# Parameter-free Online Optimization
## Part 1

Francesco Orabona    Ashok Cutkosky

ICML 2020

# Motivation

- Most of the algorithms we know/use in machine learning have parameters

# Motivation

- Most of the algorithms we know/use in machine learning have parameters
  - E.g. regularization parameter in LASSO, $k$ in $k$-Nearest Neighbourhood, topology of the networks in deep learning

# Motivation

- Most of the algorithms we know/use in machine learning have parameters
  - E.g. regularization parameter in LASSO, $k$ in $k$-Nearest Neighbourhood, topology of the networks in deep learning
- Most of the time, a large enough validation set can be used to tune the parameters

# Motivation

- Most of the algorithms we know/use in machine learning have parameters
  - E.g. regularization parameter in LASSO, $k$ in $k$-Nearest Neighbourhood, topology of the networks in deep learning
- Most of the time, a large enough validation set can be used to tune the parameters
- But, at what computational price? Are they really necessary?

# Motivation

- Most of the algorithms we know/use in machine learning have parameters
  - E.g. regularization parameter in LASSO, $k$ in $k$-Nearest Neighbourhood, topology of the networks in deep learning
- Most of the time, a large enough validation set can be used to tune the parameters
- But, at what computational price? Are they really necessary?
- Are you ignoring some computational/sample complexities?

# Motivation

- Most of the algorithms we know/use in machine learning have parameters
  - E.g. regularization parameter in LASSO, $k$ in $k$-Nearest Neighbourhood, topology of the networks in deep learning
- Most of the time, a large enough validation set can be used to tune the parameters
- But, at what computational price? Are they really necessary?
- Are you ignoring some computational/sample complexities?

- When is the last time you needed to tune the learning rate to invert a matrix?

# How ML Should Be

- No parameters to tune
- No humans in the loop
- Some guarantees

# How ML Should Be

- No parameters to tune
- No humans in the loop
- Some guarantees

- Sometimes this is possible
- This tutorial is about a method to achieve it in many interesting cases

# Learning Rates are Unavoidable?

# Learning Rates are Unavoidable?

- Let's consider an example
- You have 11 real numbers $y_1, \ldots y_{11}$
- We want to solve $\min_x \sum_{i=1}^{11} |x - y_i|$
- You might use SGD
- It will converge, but you have to set a learning rate...

# Learning Rates are Unavoidable?

- Let's consider an example
- You have 11 real numbers $y_1, \ldots y_{11}$
- We want to solve $\min_x \sum_{i=1}^{11} |x - y_i|$
- You might use SGD
- It will converge, but you have to set a learning rate...

- Alternative: you just need the median, so sort the numbers and pick the one in the middle...

# 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

# Learning Objectives

- Explain the problems behind learning rates
- Intuition behind recent parameter-free approaches
- Stripped-down algorithms and proofs
- Trick&tips for practical implementations

Overall, democratize these new methods and expand the number of people working in this field

# 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

# Stochastic Optimization of Convex Functions

- We want to solve the problem

$$\min_{\boldsymbol{x} \in X} \; F(\boldsymbol{x}),$$

  where $F$ is a convex function

- We will also assume $F$ to be $G$-Lipschitz, i.e., $|F(\boldsymbol{x}) - F(\boldsymbol{y})| \leq G\|\boldsymbol{x} - \boldsymbol{y}\|$
- For the most part of this tutorial, we will not assume much more than this

# Stochastic Gradients

- We have access to an first-order stochastic oracle
- It gives us noisy estimate $\boldsymbol{g}_t$ of the gradient of $F$ in a point $\boldsymbol{x}_t$
- We will assume unbiasedness: $\mathbb{E}[\boldsymbol{g}_t | \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{t-1}] = \nabla F(\boldsymbol{x}_t)$

⚠ Most of the things we say work also for non-diffentiable functions using subgradients

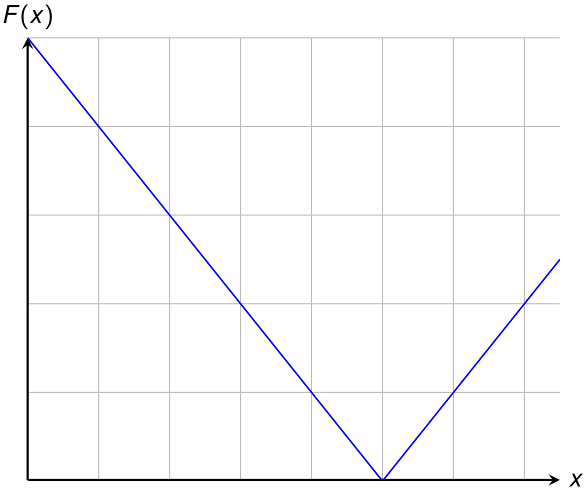
# The Most Used Algorithm in ML: SGD

- The simplest algorithm that we can use in our setting is Stochastic Gradient Descent (SGD)
- Iteratively it moves in the negative direction of the gradient:

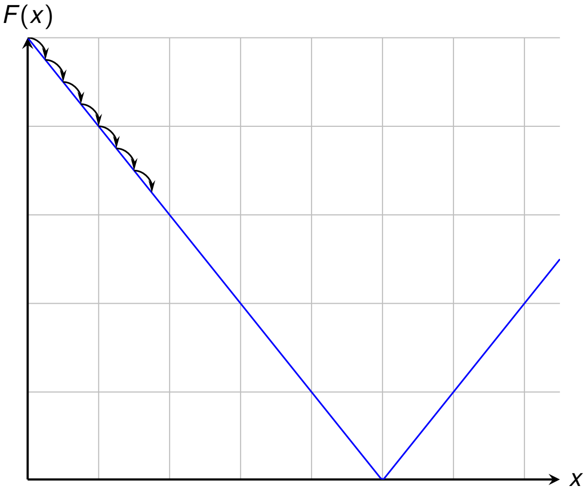
$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \boldsymbol{g}_t$$

- $\eta_t$ is the learning rate or step size
- Choosing the correct learning rate is a black art...
- Why is it so difficult?

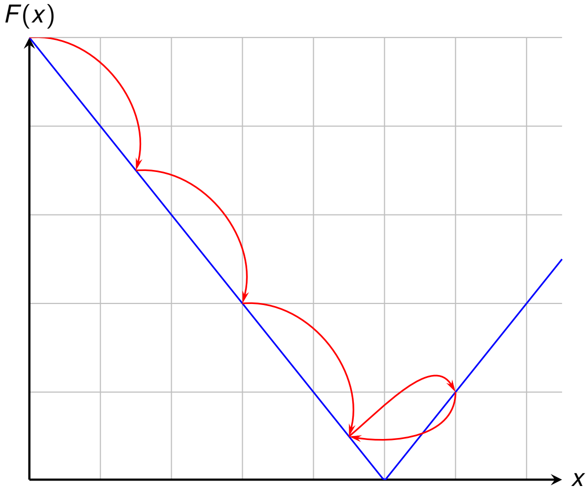
# Minimize $F(x) = |x - 10|$



# First Problem: The Learning Rate Sets Your Maximum Precision

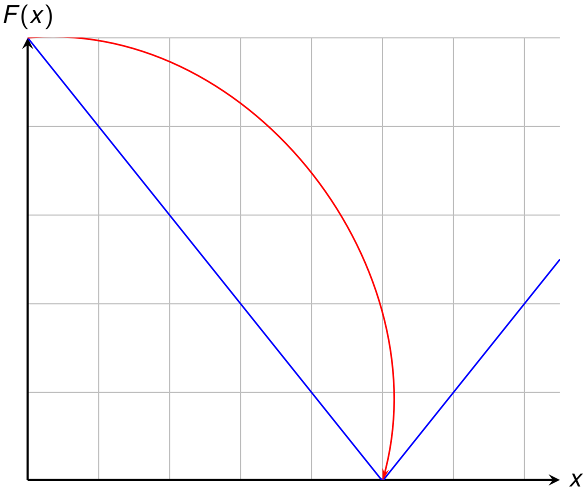


# First Problem: The Learning Rate Sets Your Maximum Precision




⚠ If you want to converge at least in the limit, learning rates cannot be fixed

# One Jump?

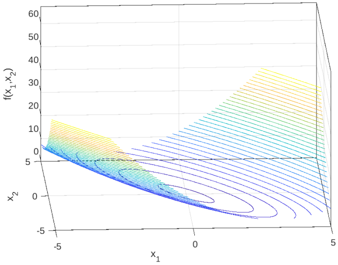


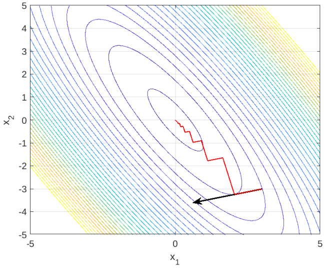
# No One Jump!

- First, in the stochastic setting, line search is not possible because you do not have the exact gradient nor can you exactly compute the function value
  - You need strong assumptions (and parameters...) to make stochastic line search work [Paquette&Scheinberg, arXiv'18; Vaswani et al., NeurIPS'19]

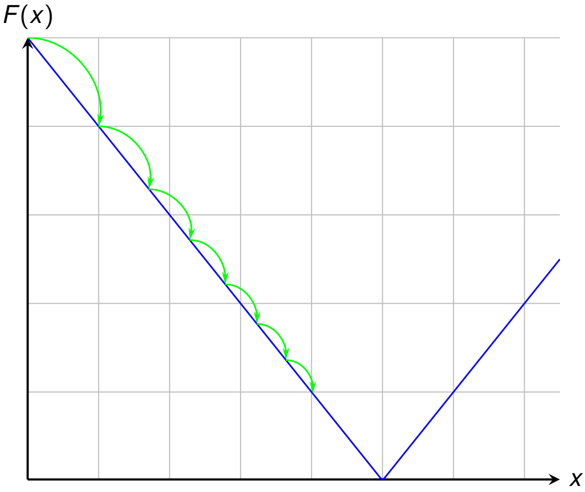
# No One Jump!

- First, in the stochastic setting, line search is not possible because you do not have the exact gradient nor can you exactly compute the function value
  - You need strong assumptions (and parameters...) to make stochastic line search work [Paquette&Scheinberg, arXiv'18; Vaswani et al., NeurIPS'19]
- Even in 2d, the gradients does not necessarily point towards the minimum



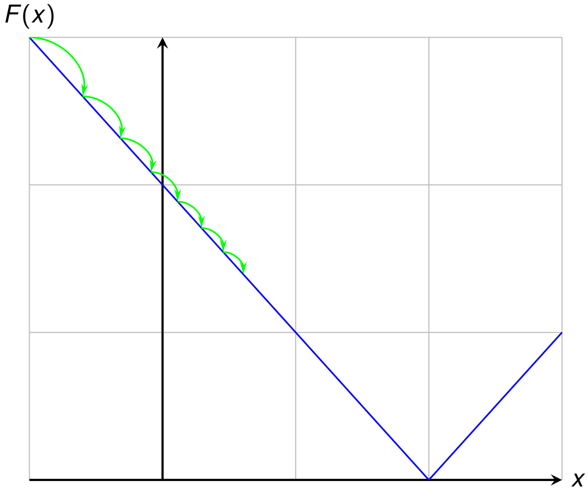



# Solution: Decreasing Learning Rates



# Second Problem: Distance to the Optimum

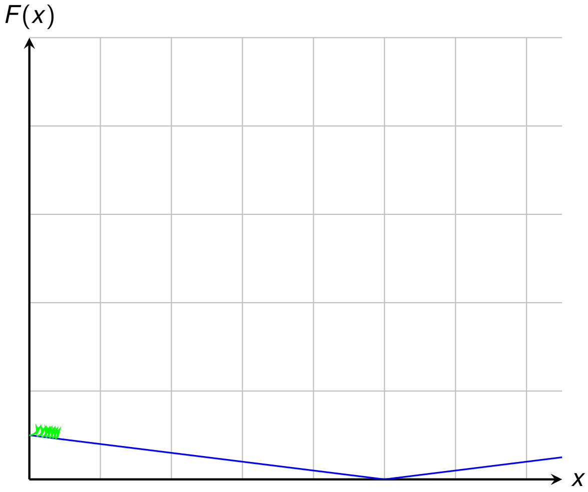
Starting from -5, our choice of the learning rate is not good anymore!




⚠ Optimal learning rate depends on distance from the optimum

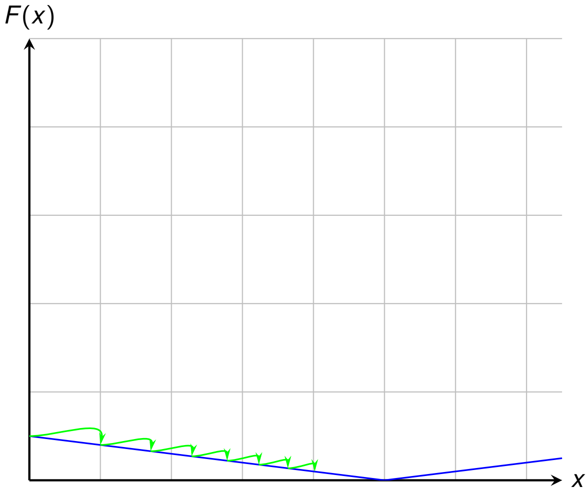
# Third Problem: Scale

Changing the function to $\color{red}{0.1}|x - 10|$ again ruins the convergence




⚠ Optimal learning rate depends on the scale (=Lipschitz constant)

# What About “Adaptive” Algorithms?




⚠ AdaGrad adapts to the scale of <u>each</u> coordinate, but does not adapt to distance from the optimum!

# Take Home Messages on Learning Rates

Learning rates...

1. ...must decrease
2. ...depend on the distance to the optimal solution
3. ...depend on the scale

# The Theory Confirms our Intuition

- Use a decreasing learning rate, $\eta_t = \frac{\alpha}{\sqrt{t}}$
- Convergence after $T$ iterations is $\mathcal{O}\left(\frac{1}{\sqrt{T}}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|^2}{\alpha} + \alpha G^2\right)\right)$
- $\boldsymbol{x}^\star$ is the best solution
- $G$ the bound on the norm of the stochastic gradients

# The Theory Confirms our Intuition

- Use a decreasing learning rate, $\eta_t = \frac{\alpha}{\sqrt{t}}$
- Convergence after $T$ iterations is $\mathcal{O}\left(\frac{1}{\sqrt{T}}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|^2}{\alpha} + \alpha G^2\right)\right)$
- $\boldsymbol{x}^\star$ is the best solution
- $G$ the bound on the norm of the stochastic gradients

- The optimal learning rate is with $\alpha = \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{G}$ that would give a rate of $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\right)$

# The Theory Confirms our Intuition

- Use a decreasing learning rate, $\eta_t = \frac{\alpha}{\sqrt{t}}$
- Convergence after $T$ iterations is $\mathcal{O}\left(\frac{1}{\sqrt{T}}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|^2}{\alpha} + \alpha G^2\right)\right)$
- $\boldsymbol{x}^\star$ is the best solution
- $G$ the bound on the norm of the stochastic gradients

- The optimal learning rate is with $\alpha = \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{G}$ that would give a rate of $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\right)$
- ...but you don't know $\boldsymbol{x}^\star$...

# The Theory Confirms our Intuition

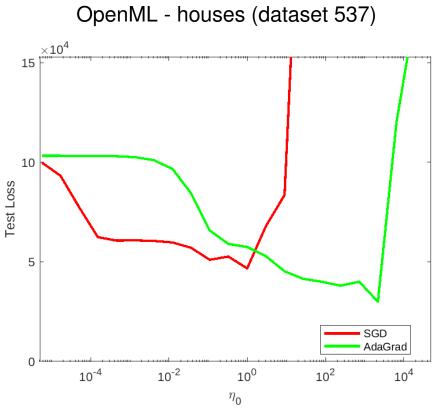
- Use a decreasing learning rate, $\eta_t = \frac{\alpha}{\sqrt{t}}$
- Convergence after $T$ iterations is $\mathcal{O}\left(\frac{1}{\sqrt{T}}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|^2}{\alpha} + \alpha G^2\right)\right)$
- $\boldsymbol{x}^\star$ is the best solution
- $G$ the bound on the norm of the stochastic gradients

- The optimal learning rate is with $\alpha = \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{G}$ that would give a rate of $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\right)$
- ...but you don't know $\boldsymbol{x}^\star$...

- Only if the domain is bounded, you can (loosely) upper bound $\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|$
  - Not-a-solution solution: Assume your domain is bounded or bounded iterates

# I Hate Learning Rates

- Finding the right one is critical!
- The only existing solution: tries different learning rates



# Lower Bounds

## Theorem

- $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\right)$ *cannot be achieved, unless we know* $\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|$

[Streeter and McMahan, NeurIPS'12; Orabona, NeurIPS'13; Cutkosky&Boahen, COLT'17]

# Lower Bounds

## Theorem

- $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\right)$ *cannot be achieved, unless we know* $\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|$
- *we can achieve* $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\sqrt{\log(G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|\sqrt{T}+1)} + \frac{1}{T}\right)$

[Streeter and McMahan, NeurIPS'12; Orabona, NeurIPS'13; Cutkosky&Boahen, COLT'17]

# Lower Bounds

## Theorem

- $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\right)$ *cannot be achieved, unless we know* $\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|$
- *we can achieve* $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\sqrt{\log(G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|\sqrt{T}+1)} + \frac{1}{T}\right)$
- *or* $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\sqrt{\log(G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|+1)} + \frac{1}{\sqrt{T}}\right)$

[Streeter and McMahan, NeurIPS'12; Orabona, NeurIPS'13; Cutkosky&Boahen, COLT'17]

# Lower Bounds

## Theorem

- $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\right)$ *cannot be achieved, unless we know* $\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|$
- *we can achieve* $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\sqrt{\log(G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|\sqrt{T}+1)} + \frac{1}{T}\right)$
- *or* $\mathcal{O}\left(\frac{G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{\sqrt{T}}\sqrt{\log(G\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|+1)} + \frac{1}{\sqrt{T}}\right)$

- The logarithmic term is the price you have to pay because you don't know how far is the optimal solution
- Think of it as the complexity of binary searching for the optimal learning rate

[Streeter and McMahan, NeurIPS'12; Orabona, NeurIPS'13; Cutkosky&Boahen, COLT'17]

# Our Tools

- Before solving this problem, we need some tools
- We will use Online Convex Optimization algorithms

- Online Optimization $\neq$ Stochastic Optimization
- Online Optimization is actually more general than Stochastic Optimization!

# From Online Learning to Stochastic Optimization

Online Learning

1. In each round, output $\boldsymbol{x}_t \in X$
2. Pay $\ell_t(\boldsymbol{x}_t)$
3. Receive $\boldsymbol{g}_t = \nabla \ell_t(\boldsymbol{x}_t)$
4. Update $\boldsymbol{x}_{t+1}$

[Cesa-Bianchi&Lugosi, 2006; Shalev-Shwartz, 2012; Hazan, 2016; Orabona, arXiv'19]

# From Online Learning to Stochastic Optimization

Online Learning

1. In each round, output $\boldsymbol{x}_t \in X$ — Choose $\boldsymbol{x}_t$ before observing $\ell_t$
2. Pay $\ell_t(\boldsymbol{x}_t)$
3. Receive $\boldsymbol{g}_t = \nabla \ell_t(\boldsymbol{x}_t)$
4. Update $\boldsymbol{x}_{t+1}$

[Cesa-Bianchi&Lugosi, 2006; Shalev-Shwartz, 2012; Hazan, 2016; Orabona, arXiv'19]

# From Online Learning to Stochastic Optimization

Online Learning

1. In each round, output $\boldsymbol{x}_t \in X$
2. Pay $\ell_t(\boldsymbol{x}_t)$
3. Receive $\boldsymbol{g}_t = \nabla \ell_t(\boldsymbol{x}_t)$
4. Update $\boldsymbol{x}_{t+1}$

Choose $\boldsymbol{x}_t$ before observing $\ell_t$

No assumptions on how $\ell_t$ is generated!

[Cesa-Bianchi&Lugosi, 2006; Shalev-Shwartz, 2012; Hazan, 2016; Orabona, arXiv'19]

# From Online Learning to Stochastic Optimization

Online Learning

1. In each round, output $\boldsymbol{x}_t \in X$
2. Pay $\ell_t(\boldsymbol{x}_t)$
3. Receive $\boldsymbol{g}_t = \nabla \ell_t(\boldsymbol{x}_t)$
4. Update $\boldsymbol{x}_{t+1}$

Choose $\boldsymbol{x}_t$ before observing $\ell_t$

No assumptions on how $\ell_t$ is generated!

Regret minimization

$$\min_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_T \in X} \sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) \qquad \text{equivalently} \qquad \min_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_T \in X} \underbrace{\sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{x}^\star)}_{Regret_T(\boldsymbol{x}^\star)}$$

- Typically, we can design Online algorithms where the $Regret_T(\boldsymbol{x}^\star) = O(\sqrt{T})$
- Harder setting, easier analysis, same results :)

[Cesa-Bianchi&Lugosi, 2006; Shalev-Shwartz, 2012; Hazan, 2016; Orabona, arXiv'19]

# From Online to Stochastic (or Batch) Optimization

**for** $t = 1$ **to** $T$ **do**
  Get $\boldsymbol{x}_t$ from Online Learning Algorithm
  Receive stochastic gradient $\boldsymbol{g}_t$ such that $\mathbb{E}_t[\boldsymbol{g}_t] = \nabla F(\boldsymbol{x}_t)$
  Pass loss $\ell_t(\boldsymbol{x}) = \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle$ to Online Learning Algorithm
**end for**
**return** $\bar{\boldsymbol{x}}_T = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{x}_t$

## Theorem

$$\mathbb{E}[F(\bar{\boldsymbol{x}}_T)] - F(\boldsymbol{x}^\star) \leq \frac{\mathbb{E}[Regret_T(\boldsymbol{x}^\star)]}{T}$$

In words: If you have an online algorithm able to work just with linear losses, you have a stochastic optimization algorithm for any convex function!

[Cesa-Bianchi et al., IEEE Trans. Inf. Theory 2004]

## Some Famous Online Learning Algorithms

- Online Gradient Descent [Zinkevich, ICML'03]
- AdaGrad [Duchi et al., COLT'10, JMRL'11]
- AMSGrad [Reddi et al., ICLR'18]

These algorithms are designed to work in the adversarial setting and have a $O(\sqrt{T})$ regret bound

Hence, they can also be used as stochastic optimization algorithms with a $O(\frac{1}{\sqrt{T}})$ convergence rate

## Summary of Part 1

1. Learning rates are difficult to set because they depends on quantities you don't know nor you can easily estimate
2. In particular, the additional logarithmic dependency on the distance to the optimal solution is unavoidable
3. Online Optimization gives us a set of powerful tools to attack this problem