- We need to prove that $-\sum_{t=1}^{T} g_t x_t \geq \psi_T^\star\left(-\sum_{t=1}^{T} g_t\right)$
- In 1d, set $c_t = -g_t$ and assume $|g_t| \leq 1$ then we have it!
- It works in the vector case too

[Krichevsky&Trofimov, 1981][Orabona&Pal, NeurIPS'16]

Extensions

- It works in any number of dimensions, even Hilbert spaces

$$\boldsymbol{x}_t = \boldsymbol{x}_0 + \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t} \left(1 - \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x}_i \rangle \right)$$

- It works with stochastic subgradients

- Variant that does not need to know the Lipschitz constant [Cutkosky, COLT'19]
- Variant works with constrained sets [Cutkosky&Orabona, COLT'18]
- Variant adapts to the strong convexity in the stochastic setting (bounded stochastic subgradients and domain) [Cutkosky&Orabona, COLT'18]

Surprising Applications of Online Learning

Online Learning is Much More than Online Learning

- Online Convex Optimization might seem only concerned with losses&regret
- In reality, it is about proving inequalities on arbitrary sequences of data
- Sometimes, the inequalities are more important than the algorithms

- Here, I'll try to convince you of this view

From Online Convex Optimization to Non-Convex Non-Smooth Optimization

a.k.a.

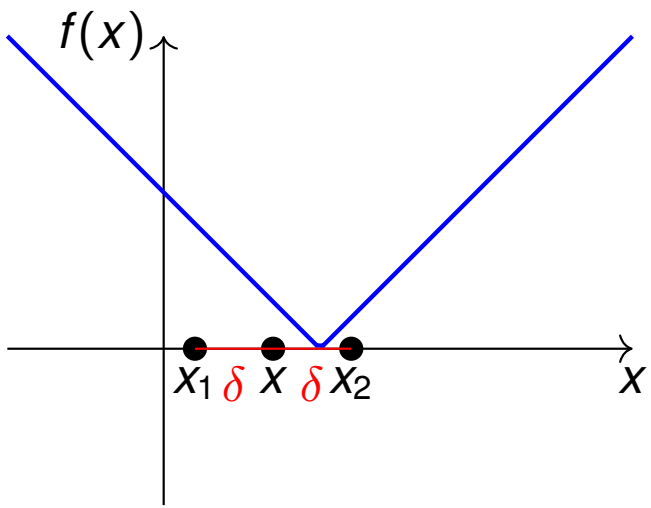
Online losses can be anything

Non-convex Optimization

- For convex optimization, we study $F(\boldsymbol{x}_T) - F(\boldsymbol{u})$
- For non-convex smooth optimization, we study $\mathbb{E}_i[\|\nabla F(\boldsymbol{x}_i)\|_2^2]$
- What can we do for non-smooth non-convex? Example: ConvNets with ReLUs

Definition (Zhang et al. ICML'20)

A point $\boldsymbol{x}$ is an (δ, ϵ)-stationary point of an almost-everywhere differentiable function F if there is a finite subset S of the ball of radius δ centered at $\boldsymbol{x}$ such that for $\boldsymbol{y}$ selected uniformly at random from S, $\mathbb{E}[\boldsymbol{y}] = \boldsymbol{x}$ and $\|\mathbb{E}[\nabla F(\boldsymbol{y})]\| \leq \epsilon$



If δ is small enough, it codifies our intuition on points close to a minimum

Well-Behaved Functions

We will assume that the functions are well-behaved in the sense that

$$F(\boldsymbol{y}) - F(\boldsymbol{x}) = \int_0^1 \langle \nabla F(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})), \boldsymbol{y} - \boldsymbol{x} \rangle dt$$

Up to perturbing the function with some noise, this holds for locally Lipschitz functions

Using OCO for Non-convex Optimization

Require: An OCO algorithm, duration of cycle K, initial point $\boldsymbol{x}_0$

$j = 0$
for $t = 1$ **to** T **do**
  if $mod(t, K) == 1$ **then**
    Reset OCO algorithm
    $j = j + 1$
    $\bar{\boldsymbol{x}}_j = \boldsymbol{0}$
  end if
  Receive $\boldsymbol{m}_t$ from OCO algorithm
  $\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \boldsymbol{m}_t$
  Sample s_t uniformly in $[0, 1]$
  $\boldsymbol{x}'_t = \boldsymbol{x}_{t-1} - s_t \boldsymbol{m}_t$
  Pass $\ell_t(\boldsymbol{x}) = -\langle \nabla F(\boldsymbol{x}'_t), \boldsymbol{x} \rangle$ to OCO algorithm
  $\bar{\boldsymbol{x}}_j = \bar{\boldsymbol{x}}_j + \boldsymbol{x}'_t / K$
end for
return $\bar{\boldsymbol{x}}_J$ where J is drawn uniformly at random between 1 and T/K

- **Important:** The OCO algorithm decides the updates not the iterates

[Cutkosky et al., ICML'23]

Main Result

Theorem

Let the OCO algorithm be OGD over the L_2 ball of radius D. Then, we have

$$\mathbb{E}\left[\frac{1}{T/K}\sum_{i=1}^{T/K}\left\|\frac{1}{K}\sum_{t=1}^{K}\nabla F(\boldsymbol{x}'_{(i-1)K+t})\right\|_2\right] \leq \frac{F(\boldsymbol{x}_0)-\inf_{\boldsymbol{x}} F(\boldsymbol{x})}{DT}+\frac{1}{\sqrt{K}}$$

Moreover, set $D=\delta/K$, $K=\left(\frac{T\delta}{F(\boldsymbol{x}_0)-\inf_{\boldsymbol{x}} F(\boldsymbol{x})}\right)^{\frac{2}{3}}$, and return $\bar{\boldsymbol{x}}_J$ where J is uniformly at random, then in expectation $\bar{\boldsymbol{x}}_j$ is $(\delta, \mathcal{O}((T\delta)^{-\frac{1}{3}}))$-stationary point.

- The choice of D: $\bar{\boldsymbol{x}}_j$ is the average of K points contained in ball of radius δ
- With the chosen D, we have

$$\frac{F(\boldsymbol{x}_0)-\inf_{\boldsymbol{x}} F(\boldsymbol{x})}{DT}+\frac{1}{\sqrt{K}} = \frac{K(F(\boldsymbol{x}_0)-\inf_{\boldsymbol{x}} F(\boldsymbol{x}))}{T\delta}+\frac{1}{\sqrt{K}}$$

- $\mathbb{E}\left[\frac{1}{T/K}\sum_{i=1}^{T/K}\left\|\frac{1}{K}\sum_{t=1}^{K}\nabla F(\boldsymbol{x}'_{(i-1)K+t})\right\|_2\right] =$
  $\mathbb{E}\left[\left\|\frac{1}{K}\sum_{t=1}^{K}\nabla F(\boldsymbol{x}'_{(J-1)K+t})\right\|_2\right] = \mathcal{O}\left((T\delta)^{-\frac{1}{3}}\right)$

[Cutkosky et al., ICML'23]

From Function Value to Gradients

In all optimization analyses we need to link function values to gradients:

- Convex functions: $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle$
- Non-convex M-smooth: $f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{M}{2} \|\boldsymbol{x} - \boldsymbol{y}\|^2$

- What can we use for non-convex non-smooth?

Key Observation

- We evaluate the gradient in $\boldsymbol{x}'_t = \boldsymbol{x}_{t-1} - s_t \boldsymbol{m}_t = \boldsymbol{x}_{t-1} + s_t(\boldsymbol{x}_t - \boldsymbol{x}_{t-1})$
- Hence, we have

$$\mathbb{E}_{s_t} \nabla F(\boldsymbol{x}'_t) = \int_0^1 \nabla F(\boldsymbol{x}_{t-1} + t(\boldsymbol{x}_t - \boldsymbol{x}_{t-1}))dt$$

- This allows us to say that

$$\begin{aligned} F(\boldsymbol{x}_t) - F(\boldsymbol{x}_{t-1}) &= \int_0^1 \langle \nabla F(\boldsymbol{x}_{t-1} + t(\boldsymbol{x}_t - \boldsymbol{x}_{t-1})), \boldsymbol{x}_t - \boldsymbol{x}_{t-1} \rangle dt \\ &= \langle \mathbb{E}_{s_t}[\nabla F(\boldsymbol{x}'_t)], \boldsymbol{x}_t - \boldsymbol{x}_{t-1} \rangle \end{aligned}$$

- This holds without assuming convexity nor smoothness!

[Zhang et al. ICML'20][Cutkosky et al., ICML'23]

Proof

Proof.

Using the key observation, for the first cycle we have

$$\begin{aligned} F(\boldsymbol{x}_t) - F(\boldsymbol{x}_{t-1}) &= \langle \mathbb{E}_{s_t}[\nabla F(\boldsymbol{x}'_t)], \boldsymbol{x}_t - \boldsymbol{x}_{t-1} \rangle = -\langle \mathbb{E}_{s_t}[\nabla F(\boldsymbol{x}'_t)], \boldsymbol{m}_t \rangle \\ &= \langle -\mathbb{E}_{s_t}[\nabla F(\boldsymbol{x}'_t)], \boldsymbol{m}_t - \boldsymbol{u} \rangle - \langle \mathbb{E}_{s_t}[\nabla F(\boldsymbol{x}'_t)], \boldsymbol{u} \rangle \end{aligned}$$

Taking full expectation, summing over $t = 1, \ldots, K$, for any $\|\boldsymbol{u}\|_2 \leq D$, we have

$$\begin{aligned} \mathbb{E}[F(\boldsymbol{x}_K)] - F(\boldsymbol{x}_0) &= \underbrace{\mathbb{E}\left[\sum_{t=1}^{K} \langle -\nabla F(\boldsymbol{x}'_t), \boldsymbol{m}_t - \boldsymbol{u} \rangle\right]}_{Regret_K(\boldsymbol{u})} - \mathbb{E}\left[\sum_{t=1}^{K} \langle \nabla F(\boldsymbol{x}'_t), \boldsymbol{u} \rangle\right] \\ &\leq D\sqrt{K} - \mathbb{E}\left[\sum_{t=1}^{K} \langle \nabla F(\boldsymbol{x}'_t), \boldsymbol{u} \rangle\right] \end{aligned}$$

Choose $\boldsymbol{u} = D \frac{\sum_{t=1}^{K} \nabla F(\boldsymbol{x}'_t)}{\left\| \sum_{t=1}^{K} \nabla F(\boldsymbol{x}'_t) \right\|_2}$ to have $\sum_{t=1}^{K} \langle \nabla F(\boldsymbol{x}'_t), \boldsymbol{u} \rangle = -D \left\| \sum_{t=1}^{K} \nabla F(\boldsymbol{x}'_t) \right\|_2$.
Summing over the cycles and dividing by DT ends the proof. □

Only a Hack or Something Fundamental?

One might wonder if the above reduction is only a hack or it discovers something more fundamental.

One way to convince you is to take a look at the resulting procedure

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{x}_{t-1} - \boldsymbol{m}_t \\ \boldsymbol{g}_t &= \nabla F(\boldsymbol{x}_t + (s_t - 1)\boldsymbol{m}_t) \\ \boldsymbol{m}_{t+1} &= \mathit{Clip}_D(\boldsymbol{m}_t + \eta \boldsymbol{g}_t) \end{aligned}$$

We recovered a version of SGD with momentum and clipping! The only really different part is that we perturb the iterate a bit before calculating the gradient

From Online Betting to PAC-Bayes Bounds

a.k.a.

The existence of online algorithms implies generalization bounds

PAC-Bayes Bounds

Definitions:

$$\text{Risk}(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim P}[\ell(y, f_{\boldsymbol{\theta}}(x))] \quad \text{(True risk of } \boldsymbol{\theta}\text{)}$$

$$\text{Risk}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, f_{\boldsymbol{\theta}}(X_i)) \quad \text{(Empirical risk of } \boldsymbol{\theta}\text{)}$$

Assumption:
$0 \leq \ell \leq 1$

Theorem (McAllester, COLT'98)

Fix a prior distribution $\pi \in \mathcal{M}(\Theta)$. With probability at least $1-\delta$ on the data $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, for any probability distribution ρ learnt on the data,

$$\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\text{Risk}(\boldsymbol{\theta})] \leq \mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\text{Risk}_n(\boldsymbol{\theta})] + \sqrt{\frac{\text{KL}(\rho||\pi) + \ln \frac{2\sqrt{n}}{\delta}}{2n}}$$

Our Theorem

Theorem

Define optimal 'log-wealth' function:

$$\psi_n^\star(\boldsymbol{\theta}) := \max_{\lambda \in [-\frac{1}{1-\mathsf{Risk}(\boldsymbol{\theta})}, \frac{1}{\mathsf{Risk}(\boldsymbol{\theta})}]} \sum_{i=1}^{n} \ln[1 + \lambda(\ell(Y_i, f_{\boldsymbol{\theta}}(X_i)) - \mathsf{Risk}(\boldsymbol{\theta}))] \; .$$

*Fix $\pi \in \mathcal{M}(\Theta)$, then with probability at least $1 - \delta$, **simultaneously** for all n and ρ,*

$$\mathbb{E}_{\boldsymbol{\theta} \sim \rho}[\psi_n^\star(\boldsymbol{\theta})] \leq \mathsf{KL}(\rho \| \pi) + \ln \frac{\sqrt{n}}{\delta} \; .$$

[Jang et al., COLT'23]

Consequences

I. By relaxing the 'log-wealth' term this inequality implies:

- McAllester's inequality [McAllester, COLT'98]
- Empirical Bernstein's PAC-Bayes inequality [Tolstikhin&Seldin, NeurIPS'13]
- Maurer's inequality of Bernoulli r.v.'s [Maurer, arXiv'04]
- Unexpected Bernstein's inequality [Mhammedi et al., NeurIPS'19]

II. With no relaxations, we can compute confidence sequences on $\mu_{\boldsymbol{\theta}}$ **efficiently**

Examples of Relaxations

Our inequality:

$$\mathbb{E}_{\theta\sim\rho}[\psi_n^\star(\boldsymbol{\theta})] := \mathbb{E}_{\theta\sim\rho}\left[\max_{\lambda\in[-\frac{1}{1-\mathrm{Risk}(\boldsymbol{\theta})},\frac{1}{\mathrm{Risk}(\boldsymbol{\theta})}]} \sum_{i=1}^{n} \ln(1+\lambda(\ell(Y_i, f_{\boldsymbol{\theta}}(X_i)) - \mathrm{Risk}(\boldsymbol{\theta})))\right]$$
$$\leq \mathrm{KL}(\rho\|\pi) + \ln\frac{\sqrt{n}}{\delta}$$

- $\ln(1+x) \geq x - x^2$ for $x \geq -0.68$ gives

  $$|\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathrm{Risk}(\boldsymbol{\theta})] - \mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathrm{Risk}_n(\boldsymbol{\theta})]| \leq 2\sqrt{\frac{\mathrm{KL}(\rho\|\pi)+\ln\frac{\sqrt{n}}{\delta}}{n}} \Rightarrow \text{McAllester's bound!}$$

- By convexity, $\max_\lambda \sum_{i=1}^n \ln\left(1+\lambda(X_i-\mu)\right) \geq n\,\mathrm{kl}(\hat{\mu},\mu)$, that gives

  $$\mathrm{kl}\Big(\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathrm{Risk}_n(\boldsymbol{\theta})], \mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathrm{Risk}(\boldsymbol{\theta})]\Big) \leq \frac{\mathrm{KL}(\rho\|\pi)+\ln\frac{\sqrt{n}}{\delta}}{n} \Rightarrow \text{Maurer's bound!}$$

- Similarly, you can get the other bounds too

Let's Start from the Classic Proof

For any $\rho \ll \pi$ and measurable F:

$$\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[F(\boldsymbol{\theta})] \leq \mathsf{KL}(\rho\|\pi) + \ln \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[e^{F(\boldsymbol{\theta})}] \qquad \text{(Change-of-measure)}$$

For some fixed $\lambda > 0$ choose $F(\boldsymbol{\theta}) = \lambda(\mathsf{Risk}(\boldsymbol{\theta}) - \mathsf{Risk}_\mathsf{n}(\boldsymbol{\theta}))$. Then,

$$\begin{aligned}\lambda\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathsf{Risk}(\boldsymbol{\theta}) - \mathsf{Risk}_\mathsf{n}(\boldsymbol{\theta})] &\leq \mathsf{KL}(\rho\|\pi) + \ln \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[e^{\lambda(\mathsf{Risk}(\boldsymbol{\theta})-\mathsf{Risk}_\mathsf{n}(\boldsymbol{\theta}))}] \\ &\leq \mathsf{KL}(\rho\|\pi) + \ln\frac{1}{\delta} + \ln \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[\mathbb{E}e^{\lambda(\mathsf{Risk}(\boldsymbol{\theta})-\mathsf{Risk}_\mathsf{n}(\boldsymbol{\theta}))}] \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Markov)}\end{aligned}$$

Concentration!
E.g., Hoeffding's lemma + tuning over λ gives McAllester's inequality.

PAC-Bayes from a Betting Game

Standard approach: λ is fixed. **Idea**: tune λ based on data...

... using an online betting game:

- A fictitious betting algorithm starts with wealth 1
- At round $i = 1, \ldots, n$ it bets a signed fraction of its wealth $B_i(\boldsymbol{\theta})$
- Observes outcome $\Delta_i(\boldsymbol{\theta}) := \ell(Y_i, f_{\boldsymbol{\theta}}(X_i)) - \mathrm{Risk}(\boldsymbol{\theta})$
- Then its log wealth is $\psi_n(\boldsymbol{\theta}) := \sum_{i=1}^n \ln(1 + B_i(\boldsymbol{\theta})\Delta_i(\boldsymbol{\theta}))$
- The regret of the algorithm is controlled, as before:

$$\psi_n^\star(\boldsymbol{\theta}) - \psi_n(\boldsymbol{\theta}) \leq \ln \sqrt{n}, \quad \forall \boldsymbol{\theta}$$

where $\psi_n^\star(\boldsymbol{\theta})$ is the log-wealth of the optimal constant betting fraction:

$$\psi_n^\star(\boldsymbol{\theta}) = \max_{\lambda \in [-\frac{1}{1-\mathrm{Risk}(\boldsymbol{\theta})}, \frac{1}{\mathrm{Risk}(\boldsymbol{\theta})}]} \sum_{i=1}^n \ln[1 + \lambda(\ell(Y_i, f_{\boldsymbol{\theta}}(X_i)) - \mathrm{Risk}(\boldsymbol{\theta}))]$$

New Proof

For any $\rho \ll \pi$ and measurable F:

$$\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[F(\boldsymbol{\theta})] \leq \mathrm{KL}(\rho\|\pi) + \ln \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[e^{F(\boldsymbol{\theta})}] \qquad \text{(Change-of-measure)}$$

Choose $F(\boldsymbol{\theta}) = \psi_n^\star(\boldsymbol{\theta})$ (optimal log-wealth). Then,

$$\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\psi_n^\star(\boldsymbol{\theta})] \leq \mathrm{KL}(\rho\|\pi) + \ln \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[e^{\psi_n^\star(\boldsymbol{\theta})}]$$

$$e^{\psi_n^\star(\boldsymbol{\theta})} = \mathrm{OptimalWealth} \leq \mathrm{WealthAnyOnlineAlgorithmA}_n \cdot \exp(\mathit{Regret}_n(A))$$

Concentration: $\mathrm{WealthAnyOnlineAlgorithmA}_n$ is a non-negative martingale

$$\Pr\left\{\sup_{n\geq 0} \mathrm{WealthAnyOnlineAlgorithmA}_n \geq \frac{1}{\delta}\right\} \leq \delta \qquad \text{(Ville's inequality)}$$

Putting all together, with probability at least $1-\delta$, we have

$$\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\psi_n^\star(\boldsymbol{\theta})] \leq \mathrm{KL}(\rho\|\pi) + \ln\left(\frac{1}{\delta}\exp(\ln\sqrt{n})\right) = \mathrm{KL}(\rho\|\pi) + \ln\frac{1}{\delta} + \ln\sqrt{n}$$

Normalized Gradients for All

a.k.a.

You don’t lose anything in using the adversarial setting

Two Assumptions, Two Learning Rates

- Consider M-smooth functions, that is, $\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\|_\star \leq M\|\boldsymbol{x} - \boldsymbol{y}\|$
- It is well-known that gradient descent converges as $\mathcal{O}(1/T)$ on smooth functions, with learning rate $\eta = \frac{1}{M}$
- On the other hand, we only have $\mathcal{O}(1/\sqrt{T})$ on Lipschitz functions with learning rate $\eta \propto \frac{1}{\sqrt{T}}$
- Two different assumptions need two different learning rates

- Can I have a single algorithm that gives both rates, without anything to tune?
- Is there anything in between these two rates?

Hölder Smoothness

- Let's generalize smoothness: $\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\|_\star \leq M_\nu \|\boldsymbol{x} - \boldsymbol{y}\|^\nu$, for $\nu \in [0, 1]$
- $\nu = 0$ is for Lipschitz functions
- $\nu = 1$ is the usual smooth assumption

- For smooth function, we have $\|\nabla F(\boldsymbol{x})\|_\star^2 \leq 2M(F(\boldsymbol{x}) - F(\boldsymbol{x}^\star))$
- For Hölder-smooth functions, we have $\|\nabla F(\boldsymbol{x})\|_\star^{1+\frac{1}{\nu}} \leq \left(1 + \frac{1}{\nu}\right) M_\nu^{\frac{1}{\nu}} (F(\boldsymbol{x}) - F(\boldsymbol{x}^\star))$

Normalized Gradient for All

- Very simple idea: take your favourite first-order optimization algorithm
- Use normalized gradients instead of gradients

- If your algorithm satisfies a regret guarantee on Lipschitz losses, then it automatically adapts to all the Hölder-smooth functions!

Formal Guarantee

Theorem

Suppose that we have an algorithm that guarantees

$$\sum_{t=1}^{T} \langle \boldsymbol{q}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq R_T(\boldsymbol{x}^\star)$$

for any sequence of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_T$ *such that* $\|\boldsymbol{q}_t\|_\star \leq 1$. *Also, assume* F *is* M_ν *Hölder-smooth.*
Then, the same algorithm fed with $\boldsymbol{q}_t = \frac{\nabla F(\boldsymbol{x}_t)}{\|\nabla F(\boldsymbol{x}_t)\|_\star}$ *guarantees*

$$F(\bar{\boldsymbol{x}}_T) - F(\boldsymbol{x}^\star) \leq 2M_\nu \left(\frac{R_T(\boldsymbol{x}^\star)}{T} \right)^{1+\nu}$$

where $\bar{\boldsymbol{x}}_T = \frac{1}{\sum_{t=1}^T \frac{1}{\|\nabla F(\boldsymbol{x}_t)\|_\star}} \sum_{t=1}^T \frac{\boldsymbol{x}_t}{\|\nabla F(\boldsymbol{x}_t)\|_\star}$

[Levy, NeurIPS'17], more general result in [Orabona, arXiv'23]

Examples

- Consider Gradient Descent with $\eta = \frac{1}{\sqrt{T}}$, it satisfies

$$\sum_{t=1}^{T} \langle \boldsymbol{q}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \frac{\sqrt{T}}{2} \|\boldsymbol{x}_1 - \boldsymbol{x}^\star\|_2^2 + \frac{\sqrt{T}}{2}$$

- How do we know it? Because we proved it yesterday!
- So, Gradient Descent with normalized gradients satisfies

$$F(\bar{\boldsymbol{x}}_T) - F(\boldsymbol{x}^\star) \leq 2M_\nu \left(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}^\star\|_2^2 + 1}{2\sqrt{T}} \right)^{1+\nu}$$

- Known since Levy [NeurIPS'17] for $\nu \in \{0, 1\}$
- Strictly more general than Mishchenko and Malitsky [ICML'20] that works only for $\nu = 1$
- Rediscovered in DoGW [Khaled et al. NeurIPS'23] for $\nu \in \{0, 1\}$

Examples

- Consider the KT algorithm I mentioned before, it satisfies that

$$\sum_{t=1}^{T}\langle \boldsymbol{q}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle \leq \mathcal{O}(\|\boldsymbol{x}_1 - \boldsymbol{x}^\star\|_2\sqrt{T\ln(T\|\boldsymbol{x}_1 - \boldsymbol{x}^\star\|_2 + 1)})$$

- So, KT with normalized gradients satisfies

$$F(\bar{\boldsymbol{x}}_T) - F(\boldsymbol{x}^\star) \leq \mathcal{O}\left(M_\nu\left(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}^\star\|_2\sqrt{T\ln(T\|\boldsymbol{x}_1 - \boldsymbol{x}^\star\|_2 + 1)}}{\sqrt{T}}\right)^{1+\nu}\right)$$

- Known since Orabona&Pal [arXiv'21]
- Rediscovered in DoGW [Khaled et al. NeurIPS'23] for $\nu = 0$ and $\nu = 1$

Proof

$$\sum_{t=1}^{T} \frac{F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)}{\|\nabla F(\boldsymbol{x}_t)\|_\star} \overset{\text{Convexity}}{\leq} \sum_{t=1}^{T} \left\langle \frac{\nabla F(\boldsymbol{x}_t)}{\|\nabla F(\boldsymbol{x}_t)\|_\star}, \boldsymbol{x}_t - \boldsymbol{x}^\star \right\rangle \overset{\text{Assumption}}{\leq} R_T(\boldsymbol{x}^\star) \qquad (*)$$

Using $\bar{\boldsymbol{x}}_T = \frac{1}{\sum_{t=1}^{T} \frac{1}{\|\nabla F(\boldsymbol{x}_t)\|_\star}} \sum_{t=1}^{T} \frac{\boldsymbol{x}_t}{\|\nabla F(\boldsymbol{x}_t)\|_\star}$, we have

$$F(\bar{\boldsymbol{x}}_T) - F(\boldsymbol{x}^\star) \overset{(*)\text{ + Jensen}}{\leq} \frac{1}{\sum_{t=1}^{T} \frac{1}{\|\nabla F(\boldsymbol{x}_t)\|_\star}} R_T(\boldsymbol{x}^\star) \overset{\text{HM-GM}}{\leq} \frac{R_T(\boldsymbol{x}^\star)}{T} \left(\prod_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_\star \right)^{\frac{1}{T}}$$

$$= \frac{R_T(\boldsymbol{x}^\star)}{T} \left(\prod_{t=1}^{T} \frac{\|\nabla F(\boldsymbol{x}_t)\|_\star^{1+\frac{1}{\nu}}}{\|\nabla F(\boldsymbol{x}_t)\|_\star} \right)^{\frac{\nu}{T}}$$

$$\overset{\text{smoothness}}{\leq} \frac{R_T(\boldsymbol{x}^\star)}{T} \left(\prod_{t=1}^{T} \frac{\left(1+\frac{1}{\nu}\right) M_\nu^{\frac{1}{\nu}} \left(F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right)}{\|\nabla F(\boldsymbol{x}_t)\|_\star} \right)^{\frac{\nu}{T}}$$

$$\overset{\text{GM-AM}}{\leq} \frac{\left(1+\frac{1}{\nu}\right)^\nu M_\nu R_T(\boldsymbol{x}^\star)}{T} \left(\frac{1}{T} \frac{\sum_{t=1}^{T} (F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star))}{\|\nabla F(\boldsymbol{x}_t)\|_\star} \right)^{\nu} \overset{(*)}{\leq} 2M_\nu \left(\frac{R_T(\boldsymbol{x}^\star)}{T} \right)^{\nu+1}$$

Even More Surprising Applications

- Rademacher complexity bounds from Online Learning [Kakade et al., NeurIPS'08]
- Better-than-KL PAC-Bayes bounds [Kuzborskij et al., COLT'24]
- Parameter-free sampling [Sharrock&Nemeth, ICML'23][Sharrock et al. NeurIPS'23]

- Confidence intervals through betting (my talk in the workshop)

Summary

- Basic concepts and definitions of Online Learning
- OMD&FTRL
- Parameter-free algorithms
- Connection between regret and betting, PAC-Bayes bounds, and adaptation

Thank you!

Website: `https://francesco.orabona.com`

Blog: `https://parameterfree.com`

X/Twitter: @bremen79