Parameter-free Online Optimization
Part 3

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

Previously

- We saw that stochastic optimization can be solved via online linear optimization:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \mathbb{E}\left[\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle\right]$$

$$\mathbb{E}[\boldsymbol{g}_t] = \nabla F(\boldsymbol{x}_t)$$

Previously

- We saw that stochastic optimization can be solved via online linear optimization:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \mathbb{E}\left[\sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle\right]$$

$$\mathbb{E}[\boldsymbol{g}_t] = \nabla F(\boldsymbol{x}_t)$$

- The optimally tuned gradient descent guarantees error $\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\| G}{\sqrt{T}}$ using learning rate $\eta = \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{G\sqrt{T}}$.

Previously

- We saw that stochastic optimization can be solved via online linear optimization:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \mathbb{E}\left[\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle\right]$$
$$\mathbb{E}[\boldsymbol{g}_t] = \nabla F(\boldsymbol{x}_t)$$

- The optimally tuned gradient descent guarantees error $\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\| G}{\sqrt{T}}$ using learning rate $\eta = \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{G\sqrt{T}}$.
- A parameter-free algorithm can obtain $\tilde{O}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\| G}{\sqrt{T}}\right)$. The logarithmic dependencies are tight.

Previously

- We saw that stochastic optimization can be solved via online linear optimization:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \mathbb{E}\left[\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle\right]$$

$$\mathbb{E}[\boldsymbol{g}_t] = \nabla F(\boldsymbol{x}_t)$$

- The optimally tuned gradient descent guarantees error $\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\| G}{\sqrt{T}}$ using learning rate $\eta = \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\|}{G\sqrt{T}}$.
- A parameter-free algorithm can obtain $\tilde{O}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_1\| G}{\sqrt{T}}\right)$. The logarithmic dependencies are tight.
- What next?

In This Session:

Building algorithms that go beyond worst-case tuning.

In This Session:

Building algorithms that go beyond worst-case tuning.

1. How to build an “adaptive” parameter-free algorithm.

In This Session:

Building algorithms that go beyond worst-case tuning.

1. How to build an “adaptive” parameter-free algorithm.
2. Automatically improving when $\boldsymbol{x}^\star$ is sparse.

In This Session:

Building algorithms that go beyond worst-case tuning.

1. How to build an “adaptive” parameter-free algorithm.
2. Automatically improving when $\boldsymbol{x}^\star$ is sparse.
3. Automatically improving when $F(\boldsymbol{x})$ is smooth.

In This Session:

Building algorithms that go beyond worst-case tuning.

1. How to build an "adaptive" parameter-free algorithm.
2. Automatically improving when $\boldsymbol{x}^\star$ is sparse.
3. Automatically improving when $F(\boldsymbol{x})$ is smooth.
4. Automatically improving when $F(\boldsymbol{x})$ is strongly-convex.

In This Session:

Building algorithms that go beyond worst-case tuning.

1. How to build an “adaptive” parameter-free algorithm.
2. Automatically improving when $\boldsymbol{x}^\star$ is sparse.
3. Automatically improving when $F(\boldsymbol{x})$ is smooth.
4. Automatically improving when $F(\boldsymbol{x})$ is strongly-convex.
5. Dependencies on Lipschitz constants G.

In This Session:

Building algorithms that go beyond worst-case tuning.

1. How to build an "adaptive" parameter-free algorithm.
2. Automatically improving when $\boldsymbol{x}^\star$ is sparse.
3. Automatically improving when $F(\boldsymbol{x})$ is smooth.
4. Automatically improving when $F(\boldsymbol{x})$ is strongly-convex.
5. Dependencies on Lipschitz constants G.
   - We will assume $\boldsymbol{x}_1 = 0$ in this section to avoid writing $\boldsymbol{x}^\star - \boldsymbol{x}_1$ too much.

Second-Order Regret Bound

We will build an online linear optimization algorithm that obtains:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \le \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

This is the “AdaGrad version” of the KT-estimator regret bound $\tilde{O}\left(\|\boldsymbol{x}^\star\| G\sqrt{T} \right)$.

[Cutkosky&Orabona, COLT’18; Kempka et al., ICML’19; Van der Hoeven, NeurIPS’19; Mhammedi&Koolen, COLT’20]

Second-Order Regret Bound

We will build an online linear optimization algorithm that obtains:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

This is the “AdaGrad version” of the KT-estimator regret bound $\tilde{O}\left(\|\boldsymbol{x}^\star\| G\sqrt{T}\right)$.

⚠ This is called “second order” because of the $\|\boldsymbol{g}_t\|^2$ - it does NOT make any use of the Hessian.

[Cutkosky&Orabona, COLT’18; Kempka et al., ICML’19; Van der Hoeven, NeurIPS’19; Mhammedi&Koolen, COLT’20]

Second-Order Regret Bound

We will build an online linear optimization algorithm that obtains:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

This is the "AdaGrad version" of the KT-estimator regret bound $\tilde{O}\left(\|\boldsymbol{x}^\star\| G\sqrt{T}\right)$.

⚠ This is called "second order" because of the $\|\boldsymbol{g}_t\|^2$ - it does NOT make any use of the Hessian.

Let us see how to obtain this bound. For simplicity, we will assume $G = 1$.

[Cutkosky&Orabona, COLT'18; Kempka et al., ICML'19; Van der Hoeven, NeurIPS'19; Mhammedi&Koolen, COLT'20]

Obtaining the Second-Order Bound (in 1d)

Recall the basic coin-betting strategy:

Obtaining the Second-Order Bound (in 1d)

Recall the basic coin-betting strategy:

1. Think of the gradient g_t as specifying (continuous) outcomes of “coin flips”. At each iteration, we earn money $-g_t x_t$.

Obtaining the Second-Order Bound (in 1d)

Recall the basic coin-betting strategy:

1. Think of the gradient g_t as specifying (continuous) outcomes of "coin flips". At each iteration, we earn money $-g_t x_t$.
2. Keep track of the wealth: $\text{Wealth}_t = 1 - \sum_{i=1}^{t} g_i x_i$.

Obtaining the Second-Order Bound (in 1d)

Recall the basic coin-betting strategy:

1. Think of the gradient g_t as specifying (continuous) outcomes of "coin flips". At each iteration, we earn money $-g_t x_t$.
2. Keep track of the wealth: $\text{Wealth}_t = 1 - \sum_{i=1}^{t} g_i x_i$.
3. The output x_t corresponds to "betting" some money on the value of g_t.

Obtaining the Second-Order Bound (in 1d)

Recall the basic coin-betting strategy:

1. Think of the gradient g_t as specifying (continuous) outcomes of "coin flips". At each iteration, we earn money $-g_t x_t$.
2. Keep track of the wealth: $\text{Wealth}_t = 1 - \sum_{i=1}^{t} g_i x_i$.
3. The output x_t corresponds to "betting" some money on the value of g_t.
4. If we can make $\text{Wealth}_T \geq H(\sum_{t=1}^{T} g_t)$ for any function H, then regret is bounded:

$$\sum_{t=1}^{T} g_t(x_t - x^\star) \leq 1 + H^\star(x^\star)$$

where $H^\star$ is the Fenchel conjugate of H.

Obtaining the Second-Order Bound (in 1d)

Recall the basic coin-betting strategy:

1. Think of the gradient g_t as specifying (continuous) outcomes of “coin flips”. At each iteration, we earn money $-g_t x_t$.
2. Keep track of the wealth: $\text{Wealth}_t = 1 - \sum_{i=1}^{t} g_i x_i$.
3. The output x_t corresponds to “betting” some money on the value of g_t.
4. If we can make $\text{Wealth}_T \geq H(\sum_{t=1}^{T} g_t)$ for any function H, then regret is bounded:

$$\sum_{t=1}^{T} g_t(x_t - x^\star) \leq 1 + H^\star(x^\star)$$

   where $H^\star$ is the Fenchel conjugate of H.

- You are not allowed to borrow money: $x_t = \beta_t \text{Wealth}_{t-1}$, where $\beta_t \in [-1, 1]$ is called the “betting fraction”.

Obtaining the Second-Order Bound (in 1d)

Recall the basic coin-betting strategy:

1. Think of the gradient g_t as specifying (continuous) outcomes of "coin flips". At each iteration, we earn money $-g_t x_t$.
2. Keep track of the wealth: $\text{Wealth}_t = 1 - \sum_{i=1}^t g_i x_i$.
3. The output x_t corresponds to "betting" some money on the value of g_t.
4. If we can make $\text{Wealth}_T \geq H(\sum_{t=1}^T g_t)$ for any function H, then regret is bounded:

$$\sum_{t=1}^{T} g_t(x_t - x^\star) \leq 1 + H^\star(x^\star)$$

where $H^\star$ is the Fenchel conjugate of H.

- You are not allowed to borrow money: $x_t = \beta_t \text{Wealth}_{t-1}$, where $\beta_t \in [-1, 1]$ is called the "betting fraction".
- This is actually no restriction: any algorithm that guarantees $Regret_T(0) = O(1)$ must be a coin-betting algorithm.

Obtaining the Second-Order Bound (in 1d)

Strategy:

1. Compute a near-optimal fixed $\beta^\star$ with foresight (similar to optimal Kelly bet).

Obtaining the Second-Order Bound (in 1d)

Strategy:

1. Compute a near-optimal fixed $\beta^\star$ with foresight (similar to optimal Kelly bet).
2. Observe that this fixed $\beta^\star$ yields the second-order bound.

Obtaining the Second-Order Bound (in 1d)

Strategy:

1. Compute a near-optimal fixed $\beta^\star$ with foresight (similar to optimal Kelly bet).
2. Observe that this fixed $\beta^\star$ yields the second-order bound.
3. Find a way to generate β_t on-the-fly that does nearly as well as $\beta^\star$.

Calculating the Wealth

Since $x_t = \beta_t \text{Wealth}_{t-1}$, we have a recursion for Wealth_T

Calculating the Wealth

Since $x_t = \beta_t \text{Wealth}_{t-1}$, we have a recursion for Wealth_T

$$\text{Wealth}_T = 1 - \sum_{t=1}^{T} g_t x_t = \text{Wealth}_{T-1} - g_T x_T$$

Calculating the Wealth

Since $x_t = \beta_t \text{Wealth}_{t-1}$, we have a recursion for Wealth_T

$$\begin{aligned}\text{Wealth}_T &= 1 - \sum_{t=1}^{T} g_t x_t = \text{Wealth}_{T-1} - g_T x_T \\ &= \text{Wealth}_{T-1}(1 - \beta_T g_T) = \prod_{t=1}^{T}(1 - \beta_t g_t)\end{aligned}$$

Calculating the Wealth

Since $x_t = \beta_t \text{Wealth}_{t-1}$, we have a recursion for Wealth_T

$$\begin{aligned}\text{Wealth}_T &= 1 - \sum_{t=1}^{T} g_t x_t = \text{Wealth}_{T-1} - g_T x_T \\ &= \text{Wealth}_{T-1}(1 - \beta_T g_T) = \prod_{t=1}^{T}(1 - \beta_t g_t)\end{aligned}$$

Whenever you see a product, it's natural to take the logarithm:

$$\log(\text{Wealth}_T) = \sum_{t=1}^{T} \log(1 - \beta_t g_t)$$

A Good Fixed Betting Fraction

Let us guess $\beta^\star = -\frac{\sum_{t=1}^T g_t}{\sum_{t=1}^T g_t^2 + 2\left|\sum_{t=1}^T g_t\right|}$. This yields $\beta^\star \in [-1/2, 1/2]$, and so

A Good Fixed Betting Fraction

Let us guess $\beta^\star = -\frac{\sum_{t=1}^T g_t}{\sum_{t=1}^T g_t^2 + 2\left|\sum_{t=1}^T g_t\right|}$. This yields $\beta^\star \in [-1/2, 1/2]$, and so

$$\log(\text{Wealth}_T) = \sum_{t=1}^T \log(1 - \beta^\star g_t)$$

apply a Taylor expansion of $\log$ using $\beta^\star g_t \in [-1/2, 1/2]$.

A Good Fixed Betting Fraction

Let us guess $\beta^\star = -\frac{\sum_{t=1}^T g_t}{\sum_{t=1}^T g_t^2 + 2\left|\sum_{t=1}^T g_t\right|}$. This yields $\beta^\star \in [-1/2, 1/2]$, and so

$$\log(\text{Wealth}_T) = \sum_{t=1}^T \log(1 - \beta^\star g_t)$$

apply a Taylor expansion of $\log$ using $\beta^\star g_t \in [-1/2, 1/2]$.

$$\geq \sum_{t=1}^T \left[-\beta^\star g_t - (\beta^\star g_t)^2\right] \geq \frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}$$

A Good Fixed Betting Fraction

Let us guess $\beta^\star = -\frac{\sum_{t=1}^T g_t}{\sum_{t=1}^T g_t^2 + 2\left|\sum_{t=1}^T g_t\right|}$. This yields $\beta^\star \in [-1/2, 1/2]$, and so

$$\log(\text{Wealth}_T) = \sum_{t=1}^T \log(1 - \beta^\star g_t)$$

apply a Taylor expansion of $\log$ using $\beta^\star g_t \in [-1/2, 1/2]$.

$$\geq \sum_{t=1}^T \left[-\beta^\star g_t - (\beta^\star g_t)^2\right] \geq \frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}$$

$$\text{Wealth}_T \geq \exp\left[\frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}\right]$$

A Good Fixed Betting Fraction

Let us guess $\beta^\star = -\frac{\sum_{t=1}^T g_t}{\sum_{t=1}^T g_t^2 + 2\left|\sum_{t=1}^T g_t\right|}$. This yields $\beta^\star \in [-1/2, 1/2]$, and so

$$\log(\text{Wealth}_T) = \sum_{t=1}^T \log(1 - \beta^\star g_t)$$

apply a Taylor expansion of $\log$ using $\beta^\star g_t \in [-1/2, 1/2]$.

$$\geq \sum_{t=1}^T \left[-\beta^\star g_t - (\beta^\star g_t)^2\right] \geq \frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}$$

$$\text{Wealth}_T \geq \exp\left[\frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}\right] := H\left(\sum_{t=1}^T g_t\right)$$

A Good Fixed Betting Fraction

$$\text{Wealth}_T \geq \exp\left[\frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}\right] := H\left(\sum_{t=1}^T g_t\right)$$

A Good Fixed Betting Fraction

$$\text{Wealth}_T \geq \exp\left[\frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}\right] := H\left(\sum_{t=1}^T g_t\right)$$

$$\sum_{t=1}^T g_t(x_t - x^\star) \leq 1 + H^\star(x^\star) = \tilde{O}\left(|x^\star|\sqrt{\sum_{t=1}^T g_t^2}\right)$$

The Story So Far

1. There is a fixed betting fraction $\beta^\star$ that would guarantee a second-order regret bound.
2. How can we learn about $\beta^\star$ on-the-fly?

Online Learning Inside Online Learning

Compute the gap between the log wealth we obtain vs the log wealth of $\beta^\star$:

Online Learning Inside Online Learning

Compute the gap between the log wealth we obtain vs the log wealth of $\beta^\star$:

$$\log(\text{Wealth}^\star_T) - \log(\text{Wealth}_T) = \sum_{t=1}^{T} \log(1 - \beta^\star g_t) - \sum_{t=1}^{T} \log(1 - \beta_t g_t)$$

[Cutkosky&Orabona, COLT'18]

Online Learning Inside Online Learning

Compute the gap between the log wealth we obtain vs the log wealth of $\beta^\star$:

$$\log(\text{Wealth}^\star_T) - \log(\text{Wealth}_T) = \sum_{t=1}^{T} \log(1 - \beta^\star g_t) - \sum_{t=1}^{T} \log(1 - \beta_t g_t)$$

Define $\ell_t(\beta) = -\log(1 - \beta g_t)$:

$$:= \sum_{t=1}^{T} \ell_t(\beta_t) - \sum_{t=1}^{T} \ell_t(\beta^\star)$$

[Cutkosky&Orabona, COLT'18]

Online Learning Inside Online Learning

Compute the gap between the log wealth we obtain vs the log wealth of $\beta^\star$:

$$\log(\text{Wealth}^\star_T) - \log(\text{Wealth}_T) = \sum_{t=1}^{T} \log(1 - \beta^\star g_t) - \sum_{t=1}^{T} \log(1 - \beta_t g_t)$$

Define $\ell_t(\beta) = -\log(1 - \beta g_t)$:

$$:= \sum_{t=1}^{T} \ell_t(\beta_t) - \sum_{t=1}^{T} \ell_t(\beta^\star)$$

ℓ_t is exp-concave! This means that choosing β_t by gradient descent on ℓ_t (e.g. via Online Newton Step [Hazan et al., MLJ'07]) yields:

$$\leq \log(T)$$

$$\text{Wealth}_T \geq \frac{\text{Wealth}^\star_T}{T}$$

[Cutkosky&Orabona, COLT'18]

Wrapping Up Second Order Bound

Now we have

$$\begin{aligned}\text{Wealth}_T &\geq \frac{\text{Wealth}_T^\star}{T} \\ &\geq \frac{1}{T}\exp\left[\frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}\right] := H\left(\sum_{t=1}^T g_t\right)\end{aligned}$$

$$\sum_{t=1}^T g_t(x_t - x^\star) \leq 1 + H^\star(x^\star) = \tilde{O}\left(|x^\star|\sqrt{\sum_{t=1}^T g_t^2}\right)$$

Wrapping Up Second Order Bound

Now we have

$$\begin{aligned}\text{Wealth}_T &\geq \frac{\text{Wealth}^\star_T}{T} \\ &\geq \frac{1}{T}\exp\left[\frac{\left(\sum_{t=1}^T g_t\right)^2}{2\sum_{t=1}^T g_t^2 + 4\left|\sum_{t=1}^T g_t\right|}\right] := H\left(\sum_{t=1}^T g_t\right)\end{aligned}$$

$$\sum_{t=1}^T g_t(x_t - x^\star) \leq 1 + H^\star(x^\star) = \tilde{O}\left(|x^\star|\sqrt{\sum_{t=1}^T g_t^2}\right)$$

The $\frac{1}{T}$ only changes a $\log(1+|x^\star|\sqrt{T})$ to $\log(1+|x^\star|T)$.

[Cutkosky&Orabona, COLT'18]

Full Second-Order Bound Algorithm

$\beta_1 = 0, \ \text{Wealth}_0 = 1$
for $t = 1$ **to** T **do**
  Play $x_t = \beta_t \text{Wealth}_{t-1}$
  Get gradient g_t, define $\ell_t(\beta) = -\log(1 - \beta g_t)$
  Compute $z_t = \ell_t'(\beta_t) = \frac{g_t}{1-\beta g_t}$
  Set $\beta_{t+1} = \text{clip}\left(\beta_t - \frac{z_t}{1+\sum_{i=1}^t z_i^2}, -\frac{1}{2}, \frac{1}{2}\right)$
  This is the ONS update
  Set $\text{Wealth}_t = \text{Wealth}_{t-1} - g_t x_t$
end for

Theorem

The above algorithm guarantees

$$\sum_{t=1}^{T} g_t(x_t - x^\star) \leq \tilde{O}\left(|x^\star| \sqrt{\sum_{t=1}^{T} g_t^2 \log(|x^\star| T)}\right)$$

Applications

Now we are armed with an second-order 1-d regret bound. We'll show how it can be easily used to accomplish some interesting tasks:

Applications

Now we are armed with an second-order 1-d regret bound. We'll show how it can be easily used to accomplish some interesting tasks:

1. A bound that depends on the sparsity of $\boldsymbol{x}^\star$.

Applications

Now we are armed with an second-order 1-d regret bound. We'll show how it can be easily used to accomplish some interesting tasks:

1. A bound that depends on the sparsity of $\boldsymbol{x}^\star$.
2. A bound that automatically adapts to the smoothness and variance parameters.

Applications

Now we are armed with an second-order 1-d regret bound. We'll show how it can be easily used to accomplish some interesting tasks:

1. A bound that depends on the sparsity of $\boldsymbol{x}^\star$.
2. A bound that automatically adapts to the smoothness and variance parameters.
3. A bound that automatically adapts to strong-convexity parameters.

Sparsity via Per-Coordinate Updates

Remember one way to make a d-dimensional algorithm from a 1-dimensional algorithm, ala AdaGrad:

$$\sum_{t=1}^{T} \langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle = \sum_{i=1}^{d} \underbrace{\sum_{t=1}^{T} g_{t,i}(x_{t,i} - x_i^\star)}_{\text{Set } \boldsymbol{x}_{t,i} \text{ via a 1-d learner.}}$$

$$\leq \underbrace{\tilde{O}\left(\sum_{i=1}^{d} |x_i^\star| \sqrt{\sum_{t=1}^{T} g_{t,i}^2}\right)}_{\text{The } \tilde{O} \text{ now hides a } \log(d) \text{ factor.}}$$

Sparsity via Per-Coordinate Updates

Remember one way to make a d-dimensional algorithm from a 1-dimensional algorithm, ala AdaGrad:

$$\sum_{t=1}^{T}\langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle = \sum_{i=1}^{d}\underbrace{\sum_{t=1}^{T} g_{t,i}(x_{t,i} - x_i^\star)}_{\text{Set } \boldsymbol{x}_{t,i} \text{ via a 1-d learner.}}$$

$$\leq \underbrace{\tilde{O}\left(\sum_{i=1}^{d}|x_i^\star|\sqrt{\sum_{t=1}^{T} g_{t,i}^2}\right)}_{\text{The } \tilde{O} \text{ now hides a } \log(d) \text{ factor.}}$$

Contrast with the (optimally tuned) AdaGrad guarantee:

$$\sum_{t=1}^{T}\langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle \leq O\left(\max_i |x_i^\star| \sum_{i=1}^{d}\sqrt{\sum_{t=1}^{T} g_{t,i}^2}\right)$$

Sparsity via Per-Coordinate Updates

What happens if some coordinates $x_i^\star$ are 0?

- Suppose $\|\boldsymbol{x}^\star\|_\infty \leq 1$, $\|\boldsymbol{g}_t\|_\infty \leq 1$, and $\boldsymbol{x}^\star$ is S-sparse. Then:

Sparsity via Per-Coordinate Updates

What happens if some coordinates $x_i^\star$ are 0?

- Suppose $\|\boldsymbol{x}^\star\|_\infty \le 1$, $\|\boldsymbol{g}_t\|_\infty \le 1$, and $\boldsymbol{x}^\star$ is S-sparse. Then:

$$\begin{aligned}
\sum_{t=1}^{T} \langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle &\le \tilde{O}\left(\sum_{i=1}^{d} |x_i^\star| \sqrt{\sum_{t=1}^{T} g_{t,i}^2} \right) \\
&\le \tilde{O}\left(\|\boldsymbol{x}^\star\|_1 \sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|_\infty^2} \right) \\
&\le S\sqrt{T}
\end{aligned}$$

More Properties of the Per-Coordinate Update

Per-Coordinate updates also achieve the L_2-norm bound:

$$\sum_{t=1}^{T}\langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle \le \tilde{O}\left(\sum_{i=1}^{d}|x_i^\star|\sqrt{\sum_{t=1}^{T} g_{t,i}^2}\right)$$

$$\text{apply Cauchy-Schwarz} \le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2}\right)$$

Both of these properties leverage the automatic adaptivity to each individual $x_i^\star$.

More Properties of the Per-Coordinate Update

Per-Coordinate updates also achieve the L_2-norm bound:

$$\sum_{t=1}^{T} \langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\sum_{i=1}^{d} |x_i^\star| \sqrt{\sum_{t=1}^{T} g_{t,i}^2} \right)$$

$$\text{apply Cauchy-Schwarz} \leq \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

Both of these properties leverage the automatic adaptivity to each individual $x_i^\star$.

⚠ This is not the same as identifying which coordinates of $\boldsymbol{x}^\star$ are zero.

[Kotłowski, ALT'17]

Parameter-Free Experts Algorithms

- Learning with expert advice setting is a special case of online linear optimization in which both $\boldsymbol{x}_t$ and $\boldsymbol{x}^\star$ must lie in the probability simplex in $\mathbb{R}^d$.
- There are many parameter-free algorithms for this special case, starting with NormalHedge [Chaudhuri et al., NeurIPS'09], and including [Gaillard et al., COLT'14; Koolen&van Erven, COLT'15; Luo&Schapire, COLT'15; Foster et al., NeurIPS'15; Harvey et al., arXiv'20].
- In this special setting, one can hope for interesting different bounds. For example, the Squint algorithm can obtain:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\sqrt{\sum_{t=1}^{T} \sum_{i=1}^{d} |x_i^\star| (\langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle - g_{t,i})^2} \right)$$

Parameter-Free Experts Algorithms

- Learning with expert advice setting is a special case of online linear optimization in which both $\boldsymbol{x}_t$ and $\boldsymbol{x}^\star$ must lie in the probability simplex in $\mathbb{R}^d$.
- There are many parameter-free algorithms for this special case, starting with NormalHedge [Chaudhuri et al., NeurIPS'09], and including [Gaillard et al., COLT'14; Koolen&van Erven, COLT'15; Luo&Schapire, COLT'15; Foster et al., NeurIPS'15; Harvey et al., arXiv'20].
- In this special setting, one can hope for interesting different bounds. For example, the Squint algorithm can obtain:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\sqrt{\sum_{t=1}^{T} \sum_{i=1}^{d} |x_i^\star| (\langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle - g_{t,i})^2} \right)$$

which implies:

$$\sum_{t=1}^{T} \langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\sum_{i=1}^{d} |x_i^\star| \sqrt{\sum_{t=1}^{T} |g_{t,i}|} \right)$$

Parameter-Free Experts Algorithms

- Learning with expert advice setting is a special case of online linear optimization in which both $\boldsymbol{x}_t$ and $\boldsymbol{x}^\star$ must lie in the probability simplex in $\mathbb{R}^d$.
- There are many parameter-free algorithms for this special case, starting with NormalHedge [Chaudhuri et al., NeurIPS'09], and including [Gaillard et al., COLT'14; Koolen&van Erven, COLT'15; Luo&Schapire, COLT'15; Foster et al., NeurIPS'15; Harvey et al., arXiv'20].
- In this special setting, one can hope for interesting different bounds. For example, the Squint algorithm can obtain:

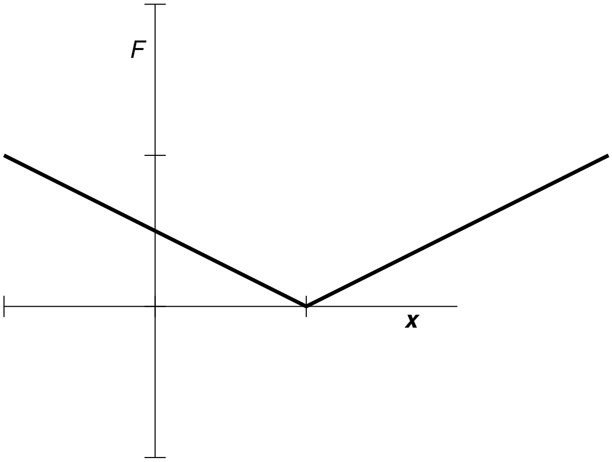
$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\sqrt{\sum_{t=1}^{T} \sum_{i=1}^{d} |x_i^\star| (\langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle - g_{t,i})^2} \right)$$

which implies:

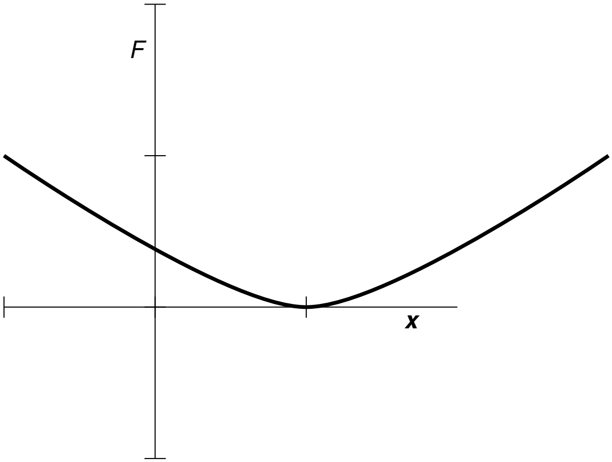
$$\sum_{t=1}^{T} \langle g_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\sum_{i=1}^{d} |x_i^\star| \sqrt{\sum_{t=1}^{T} |g_{t,i}|} \right)$$

- This does not have the $g_{t,i}^2$, but works in the simplex.

Adapting to Smoothness



Adapting to Smoothness



Adapting to Smoothness

Suppose our original objective $F(\boldsymbol{x})$ is L-smooth:

$$\nabla^2 F(\boldsymbol{x}) \preceq LI$$

We don't know what L is. Suppose also that for all t,

$$\mathrm{Var}[\boldsymbol{g}_t] \leq \sigma^2$$

In this case, the optimal learning rate for gradient descent depends on L and σ, and achieves:

$$\mathbb{E}[F(\boldsymbol{x}_{\mathrm{SGD}}) - F(\boldsymbol{x}^\star)] \leq \frac{L\|\boldsymbol{x}^\star\|^2}{T} + \frac{\sigma\|\boldsymbol{x}^\star\|}{\sqrt{T}}$$

[Ghadimi&Lan, J. of Opt.'12]

Adapting to Smoothness

- The second-order parameter-free bound automatically obtains this result up to a log factor!
- To see how, we need to look inside the online-to-batch conversion, and look beyond the linear approximation to F.

Adapting to Smoothness

- The second-order parameter-free bound automatically obtains this result up to a log factor!
- To see how, we need to look inside the online-to-batch conversion, and look beyond the linear approximation to F.
- Specifically, we need the following simple consequence of smoothness (see [Srebro et al., NeurIPS'10; Levy, NeurIPS'17] for related applications):

$$\mathbb{E}[\|\boldsymbol{g}_t\|^2] \le L(F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)) + \sigma^2$$

Adapting to Smoothness

In the online-to-batch conversion, we argued:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle\right] \text{ (convexity)}$$

Adapting to Smoothness

In the online-to-batch conversion, we argued:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \mathbb{E}\left[\sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle\right] \quad \text{(convexity)}$$

$$\text{use regret bound} \quad \le \mathbb{E}\left[\tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2}\right)\right]$$

Adapting to Smoothness

In the online-to-batch conversion, we argued:

$$
\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] &\le \mathbb{E}\left[\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle\right] \quad \text{(convexity)}\\
\text{use regret bound} \quad &\le \mathbb{E}\left[\tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2}\right)\right]\\
\text{use Jensen} \quad &\le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\mathbb{E}\left[\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2\right]}\right)
\end{aligned}
$$

Adapting to Smoothness

In the online-to-batch conversion, we argued:

$$
\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] &\le \mathbb{E}\left[\sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle\right] \quad \text{(convexity)}\\
\text{use regret bound} \quad &\le \mathbb{E}\left[\tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2}\right)\right]\\
\text{use Jensen} \quad &\le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\mathbb{E}\left[\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2\right]}\right)\\
\text{use smoothness} \quad &\le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{L\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] + T\sigma^2}\right)
\end{aligned}
$$

Adapting to Smoothness

In the online-to-batch conversion, we argued:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \mathbb{E}\left[\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle\right] \text{ (convexity)}$$

$$\text{use regret bound } \le \mathbb{E}\left[\tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2}\right)\right]$$

$$\text{use Jensen } \le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\mathbb{E}\left[\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2\right]}\right)$$

$$\text{use smoothness } \le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{L\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] + T\sigma^2}\right)$$

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \tilde{O}\left(L\|\boldsymbol{x}^\star\|^2 + \sigma\|\boldsymbol{x}^\star\|\sqrt{T}\right) \text{ (quadratic formula)}$$

Adapting to Smoothness

In the online-to-batch conversion, we argued:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \mathbb{E}\left[\sum_{t=1}^{T}\langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle\right] \quad \text{(convexity)}$$

$$\text{use regret bound} \quad \le \mathbb{E}\left[\tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2}\right)\right]$$

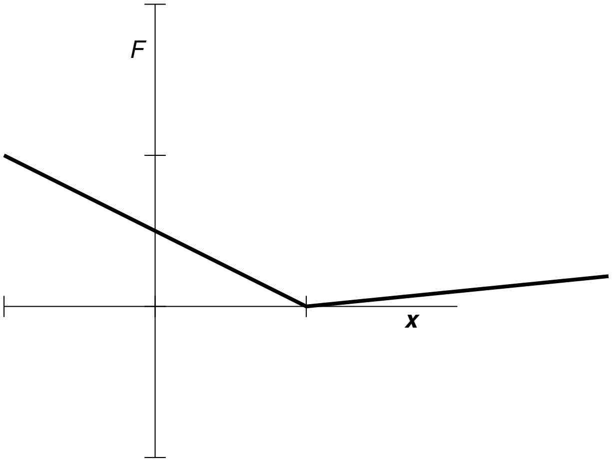
$$\text{use Jensen} \quad \le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{\mathbb{E}\left[\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2\right]}\right)$$

$$\text{use smoothness} \quad \le \tilde{O}\left(\|\boldsymbol{x}^\star\|\sqrt{L\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] + T\sigma^2}\right)$$

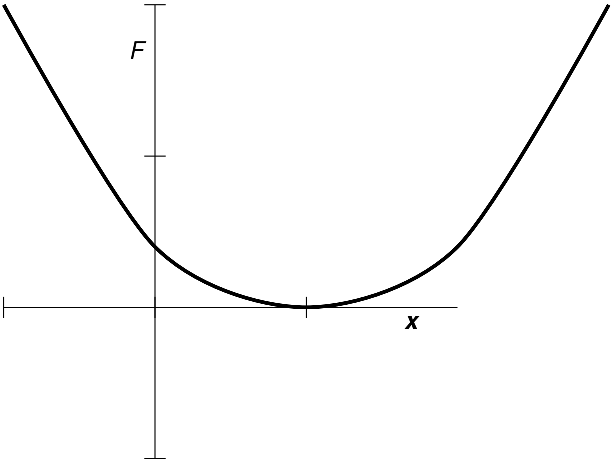
$$\mathbb{E}\left[\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star)\right] \le \tilde{O}\left(L\|\boldsymbol{x}^\star\|^2 + \sigma\|\boldsymbol{x}^\star\|\sqrt{T}\right) \quad \text{(quadratic formula)}$$

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right)\right] - F(\boldsymbol{x}^\star) \le \tilde{O}\left(\frac{L\|\boldsymbol{x}^\star\|^2}{T} + \frac{\sigma\|\boldsymbol{x}^\star\|}{\sqrt{T}}\right) \quad \text{(Jensen)}$$

Adapting to Strong Convexity



Adapting to Strong Convexity



Adapting to Strong-Convexity

- The strongly-convex losses are an important special class of optimization problems:

$$\nabla^2 F(\boldsymbol{x}) \succeq \mu I$$

- For strongly-convex losses, gradient descent with learning rate $\frac{1}{\mu t}$ guarantees [Hazan et al., MLJ'07]:

$$\mathbb{E}[F(\boldsymbol{x}_{\text{SGD}}) - F(\boldsymbol{x}^\star)] \leq \frac{\log(T)}{\mu T}$$

This requires knowledge of the parameter μ. Can we do without this?

A Regret Bound for Strong-Convexity

- MetaGrad [Koolan&van Erven, NeurIPS'16] and Maler [Wang et al., UAI'19] are online linear optimization algorithms that guarantee:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\sqrt{\sum_{t=1}^{T} \|\boldsymbol{x}_t - \boldsymbol{x}^\star\|^2 \|\boldsymbol{g}_t\|^2}\right)$$

- Amazingly, this automatically implies adaptivity to strong-convexity!
- We'll look at a simple trick for obtaining this style of bound.

Dynamics of $\boldsymbol{x}_t$

Recall:

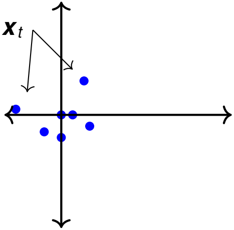
- We obtain error $\tilde{O}\left(\frac{\|\boldsymbol{x}^\star\| G}{\sqrt{T}}\right)$ (ignoring logs and constants).
- Intuitively, when $\boldsymbol{x}^\star$ is far from the starting point, we have more space to explore, so we expect to $\boldsymbol{x}_t$ to move a lot..

Dynamics of $\boldsymbol{x}_t$

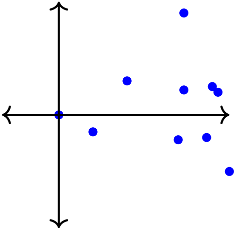
When $\boldsymbol{x}^\star$ is far from the starting point, $\boldsymbol{x}_t$ should move a lot more.

Dynamics of $\boldsymbol{x}_t$

When $\boldsymbol{x}^\star$ is far from the starting point, $\boldsymbol{x}_t$ should move a lot more.



$\boldsymbol{x}^\star = (0,0)$

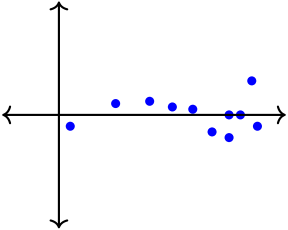


$\boldsymbol{x}^\star = (1,0)$

- Get much “tighter” distribution of $\boldsymbol{x}_t$s when $\boldsymbol{x}^\star$ is small.

Dynamics of $\boldsymbol{x}_t$

Can we have a tight distribution even when $\boldsymbol{x}^\star$ is not small?



Dynamics of $\boldsymbol{x}_t$

- We can get suboptimality

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star)\right] = \tilde{O}\left(\frac{\|\boldsymbol{x}^\star\| G}{\sqrt{T}}\right)$$

Dynamics of $\boldsymbol{x}_t$

- We can get suboptimality

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star)\right] = \tilde{O}\left(\frac{\|\boldsymbol{x}^\star\|G}{\sqrt{T}}\right)$$

- If we shift all outputs $\boldsymbol{x}_t$ by a constant $\boldsymbol{v}$, then we get

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star)\right] = \tilde{O}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{v}\|G}{\sqrt{T}}\right)$$

Dynamics of $\boldsymbol{x}_t$

- We can get suboptimality

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right)-F(\boldsymbol{x}^\star)\right]=\tilde{O}\left(\frac{\|\boldsymbol{x}^\star\|G}{\sqrt{T}}\right)$$

- If we shift all outputs $\boldsymbol{x}_t$ by a constant $\boldsymbol{v}$, then we get

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right)-F(\boldsymbol{x}^\star)\right]=\tilde{O}\left(\frac{\|\boldsymbol{x}^\star-\boldsymbol{v}\|G}{\sqrt{T}}\right)$$

- If $\boldsymbol{v}$ is close to $\boldsymbol{x}^\star$, then this is much better.

Dynamics of $\boldsymbol{x}_t$

- We can get suboptimality

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star)\right] = \tilde{O}\left(\frac{\|\boldsymbol{x}^\star\|G}{\sqrt{T}}\right)$$

- If we shift all outputs $\boldsymbol{x}_t$ by a constant $\boldsymbol{v}$, then we get

$$\mathbb{E}\left[F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right) - F(\boldsymbol{x}^\star)\right] = \tilde{O}\left(\frac{\|\boldsymbol{x}^\star - \boldsymbol{v}\|G}{\sqrt{T}}\right)$$

- If $\boldsymbol{v}$ is close to $\boldsymbol{x}^\star$, then this is much better.
- So how can we find a closer $\boldsymbol{v}$?

Using Strong Convexity

We need to move beyond the naive linear approximation $F(\boldsymbol{x}_t) - F(\boldsymbol{x}^\star) \le \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle$. Strongly-convex functions also satisfy:

$$F(\boldsymbol{x}) - F(\boldsymbol{x}^\star) \ge \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{x}^\star\|^2$$

This means that <u>points with low error are close to $\boldsymbol{x}^\star$</u>.

Adapting to Strong Convexity

- Small function value means that we are close to $\boldsymbol{x}^\star$.

[Cutkosky&Boahen, NeurIPS'17; Cutkosky&Orabona, COLT'18]

Adapting to Strong Convexity

- Small function value means that we are close to $\boldsymbol{x}^\star$.
- The "average iterate" $\frac{\sum_{t=1}^{T} \boldsymbol{x}_t}{T}$ has small function value, so it must be close to $\boldsymbol{x}^\star$!

[Cutkosky&Boahen, NeurIPS'17; Cutkosky&Orabona, COLT'18]

Adapting to Strong Convexity

- Small function value means that we are close to $\boldsymbol{x}^\star$.
- The "average iterate" $\frac{\sum_{t=1}^T \boldsymbol{x}_t}{T}$ has small function value, so it must be close to $\boldsymbol{x}^\star$! We should try using a running average for v:

[Cutkosky&Boahen, NeurIPS'17; Cutkosky&Orabona, COLT'18]

Adapting to Strong Convexity

- Small function value means that we are close to $\boldsymbol{x}^\star$.
- The "average iterate" $\frac{\sum_{t=1}^T \boldsymbol{x}_t}{T}$ has small function value, so it must be close to $\boldsymbol{x}^\star$! We should try using a running average for v:

$$\hat{\boldsymbol{x}}_{t+1} = \text{Parameter-free Output}$$
$$\boldsymbol{x}_{t+1} = \hat{\boldsymbol{x}}_{t+1} + \underbrace{\frac{\sum_{i=1}^t \|\boldsymbol{g}_i\|^2 \boldsymbol{x}_i}{\sum_{i=1}^t \|\boldsymbol{g}_i\|^2}}_{\text{Can use } \sum_{i=1}^t \boldsymbol{x}_i/t \text{ too.}}$$

[Cutkosky&Boahen, NeurIPS'17; Cutkosky&Orabona, COLT'18]

Adapting to Strong Convexity

- Small function value means that we are close to $\boldsymbol{x}^\star$.
- The "average iterate" $\frac{\sum_{t=1}^T \boldsymbol{x}_t}{T}$ has small function value, so it must be close to $\boldsymbol{x}^\star$! We should try using a running average for v:

$$\hat{\boldsymbol{x}}_{t+1} = \text{Parameter-free Output}$$
$$\boldsymbol{x}_{t+1} = \hat{\boldsymbol{x}}_{t+1} + \underbrace{\frac{\sum_{i=1}^t \|\boldsymbol{g}_i\|^2 \boldsymbol{x}_i}{\sum_{i=1}^t \|\boldsymbol{g}_i\|^2}}_{\text{Can use } \sum_{i=1}^t \boldsymbol{x}_i/t \text{ too.}}$$

- Via some careful algebra, we get:

$$\sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \le \tilde{O}\left(\sqrt{\sum_{t=1}^T \|\boldsymbol{x}_t - \boldsymbol{x}^\star\|^2 \|\boldsymbol{g}_t\|^2}\right)$$

This implies $\tilde{O}\left(\frac{1}{\mu T}\right)$ error in strongly-convex problems.

[Cutkosky&Boahen, NeurIPS'17; Cutkosky&Orabona, COLT'18]

Adapting to Strong Convexity

- Small function value means that we are close to $\boldsymbol{x}^\star$.
- The "average iterate" $\frac{\sum_{t=1}^T \boldsymbol{x}_t}{T}$ has small function value, so it must be close to $\boldsymbol{x}^\star$! We should try using a running average for v:

$$\hat{\boldsymbol{x}}_{t+1} = \text{Parameter-free Output}$$
$$\boldsymbol{x}_{t+1} = \hat{\boldsymbol{x}}_{t+1} + \underbrace{\frac{\sum_{i=1}^t \|\boldsymbol{g}_i\|^2 \boldsymbol{x}_i}{\sum_{i=1}^t \|\boldsymbol{g}_i\|^2}}_{\text{Can use } \sum_{i=1}^t \boldsymbol{x}_i/t \text{ too.}}$$

- Via some careful algebra, we get:

$$\sum_{t=1}^T \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \le \tilde{O}\left(\sqrt{\sum_{t=1}^T \|\boldsymbol{x}_t - \boldsymbol{x}^\star\|^2 \|\boldsymbol{g}_t\|^2}\right)$$

  This implies $\tilde{O}\left(\frac{1}{\mu T}\right)$ error in strongly-convex problems.
- Adding the average works for <u>any</u> parameter-free algorithm.

[Cutkosky&Boahen, NeurIPS'17; Cutkosky&Orabona, COLT'18]

Dealing with Unknown G

- We have been assuming $G = 1$. In practice, one can usually guess a good upper bound for G by just looking at the first few gradients.

Dealing with Unknown G

- We have been assuming $G = 1$. In practice, one can usually guess a good upper bound for G by just looking at the first few gradients.
- The guess G can be wildly wrong before anything bad happens:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{G^2 + \sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

Dealing with Unknown G

- We have been assuming $G = 1$. In practice, one can usually guess a good upper bound for G by just looking at the first few gradients.
- The guess G can be wildly wrong before anything bad happens:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{G^2 + \sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

- It is “safe” for G to over-estimate by a factor of $O(\sqrt{T})$.

Dealing with Unknown G

- We have been assuming $G = 1$. In practice, one can usually guess a good upper bound for G by just looking at the first few gradients.
- The guess G can be wildly wrong before anything bad happens:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{G^2 + \sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

- It is “safe” for G to over-estimate by a factor of $O(\sqrt{T})$.
- In contrast, the classical adaptive gradient approach stipulates a known bound $B \geq \|\boldsymbol{x}^\star\|$ to guarantee:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \leq O\left(B \sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

Dealing with Unknown G

- We have been assuming $G = 1$. In practice, one can usually guess a good upper bound for G by just looking at the first few gradients.
- The guess G can be wildly wrong before anything bad happens:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \le \tilde{O}\left(\|\boldsymbol{x}^\star\| \sqrt{G^2 + \sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

- It is "safe" for G to over-estimate by a factor of $O(\sqrt{T})$.
- In contrast, the classical adaptive gradient approach stipulates a known bound $B \ge \|\boldsymbol{x}^\star\|$ to guarantee:

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}^\star \rangle \le O\left(B \sqrt{\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2} \right)$$

- This pays linearly for any over-estimate in B.

Dealing with Unknown G

- What about the true worst-case bound?

Dealing with Unknown G

- What about the true worst-case bound?
- If we do not know G, an adversary can force regret $\tilde{\Omega}(\|\boldsymbol{x}^\star\|G\sqrt{T} + G\|\boldsymbol{x}^\star\|^3)$ [Mhammedi&Koolen, COLT'20].

Dealing with Unknown G

- What about the true worst-case bound?
- If we do not know G, an adversary can force regret $\tilde{\Omega}(\|\boldsymbol{x}^\star\|G\sqrt{T} + G\|\boldsymbol{x}^\star\|^3)$ [Mhammedi&Koolen, COLT'20].
  - Note that the extra $\|\boldsymbol{x}^\star\|^3$ term does not depend on T.

Dealing with Unknown G

- What about the true worst-case bound?
- If we do not know G, an adversary can force regret $\tilde{\Omega}(\|\boldsymbol{x}^\star\| G\sqrt{T} + G\|\boldsymbol{x}^\star\|^3)$ [Mhammedi&Koolen, COLT'20].
  - Note that the extra $\|\boldsymbol{x}^\star\|^3$ term <u>does not depend on T</u>.
  - It is unclear how "real" this lower bound is in real stochastic environments.

Dealing with Unknown G

- What about the true worst-case bound?
- If we do not know G, an adversary can force regret $\tilde{\Omega}(\|\boldsymbol{x}^\star\| G\sqrt{T} + G\|\boldsymbol{x}^\star\|^3)$ [Mhammedi&Koolen, COLT'20].
  - Note that the extra $\|\boldsymbol{x}^\star\|^3$ term <u>does not depend on T</u>.
  - It is unclear how "real" this lower bound is in real stochastic environments.
- The bound is tight. It can be obtained "lying" to the algorithm by feeding $\boldsymbol{g}_t$ clipped to have norm at most $\max_{i<t} \|\boldsymbol{g}_i\|$ [Cutkosky, COLT'19; Mhammedi&Koolen, COLT'20].

Summary of Part 3

Once you know how to make parameter-free algorithms, it is surprisingly easy to build extra results.

1. Adapt to sparse $\boldsymbol{x}^\star$.
2. Adapt to smoothness.
3. Adapt to strong-convexity.
4. Incorporate unknown G.