Parameter-free Online Optimization
Part 2

Francesco Orabona   Ashok Cutkosky

ICML 2020

Outline of the Tutorial

- Part 1: Stochastic and Online Convex Optimization
- **Part 2: Parameter-free Convex Optimization**
- Part 3: More Adaptivity and Applications
- Part 4: Implementation, Experiments, Open Problems

Previously

- We saw why tuning learning rates is difficult
- In particular, the distance from to the optimum is unknown and it cannot be easily estimated
- We want to achieve $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\sqrt{\log(G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|\sqrt{T}+1)} + \frac{1}{T}\right)$

Previously

- We saw why tuning learning rates is difficult
- In particular, the distance from to the optimum is unknown and it cannot be easily estimated
- We want to achieve $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\sqrt{\log(G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|\sqrt{T}+1)} + \frac{1}{T}\right)$
- What next?

How Would Nemirovski Remove the Learning Rate? (1)

Idea: Try a grid of learning rates, but with a budget for each one, so that overall you spend the same time as using only 1 learning rate.

- Procedure $M(\alpha, T)$: Run GD for T iterations, with stepsizes $\eta_t = \frac{\alpha}{\sqrt{T}\|\nabla F(\boldsymbol{x}_t)\|}$, starting at the origin, and return $\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t$
- For $n = 1, 2, \ldots$ let us run one after one the procedures $M(\alpha = 1, T = 2^n), M(\alpha = 2, T = 2^n), M(\alpha = 4, T = 2^n), M(\alpha = 8, T = 2^n)$, ..., $M(\alpha = 2^n, T = 2^n)$ and return the one with the smallest value of F
- Recall that we need $\alpha \approx \|\boldsymbol{x}^\star\|$
- Set m be such that $2^{m-1} \leq \|\boldsymbol{x}^\star\| \leq 2^m$

[Nemirovski, Personal Communication 2013]

How Would Nemirovski Remove the Learning Rate? (2)

- When $n \geq m$, the n-th batch will produce a solution of accuracy $O(1)G\|\boldsymbol{x}^\star\|2^{-n/2}$ – this is what you will get when running $M(\alpha = 2^m, T = 2^n)$
- How long does it take?
- The total number of steps, from the beginning till executing n-th batch will be $T(n) = O(1)n2^n$
- So, when the total number of steps T satisfies $T \geq O(1)m2^m = O(1)\|\boldsymbol{x}^\star\| \ln(\|\boldsymbol{x}^\star\| + 1)$, the inaccuracy will be at most $O(1)\frac{\|\boldsymbol{x}^\star\|G\sqrt{\ln T}}{\sqrt{T}}$

[Nemirovski, Personal Communication 2013]

Not a Satisfying Solution...

- ✓ The construction tells us in principle there is an algorithm that finds the optimal learning rate
- × It does not work in the stochastic setting: one of the runs was good, but you don't know which one
- × You "waste" computation trying wrong learning rates
- × It does not work in practice

Second Alternative: Doubling Trick

Require: Initial Learning rate η_0

$x_1 = 0, Q_0 = 0$

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

  Output x_t

  Receive gradient $g_t = \nabla F(x_t)$

  $Q_t = Q_{t-1} - g_t x_t$

  if $Q_t \le \eta_{t-1} T$ **then**

    $\eta_t = \eta_{t-1}$

  else

    $x_t = 0$

    $\eta_t = 2\eta_{t-1}$

    $Q_t = 0$

  end if

  $x_{t+1} = x_t - \eta_t g_t$

end for

✓ It works in the stochastic and online setting

× The bound is suboptimal

× It does not really work (due to the resets)

[Streeter&McMahan, NeurIPS'12]

Third Alternative: Multiple Learning Rates in “Parallel”

- Instead of trying each single learning rate on its own, we might try all of them at the same time
- We run $N = O(\log T)$ SGD procedures, each with a different learning rate
- We feed the N algorithms with surrogate gradients and we merge their outputs at each step
- This approach was first proposed in the MetaGrad [van Erven&Koolen, NeurIPS’16] and Maler [Wang et al., UAI’19]
- Foster et al. [NeurIPS’17] proposed a similar approach that works for unbounded domains too

✓ General approach: you can use any base optimizer, not only SGD

× MetaGrad and Maler work only for bounded domains

× These approaches are a bit cumbersome: you do need to run all the algorithms at the same time

Fourth Alternative: FTRL/RDA Approach

- We would like the obtain the optimal convergence rate $O\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|\sqrt{\ln(\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|\sqrt{T})}}{\sqrt{T}} + \frac{1}{T}\right)$
- Equivalently, we would like the obtain the optimal regret $Regret_T(\boldsymbol{u}) \leq O\left(\|\boldsymbol{u}\|\sqrt{T\ \ln(\|\boldsymbol{u}\|\sqrt{T})} + 1\right)$
- So, we can use Follow-the-Regularized-Leader (aka Regularizered Dual Averaging) with time-varying regularizer $\psi_t(\boldsymbol{x}) \approx \|\boldsymbol{x}\|\sqrt{t\ \ln(\|\boldsymbol{x}\|\sqrt{t})}$

$$\boldsymbol{x}_t \approx \underset{\boldsymbol{x}}{argmin}\ \|\boldsymbol{x}\|\sqrt{t\ \ln(\|\boldsymbol{u}\|\sqrt{t})} + \sum_{i=1}^{t-1}\langle \boldsymbol{x}, \boldsymbol{g}_i\rangle$$

✓ It works!

× Very difficult to analyze and explain

[Streeter&McMahan, NeurIPS'12; McMahan&Abernethy, NeurIPS'13; Orabona, NeurIPS'13, NeurIPS'14; McMahan&Orabona, COLT'14; Foster et al., NeurIPS'15; Foster et al., COLT'18]

Last Alternative: Coin-betting Approach

This is the same as the FTRL approach, but:

- ✓ Easy
- ✓ Gives a new interpretation to optimization as gambling/compression/prediction with log loss
- ✓ Works very well :)
- × Very different from the usual optimization algorithms

[Orabona&Pal, NeurIPS'16; Cutkosky&Orabona, COLT'18]

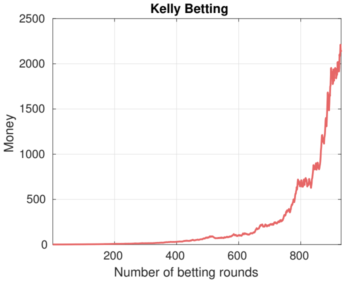
Betting on a Coin

- Start with \$1
- Bet x_t money on head ($x_t > 0$) or tails ($x_t < 0$)
  - Cannot borrow money
- Win or lose depending on the outcome of the coin $c_t \in \{-1, 1\}$
- $Wealth_t = Wealth_{t-1} + x_t c_t$

Aim: Maximize gain on all sequences where the number of tails and head are different

Optimal Betting Strategy for a Stochastic Coin: Kelly Betting (1956)

- Known problem in economics.
- Bet a fraction of your money equal to $2p - 1 = \mathbb{E}[c_t]$ on tail at each round.
  - E.g. $p = 0.51 \Rightarrow$ bet 2%
- Expected log wealth is linear in time.



Non-Stochastic setting, but Knowing the Future

- Non-stochastic setting, T rounds.
- Assume we bet a <u>fixed fraction</u> of money at each round.
- What is the optimal fraction?

[McMahan&Abernethy, NeurIPS'13; Orabona&Pal, NeurIPS'16]

Non-Stochastic setting, but Knowing the Future

- Non-stochastic setting, T rounds.
- Assume we bet a fixed fraction of money at each round.
- What is the optimal fraction?

- The optimal signed fraction at each time step t is $\frac{\sum_{i=1}^T c_i}{T}$.
- Hence you bet $\frac{\sum_{i=1}^T c_i}{T} \cdot Wealth_{t-1}$.
- Winnings are exponential:

$$\text{Winnings} > \exp\left(\frac{\left(\sum_{t=1}^T c_t\right)^2}{2T}\right)$$

[McMahan&Abernethy, NeurIPS'13; Orabona&Pal, NeurIPS'16]

Worst case Optimal Betting Strategy: Krichevsky-Trofimov Bettor

- What if we don't know the future?

[Orabona&Pal, NeurIPS'16]

Worst case Optimal Betting Strategy: Krichevsky-Trofimov Bettor

- What if we don't know the future?
- Estimate "probability" of heads with Krichevsky-Trofimov (KT) estimator: $\frac{\frac{1}{2} + \#\text{ of heads in } t \text{ rounds}}{t+1}$
  - No stochastic assumptions: KT has optimal regret w.r.t. the log loss
  - Not the only possible strategy, but the simplest one

[Orabona&Pal, NeurIPS'16]

Worst case Optimal Betting Strategy: Krichevsky-Trofimov Bettor

- What if we don't know the future?
- Estimate "probability" of heads with Krichevsky-Trofimov (KT) estimator: $\frac{\frac{1}{2} + \text{\# of heads in } t \text{ rounds}}{t+1}$
  - No stochastic assumptions: KT has optimal regret w.r.t. the log loss
  - Not the only possible strategy, but the simplest one
- Hence, on round t bet a signed fraction of your money equal to $\frac{\sum_{i=1}^{t-1} c_i}{t}$

[Orabona&Pal, NeurIPS'16]

Worst case Optimal Betting Strategy: Krichevsky-Trofimov Bettor

- What if we don't know the future?
- Estimate "probability" of heads with Krichevsky-Trofimov (KT) estimator: $\frac{\frac{1}{2}+\#\text{ of heads in } t \text{ rounds}}{t+1}$
  - No stochastic assumptions: KT has optimal regret w.r.t. the log loss
  - Not the only possible strategy, but the simplest one
- Hence, on round t bet a signed fraction of your money equal to $\frac{\sum_{i=1}^{t-1} c_i}{t}$
- Still exponential

$$\text{Winnings of KT Bettor } \geq \frac{\exp\left(\frac{\left(\sum_{t=1}^{T} c_t\right)^2}{2T}\right)}{2\sqrt{T}}$$

[Orabona&Pal, NeurIPS'16]

Worst case Optimal Betting Strategy: Krichevsky-Trofimov Bettor

- What if we don't know the future?
- Estimate "probability" of heads with Krichevsky-Trofimov (KT) estimator: $\frac{\frac{1}{2}+\text{\# of heads in } t \text{ rounds}}{t+1}$
  - No stochastic assumptions: KT has optimal regret w.r.t. the log loss
  - Not the only possible strategy, but the simplest one
- Hence, on round t bet a signed fraction of your money equal to $\frac{\sum_{i=1}^{t-1} c_i}{t}$
- Still exponential

$$\text{Winnings of KT Bettor} \geq \frac{\exp\left(\frac{(\sum_{t=1}^{T} c_t)^2}{2T}\right)}{2\sqrt{T}} = \frac{\text{Winnings knowing the future}}{2\sqrt{T}}$$

[Orabona&Pal, NeurIPS'16]

Worst case Optimal Betting Strategy: Krichevsky-Trofimov Bettor

- What if we don't know the future?
- Estimate "probability" of heads with Krichevsky-Trofimov (KT) estimator: $\frac{\frac{1}{2} + \text{\# of heads in } t \text{ rounds}}{t+1}$
  - No stochastic assumptions: KT has optimal regret w.r.t. the log loss
  - Not the only possible strategy, but the simplest one
- Hence, on round t bet a signed fraction of your money equal to $\frac{\sum_{i=1}^{t-1} c_i}{t}$
- Still exponential

$$\text{Winnings of KT Bettor} \geq \frac{\exp\left(\frac{(\sum_{t=1}^{T} c_t)^2}{2T}\right)}{2\sqrt{T}} = \frac{\text{Winnings knowing the future}}{2\sqrt{T}}$$

Coin-betting is solvable with a parameter-free optimal strategy, but what is the connection with SGD?

[Orabona&Pal, NeurIPS'16]

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^{T} c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^{T} c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^{T} c_t) = C \exp((\sum_{t=1}^{T} c_t)^2 / T) / \sqrt{T})$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^{T} c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^{T} c_t) = C \exp((\sum_{t=1}^{T} c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^{T} c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^{T} c_t) = C \exp((\sum_{t=1}^{T} c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$
- Let's set a betting game: bet x_t dollars on $c_t = -g_t = -\nabla F(x_t) \in \{-1, +1\}$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^T c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^T c_t) = C \exp((\sum_{t=1}^T c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$
- Let's set a betting game: bet x_t dollars on $c_t = -g_t = -\nabla F(x_t) \in \{-1, +1\}$
- Claim: the average of the bets will converge to the minimum of $F(x)$ at a rate that depends on how good is our betting strategy!

$$F\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - F(x^\star)$$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^{T} c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^{T} c_t) = C \exp((\sum_{t=1}^{T} c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$
- Let's set a betting game: bet x_t dollars on $c_t = -g_t = -\nabla F(x_t) \in \{-1, +1\}$
- Claim: the average of the bets will converge to the minimum of $F(x)$ at a rate that depends on how good is our betting strategy!

$$F\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - F(x^\star) \stackrel{\text{Jensen}}{\leq} \frac{1}{T}\sum_{t=1}^{T} F(x_t) - F(x^\star)$$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^{T} c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^{T} c_t) = C \exp((\sum_{t=1}^{T} c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$
- Let's set a betting game: bet x_t dollars on $c_t = -g_t = -\nabla F(x_t) \in \{-1, +1\}$
- Claim: the average of the bets will converge to the minimum of $F(x)$ at a rate that depends on how good is our betting strategy!

$$F\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - F(x^\star) \overset{\text{Jensen}}{\leq} \frac{1}{T}\sum_{t=1}^{T} F(x_t) - F(x^\star) \overset{\text{Convexity}}{\leq} \frac{1}{T}\left(\sum_{t=1}^{T} c_t x^\star - \sum_{t=1}^{T} c_t x_t\right)$$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^T c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^T c_t) = C \exp((\sum_{t=1}^T c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$
- Let's set a betting game: bet x_t dollars on $c_t = -g_t = -\nabla F(x_t) \in \{-1, +1\}$
- Claim: the average of the bets will converge to the minimum of $F(x)$ at a rate that depends on how good is our betting strategy!

$$F\left(\frac{1}{T}\sum_{t=1}^T x_t\right) - F(x^\star) \overset{\text{Jensen}}{\leq} \frac{1}{T}\sum_{t=1}^T F(x_t) - F(x^\star) \overset{\text{Convexity}}{\leq} \frac{1}{T}\left(\sum_{t=1}^T c_t x^\star - \sum_{t=1}^T c_t x_t\right)$$

$$\overset{\text{Def. } H}{\leq} \tfrac{1}{T}\left(\sum_{t=1}^T c_t x^\star - H\left(\sum_{t=1}^T c_t\right) + 1\right)$$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^T c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^T c_t) = C \exp((\sum_{t=1}^T c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$
- Let's set a betting game: bet x_t dollars on $c_t = -g_t = -\nabla F(x_t) \in \{-1, +1\}$
- Claim: the average of the bets will converge to the minimum of $F(x)$ at a rate that depends on how good is our betting strategy!

$$F\left(\frac{1}{T}\sum_{t=1}^T x_t\right) - F(x^\star) \overset{\text{Jensen}}{\leq} \frac{1}{T}\sum_{t=1}^T F(x_t) - F(x^\star) \overset{\text{Convexity}}{\leq} \frac{1}{T}\left(\sum_{t=1}^T c_t x^\star - \sum_{t=1}^T c_t x_t\right)$$

$$\overset{\text{Def. } H}{\leq} \tfrac{1}{T}\left(\sum_{t=1}^T c_t x^\star - H\left(\sum_{t=1}^T c_t\right) + 1\right) \overset{\text{Max}}{\leq} \tfrac{1}{T}\left(1 + \max_v \ v x^\star - H(v)\right)$$

1D Optimization through Betting

- Assume that there exists a function $H(\cdot)$ such that our betting strategy will guarantee that the wealth after T rounds will be at least $H(\sum_{t=1}^T c_t)$ for any arbitrary sequence $c_1, \cdots, c_T$
  - For KT $H(\sum_{t=1}^T c_t) = C \exp((\sum_{t=1}^T c_t)^2/T)/\sqrt{T})$
- We want to minimize $F(x) = |x - 10|$
- Let's set a betting game: bet x_t dollars on $c_t = -g_t = -\nabla F(x_t) \in \{-1, +1\}$
- Claim: the average of the bets will converge to the minimum of $F(x)$ at a rate that depends on how good is our betting strategy!

$$F\left(\frac{1}{T}\sum_{t=1}^T x_t\right) - F(x^\star) \overset{\text{Jensen}}{\leq} \frac{1}{T}\sum_{t=1}^T F(x_t) - F(x^\star) \overset{\text{Convexity}}{\leq} \frac{1}{T}\left(\sum_{t=1}^T c_t x^\star - \sum_{t=1}^T c_t x_t\right)$$

$$\overset{\text{Def. } H}{\leq} \tfrac{1}{T}\left(\sum_{t=1}^T c_t x^\star - H\left(\sum_{t=1}^T c_t\right) + 1\right) \overset{\text{Max}}{\leq} \tfrac{1}{T}\left(1 + \max_v \ v x^\star - H(v)\right) \overset{\text{Def } H^*}{=} \frac{H^*(x^\star)+1}{T}$$

KT as an Optimization Algorithm

- Assume $\nabla F(x_t) \in \{-1, +1\}$
- $x_t = \frac{\sum_{i=1}^{t-1} c_i}{t} Wealth_{t-1} = \frac{-\sum_{i=1}^{t-1} g_i}{t}(1 - \sum_{i=1}^{t-1} g_i \cdot x_i)$

Theorem (Informal)

KT betting in 1-d guarantees

$$F\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - F(x^\star) \leq C\frac{|x^\star|\sqrt{\log(T)}}{\sqrt{T}}$$

[Orabona&Pal, NeurIPS'16]

KT as an Optimization Algorithm

- Assume the function $F(\boldsymbol{x})$ is 1-Lipschitz ($G = 1$)
- $\boldsymbol{x}_t = \frac{\sum_{i=1}^{t-1} \boldsymbol{c}_i}{t} Wealth_{t-1} = \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t}(1 - \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x}_i \rangle)$

Theorem (Informal)

KT betting in Hilbert spaces guarantees

$$F\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star) \leq C\frac{\|\boldsymbol{x}^\star\|\sqrt{\log(T)}}{\sqrt{T}}$$

Proof idea:

- "Worst" direction for gradient at time t is parallel to $\sum_{i=1}^{t-1} \boldsymbol{g}_i$
- "Worst" gradient have $\|\boldsymbol{g}_t\| = 1$

[Orabona&Pal, NeurIPS'16]

KT as an Optimization Algorithm

- Assume the function $F(\boldsymbol{x})$ is 1-Lipschitz ($G = 1$)
- $\boldsymbol{x}_t = \frac{\sum_{i=1}^{t-1} \boldsymbol{c}_i}{t} \mathit{Wealth}_{t-1} = \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t}(1 - \sum_{i=1}^{t-1}\langle \boldsymbol{g}_i, \boldsymbol{x}_i\rangle)$

Theorem (Informal)

KT betting in Hilbert spaces guarantees

$$F\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star) \leq \min_{\alpha} C \frac{\left(\frac{\|\boldsymbol{x}^\star\|^2}{\alpha} + \alpha\right)\sqrt{\log(T)}}{\sqrt{T}}$$

KT as an Optimization Algorithm

- Assume the function $F(\boldsymbol{x})$ is 1-Lipschitz ($G = 1$)
- $\boldsymbol{x}_t = \frac{\sum_{i=1}^{t-1} \boldsymbol{c}_i}{t} \textit{Wealth}_{t-1} = \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t} (1 - \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x}_i \rangle)$

Theorem (Informal)

KT betting in Hilbert spaces guarantees

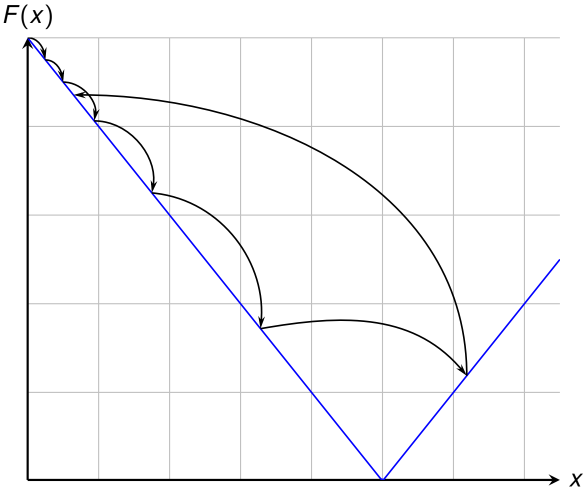
$$F\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star) \leq \min_{\alpha} C \frac{\left(\frac{\|\boldsymbol{x}^\star\|^2}{\alpha} + \alpha\right)\sqrt{\log(T)}}{\sqrt{T}}$$

- Compare it to the Gradient Descent guarantee with learning rate $\frac{\alpha}{\sqrt{t}}$

$$F\left(\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star) \leq C' \frac{\frac{\|\boldsymbol{x}^\star\|^2}{\alpha} + \alpha}{\sqrt{T}}$$

[Orabona&Pal, NeurIPS'16]

How does the Betting Approach Work?



Constant Betting Case

- We said the optimal learning rate is $\eta_t = \frac{\|\boldsymbol{x}^\star\|}{\sqrt{t}}$, when $\boldsymbol{x}_1 = \mathbf{0}$
- Can we approximate it with $\eta_t = \frac{\|\boldsymbol{x}_t\|}{\sqrt{T}}$ and use $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\|\boldsymbol{x}_t\|}{\sqrt{T}}\boldsymbol{g}_t$?
  [You et al., arXiv'17; You et al., ICPP'18; Bernstein et al., arXiv'20]

⚠ Such update rule would require $\boldsymbol{x}_1 \neq \mathbf{0}$

Constant Betting Case

- We said the optimal learning rate is $\eta_t = \frac{\|\boldsymbol{x}^\star\|}{\sqrt{t}}$, when $\boldsymbol{x}_1 = \mathbf{0}$
- Can we approximate it with $\eta_t = \frac{\|\boldsymbol{x}_t\|}{\sqrt{T}}$ and use $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\|\boldsymbol{x}_t\|}{\sqrt{T}} \boldsymbol{g}_t$?
  [You et al., arXiv'17; You et al., ICPP'18; Bernstein et al., arXiv'20]

⚠ Such update rule would require $\boldsymbol{x}_1 \neq \mathbf{0}$

- Use a betting view: constant betting fraction of $\frac{1}{\sqrt{T}}$, i.e., $x_t = \frac{1}{\sqrt{T}} Wealth_{t-1}$

⚠ x_t is always positive

- $Wealth_t = Wealth_{t-1} + x_t c_t = Wealth_{t-1}(1 - \frac{1}{\sqrt{T}} g_t)$
- $x_{t+1} = \frac{1}{\sqrt{T}} Wealth_t = \frac{1}{\sqrt{T}} Wealth_{t-1}(1 - \frac{1}{\sqrt{T}} g_t) = x_t(1 - \frac{1}{\sqrt{T}} g_t) = x_t - \frac{x_t}{\sqrt{T}} g_t$

Constant Betting Case

- We said the optimal learning rate is $\eta_t = \frac{\|\boldsymbol{x}^\star\|}{\sqrt{t}}$, when $\boldsymbol{x}_1 = \mathbf{0}$
- Can we approximate it with $\eta_t = \frac{\|\boldsymbol{x}_t\|}{\sqrt{T}}$ and use $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\|\boldsymbol{x}_t\|}{\sqrt{T}} \boldsymbol{g}_t$?
  [You et al., arXiv'17; You et al., ICPP'18; Bernstein et al., arXiv'20]

⚠ Such update rule would require $\boldsymbol{x}_1 \neq \mathbf{0}$

- Use a betting view: constant betting fraction of $\frac{1}{\sqrt{T}}$, i.e., $x_t = \frac{1}{\sqrt{T}} Wealth_{t-1}$

⚠ x_t is always positive

- $Wealth_t = Wealth_{t-1} + x_t c_t = Wealth_{t-1}(1 - \frac{1}{\sqrt{T}} g_t)$
- $x_{t+1} = \frac{1}{\sqrt{T}} Wealth_t = \frac{1}{\sqrt{T}} Wealth_{t-1}(1 - \frac{1}{\sqrt{T}} g_t) = x_t(1 - \frac{1}{\sqrt{T}} g_t) = x_t - \frac{x_t}{\sqrt{T}} g_t$

- Wealth lower bound: $Wealth_T = \prod_{t=1}^T (1 - \frac{1}{\sqrt{T}} g_t) \geq \exp(-\frac{1}{\sqrt{T}} \sum_{t=1}^T g_t - 1)$
- We get almost the optimal rate: $O(\frac{x^\star}{\sqrt{T}} \ln(x^\star \sqrt{T}))$ with $x^\star \geq 0$

[No citation: this is new]

How to Go from 1d to $\mathbb{R}^d$?

We have proposed 3 possible solutions:

- Under certain technical conditions, the algorithm can be transformed simply using inner products instead of multiplications:
  $\boldsymbol{x}_t = \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t} \left(1 - \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x}_i \rangle\right)$.

[Streeter&McMahan, NeurIPS'12; Orabona&Pal, NeurIPS'16; Cutkosky&Orabona, COLT'18]

How to Go from 1d to $\mathbb{R}^d$?

We have proposed 3 possible solutions:

- Under certain technical conditions, the algorithm can be transformed simply using inner products instead of multiplications:
  $\boldsymbol{x}_t = \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t}\left(1 - \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x}_i \rangle\right)$.

- Just using a coordinate-wise reduction: $x_{t,j} = \frac{-\sum_{i=1}^{t-1} g_{i,j}}{t}\left(1 - \sum_{i=1}^{t-1} g_{i,j} x_{i,j}\right)$.

[Streeter&McMahan, NeurIPS'12; Orabona&Pal, NeurIPS'16; Cutkosky&Orabona, COLT'18]

How to Go from 1d to $\mathbb{R}^d$?

We have propose 3 possible solutions:

- Under certain technical conditions, the algorithm can be transformed simply using inner products instead of multiplications: $\boldsymbol{x}_t = \frac{-\sum_{i=1}^{t-1} \boldsymbol{g}_i}{t}\left(1 - \sum_{i=1}^{t-1} \langle \boldsymbol{g}_i, \boldsymbol{x}_i \rangle\right)$.
- Just using a coordinate-wise reduction: $x_{t,j} = \frac{-\sum_{i=1}^{t-1} g_{i,j}}{t}\left(1 - \sum_{i=1}^{t-1} g_{i,j} x_{i,j}\right)$.
- Optimize the magnitude and direction of $\boldsymbol{x}_t$ independently

[Streeter&McMahan, NeurIPS'12; Orabona&Pal, NeurIPS'16; Cutkosky&Orabona, COLT'18]

Frustratingly Easy Magnitude and Direction Decomposition

Require: 1d algorithm $\mathcal{A}_{1D}$, unit ball $S \subset R^d$ algorithm $\mathcal{A}_S$

1: **for** $t = 1$ **to** T **do**
2: Get point $z_t \in \mathbb{R}$ from $\mathcal{A}_{1D}$
3: Get point $\boldsymbol{y}_t \in S$ from $\mathcal{A}_S$
4: Play $\boldsymbol{x}_t = z_t \boldsymbol{y}_t \in R^d$
5: Receive gradient $\boldsymbol{g}_t$
6: Send $\boldsymbol{g}_t$ as the t-th gradient to $\mathcal{A}_S$
7: Send $\langle \boldsymbol{g}_t, \boldsymbol{y}_t \rangle$ as the t-th gradient to $\mathcal{A}_{1D}$
8: **end for**

$$
\begin{aligned}
Regret_T(\boldsymbol{x}^\star) &= \sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle = \sum_{t=1}^T \langle \boldsymbol{g}_t, z_t \boldsymbol{y}_t \rangle - \langle \boldsymbol{g}_t, \boldsymbol{x}^\star \rangle \\
&= \sum_{t=1}^T \underbrace{(\langle \boldsymbol{g}_t, \boldsymbol{y}_t \rangle z_t - \langle \boldsymbol{g}_t, \boldsymbol{y}_t \rangle \|\boldsymbol{x}^\star\|)}_{\text{regret of } \mathcal{A}_{1D} \text{ at } \|\boldsymbol{x}^\star\| \in \mathbb{R}} + \sum_{t=1}^T \|\boldsymbol{x}^\star\| \underbrace{\left(\langle \boldsymbol{g}_t, \boldsymbol{y}_t \rangle - \left\langle \boldsymbol{g}_t, \frac{\boldsymbol{x}^\star}{\|\boldsymbol{x}^\star\|} \right\rangle \right)}_{\text{regret of } \mathcal{A}_S \text{ at } \frac{\boldsymbol{x}^\star}{\|\boldsymbol{x}^\star\|} \in S} \\
&= Regret_T^{\mathcal{A}_{1D}}(\|\boldsymbol{x}^\star\|) + \|\boldsymbol{x}^\star\| Regret_T^{\mathcal{A}_S}\left(\frac{\boldsymbol{x}^\star}{\|\boldsymbol{x}^\star\|} \right)
\end{aligned}
$$

[Cutkosky&Orabona,COLT'18]

Summary of Part 2

1. Removing learning rates is indeed possible
2. You can make an exponential amount of money betting on a non-stochastic coin with a parameter-free algorithm
3. You can reduce optimization to betting on a coin
4. Hence, you can use a betting algorithm as a parameter-free optimization algorithm