Randomized Smoothing Techniques in Optimization

John Duchi

Based on joint work with Peter Bartlett, Michael Jordan,
Martin Wainwright, Andre Wibisono

Stanford University

Information Systems Laboratory Seminar
October 2014

Problem Statement

Goal: solve the following problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned}$$

Problem Statement

Goal: solve the following problem

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{X} \end{array}$$

Often, we will assume

$$f(x) := \frac{1}{n} \sum_{i=1}^{n} F(x; \xi_i) \quad \text{or} \quad f(x) := \mathbb{E}[F(x; \xi)]$$

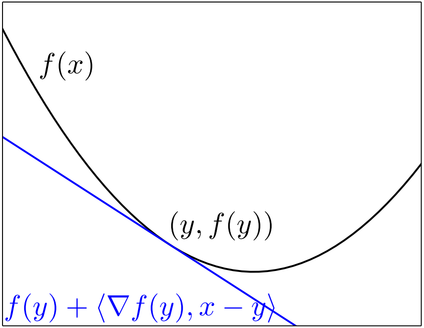
Gradient Descent

Goal: solve

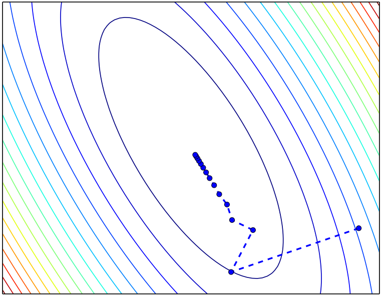
$$\text{minimize} \quad f(x)$$

Technique: go down the slope,

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

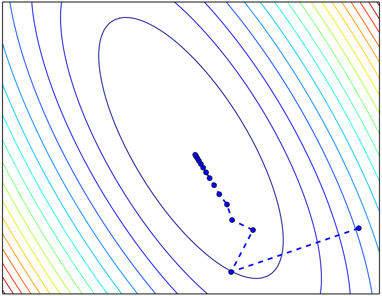


When is optimization easy?

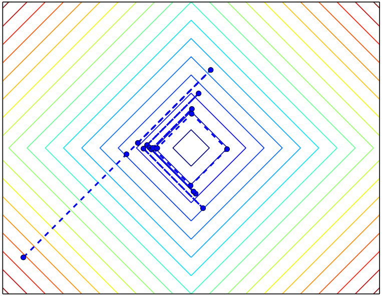


Easy problem: function is convex, nice and smooth

When is optimization easy?

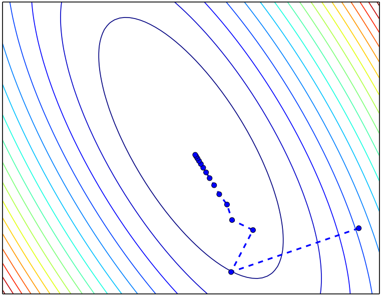


Easy problem: function is convex, nice and smooth

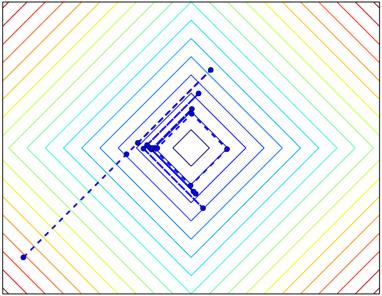


Not so easy problem: function is non-smooth

When is optimization easy?



Easy problem: function is convex, nice and smooth



Not so easy problem: function is non-smooth

Even harder problems:

- We cannot compute gradients $\nabla f(x)$
- Function f is non-convex and non-smooth

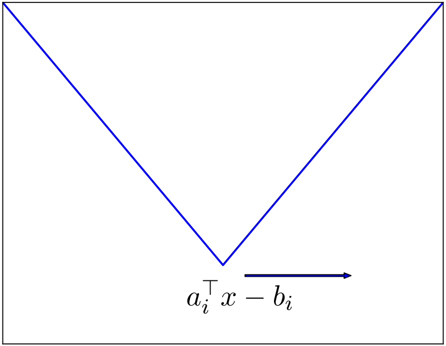
Example 1: Robust regression

- Data in pairs $\xi_i = (a_i, b_i) \in \mathbb{R}^d \times \mathbb{R}$
- Want to estimate $b_i \approx a_i^\top x$

Example 1: Robust regression

- Data in pairs $\xi_i = (a_i, b_i) \in \mathbb{R}^d \times \mathbb{R}$
- Want to estimate $b_i \approx a_i^\top x$
- To avoid outliers, minimize

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} |a_i^\top x - b_i| = \frac{1}{n}\|Ax - b\|_1$$

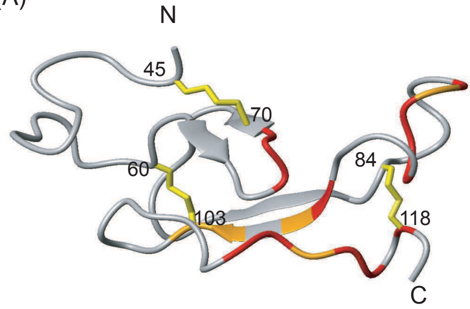


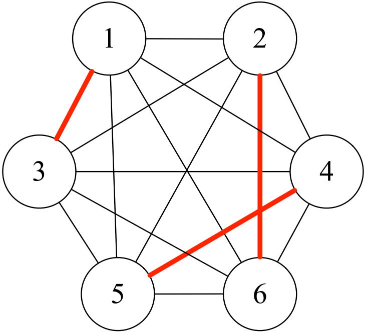
Example 2: Protein Structure Prediction

RSCCPCYWGGCPWGQNCYPEGCSGPKV

1 2 3 4 5 6

(A)

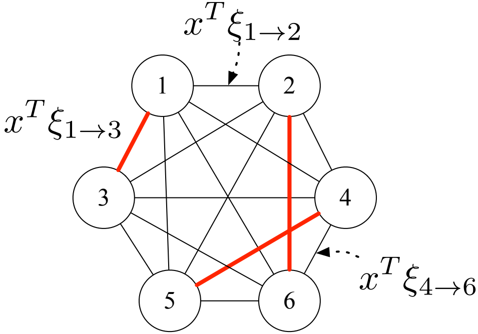




(Grace et al., PNAS 2004)

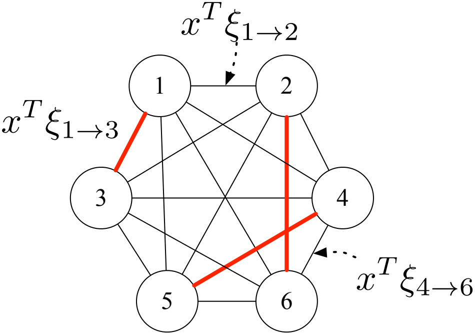
Protein Structure Prediction

Featurize edges e in graph: vector ξ_e. Labels y are *matching* in a graph, set $\mathcal{V}$ is *all matchings*.



Protein Structure Prediction

Featurize edges e in graph: vector ξ_e. Labels y are *matching* in a graph, set $\mathcal{V}$ is *all matchings*.

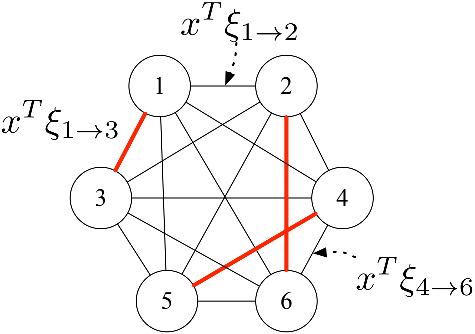


Goal: Learn weights x so that

$$\operatorname*{argmax}_{\widehat{\nu}\in\mathcal{V}} \left\{ \sum_{e\in\widehat{\nu}} \xi_e^\top x \right\} = \nu$$

i.e. learn x so maximum matching in graph with edge weights $x^\top \xi_e$ is correct

Protein Structure Prediction

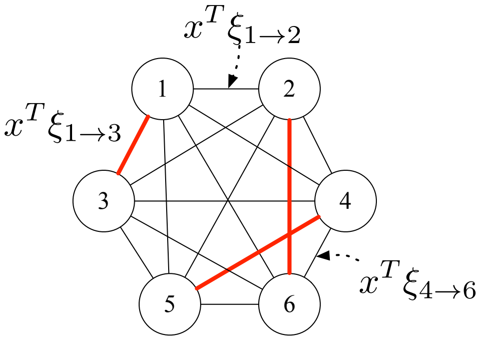


Goal: Learn weights x so that

$$\operatorname*{argmax}_{\widehat{\nu}\in\mathcal{V}} \left\{ \sum_{e\in\widehat{\nu}} \xi_e^\top x \right\} = \nu$$

i.e. learn x so maximum matching in graph with edge weights $x^\top \xi_e$ is correct

Protein Structure Prediction



Goal: Learn weights x so that

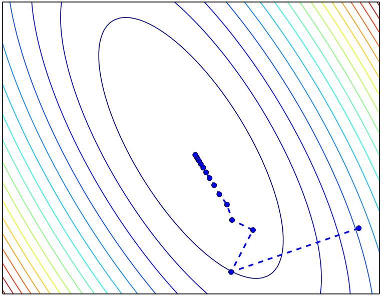
$$\underset{\widehat{\nu}\in\mathcal{V}}{\operatorname{argmax}}\left\{\sum_{e\in\widehat{\nu}}\xi_e^\top x\right\} = \nu$$

i.e. learn x so maximum matching in graph with edge weights $x^\top\xi_e$ is correct

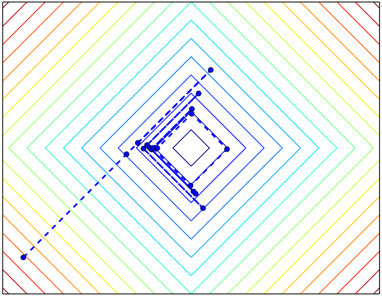
Loss function: $L(\nu,\widehat{\nu})$ is *number of disagreements* in matchings

$$F(x;\{\xi,\nu\}) := \max_{\widehat{\nu}\in\mathcal{V}}\left(L(\nu,\widehat{\nu}) + x^\top\sum_{e\in\widehat{\nu}}\xi_e - x^\top\sum_{e\in\nu}\xi_e\right).$$

When is optimization easy?



Easy problem: function is convex, nice and smooth



Not so easy problem: function is non-smooth

Even harder problems:

- We cannot compute gradients $\nabla f(x)$
- Function f is non-convex and non-smooth

One technique to address many of these

Instead of only using $f(x)$ and $\nabla f(x)$ to solve

$$\text{minimize} \quad f(x),$$

get more *global* information

One technique to address many of these

Instead of only using $f(x)$ and $\nabla f(x)$ to solve

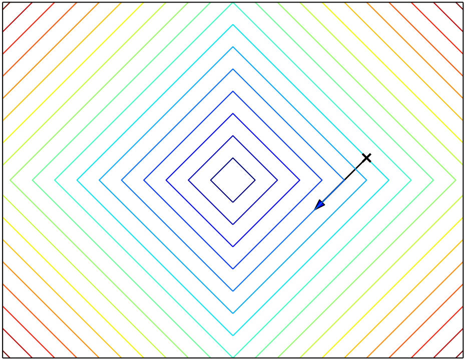
$$\text{minimize} \quad f(x),$$

get more *global* information

Let Z be a random variable, and for small u, look at f near points

$$f(x + uZ),$$

where u is small



One technique to address many of these

Instead of only using $f(x)$ and $\nabla f(x)$ to solve

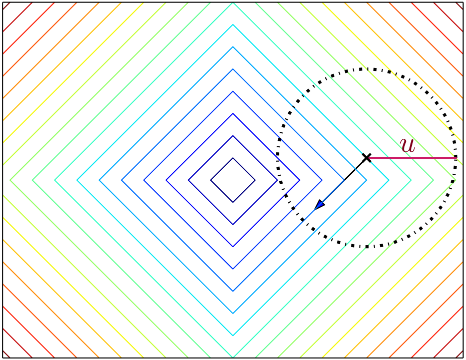
$$\text{minimize} \quad f(x),$$

get more *global* information

Let Z be a random variable, and for small u, look at f near points

$$f(x + uZ),$$

where u is small



One technique to address many of these

Instead of only using $f(x)$ and $\nabla f(x)$ to solve

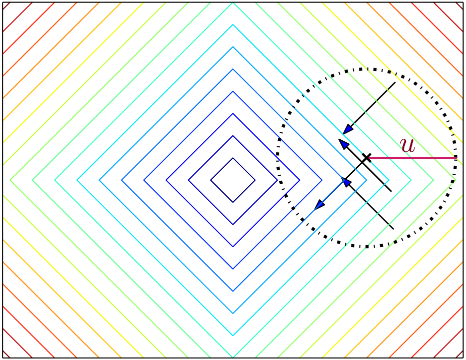
$$\text{minimize} \quad f(x),$$

get more *global* information

Let Z be a random variable, and for small u, look at f near points

$$f(x + uZ),$$

where u is small



Three instances

- Solving previously unsolvable problems [Burke, Lewis, Overton 2005]
  - Non-smooth, non-convex problems

Three instances

I Solving previously unsolvable problems [Burke, Lewis, Overton 2005]
- Non-smooth, non-convex problems

II Optimal convergence guarantees for problems with existing algorithms [D., Jordan, Wainwright, Wibisono 2014]
- Smooth and non-smooth zero order stochastic and non-stochastic optimization problems

Three instances

I Solving previously unsolvable problems [Burke, Lewis, Overton 2005]

- Non-smooth, non-convex problems

II Optimal convergence guarantees for problems with existing algorithms [D., Jordan, Wainwright, Wibisono 2014]

- Smooth and non-smooth zero order stochastic and non-stochastic optimization problems

III Parallelism: really fast solutions for large scale problems [D., Bartlett, Wainwright 2013]

- Smooth and non-smooth stochastic optimization problems

Instance I: Gradient Sampling Algorithm

Problem: Solve

$$\underset{x \in \mathcal{X}}{\text{minimize}} \; f(x)$$

where f is potentially non-smooth and non-convex (but assume it is continuous and a.e. differentiable) [Burke, Lewis, Overton 2005]

Instance I: Gradient Sampling Algorithm

Problem: Solve

$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f(x)$$

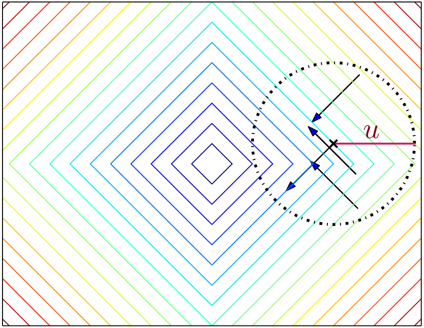
where f is potentially non-smooth and non-convex (but assume it is continuous and a.e. differentiable) [Burke, Lewis, Overton 2005]

At each iteration t,

- Draw $Z_1, \ldots, Z_m$ i.i.d. $\|Z_i\| \leq 1$
- Set $g_t^i = \nabla f(x_t + u Z_i)$
- Set gradient g_t as

$$g_t = \operatorname{argmin}_g \left\{ \|g\|_2^2 : \begin{array}{c} g = \sum_i \lambda_i g_t^i \\ \lambda \geq 0, \sum_i \lambda_i = 1 \end{array} \right\}$$

- Update $x_{t+1} = x_t - \alpha g_t$, where $\alpha > 0$ chosen by line search



Instance I: Gradient Sampling Algorithm

Problem: Solve

$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f(x)$$

where f is potentially non-smooth and non-convex (but assume it is continuous and a.e. differentiable) [Burke, Lewis, Overton 2005]
Define the set

$$G_u(x) := \mathrm{Conv}\left\{ \nabla f(x') : \left\| x' - x \right\|_2 \leq u, \;\; \nabla f(x') \text{ exists} \right\}$$

Proposition (Burke, Lewis, Overton):
There exist cluster points $\bar{x}$ of the sequence x_t, and for any such cluster point,

$$0 \in G_u(\bar{x})$$

Instance II: Zero Order Optimization

Problem: We want to solve

$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f(x) = \mathbb{E}[F(x; \xi)]$$

but we are only allowed to observe function values $f(x)$ (or $F(x; \xi)$)

Instance II: Zero Order Optimization

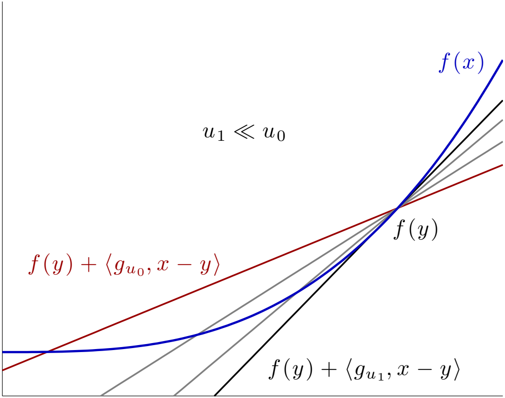
Problem: We want to solve

$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f(x) = \mathbb{E}[F(x; \xi)]$$

but we are only allowed to observe function values $f(x)$ (or $F(x; \xi)$)

Idea: Approximate gradient by function differences

$$f'(y) \approx g_u := \frac{f(y+u) - f(y)}{u}$$



Instance II: Zero Order Optimization

Problem: We want to solve

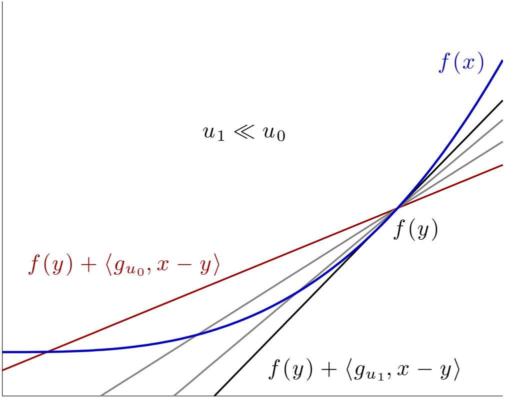
$$\underset{x \in \mathcal{X}}{\text{minimize}} \; f(x) = \mathbb{E}[F(x;\xi)]$$

but we are only allowed to observe function values $f(x)$ (or $F(x;\xi)$)

Idea: Approximate gradient by function differences

$$f'(y) \approx g_u := \frac{f(y+u) - f(y)}{u}$$

- Long history in optimization: Kiefer-Wolfowitz, Spall, Robbins-Monroe



Instance II: Zero Order Optimization

Problem: We want to solve

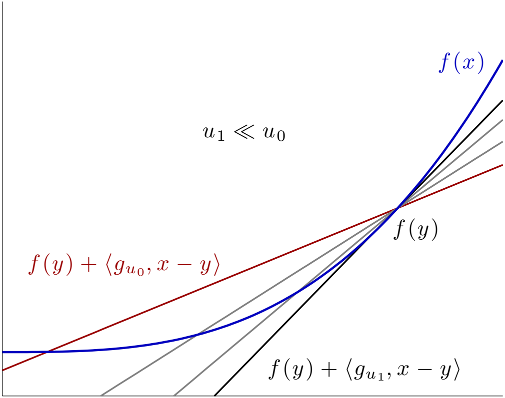
$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f(x) = \mathbb{E}[F(x;\xi)]$$

but we are only allowed to observe function values $f(x)$ (or $F(x;\xi)$)

Idea: Approximate gradient by function differences

$$f'(y) \approx g_u := \frac{f(y+u) - f(y)}{u}$$

- Long history in optimization: Kiefer-Wolfowitz, Spall, Robbins-Monroe
- Can randomized perturbations give insights?



Stochastic Gradient Descent

Algorithm: At iteration t

- Choose random ξ, set

$$g_t = \nabla F(x_t; \xi_i)$$

- Update

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

Stochastic Gradient Descent

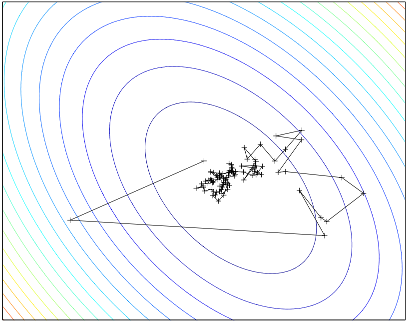
Algorithm: At iteration t

- Choose random ξ, set

$$g_t = \nabla F(x_t; \xi_i)$$

- Update

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



Stochastic Gradient Descent

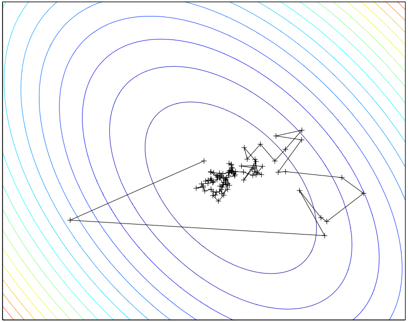
Algorithm: At iteration t

- Choose random ξ, set

$$g_t = \nabla F(x_t; \xi_i)$$

- Update

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



Theorem (Russians): Let $\widehat{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ and assume $R \geq \|x^* - x_1\|_2$, $G^2 \geq \mathbb{E}[\|g_t\|_2^2]$. Then

$$\mathbb{E}[f(\widehat{x}_T) - f(x^*)] \leq RG\frac{1}{\sqrt{T}}$$

Stochastic Gradient Descent

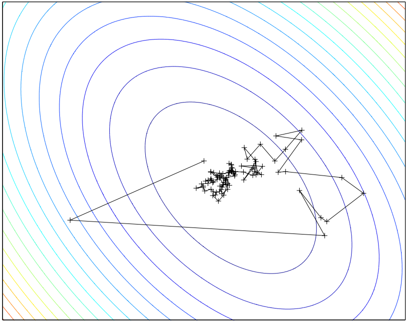
Algorithm: At iteration t

- Choose random ξ, set

$$g_t = \nabla F(x_t; \xi_i)$$

- Update

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



Theorem (Russians): Let $\widehat{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ and assume $R \geq \|x^* - x_1\|_2$, $G^2 \geq \mathbb{E}[\|g_t\|_2^2]$. Then

$$\mathbb{E}[f(\widehat{x}_T) - f(x^*)] \leq RG\frac{1}{\sqrt{T}}$$

Note: Dependence on G important

Derivative-free gradient descent

$$\boxed{\mathbb{E}[f(\widehat{x}_T) - f(x^*)] \leq RG\frac{1}{\sqrt{T}}}$$

Question: How well can we estimate gradient ∇f using only function differences? And how small is the norm of this estimate?

Derivative-free gradient descent

$$\mathbb{E}[f(\widehat{x}_T) - f(x^*)] \leq RG\frac{1}{\sqrt{T}}$$

Question: How well can we estimate gradient ∇f using only function differences? And how small is the norm of this estimate?

First idea gradient estimator:

- Sample $Z \sim \mu$ satisfying $\mathbb{E}_\mu[ZZ^\top] = I_{d\times d}$
- Gradient estimator at x:

$$g = \frac{f(x + uZ) - f(x)}{u} Z$$

Perform gradient descent using these g

Two-point gradient estimates

- At any point x and any direction z, for small $u > 0$

$$\frac{f(x+uz) - f(x)}{u} \approx f'(x,z) := \lim_{h \downarrow 0} \frac{f(x+hz) - f(x)}{h}$$

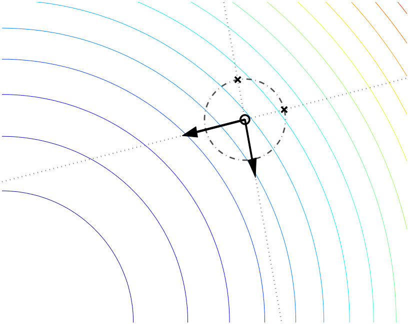
- If $\nabla f(x)$ exists, $f'(x,z) = \langle \nabla f(x), z \rangle$
- If $\mathbb{E}[ZZ^\top] = I$, then $\mathbb{E}[f'(x,Z)Z] = \mathbb{E}[ZZ^\top \nabla f(x)] = \nabla f(x)$

Two-point gradient estimates

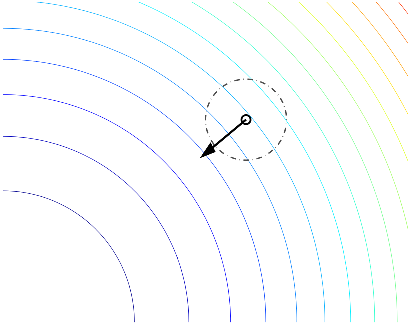
- At any point x and any direction z, for small $u > 0$

$$\frac{f(x+uz)-f(x)}{u} \approx f'(x,z) := \lim_{h \downarrow 0} \frac{f(x+hz)-f(x)}{h}$$

- If $\nabla f(x)$ exists, $f'(x,z) = \langle \nabla f(x), z \rangle$
- If $\mathbb{E}[ZZ^\top] = I$, then $\mathbb{E}[f'(x,Z)Z] = \mathbb{E}[ZZ^\top \nabla f(x)] = \nabla f(x)$



Random estimates



Average $\approx \nabla f$

Two-point stochastic gradient: differentiable functions

To solve d-dimensional problem

$$\underset{x \in \mathcal{X} \subset \mathbb{R}^d}{\text{minimize}} \; f(x) := \mathbb{E}[F(x;\xi)]$$

Algorithm: Iterate

- Draw ξ according to distribution, draw $Z \sim \mu$ with $\mathrm{Cov}(Z) = I$
- Set $u_t = u/t$ and

$$g_t = \frac{F(x_t + u_t Z;\xi) - F(x_t;\xi)}{u_t} Z$$

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

Two-point stochastic gradient: differentiable functions

To solve d-dimensional problem

$$\underset{x \in \mathcal{X} \subset \mathbb{R}^d}{\text{minimize}} \; f(x) := \mathbb{E}[F(x;\xi)]$$

Algorithm: Iterate

- Draw ξ according to distribution, draw $Z \sim \mu$ with $\mathrm{Cov}(Z) = I$
- Set $u_t = u/t$ and

$$g_t = \frac{F(x_t + u_t Z; \xi) - F(x_t; \xi)}{u_t} Z$$

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

Theorem (D., Jordan, Wainwright, Wibisono)**:** With appropriate α, if $R \geq \|x^* - x_1\|_2$ and $\mathbb{E}[\|\nabla F(x;\xi)\|_2^2] \leq G^2$ for all x, then

$$\mathbb{E}[f(\widehat{x}_T) - f(x^*)] \leq RG \cdot \frac{\sqrt{d}}{\sqrt{T}} + O\left(u^2 \frac{\log T}{T}\right).$$

Comparisons to knowing gradient

Convergence rate scaling

$$RG\frac{1}{\sqrt{T}} \quad \text{versus} \quad RG\frac{\sqrt{d}}{\sqrt{T}}$$

Comparisons to knowing gradient

Convergence rate scaling

$$RG\frac{1}{\sqrt{T}} \quad \text{versus} \quad RG\frac{\sqrt{d}}{\sqrt{T}}$$

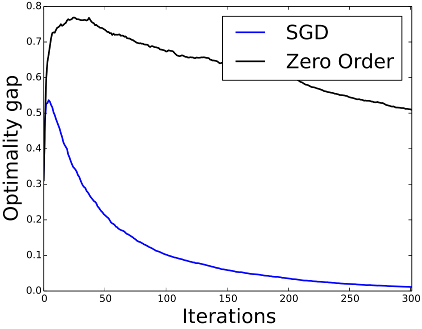
- To achieve ϵ-accuracy, $1/\epsilon^2$ versus d/ϵ^2

Comparisons to knowing gradient

Convergence rate scaling

$$RG\frac{1}{\sqrt{T}} \quad \text{versus} \quad RG\frac{\sqrt{d}}{\sqrt{T}}$$

- To achieve ϵ-accuracy, $1/\epsilon^2$ versus d/ϵ^2

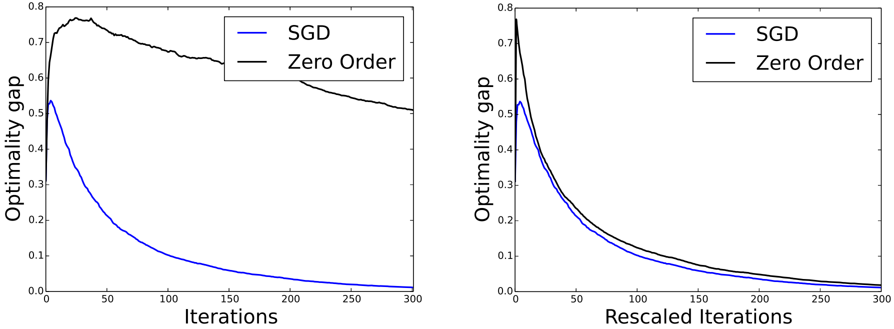


Comparisons to knowing gradient

Convergence rate scaling

$$RG\frac{1}{\sqrt{T}} \quad \text{versus} \quad RG\frac{\sqrt{d}}{\sqrt{T}}$$

- To achieve ϵ-accuracy, $1/\epsilon^2$ versus d/ϵ^2



Two-point stochastic gradient: non-differentiable functions

Problem: If f is non-differentiable, "kinks" make estimates too large

Two-point stochastic gradient: non-differentiable functions

Problem: If f is non-differentiable, "kinks" make estimates too large

Two-point stochastic gradient: non-differentiable functions

Problem: If f is non-differentiable, "kinks" make estimates too large

Example: Let $f(x) = \|x\|_2$. Then if $\mathbb{E}[ZZ^\top] = I_{d\times d}$, at $x = 0$

$$\mathbb{E}[\|(f(uZ) - 0)Z/u\|_2^2] = \mathbb{E}[\|Z\|_2^2 \, \|Z\|_2^2] \geq d^2$$

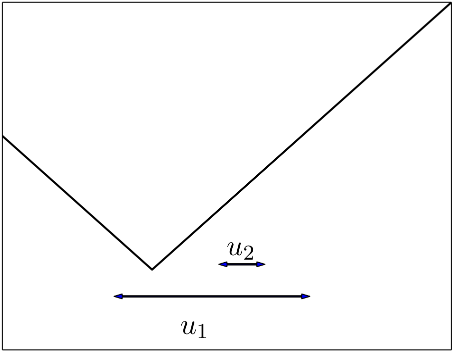
Two-point stochastic gradient: non-differentiable functions

Problem: If f is non-differentiable, "kinks" make estimates too large

Example: Let $f(x) = \|x\|_2$. Then if $\mathbb{E}[ZZ^\top] = I_{d\times d}$, at $x = 0$

$$\mathbb{E}[\|(f(uZ) - 0)Z/u\|_2^2] = \mathbb{E}[\|Z\|_2^2 \, \|Z\|_2^2] \geq d^2$$

Idea: More randomization!



Two-point stochastic gradient: non-differentiable functions

Problem: If f is non-differentiable, "kinks" make estimates too large

Proposition (D., Jordan, Wainwright, Wibisono)**:** If Z_1, Z_2 are $N(0, I_{d\times d})$ or uniform on $\|z\|_2 \le \sqrt{d}$, then

$$g := \frac{f(x + u_1 Z_1 + u_2 Z_2) - f(x + u_1 Z_1)}{u_2} Z_2$$

satisfies

$$\mathbb{E}[g] = \nabla f_{u_1}(x) + \mathcal{O}(u_2/u_1) \quad \text{and} \quad \mathbb{E}[\|g\|_2^2] \le d\left(\sqrt{\frac{u_2}{u_1}} d + \log(2d)\right).$$

Two-point stochastic gradient: non-differentiable functions

Problem: If f is non-differentiable, "kinks" make estimates too large

Proposition (D., Jordan, Wainwright, Wibisono)**:** If Z_1, Z_2 are $N(0, I_{d\times d})$ or uniform on $\|z\|_2 \le \sqrt{d}$, then

$$g := \frac{f(x + u_1 Z_1 + u_2 Z_2) - f(x + u_1 Z_1)}{u_2} Z_2$$

satisfies

$$\mathbb{E}[g] = \nabla f_{u_1}(x) + \mathcal{O}(u_2/u_1) \quad \text{and} \quad \mathbb{E}[\|g\|_2^2] \le d\left(\sqrt{\frac{u_2}{u_1}}d + \log(2d)\right).$$

Note: If $u_2/u_1 \to 0$, scaling linear in d

Two-point sub-gradient: non-differentiable functions

To solve d-dimensional problem

$$\text{minimize } \; f(x) \quad \text{subject to } \; x \in \mathcal{X} \subset \mathbb{R}^d$$

Algorithm: Iterate

- Draw $Z_1 \sim \mu$ and $Z_2 \sim \mu$ with $\mathrm{Cov}(Z) = I$
- Set $u_{t,1} = u/t$, $u_{t,2} = u/t^2$, and

$$g_t = \frac{f(x_t + u_{t,1}Z_1 + u_{t,2}Z_2) - F(x_t + u_{t,1}Z_1)}{u_t} Z_2$$

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

Two-point sub-gradient: non-differentiable functions

To solve d-dimensional problem

$$\text{minimize } \; f(x) \quad \text{subject to } \; x \in \mathcal{X} \subset \mathbb{R}^d$$

Algorithm: Iterate

- Draw $Z_1 \sim \mu$ and $Z_2 \sim \mu$ with $\mathrm{Cov}(Z) = I$
- Set $u_{t,1} = u/t$, $u_{t,2} = u/t^2$, and

$$g_t = \frac{f(x_t + u_{t,1} Z_1 + u_{t,2} Z_2) - F(x_t + u_{t,1} Z_1)}{u_t} Z_2$$

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

Theorem (D., Jordan, Wainwright, Wibisono)**:** With appropriate α, if $R \geq \|x^* - x_1\|_2$ and $\|\partial f(x)\|_2^2 \leq G^2$ for all x, then

$$\mathbb{E}[f(\widehat{x}_T) - f(x^*)] \leq RG \cdot \frac{\sqrt{d \log d}}{\sqrt{T}} + O\left(u \frac{\log T}{T}\right).$$

Two-point sub-gradient: non-differentiable functions

To solve d-dimensional problem

$$\underset{x \in \mathcal{X} \subset \mathbb{R}^d}{\text{minimize}} \;\; f(x) := \mathbb{E}[F(x;\xi)]$$

Algorithm: Iterate

- Draw ξ according to distribution, draw $Z_1 \sim \mu$ and $Z_2 \sim \mu$ with $\mathrm{Cov}(Z) = I$
- Set $u_{t,1} = u/t$, $u_{t,2} = u/t^2$, and

$$g_t = \frac{F(x_t + u_{t,1} Z_1 + u_{t,2} Z_2; \xi) - F(x_t + u_{t,1}; \xi)}{u_t} Z_2$$

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

Two-point sub-gradient: non-differentiable functions

To solve d-dimensional problem

$$\underset{x \in \mathcal{X} \subset \mathbb{R}^d}{\text{minimize}} \; f(x) := \mathbb{E}[F(x;\xi)]$$

Algorithm: Iterate

- Draw ξ according to distribution, draw $Z_1 \sim \mu$ and $Z_2 \sim \mu$ with $\mathrm{Cov}(Z) = I$
- Set $u_{t,1} = u/t$, $u_{t,2} = u/t^2$, and

$$g_t = \frac{F(x_t + u_{t,1}Z_1 + u_{t,2}Z_2;\xi) - F(x_t + u_{t,1};\xi)}{u_t} Z_2$$

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

Corollary (D., Jordan, Wainwright, Wibisono)**:** With appropriate α, if $R \geq \|x^* - x_1\|_2$ and $\mathbb{E}[\|\nabla F(x;\xi)\|_2^2] \leq G^2$ for all x, then

$$\mathbb{E}[f(\widehat{x}_T) - f(x^*)] \leq RG \cdot \frac{\sqrt{d \log d}}{\sqrt{T}} + O\left(u \frac{\log T}{T}\right).$$

Wrapping up zero-order gradient methods

- If gradients available, convergence rates of $\sqrt{1/T}$
- If only zero order information available, in smooth and non-smooth case, convergence rates of $\sqrt{d/T}$
- Time to ϵ-accuracy: $1/\epsilon^2 \mapsto d/\epsilon^2$

Wrapping up zero-order gradient methods

- If gradients available, convergence rates of $\sqrt{1/T}$
- If only zero order information available, in smooth and non-smooth case, convergence rates of $\sqrt{d/T}$
- Time to ϵ-accuracy: $1/\epsilon^2 \mapsto d/\epsilon^2$
- **Sharpness:** In *stochastic* case, no algorithms exist that can do better than those we have provided. That is, lower bound for *all* zero-order algorithms of

$$RG\frac{\sqrt{d}}{\sqrt{T}}.$$

Wrapping up zero-order gradient methods

- If gradients available, convergence rates of $\sqrt{1/T}$
- If only zero order information available, in smooth and non-smooth case, convergence rates of $\sqrt{d/T}$
- Time to ϵ-accuracy: $1/\epsilon^2 \mapsto d/\epsilon^2$
- **Sharpness:** In *stochastic* case, no algorithms exist that can do better than those we have provided. That is, lower bound for *all* zero-order algorithms of

$$RG\frac{\sqrt{d}}{\sqrt{T}}.$$

- **Open question:** Non-stochastic lower bounds? (Sebastian Pokutta, next week.)

Instance III: Parallelization and fast algorithms

Goal: solve the following problem

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{X} \end{array}$$

where

$$f(x) := \frac{1}{n}\sum_{i=1}^{n} F(x;\xi_i) \quad \text{or} \quad f(x) := \mathbb{E}[F(x;\xi)]$$

Stochastic Gradient Descent

Problem: Tough to compute

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} F(x; \xi_i).$$

Instead: At iteration t

- Choose random ξ_i, set

$$g_t = \nabla F(x_t; \xi_i)$$

- Update

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

Stochastic Gradient Descent

Problem: Tough to compute

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} F(x;\xi_i).$$

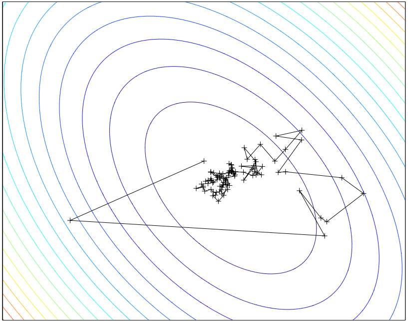
Instead: At iteration t

- Choose random ξ_i, set

$$g_t = \nabla F(x_t;\xi_i)$$

- Update

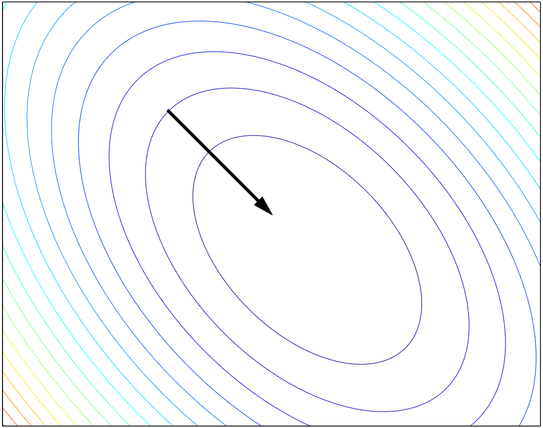
$$x_{t+1} = x_t - \alpha g_t$$



What everyone "knows" we should do

Obviously: get a lower-variance estimate of the gradient.

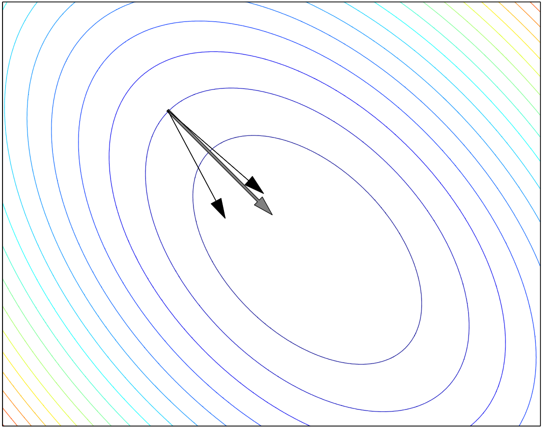
$$\text{Sample } g_{j,t} \text{ with } \mathbb{E}[g_{j,t}] = \nabla f(x_t) \quad \text{and use} \quad g_t = \frac{1}{m}\sum_{j=1}^{m} g_{j,t}$$



What everyone "knows" we should do

Obviously: get a lower-variance estimate of the gradient.

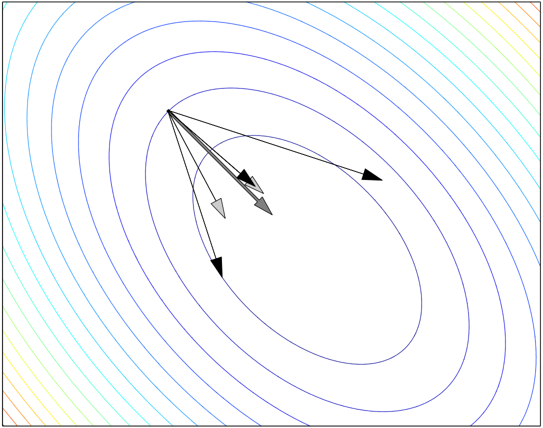
$$\text{Sample } g_{j,t} \quad \text{with } \mathbb{E}[g_{j,t}] = \nabla f(x_t) \quad \text{and use} \quad g_t = \frac{1}{m} \sum_{j=1}^{m} g_{j,t}$$



What everyone "knows" we should do

Obviously: get a lower-variance estimate of the gradient.

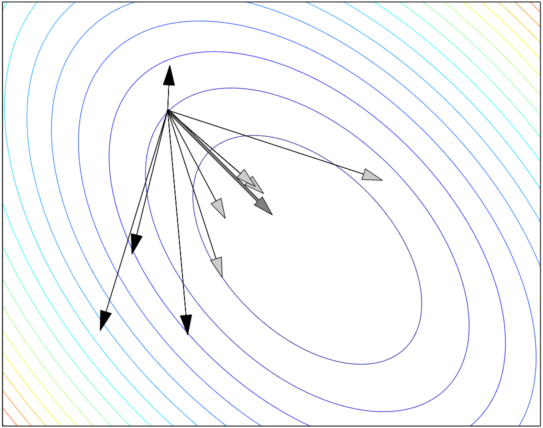
$$\text{Sample } g_{j,t} \ \text{ with } \mathbb{E}[g_{j,t}] = \nabla f(x_t) \quad \text{and use } \ g_t = \frac{1}{m}\sum_{j=1}^{m} g_{j,t}$$



What everyone “knows” we should do

Obviously: get a lower-variance estimate of the gradient.

$$\text{Sample } g_{j,t} \quad \text{with } \mathbb{E}[g_{j,t}] = \nabla f(x_t) \quad \text{and use} \quad g_t = \frac{1}{m}\sum_{j=1}^{m} g_{j,t}$$



Problem: only works for smooth functions.

Non-smooth problems we care about:

- Classification

$$F(x;\xi) = F(x;(a,b)) = \left[1 - bx^\top a\right]_+$$

- Robust regression

$$F(x;(a,b)) = \left|b - x^\top a\right|$$

Non-smooth problems we care about:

- Classification

$$F(x;\xi) = F(x;(a,b)) = \left[1 - bx^\top a\right]_+$$

- Robust regression

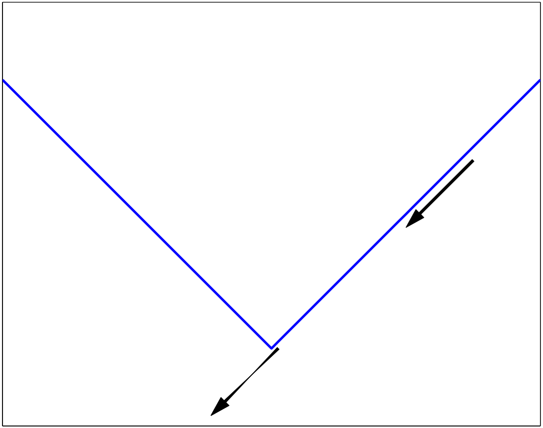
$$F(x;(a,b)) = \left|b - x^\top a\right|$$

- Structured prediction (ranking, parsing, learning matchings)

$$F(x;\{\xi,\nu\}) = \max_{\widehat{\nu}\in\mathcal{V}} \left[L(\nu,\widehat{\nu}) + x^\top \Phi(\xi,\widehat{\nu}) - x^\top \Phi(\xi,\nu)\right]$$

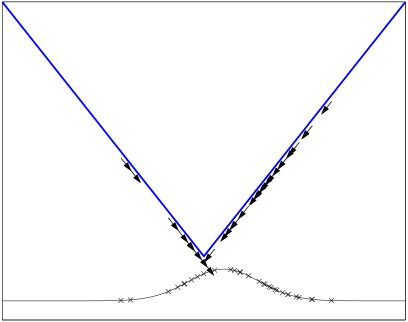
Difficulties of non-smooth

Intuition: Gradient is poor indicator of global structure



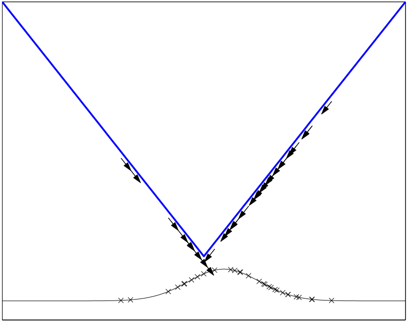
Better global estimators

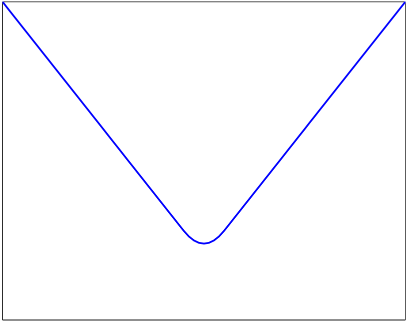
Idea: Ask for subgradients from multiple points



Better global estimators

Idea: Ask for subgradients from multiple points





The algorithm

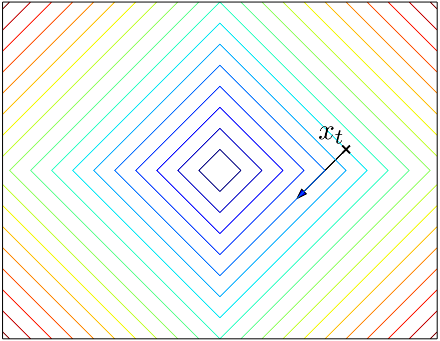
Normal approach: sample ξ at random,

$$g_{j,t} \in \partial F(x_t; \xi).$$

Our approach: add noise to x

$$g_{j,t} \in \partial F(x_t + u_t Z_j; \xi)$$

Decrease magnitude u_t over time



The algorithm

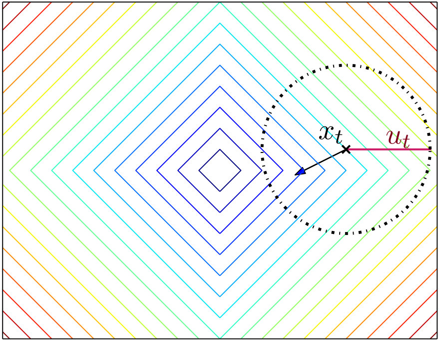
Normal approach: sample ξ at random,

$$g_{j,t} \in \partial F(x_t; \xi).$$

Our approach: add noise to x

$$g_{j,t} \in \partial F(x_t + u_t Z_j; \xi)$$

Decrease magnitude u_t over time



The algorithm

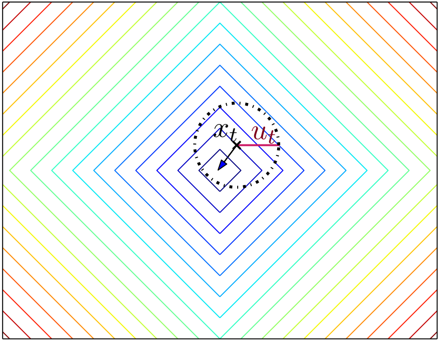
Normal approach: sample ξ at random,

$$g_{j,t} \in \partial F(x_t; \xi).$$

Our approach: add noise to x

$$g_{j,t} \in \partial F(x_t + u_t Z_j; \xi)$$

Decrease magnitude u_t over time



Algorithm

Generalization of accelerated gradient methods (Nesterov 1983, Tseng 2008, Lan 2010). Have query point and exploration point

Algorithm

Generalization of accelerated gradient methods (Nesterov 1983, Tseng 2008, Lan 2010). Have query point and exploration point

I. Get query point and gradients:

$$y_t = (1 - \theta_t) x_t + \theta_t z_t$$

Sample $\xi_{j,t}$ and $Z_{j,t}$, compute gradient approximation

$$g_t = \frac{1}{m} \sum_{j=1}^{m} g_{j,t}, \quad g_{j,t} \in \partial F(y_t + u_t Z_{j,t}; \xi_{j,t})$$

Algorithm

Generalization of accelerated gradient methods (Nesterov 1983, Tseng 2008, Lan 2010). Have query point and exploration point

I. Get query point and gradients:

$$y_t = (1-\theta_t)x_t + \theta_t z_t$$

Sample $\xi_{j,t}$ and $Z_{j,t}$, compute gradient approximation

$$g_t = \frac{1}{m}\sum_{j=1}^{m} g_{j,t}, \quad g_{j,t} \in \partial F(y_t + u_t Z_{j,t}; \xi_{j,t})$$

II. Solve for exploration point

$$z_{t+1} = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ \underbrace{\sum_{\tau=0}^{t} \frac{1}{\theta_\tau} \left[\langle g_\tau, x \rangle \right]}_{\text{Approximate } f} + \underbrace{\frac{1}{2\alpha_t} \|x\|_2^2}_{\text{Regularize}} \right\}$$

Algorithm

Generalization of accelerated gradient methods (Nesterov 1983, Tseng 2008, Lan 2010). Have query point and exploration point

I. Get query point and gradients:

$$y_t = (1 - \theta_t)x_t + \theta_t z_t$$

Sample $\xi_{j,t}$ and $Z_{j,t}$, compute gradient approximation

$$g_t = \frac{1}{m}\sum_{j=1}^{m} g_{j,t}, \quad g_{j,t} \in \partial F(y_t + u_t Z_{j,t}; \xi_{j,t})$$

II. Solve for exploration point

$$z_{t+1} = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ \underbrace{\sum_{\tau=0}^{t} \frac{1}{\theta_\tau}\left[\, \langle g_\tau, x \rangle \,\right]}_{\text{Approximate } f} + \underbrace{\frac{1}{2\alpha_t}\, \|x\|_2^2}_{\text{Regularize}} \right\}$$

III. Interpolate

$$x_{t+1} = (1 - \theta_t)x_t + \theta_t z_{t+1}$$

Theoretical Results

Objective:

$$\underset{x \in \mathcal{X}}{\text{minimize}} \; f(x) \quad \text{where } f(x) = \mathbb{E}[F(x;\xi)]$$

using m gradient samples for stochastic gradients.

Theoretical Results

Objective:

$$\underset{x \in \mathcal{X}}{\text{minimize}} \; f(x) \quad \text{where } f(x) = \mathbb{E}[F(x;\xi)]$$

using m gradient samples for stochastic gradients.

Non-strongly convex objectives:

$$f(x_T) - f(x^*) = \mathcal{O}\left(\frac{C}{T} + \frac{1}{\sqrt{Tm}}\right)$$

Theoretical Results

Objective:

$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f(x) \quad \text{where } f(x) = \mathbb{E}[F(x;\xi)]$$

using m gradient samples for stochastic gradients.

Non-strongly convex objectives:

$$f(x_T) - f(x^*) = \mathcal{O}\left(\frac{C}{T} + \frac{1}{\sqrt{Tm}}\right)$$

λ-strongly convex objectives:

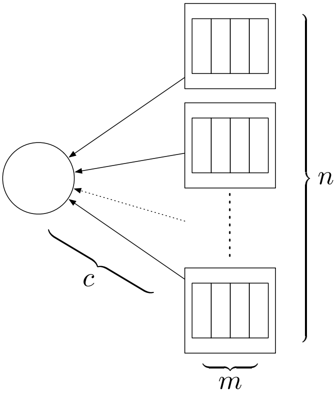
$$f(x_T) - f(x^*) = \mathcal{O}\left(\frac{C}{T^2} + \frac{1}{\lambda Tm}\right)$$

A few remarks on distributing

Convergence rate:

$$f(x_T) - f(x^*) = \mathcal{O}\left(\frac{1}{T} + \frac{1}{\sqrt{Tm}}\right)$$

- If communication is expensive, use larger batch sizes m:
  (a) Communication cost is c
  (b) n computers with batch size m
  (c) S total update steps



A few remarks on distributing

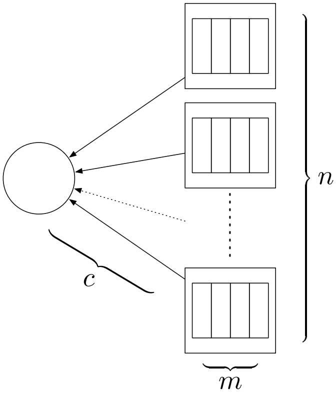
Convergence rate:

$$f(x_T) - f(x^*) = \mathcal{O}\left(\frac{1}{T} + \frac{1}{\sqrt{Tm}}\right)$$

- If communication is expensive, use larger batch sizes m:
  (a) Communication cost is c
  (b) n computers with batch size m
  (c) S total update steps

Backsolve: after $T = S(m+c)$ units of time, error is

$$\mathcal{O}\left(\frac{m+c}{T} + \frac{1}{\sqrt{Tn}} \cdot \sqrt{\frac{m+c}{m}}\right)$$

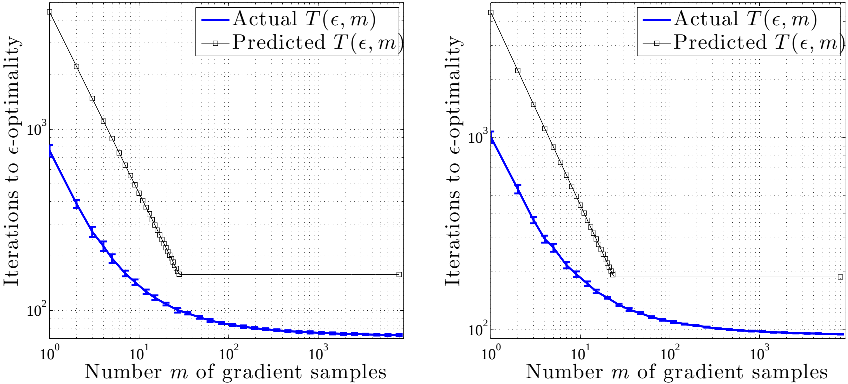


Experimental results

Iteration complexity simulations

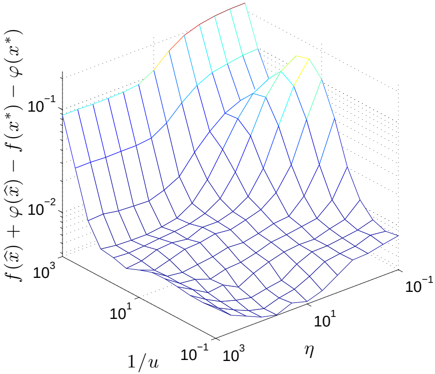
Define $T(\epsilon, m) = \min\{t \in \mathbb{N} \mid f(x_t) - f(x^*) \leq \epsilon\}$, solve robust regression problem

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} \left|a_i^\top x - b_i\right| = \frac{1}{n}\left\|Ax - b\right\|_1$$



Robustness to stepsize and smoothing

- Two parameters: smoothing parameter u, stepsize η



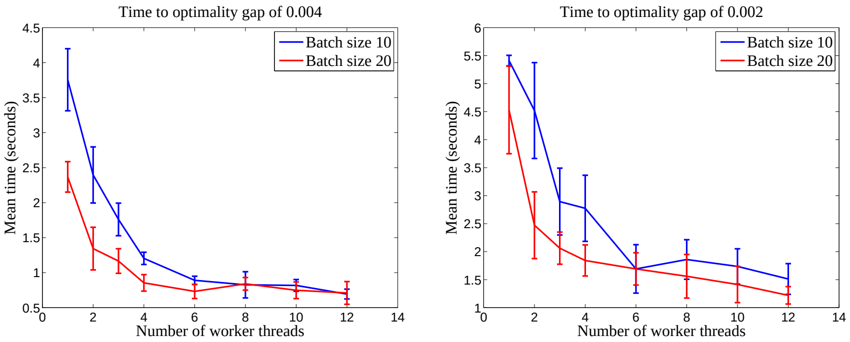
Plot: optimality gap after 2000 iterations on synthetic SVM problem

$$f(x) + \varphi(x) := \frac{1}{n} \sum_{i=1}^{n} \left[1 - \xi_i^\top x\right]_+ + \frac{\lambda}{2} \|x\|_2^2$$

Text Classification

Reuter's RCV1 dataset, time to ϵ-optimal solution for

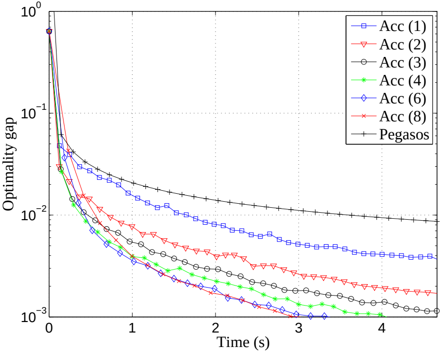
$$\frac{1}{n}\sum_{i=1}^{n}\left[1-\xi_i^\top x\right]_+ + \frac{\lambda}{2}\|x\|_2^2$$



Text Classification

Reuter's RCV1 dataset, optimization speed for minimizing

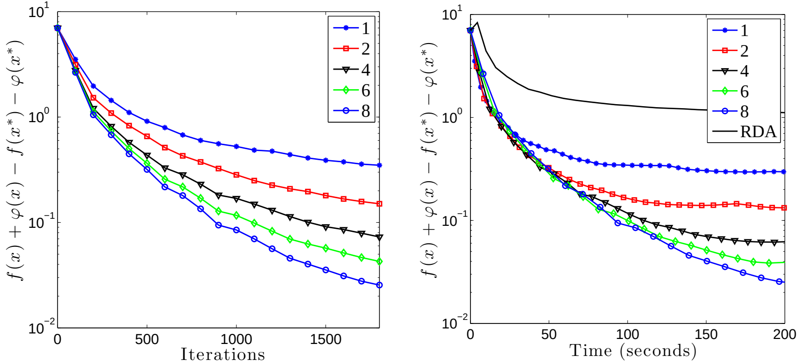
$$\frac{1}{n}\sum_{i=1}^{n}\left[1-\xi_i^\top x\right]_+ + \frac{\lambda}{2}\|x\|_2^2$$



Parsing

Penn Treebank dataset, learning weights for a hypergraph parser (here x is a sentence, $y \in \mathcal{V}$ is a parse tree)

$$\frac{1}{n}\sum_{i=1}^{n} \max_{\widehat{\nu}\in\mathcal{V}} \Big[L(\nu_i, \widehat{\nu}) + x^\top \left(\Phi(\xi_i, \widehat{\nu}) - \Phi(\xi_i, \nu_i) \right) \Big] + \frac{\lambda}{2} \|x\|_2^2 \, .$$

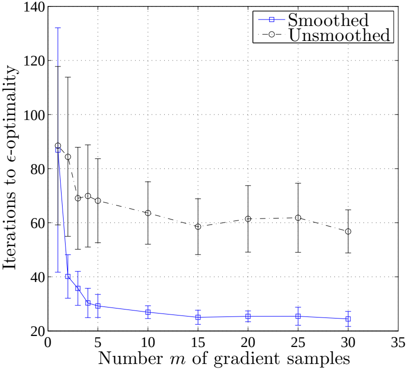


Is smoothing necessary?

Solve multiple-median problem

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \|x - \xi_i\|_1 ,$$

$\xi_i \in \{-1, 1\}^d$. Compare standard stochastic gradient:



Discussion

- Randomized smoothing allows
  - Stationary points of non-convex non-smooth problems
  - Optimal solutions in zero-order problems (including non-smooth)
  - Parallelization for non-smooth problems

Discussion

- Randomized smoothing allows
  - Stationary points of non-convex non-smooth problems
  - Optimal solutions in zero-order problems (including non-smooth)
  - Parallelization for non-smooth problems
- Current experiments in consultation with Google for large-scale parsing/translation tasks
- Open questions: non-stochastic optimality guarantees? True zero-order optimization?

Discussion

- Randomized smoothing allows
  - Stationary points of non-convex non-smooth problems
  - Optimal solutions in zero-order problems (including non-smooth)
  - Parallelization for non-smooth problems
- Current experiments in consultation with Google for large-scale parsing/translation tasks
- Open questions: non-stochastic optimality guarantees? True zero-order optimization?

Thanks!

Discussion

- Randomized smoothing allows
  - Stationary points of non-convex non-smooth problems
  - Optimal solutions in zero-order problems (including non-smooth)
  - Parallelization for non-smooth problems
- Current experiments in consultation with Google for large-scale parsing/translation tasks
- Open questions: non-stochastic optimality guarantees? True zero-order optimization?

References:

- Randomized Smoothing for Stochastic Optimization (D., Bartlett, Wainwright). *SIAM Journal on Optimization*, 22(2), pages 674–701.
- Optimal rates for zero-order convex optimization: the power of two function evaluations (D., Jordan, Wainwright, Wibisono). *arXiv:1312.2139 [math.OC]*.

Available on my webpage (`http://web.stanford.edu/~jduchi`)