

Understanding Non-convex Optimization

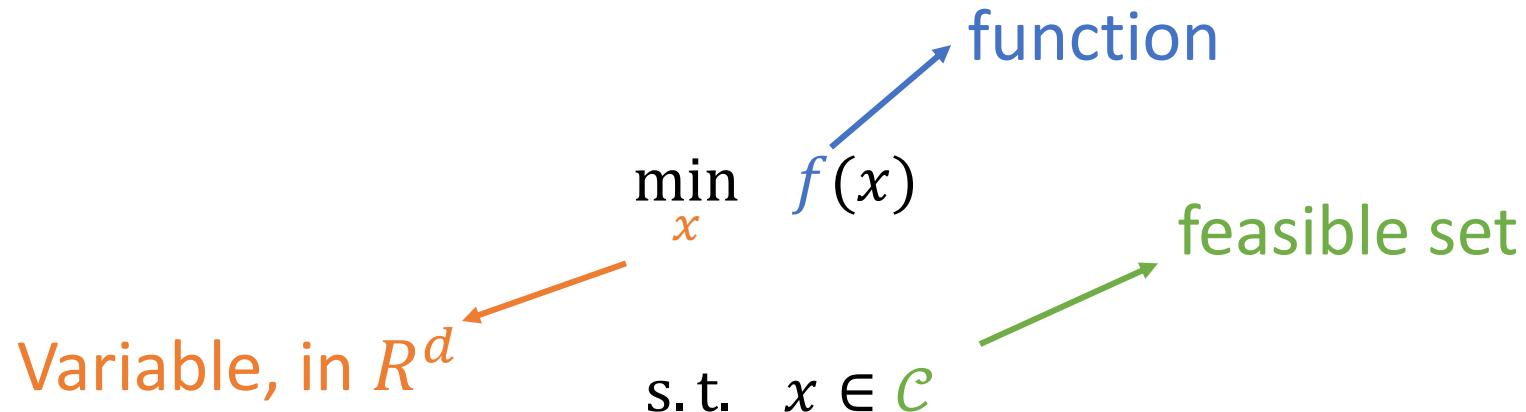
Sujay Sanghavi

UT Austin

Praneeth Netrapalli

Microsoft Research

Optimization



- In general too hard
- Convex optimization $\longleftrightarrow f()$ is a convex function, \mathcal{C} is convex set
- But “today’s problems”, and this tutorial, are non-convex
 - Our focus: non-convex problems that arise in machine learning

Outline of Tutorial

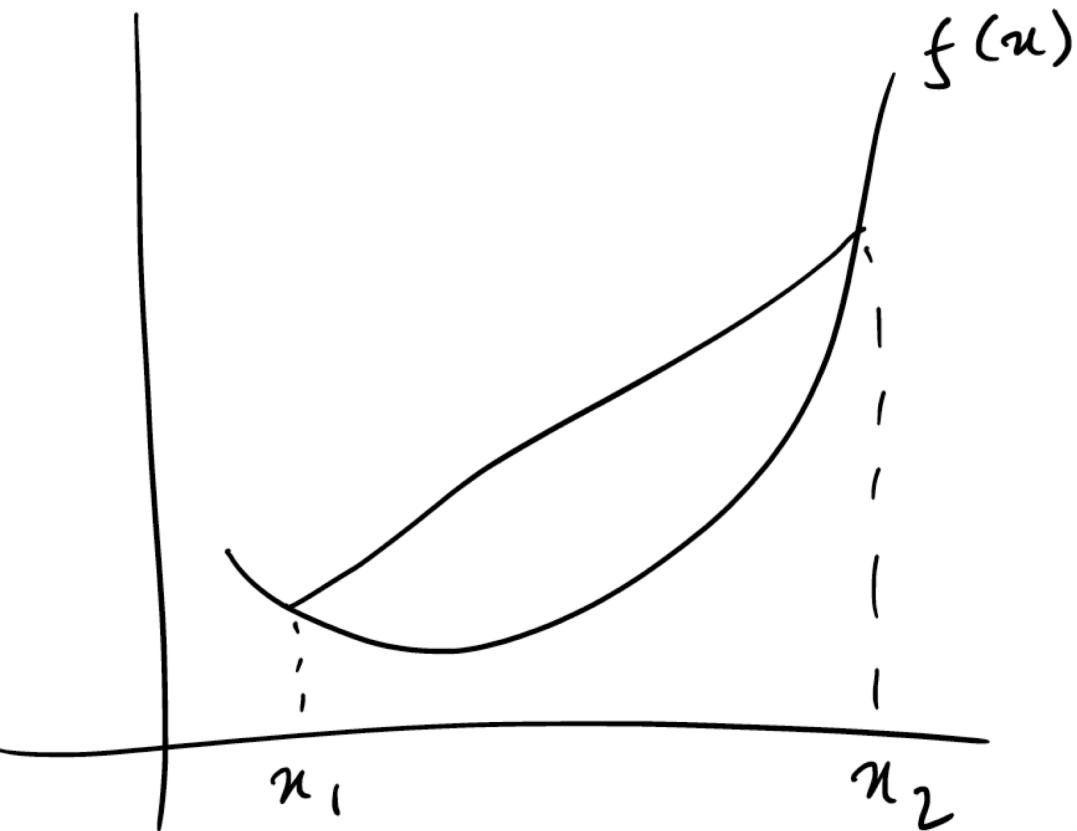
Part I (algorithms & background)

- Convex optimization (brief overview)
- Nonconvex optimization

Part II

- Example applications of nonconvex optimization
- Open directions

Convex Functions



Convex functions “lie below the line”

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

If $\nabla f(x)$ exists, convex f “lies above local linear approximation”

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) \quad \forall y$$

- GENERAL enough to capture/model many problems
- SPECIAL enough to have **fast** “off the shelf” algorithms

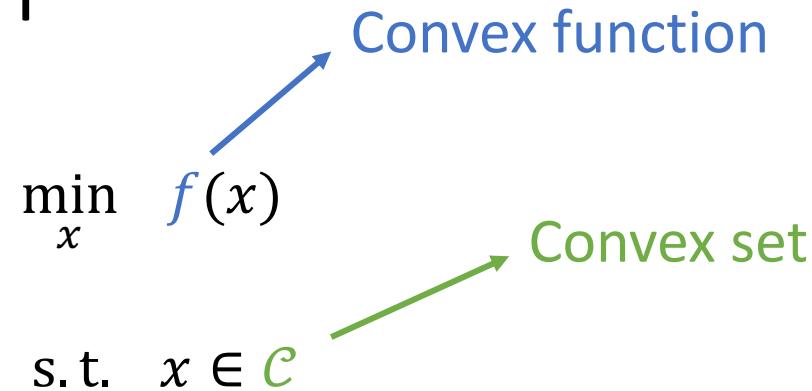
low-order polynomial complexity in dimension d

Convex Optimization

$$\begin{array}{ll} \min_x & f(x) \\ \text{s. t.} & x \in \mathcal{C} \end{array}$$

Convex function

Convex set

A diagram illustrating a convex optimization problem. It shows the mathematical formulation: $\min_x f(x)$ subject to $x \in \mathcal{C}$. Two arrows point from the text labels "Convex function" and "Convex set" to the corresponding parts of the equation: "Convex function" points to $f(x)$, and "Convex set" points to \mathcal{C} .

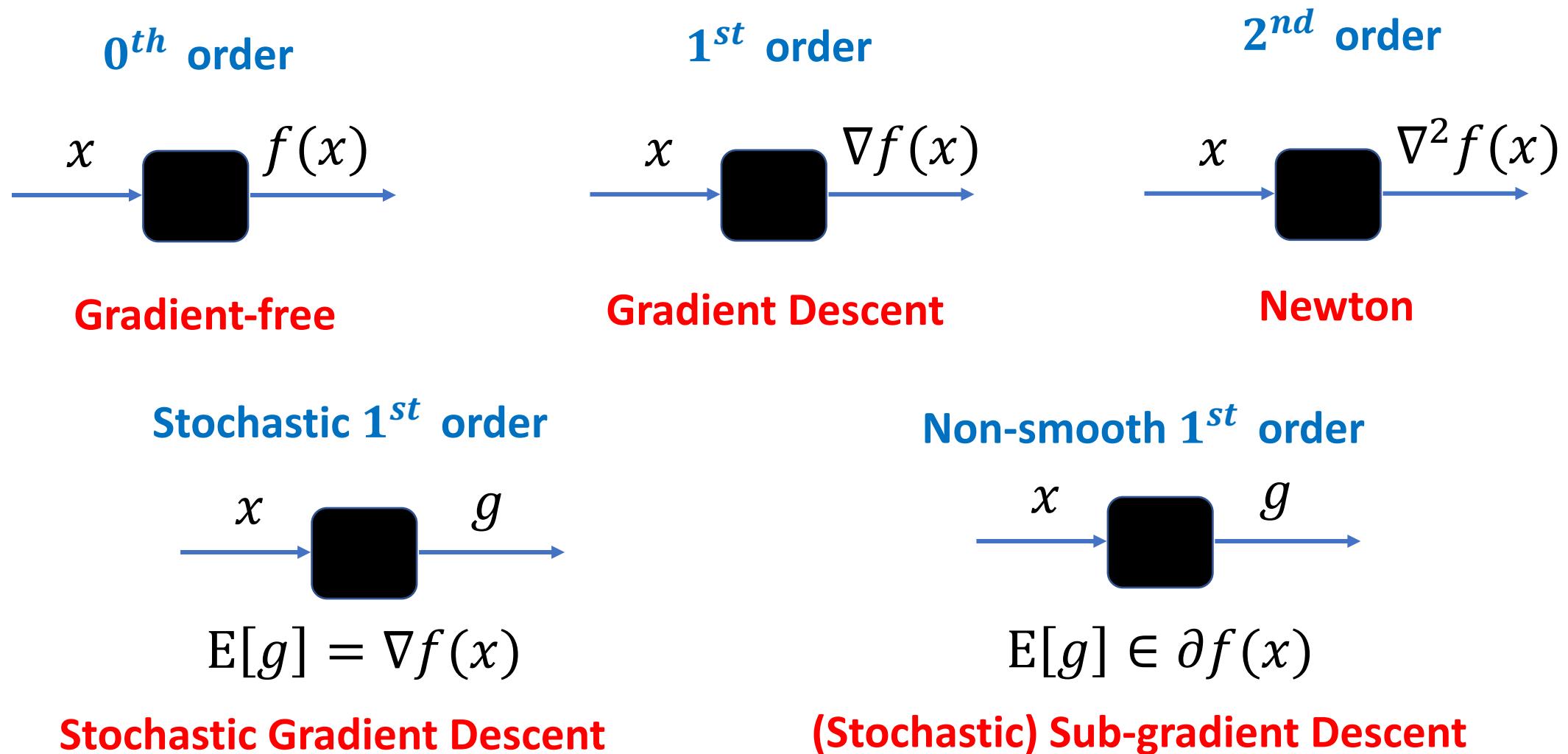
- **Key property of convex optimization:**

Local optimality  Global optimality

Proof: if x is local (but not global) opt and x^* is global, then moving from x to x^* strictly decreases f . This violates local optimality of x

- **All general-purpose methods search for locally optimal solutions**

“Black box / Oracle view” of Optimization



Gradients

For any **smooth** function,

- $-\nabla f(x)$ is a **local descent direction** at x
- x^* locally optimum  $\nabla f(x^*) = 0$

$\nabla f(x)$ exists everywhere

For convex functions, this means enough to search for **0-gradient points**

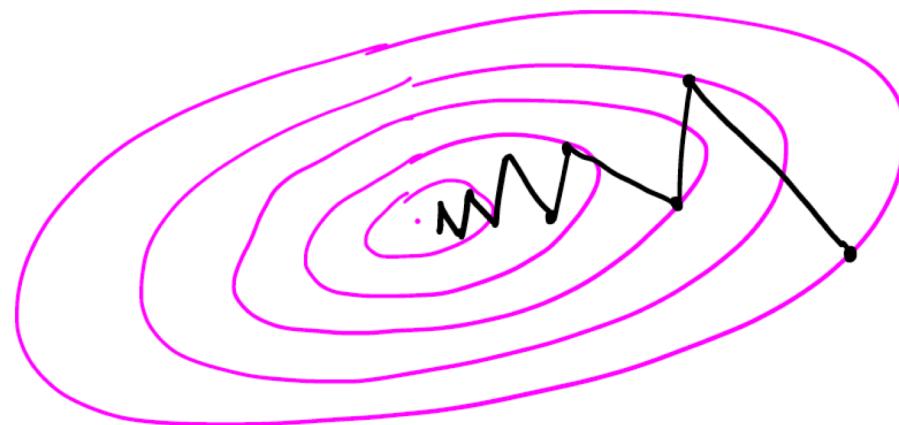
$$x^* \text{ globally optimum} \iff \nabla f(x^*) = 0$$

Gradient Descent

An **iterative** method to solve $\min_x f(x)$

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

next iterate current iterate step size current gradient



Gradient Descent

- “Off-the shelf” method that can be applied to “any” problem
 - Only need to be able to calculate gradients (i.e. 1st order oracle)
- “Only” parameters are the step sizes η_t
 - but choosing them correctly is crucially important
 - If too small, convergence too slow. If too big, divergence / oscillation
 - The more we know about f , the better we can make this choice

Gradient Descent Convergence

$f(\cdot)$ has **β -Lipschitz gradients** if

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\| \quad \forall x, y$$

β is a uniform bound

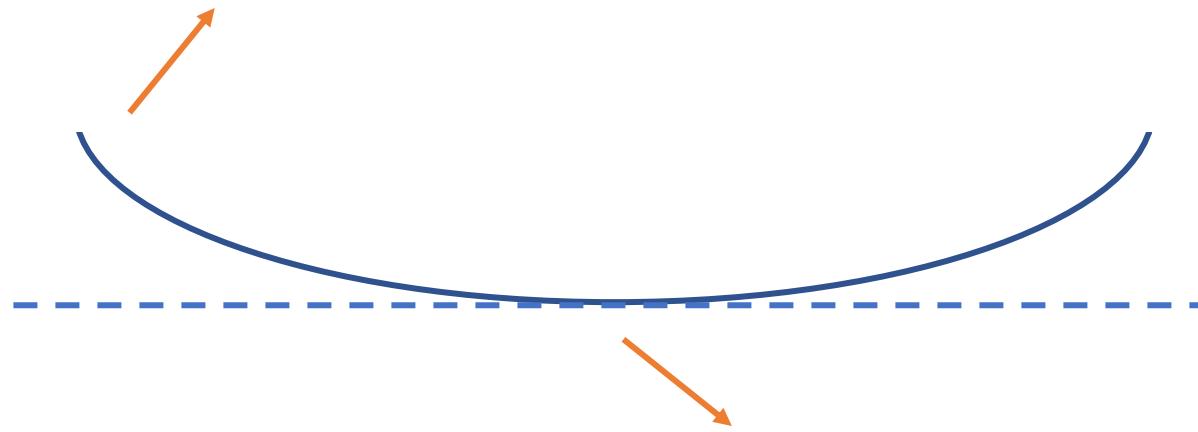
- In this case, fixed $\eta_t = 1/\beta$ gives

$$f(x_T) - f(x^*) \leq \epsilon \quad \text{once} \quad T \sim \Theta\left(\frac{1}{\epsilon}\right)$$

- This however is **not tight**: there exists a different method that is faster but only uses 1st first order oracle ...

Gradient Descent Convergence

Very steep so cannot
use big η



Very flat near x^* so small η
makes it slow

Gradient Descent Convergence

$f(\cdot)$ is **α -strongly convex** if

“cannot be too flat”

$$\|\nabla f(x) - \nabla f(y)\| \geq \alpha \|x - y\| \quad \forall x, y$$

- Now GD with fixed $\eta_t = 1/\beta$ gives

$$f(x_T) - f(x^*) \leq \epsilon \quad \text{in} \quad T \sim O\left(\frac{\log(\frac{1}{\epsilon})}{-\log(1 - \frac{\alpha}{\beta})}\right)$$



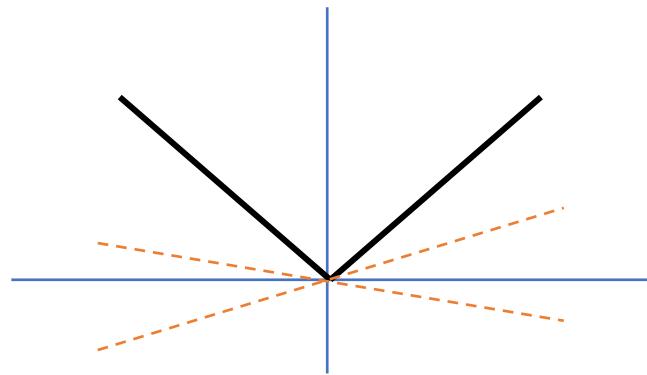
exponentially faster than
the only smooth case

Non-smooth functions

Sub-gradient of (convex) f at x is a set of “gradient like” vectors

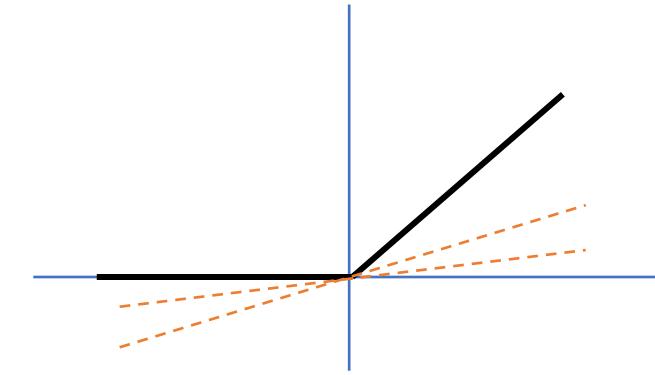
$$\{ g \in \mathbb{R}^d : f(y) \geq f(x) + g^T(y - x) \quad \forall y \}$$

Eg: $|x|$



$$\partial f(0) = \{u: -1 \leq u \leq 1\}$$

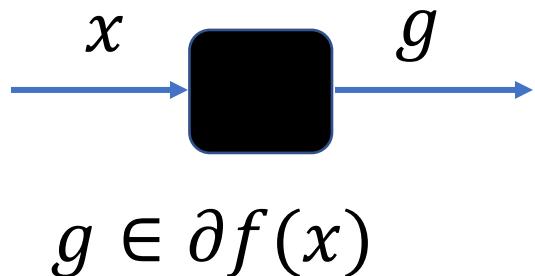
$\text{relu}(x) = \max(x, 0)$



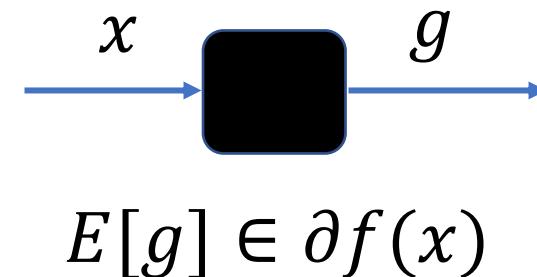
$$\partial f(0) = \{u: 0 \leq u \leq 1\}$$

Sub-gradient Descent

Non-smooth 1st order



**Stochastic Non-smooth
1st order**

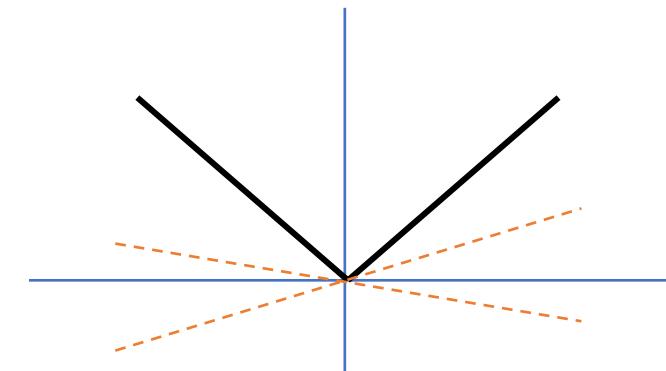


Sub-gradient Descent

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

An orange arrow points from the term $\eta_t g_t$ in the update equation to the expression $g_t \in \partial f(x_t)$.

$$g_t \in \partial f(x_t)$$

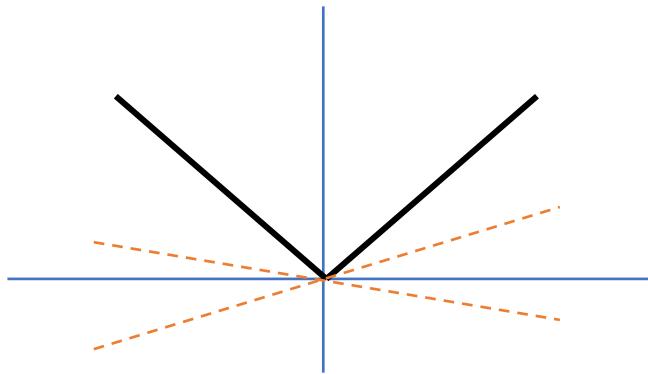


Sub-gradient Descent

- Now **cannot use fixed step size** in general

η_t has to $\rightarrow 0$, else oscillations possible

E.g. for $f(x) = |x|$, subgradient $g = \text{sign}(x)$ for $x \neq 0$



$$x_+ = x - \eta \text{sign}(x)$$

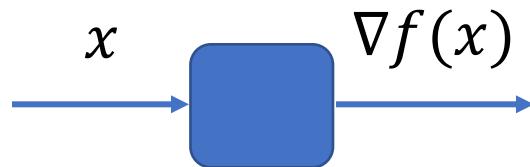
will oscillate between a and $a - \eta$
for some a that depends on initial point

Convergence slower than gradient descent because step size forced to decay:

$$f(x_T) - f(x^*) \leq \epsilon \text{ in } T \sim O(1/\epsilon^2)$$

Stochastic Gradient Descent (SGD)

Gradient Descent



But sometimes even calculating the gradient may be hard

- Large # terms

$$\min_x \frac{1}{n} \sum_{i=1}^n f_i(x)$$

n is large

- High dimensionality

$$\min_x f(x_1, \dots, x_d)$$

d is large

Stochastic Gradient Descent (SGD)

Lowering the complexity of gradient calculation via sub-sampling the sum of terms

$$x_+ = x - \eta \left(\frac{1}{n} \sum_{i=1}^n \nabla f_i(x) \right) \longrightarrow x_+ = x - \eta \left(\frac{1}{|B|} \sum_{i \in B} \nabla f_i(x) \right)$$

B is uniform subset from [n]. This is **mini-batch SGD**.

$$x_+ = x - \eta \left(\sum_{j=1}^d \frac{\partial f}{\partial x_j} e_j \right) \longrightarrow x_+ = x - \eta \left(\frac{d}{|S|} \sum_{j \in S} \frac{\partial f}{\partial x_j} e_j \right)$$

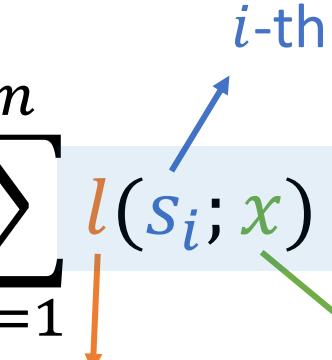
S is uniform subset from [n]. This is **block coordinate descent**.

Mini-batch SGD

Many ML problems can be reduced to optimization problems of the form

$$\min_x \frac{1}{n} \sum_{i=1}^n l(s_i; x)$$

i-th sample
loss function *weights*



$$\nabla f(x) = \frac{1}{n} \sum_{i=1}^n \nabla l(s_i; x)$$

“Batch or full” gradient

$$g = \frac{1}{|B|} \sum_{i \in B} \nabla l(s_i; x)$$

“Mini-batch” gradient

SGD and its Convergence Rate

g is a **Noisy Unbiased (sub) Gradient (NUS)** of $f(\cdot)$ at x if $E[g|x] \in \partial f(x)$

In all of the cases we have seen so far, g is NUS

SGD:

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

Suppose $\text{Var}(\|g_t\| | x_t) \leq G_t^2$

Theorem:

$$E[f(x_{\text{BEST}}^{(T)})] - f^* \leq \frac{R^2 + \sum_{t \leq T} \eta_t^2 G_t^2}{2 \sum_{t \leq T} \eta_t}$$

Where

$$R = |x_0 - x^*|$$

SGD and its Convergence Rate

$$\mathbb{E}[f^{(T)}] - f^* \leq \frac{R^2 + \sum_{t \leq T} \eta_t^2 G_t^2}{2 \sum_{t \leq T} \eta_t}$$

Idea 1: **Fixed mini-batch size** at every step $g_t = \frac{1}{|B|} \sum_{i \in B} \nabla f_i(x_t)$

$\Rightarrow G_t^2 = G^2$  B does not depend on t

1(a) Fixed step size $\eta_t = \eta$ gives $\mathbb{E}[f^{(T)}] - f^* \leq \frac{R^2}{\eta T} + \eta G^2$

Convergence to a "ball" around the optimum at rate $1/T$

SGD and its Convergence Rate

* Convergence to optimum $\Rightarrow \eta_t$ **has to** decay with time

1(b) Best choice for trading off between two error terms: $\eta_t \sim \frac{1}{\sqrt{t}}$

Gives $E[f^{(T)}] - f^* \leq O((R+G)/\sqrt{T})$ convergence rate

- much slower compared to the $O(1/T)$ rate of (full) gradient descent

- dependence on n captured by G - it is $O(\sqrt{n/|B|})$

SGD and its Convergence Rate

Fixed mini-batch size means fixed variance, which forces decaying step size and hence slow convergence

	Lipschitz f	Strongly Convex f
Full GD	$O(1/T)$	$O(c^T)$
Fixed-size mini-batch SGD	$O(1/\sqrt{T})$	$O(1/T)$

Variance reduction: keep step size η_t constant but decrease variance G_t as t increases

- (a) By increasing size of mini-batch for some of the steps
- (b) Via memory

Variance reduction in SGD

$$\min_x \frac{1}{n} \sum_{i=1}^n f_i(x)$$

Full GD: $x_{t+1} = x_t - \eta_t \frac{1}{n} \sum_i \nabla f_i(x_t)$ O(n) computations per iteration

SGD: $x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t)$ O(1) computation per iteration

(but many more iterations)
random index

Stochastic Average Gradient (Schmidt, Le Roux, Bach)

- Maintain $g_1^{(k)}, \dots, g_n^{(k)}$
- Initialize with one pass over all samples: $g_i^{(0)} = \nabla f_i(x^{(0)})$
- At step t , pick i_t randomly **(Update g 's lazily)**

current estimate for $\nabla f_n(x^{(k)})$

For the problem

$$\min_x \frac{1}{n} \sum_{i=1}^n f_i(x)$$

$$g_{i_t}^{(t)} = \nabla f_{i_t}(x^{(t-1)})$$
$$g_j^{(t)} = g_j^{(t-1)} \quad \text{for } j \neq i_t$$

- Update $x^{(k)} = x^{(k-1)} - \eta_k \frac{1}{n} \sum_{i=1}^n g_i^{(k)}$

Stochastic Average Gradient (SAG)

- Memory efficient implementation:

$$\frac{1}{n} \sum_{i=1}^n g_i^{(k)} = \frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n} + \frac{1}{n} \sum_{i=1}^n g_i^{(k-1)}$$

$$a^{(k)} = \frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n} + a^{(k-1)}$$

- $a^{(0)}$ just accumulates $\nabla f_i(x^{(0)})$'s

Stochastic Average Gradient (SAG)

NOTE: SAG not an unbiased stochastic gradient method

- Bias = $\frac{1}{n} \sum_i \nabla f_i(x^{(k)}) - \frac{1}{n} \sum_i g_i^{(k)}$
- As $k \rightarrow \infty$, $x^{(k)} \rightarrow x^*$, so Bias $\rightarrow 0$ and variance $\rightarrow 0$ as well !

However, proving this is quite involved

Theorem: If each $\nabla f_i(\cdot)$ is β -Lipschitz, SAG with fixed step size has

$$E[f^{(T)}] - f^* \leq \frac{cn}{T} [f^{(0)} - f^* + \beta \|x^{(0)} - x^*\|]$$

$O\left(\frac{1}{T}\right)$ convergence rate !

Stochastic Average Gradient

Theorem: If each f_i is also α -strongly convex

$$\mathbb{E}[f^{(T)}] - f^* \leq \left(1 - \frac{c_0\alpha}{\beta}\right)^T c_1 R$$

$c_0 < 1$, depends on n

Linear convergence
for strongly convex ..

SAG thus fixes the shortcomings of SGD wrt dependence of error on T ...

... but it is biased and hence hard to prove extensions for variants like Proximal SGD ...

Stochastic Variance Reduced Gradient

SVRG (Johnson and Zhang):

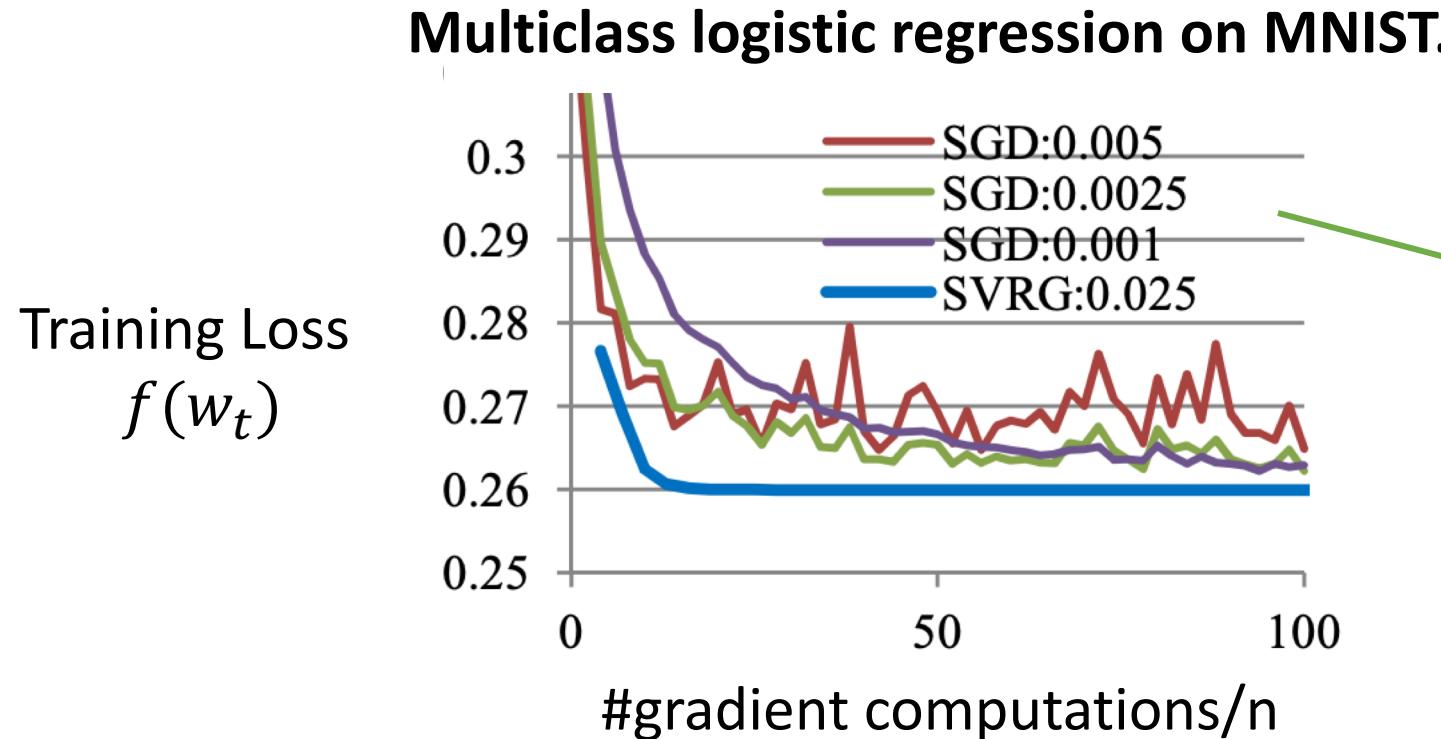
For epoch $t = 1, \dots, T$

\tilde{f}_t = full gradient

For iteration $k = 1, \dots, M$

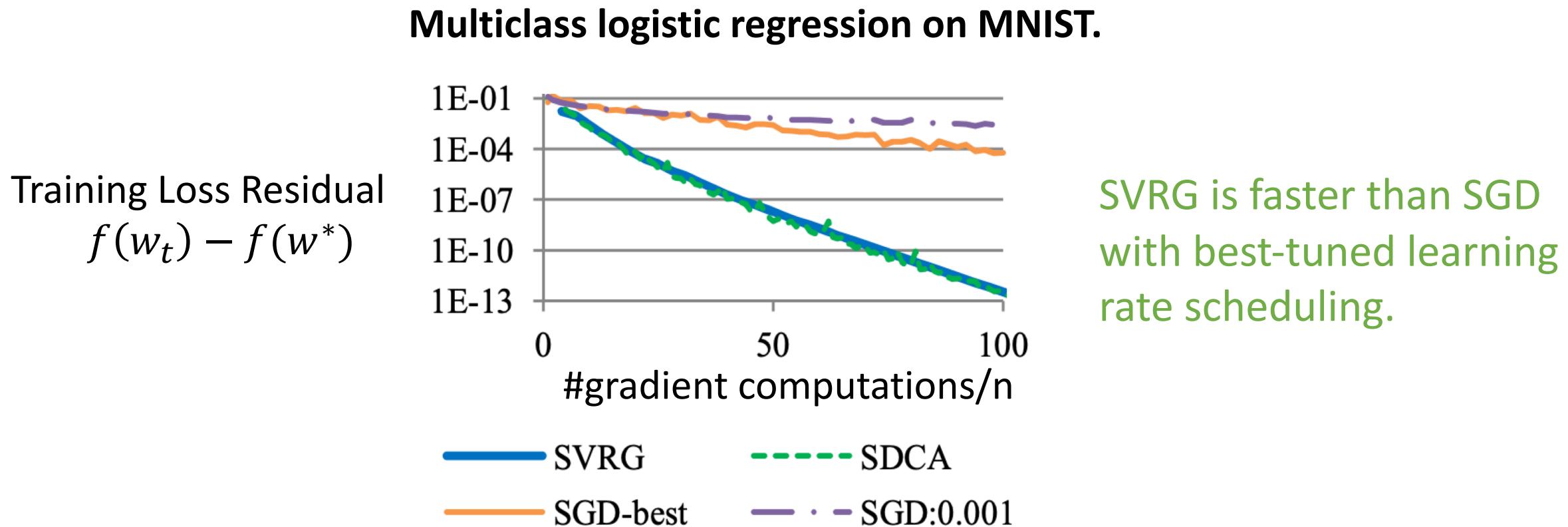
SAG-like updates

Experiments: SVRG vs SGD (Johnson and Zhang'2013)



Fixed learning rates:
SVRG can use larger
learning rate.

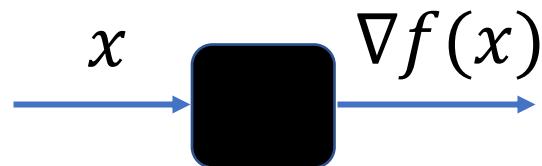
Experiments: SVRG vs SGD (Johnson and Zhang'2013)



Acceleration

Acceleration / Momentum

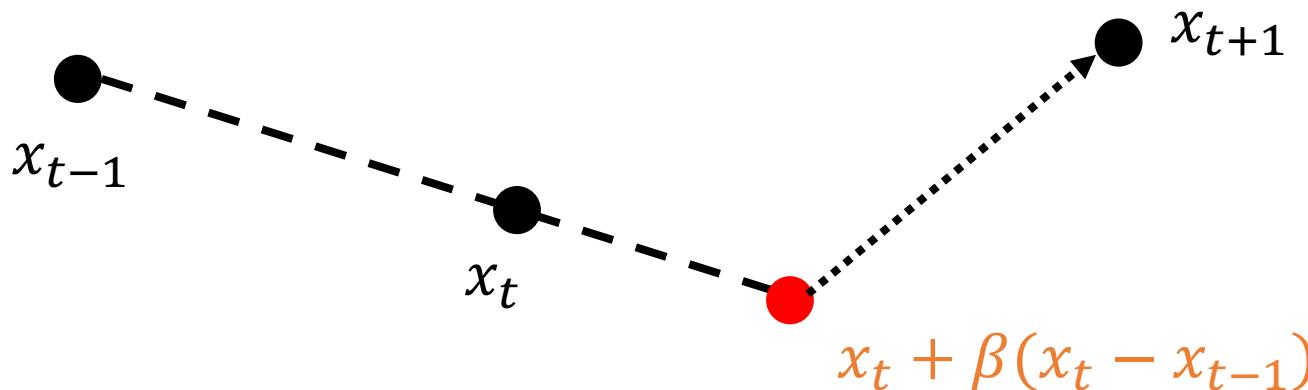
1st order



Gradient descent is one algorithm that uses 1st order oracle

But it is possible to get faster rates with this oracle ...

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



Can be viewed as a
discretization of

$$\ddot{x} + \tilde{\theta}\dot{x} + \nabla f(x) = 0$$

Convergence rates of Accelerated GD

	Smooth	Smooth & Strongly convex
Gradient Descent	$O\left(\frac{1}{\epsilon}\right)$	$O\left(\kappa \log\left(\frac{1}{\epsilon}\right)\right)$
Nesterov's Accelerated Gradient	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$	$O\left(\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right)$

Comparison of the convergence rates

Algorithm	Convergence rate
Subgradient	$O\left(\frac{1}{\epsilon^2}\right)$
ISTA	$O\left(\frac{1}{\epsilon}\right)$
FISTA	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$

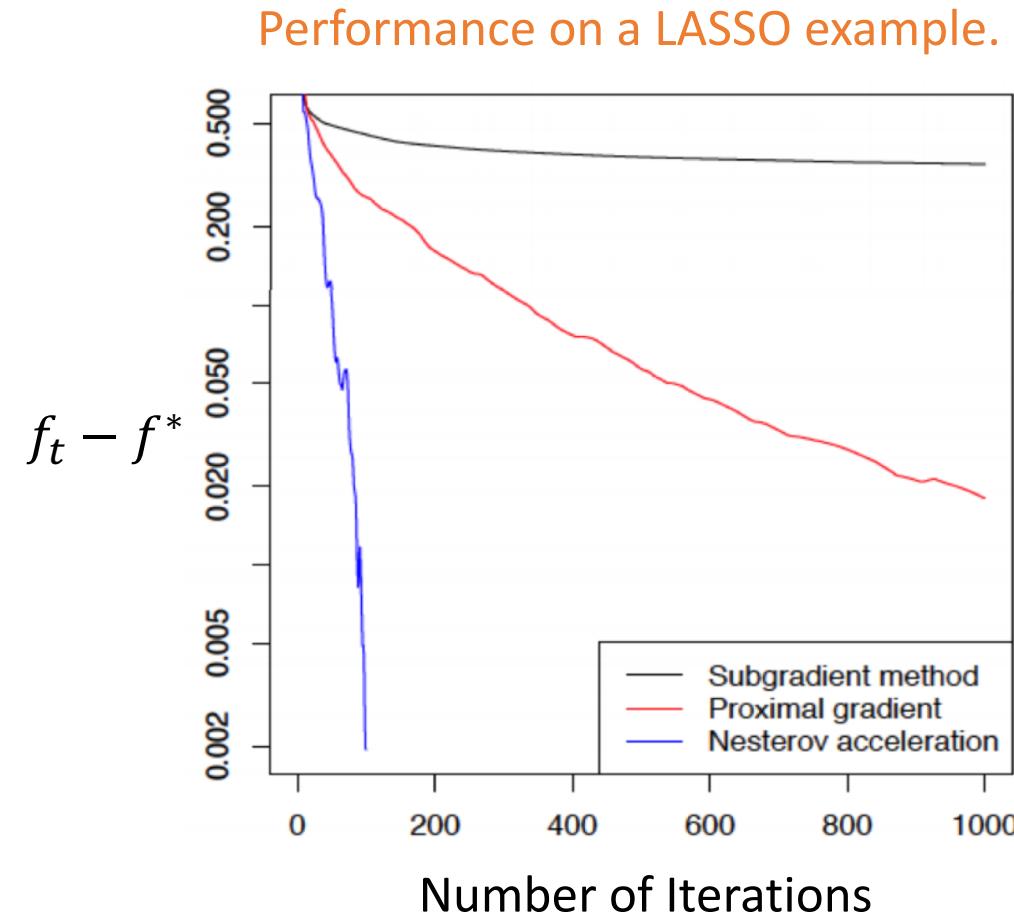


Figure taken from Tibshirani's lecture notes: <http://www.stat.cmu.edu/~ryantibs/convexopt-S15/scribes/08-prox-grad-scribed.pdf>

Summary of convex background

Nonconvex optimization

Problem: $\min_x f(x)$ $f(\cdot)$: nonconvex function

Applications: Deep learning, compressed sensing, matrix completion, dictionary learning, nonnegative matrix factorization, ...

Challenge: NP-hard in the worst case

Gradient descent (GD) [Cauchy 1847]

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

Question

How does it perform for non-convex functions?

Answer

Converges to **first order stationary points**

Definition

ϵ -First order stationary point (ϵ -FOSP) : $\|\nabla f(x)\| \leq \epsilon$

Concretely

ϵ -FOSP in $O(\epsilon^{-2})$ iterations
[Folklore]

GD for smooth functions

- **Assumption:** $f(\cdot)$ is L -smooth $\stackrel{\text{def}}{=} \nabla f(\cdot)$ is L -Lipschitz

$$\|\nabla f(x) - \nabla f(y)\| \leq L \cdot \|x - y\|, \forall x, y.$$

This implies

$$f(y) \leq \underbrace{f(x) + \langle \nabla f(x), y - x \rangle}_{\text{1}^{\text{st}} \text{ order Taylor expansion}} + \frac{L}{2} \|x - y\|^2, \forall x, y$$

1st order Taylor expansion

Quadratic upper bound

Alternate view of GD

$$\eta \leq \frac{1}{L}$$

$$x_{t+1} = \operatorname{argmin}_x f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{1}{2\eta} \|x - x_t\|^2$$

Quadratic upper bound

$$\begin{aligned} f(x_{t+1}) &\leq \min_x f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{1}{2\eta} \|x - x_t\|^2 \\ &= f(x_t) - \frac{\eta}{2} \|\nabla f(x_t)\|^2 \end{aligned}$$

Telescoping: $f(x_T) \leq f(x_0) - \frac{\eta}{2} \sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2$

Stochastic gradient descent (SGD) [Robbins, Monro 1951]

$$x_{t+1} = x_t - \eta \hat{\nabla} f(x_t); \mathbb{E}[\hat{\nabla} f(x_t)] = \nabla f(x_t)$$

Question

How does it perform?

Answer

Converges to **first order stationary points**

Definition

ϵ -First order stationary point (ϵ -FOSP) : $\|\nabla f(x)\| \leq \epsilon$

Concretely

ϵ -FOSP in $O(\epsilon^{-4})$ iterations
vs $O(\epsilon^{-2})$ for GD

Proof of convergence rate of SGD

- **Assumption:** $f(\cdot)$ is L -smooth $\stackrel{\text{def}}{=} \nabla f(\cdot)$ is L -Lipschitz
- **Assumption:** $\mathbb{E} \left[\|\widehat{\nabla} f(x) - \nabla f(x)\|^2 \right] \leq \sigma^2$

$$\mathbb{E}[f(x_{t+1})] \leq \mathbb{E}[f(\mathbf{x}_t)] - \frac{\eta}{2} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] + \frac{\eta^2 L}{2} \sigma^2$$

$$\eta \sim 1/\sqrt{T}: \quad \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] \leq O\left(\sigma/\sqrt{T}\right)$$

Finite sum problems

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- $f_i(\cdot)$ = loss on i^{th} data point; L -smooth and nonconvex
 - E.g., $f_i(\cdot) = \ell(\phi(x, a_i), b_i)$; ℓ – loss function; ϕ – model.
- Usually computing $\nabla f_i(\cdot)$ takes same amount of time for each i

GD vs SGD

ϵ — desired accuracy
 n — # component functions

	# gradient calls per iteration	# iterations
GD	n	ϵ^{-2}
SGD	1	ϵ^{-4}

Can we get best of both?

- **Main observations:**
 - Can compute exact gradients when required
 - Convergence of SGD depends on noise variance σ
- **Main idea of SVRG:** [Johnson, Zhang 2013]; [Reddi, Hefny, Sra, Poczos, Smola 2016]; [Allen-Zhu, Hazan 2016]
 - Compute full gradient in the beginning; reference point x_0
 - At x_t ,
$$\widehat{\nabla}f(x_t) \stackrel{\text{def}}{=} \nabla f_i(x_t) - \nabla f_i(x_0) + \nabla f(x_0)$$

Two potential functions

$$\sigma^2 \stackrel{\text{def}}{=} \mathbb{E} \left[\|\widehat{\nabla} f(x_t) - \nabla f(x_t)\|^2 \right] \leq L \|x_0 - x_t\|^2$$

$$\mathbb{E}[f(x_{t+1})] - \mathbb{E}[f(x_t)] \leq -\frac{\eta}{2} \mathbb{E}[\|\nabla f(x_t)\|^2] + \frac{\eta^2 L}{2} \sigma^2$$

$$\|x_{t+1} - x_t\|^2 \leq \eta^2 \mathbb{E}[\|\nabla f(x_t)\|^2] + \eta^2 L \|x_0 - x_t\|^2$$

- For $t = 0$, $\|x_0 - x_t\|^2 = 0$.
- $\|x_0 - x_t\|^2$ can increase only if $\|\nabla f(x_t)\|^2$ is large.
- But if so, $\mathbb{E}[f(x_{t+1})]$ will decrease significantly.
- Careful combination of these two potential functions.

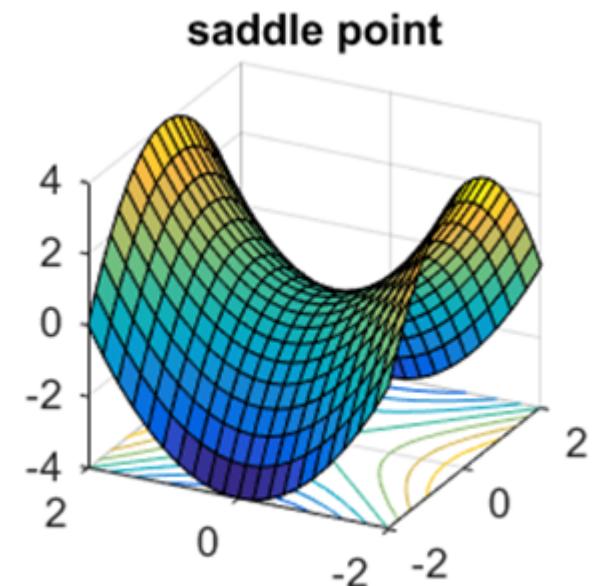
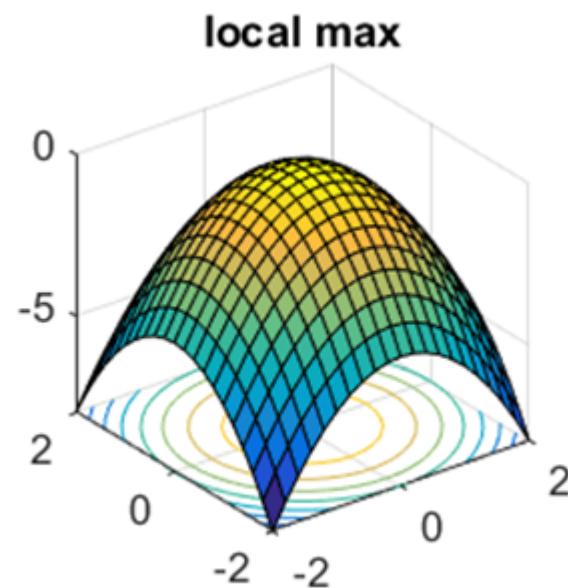
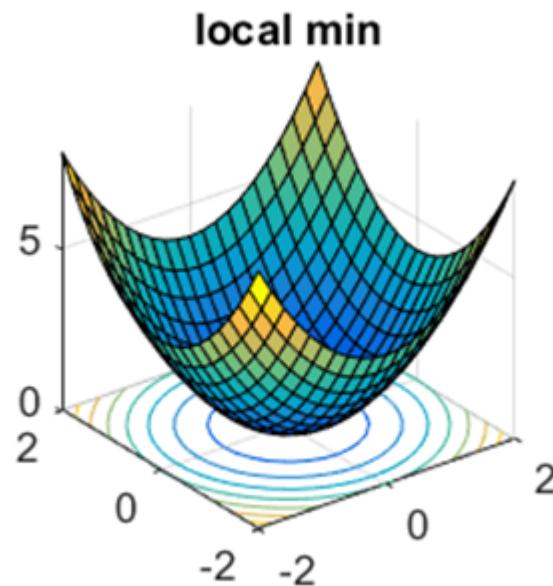
Comparison with GD/SGD

ϵ — desired accuracy
 n — # component functions

	# gradient calls per iteration	# iterations
GD	n	ϵ^{-2}
SGD	1	ϵ^{-4}
SVRG	1	$n^{2/3}\epsilon^{-2}$
SNVRG (Zhou, Xu, Gu 2018)	1	$n^{1/2}\epsilon^{-2} \wedge \epsilon^{-3}$
Spider (Fang, Li, Lin, Zhang 2018)	1	$n^{1/2}\epsilon^{-2} \wedge \epsilon^{-3}$

Other results: Ghadimi, Lan 2015; Allen-Zhu 2018

How do FOSPs look like?



Hessian PSD
 $\nabla^2 f(x) \geq 0$

Second order stationary
points (SOSP)

Hessian NSD
 $\nabla^2 f(x) \leq 0$

Hessian indefinite
 $\lambda_{\min}(\nabla^2 f(x)) \leq 0$
 $\lambda_{\max}(\nabla^2 f(x)) \geq 0$

FOSPs vs SOSPs in popular problems

- Very well studied
 - Matrix sensing [Bhojanapalli, Neyshabur, Srebro 2016]
 - Matrix completion [Ge, Lee, Ma 2016]
 - Robust PCA [Ge, Jin, Zheng 2017]
 - Tensor factorization [Ge, Huang, Jin, Yuan 2015]; [Ge, Ma 2017]
 - Smooth semidefinite programs [Boumal, Voroninski, Bandeira 2016]
 - Synchronization & community detection [Bandeira, Boumal, Voroninski 2016]; [Mei, Misiakiewicz, Montanari, Oliveira 2017]
 - Phase retrieval [Chen, Chi, Fan, Ma 2018]

Two major observations

- FOSPs: proliferation (exponential #) of saddle points
 - Recall FOSP $\triangleq \nabla f(x) = 0$
 - Gradient descent can get stuck near them
- SOSPs: not just local minima; as good as global minima
 - Recall SOSP $\triangleq \nabla f(x) = 0 \text{ & } \nabla^2 f(x) \succcurlyeq 0$

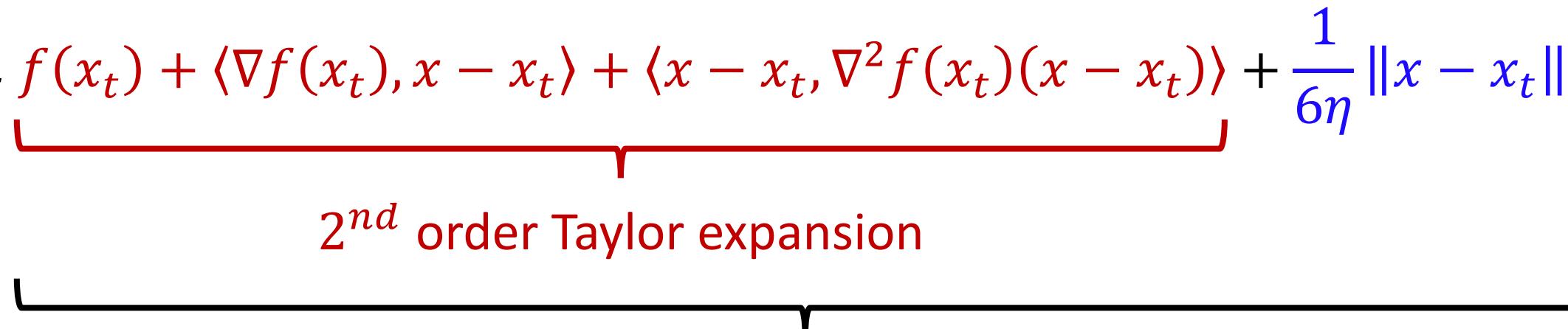
Upshot for these problems

1. FOSP not good enough
2. Finding SOSP sufficient

How to find SOSPs?

- Cubic regularization (CR) [Nesterov, Polyak 2006]

$$x_{t+1} = \operatorname{argmin}_x f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \langle x - x_t, \nabla^2 f(x_t)(x - x_t) \rangle + \frac{1}{6\eta} \|x - x_t\|^3$$


2nd order Taylor expansion
Cubic upper bound

- Contrast with GD: Minimizes quadratic upper bound

GD vs CR

	Guarantee	Per Iteration cost
CR	SOSP	Hessian computation
GD	FOSP	Gradient computation

- Hessian computation is not practical for large scale applications

Can we find SOSPs using first order (gradient) methods?

GD finds SOSPs with probability 1

- [Lee, Panageas, Piliouras, Simchowitz, Jordan, Recht 2017]
 $\text{Lebesguemeasure}(\{x_0: \text{GD from } x_0 \text{ converges to non SOSP}\}) = 0$
- However, time taken for convergence could be $\Omega(d)$ [Du, Jin, Lee, Jordan, Singh, Poczos 2017]

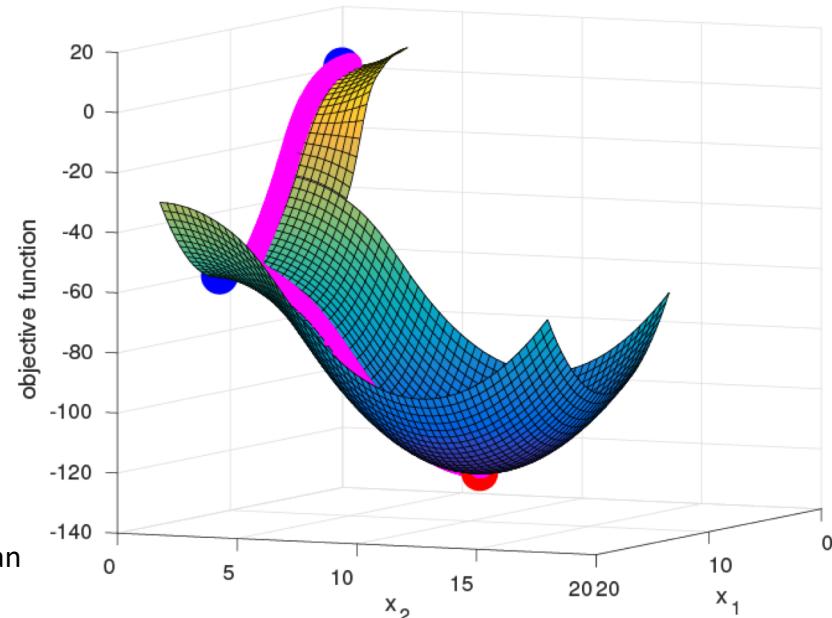


Image credit: Chi Jin & Mike Jordan

GD vs CR

	Guarantee	Per Iteration cost	# iterations
CR	SOSP	Hessian computation	$O(\epsilon^{-1.5})$
GD	SOSP	Gradient computation	$\Omega(d)$

- Convergence rate of GD too slow

Can we speed up GD for finding SOSPs?

Perturbation to the rescue!

Perturbed gradient descent (PGD)

1. **For** $t = 0, 1, \dots$ **do**
2. $x_{t+1} \leftarrow x_t - \eta \nabla f(x_t) + \xi_t$ where $\xi_t \sim \text{Unif}(B_0(\epsilon))$

	Guarantee	Per Iteration cost	# iterations
CR	SOSP	Hessian computation	$O(\epsilon^{-1.5})$
GD	SOSP	Gradient computation	$\Omega(d)$
PGD	SOSP	Gradient computation	$\tilde{O}(\epsilon^{-2})$

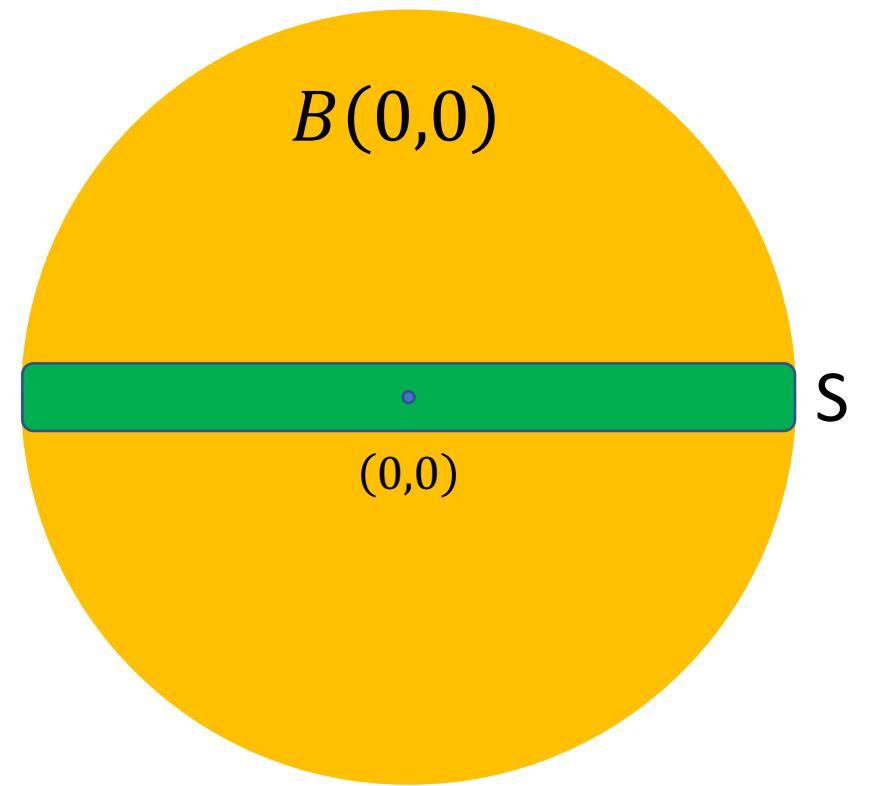
[Ge, Huang, Jin, Yuan 2015]; [Jin, Ge, Netrapalli, Kakade, Jordan 2017]

Main idea

- $S \stackrel{\text{def}}{=} \text{set of points around saddle point from where gradient descent does not escape quickly}$
- Escape $\stackrel{\text{def}}{=} \text{function value decreases significantly}$
- How much is $\text{Vol}(S)$?
- $\text{Vol}(S)$ small \Rightarrow perturbed GD escapes saddle points efficiently

Two dimensional quadratic case

- $f(x) = \frac{1}{2}x^\top \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} x$
- $\lambda_{\min}(H) = -1 < 0$
- $(0,0)$ is a saddle point
- **GD:** $x_{t+1} = \begin{bmatrix} 1 - \eta & 0 \\ 0 & 1 + \eta \end{bmatrix} x_t$
- S is a thin strip, $\text{Vol}(S)$ is small



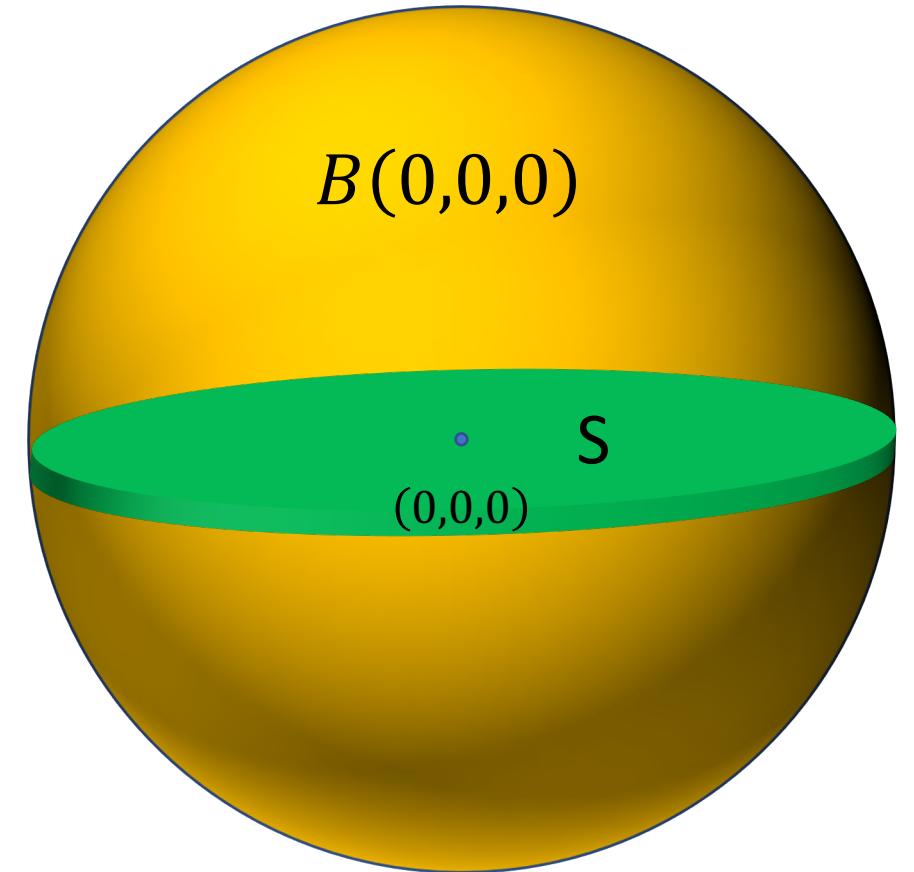
Three dimensional quadratic case

- $f(x) = \frac{1}{2}x^\top \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} x$

- $(0,0,0)$ is a saddle point

- **GD:** $x_{t+1} = \begin{bmatrix} 1 - \eta & 0 & 0 \\ 0 & 1 - \eta & 0 \\ 0 & 0 & 1 + \eta \end{bmatrix} x_t$

- S is a thin disc, $\text{Vol}(S)$ is small

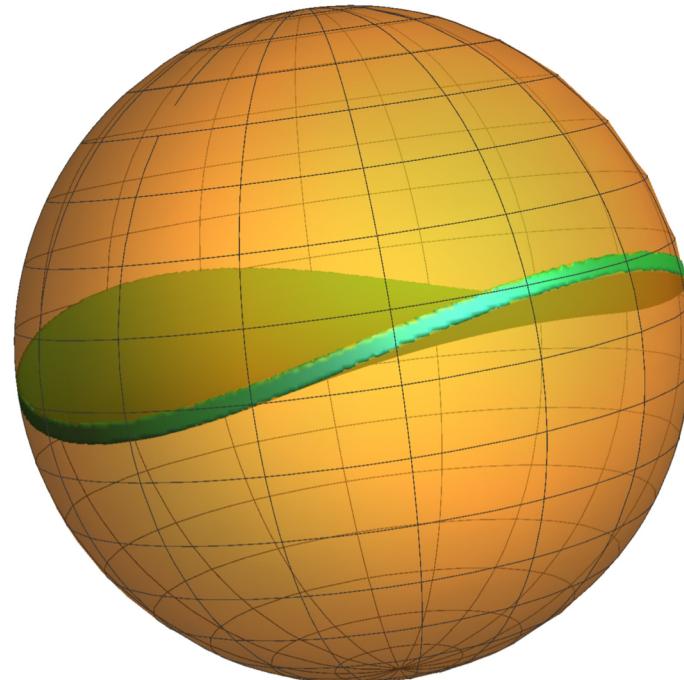


General case

Key technical lemma

$S \sim$ thin deformed disc

$\text{Vol}(S)$ is small



Can we accelerate nonconvex GD?

	Guarantee	Per Iteration cost	# iterations
CR	SOSP	Hessian computation	$O(\epsilon^{-1.5})$
GD	SOSP	Gradient computation	$\Omega(d)$
PGD	SOSP	Gradient computation	$\tilde{O}(\epsilon^{-2})$
PAGD	SOSP	Gradient computation	$O(\epsilon^{-1.75})$

[Agarwal, Allen-Zhu, Bullins, Hazan, Ma 2016]

[Carmon, Duchi, Hinder, Sidford 2016, 2017]

[Jin, Netrapalli, Jordan 2018]

Differential equation view of Accelerated GD

- AGD is a discretization of the following ODE [Polyak 1965]; [Su, Candes, Boyd 2015]

$$\ddot{x} + \tilde{\theta}\dot{x} + \nabla f(x) = 0$$

- Multiplying by \dot{x} and integrating from t_1 to t_2 gives us

$$f(x_{t_2}) + \frac{1}{2} \|\dot{x}_{t_2}\|^2 = f(x_{t_1}) + \frac{1}{2} \|\dot{x}_{t_1}\|^2 - \tilde{\theta} \int_{t_1}^{t_2} \|\dot{x}_t\|^2 dt$$

- Hamiltonian $f(x_t) + \frac{1}{2} \|\dot{x}_t\|^2$ decreases monotonically

After discretization

Iterate: x_t and velocity: $v_t := x_t - x_{t-1}$

- Hamiltonian $f(x_t) + \frac{1}{2\eta} \|v_t\|^2$ decreases monotonically if $f(\cdot)$ “*not too nonconvex*” between x_t and $x_t + v_t$
 - *too nonconvex* = *negative curvature*
 - Can increase if $f(\cdot)$ is “*too nonconvex*”
- If the function is “*too nonconvex*”, reset velocity or move in nonconvex direction – *negative curvature exploitation*

Can we accelerate nonconvex GD?

	Guarantee	Per Iteration cost	# iterations
CR	SOSP	Hessian computation	$O(\epsilon^{-1.5})$
GD	SOSP	Gradient computation	$\Omega(d)$
PGD	SOSP	Gradient computation	$\tilde{O}(\epsilon^{-2})$
PAGD	SOSP	Gradient computation	$O(\epsilon^{-1.75})$

[Agarwal, Allen-Zhu, Bullins, Hazan, Ma 2016]

[Carmon, Duchi, Hinder, Sidford 2016, 2017]

[Jin, Netrapalli, Jordan 2018]

Stochastic algorithms

Perturbed SGD (PSGD)

1. **For** $t = 0, 1, \dots$ **do**
2. $x_{t+1} \leftarrow x_t - \eta \hat{\nabla} f(x_t) + \xi_t$

- Convergence rate: $O(\epsilon^{-3.5})$ [Fang, Lin, Zhang 2019]
- $O(\epsilon^{-3})$ with variance reduction [Zhou, Xu, Gu 2018]; [Fang, Li, Lin, Zhang 2018]
- **Finite sum:** $O(\epsilon^{-3} \wedge \sqrt{n}\epsilon^{-2})$

Summary of nonconvex optimization

- Local optimality for nonconvex optimization
- For many problems, SOSPs are sufficient
- Ideas such as momentum, stochastic methods, variance reduction are useful in the nonconvex setting as well
- Complexity theory still not as mature as in convex optimization

Problems where SOSPs = global optima

- Matrix sensing [Bhojanapalli, Neyshabur, Srebro 2016]
- Matrix completion [Ge, Lee, Ma 2016]
- Robust PCA [Ge, Jin, Zheng 2017]
- Tensor factorization [Ge, Huang, Jin, Yuan 2015]; [Ge, Ma 2017]
- Smooth semidefinite programs [Boumal, Voroninski, Bandeira 2016]
- Synchronization & community detection [Bandeira, Boumal, Voroninski 2016]; [Mei, Misiakiewicz, Montanari, Oliveira 2017]
- Phase retrieval [Chen, Chi, Fan, Ma 2018]

Semi-definite programs (SDPs)

$$\min_{X \in R^{n \times n}} \langle C, X \rangle \quad s.t. \quad \begin{aligned} & \langle A_i, X \rangle = b_i, 1 \leq i \leq m \\ & X \geq 0 \end{aligned}$$

- Several applications
 - Clustering (max-cut)
 - Control
 - Sum-of-squares
 - ...
- Classical polynomial time solutions exist but can be slow
 - Interior-point methods
 - Ellipsoid method
 - Multiplicative weight update

Low rank solutions always exist!

- (Barvinok'95, Pataki'98): For **any** feasible SDP, at least one solution exists with rank
$$k^* \leq \sqrt{2m}$$
- In several applications $m \sim n$. So $k^* \ll n$.

Burer-Monteiro: Optimize in low rank space; iterations are fast!

Burer-Monteiro approach

$$\begin{aligned} & \min_{X \in R^{n \times n}} \langle C, X \rangle \\ s.t. \quad & \langle A_i, X \rangle = b_i; \ i = 1, \dots, m \\ & X \geq 0 \end{aligned}$$

$$\begin{aligned} & \min_{U \in R^{n \times k}} \langle C, UU^T \rangle \\ s.t. \quad & \langle A_i, UU^T \rangle = b_i \end{aligned}$$

$$k \sim \sqrt{m}$$

n^2 dimensional problem



nk dimensional problem

Burer-Monteiro approach

$$\begin{aligned} & \min_{X \in R^{n \times n}} \langle C, X \rangle \\ s.t. \quad & \langle A_i, X \rangle = b_i; \quad i = 1, \dots, m \\ & X \geq 0 \end{aligned}$$

$$\begin{aligned} & \min_{U \in R^{n \times k}} \langle C, UU^T \rangle \\ s.t. \quad & \langle A_i, UU^T \rangle = b_i \end{aligned}$$

$k \sim \sqrt{m}$

Penalty
Version

Penalty
parameter

$$\min_{U \in R^{n \times k}} f(U) = \langle C, UU^T \rangle + \mu \sum_i (\langle A_i, UU^T \rangle - b_i)^2$$

Nonconvex problem!

$$\min_{U \in R^{n \times k}} f(U) = \langle C, UU^T \rangle + \mu \sum_i (\langle A_i, UU^T \rangle - b_i)^2$$

Are SOSPs = global minima?

- In general, no [Bhojanapalli, Boumal, Jain, Netrapalli 2018]

Smoothed analysis

$$\min_{U \in R^{n \times k}} f(U) = \langle C + G, UU^T \rangle + \mu \sum_i (\langle A_i, UU^T \rangle - b_i)^2$$

- G : symmetric Gaussian matrix with $G_{ij} \sim N(0, \sigma^2)$

- If $k = \Omega(\sqrt{m \log 1/\epsilon})$ then with high probability

every ϵ SOSP = ϵ global optimum

[Boumal, Voroninski, Bandeira 2016]

[Bhojanapalli, Boumal, Jain, Netrapalli 2018]

Two key steps

$$\min_{U \in \mathbb{R}^{n \times k}} f(UU^\top) \quad f(\cdot) \text{ convex}$$

1. SOSP that is rank deficient is **global** optimum [Burer-Monteiro 2003]

U SOSP and $\sigma_k(U) = 0 \Rightarrow U$ is a global optimum

↳ k^{th} largest singular value of U

2. For perturbed SDPs, with probability 1, if $k \geq \sqrt{2m}$, then

all FOSPs have $\sigma_k(U) = 0$.

ϵ -FOSP $\Rightarrow \sigma_k(U)$ is small

$$\min_{U \in R^{n \times k}} f(U) = \langle C + \textcolor{red}{G}, UU^T \rangle + \mu \sum_i (\langle A_i, UU^T \rangle - b_i)^2$$

- Approximate FOSP: $\| (C + \textcolor{red}{G} + 2\mu \sum_i (\langle A_i, UU^\top \rangle - b_i) A_i) U \| \leq \epsilon$

$$\left\| C + \textcolor{red}{G} + 2\mu \sum_i (\langle A_i, UU^\top \rangle - b_i) A_i \right\|_U \leq \epsilon$$

The diagram illustrates the matrix dimensions. On the left, a large blue rectangle represents the matrix $C + \textcolor{red}{G} + 2\mu \sum_i (\langle A_i, UU^\top \rangle - b_i) A_i$, with vertical double bars indicating its height n . To its right is a smaller blue rectangle labeled U , with vertical double bars indicating its height n and horizontal double bars indicating its width k .

Aside: Lower bound on product of matrices

$$\left\| \begin{array}{c} H \\ \parallel \\ \text{blue rectangle} \end{array} \right\| \cdot \left\| \begin{array}{c} U \\ \parallel \\ n \downarrow \\ \text{blue rectangle} \\ k \rightarrow \end{array} \right\| \geq \sigma_{n-k+1}(H) \cdot \sigma_k(U)$$

$$\sigma_k(U) \leq \frac{\|HU\|}{\sigma_{n-k+1}(H)}$$

FOSP $\Rightarrow \sigma_k(U)$ is small

$\sigma_{n-k+1}(C + \textcolor{red}{G} + 2\mu \sum_i (\langle A_i, UU^\top \rangle - b_i) A_i)$ large



$\sigma_k(U)$ is small

$$\left\| C + \textcolor{red}{G} + 2\mu \sum_i (\langle A_i, UU^\top \rangle - b_i) A_i \right\|_F \leq \epsilon$$

The diagram illustrates the matrix norm of the expression. It shows two blue rectangular blocks representing matrices. The left block has height n and width k , indicated by a vertical double-headed arrow between its top and bottom edges. The right block has height n and width 1 , indicated by a vertical double-headed arrow between its top and bottom edges. The expression is enclosed in vertical double-line brackets, representing the Frobenius norm. To the right of the expression is a vertical double-headed arrow between the top and bottom edges of the rightmost block, labeled n . Below the rightmost block is a horizontal double-headed arrow between its left and right edges, labeled k . To the right of the entire expression is the inequality symbol $\leq \epsilon$.

Smallest singular values of Gaussian matrices

- $\sigma_i(G)$ denotes the i^{th} singular value of G .

$$\mathbb{P}[\sigma_n(G) = 0] = 0$$

- In general, $\sigma_{n-k}(G) \sim \frac{k}{\sqrt{n}}$.
- Can obtain large deviation bounds [Nguyen 2017]

$$\mathbb{P}\left[\sigma_{n-k}(G) < c \frac{k}{\sqrt{n}}\right] < \exp(-Ck^2 + k \log n)$$

- Can extend the above to $G + \textcolor{blue}{A}$ for any fixed matrix $\textcolor{blue}{A}$
- In this case, $\textcolor{red}{G} + C + 2\mu \sum_i (\langle A_i, UU^\top \rangle - b_i) A_i$

Summary

Several problems for which SOSPs = global optima

Convert a large convex problem into a smaller nonconvex problem

- E.g., Burer-Monteiro approach for solving SDPs
- Empirically, much faster than ellipsoid/interior point methods
- Open problem: Identify other problems which have this property

Alternating Minimization

Alternating Minimization

Applicable to a special class of problems:

those here variables can be split into two sets, i.e. $x = (u, v)$ such that

$$\min_u f(u, v) \quad \text{feasible / "easy" for fixed } v$$

$$\min_v f(u, v) \quad \text{feasible / "easy" for fixed } u$$

Alt-min: (initialize)

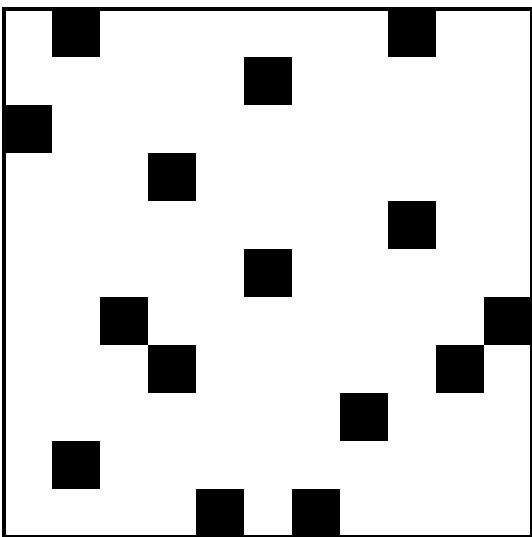
$$u_+ = \arg \min_u f(u, v)$$

$$v_+ = \arg \min_v f(u_+, v)$$

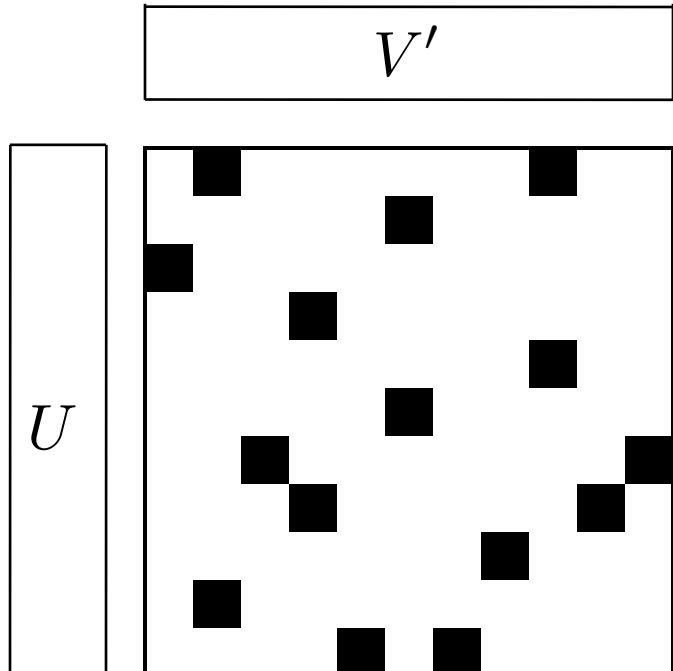
- “big” global steps, and can be much faster than “local” gradient descent
- No step size to be tuned / chosen

Matrix Completion

Find a low-rank matrix from a few (randomly sampled) elements



Matrix Completion



Find a low-rank matrix from a few (randomly sampled) elements

- Let Ω be set of sampled elements

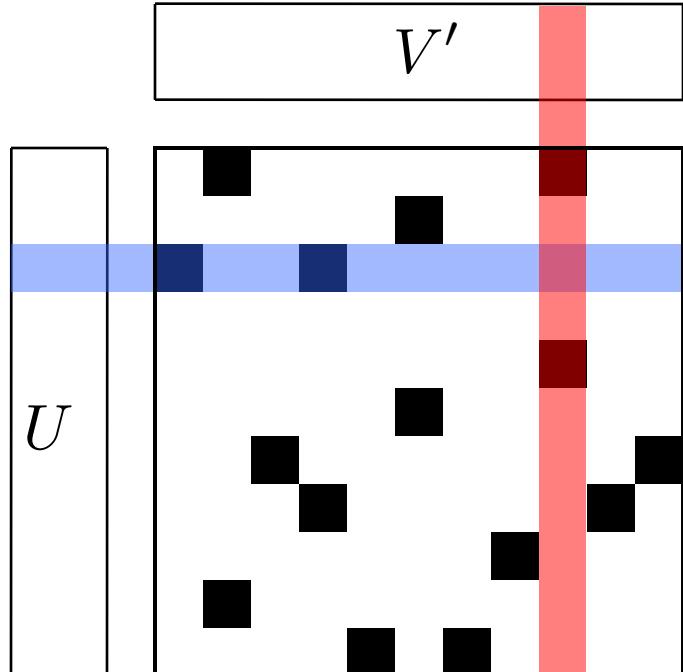
Alt min:

(1) Write as non-convex problem

$$\min_{U,V} \|\mathcal{P}_\Omega(M - UV')\|_F$$

(2) Alternately optimize U and V

AltMin for Matrix Completion



Naturally decouples into small least-squares problems

(a) For all i

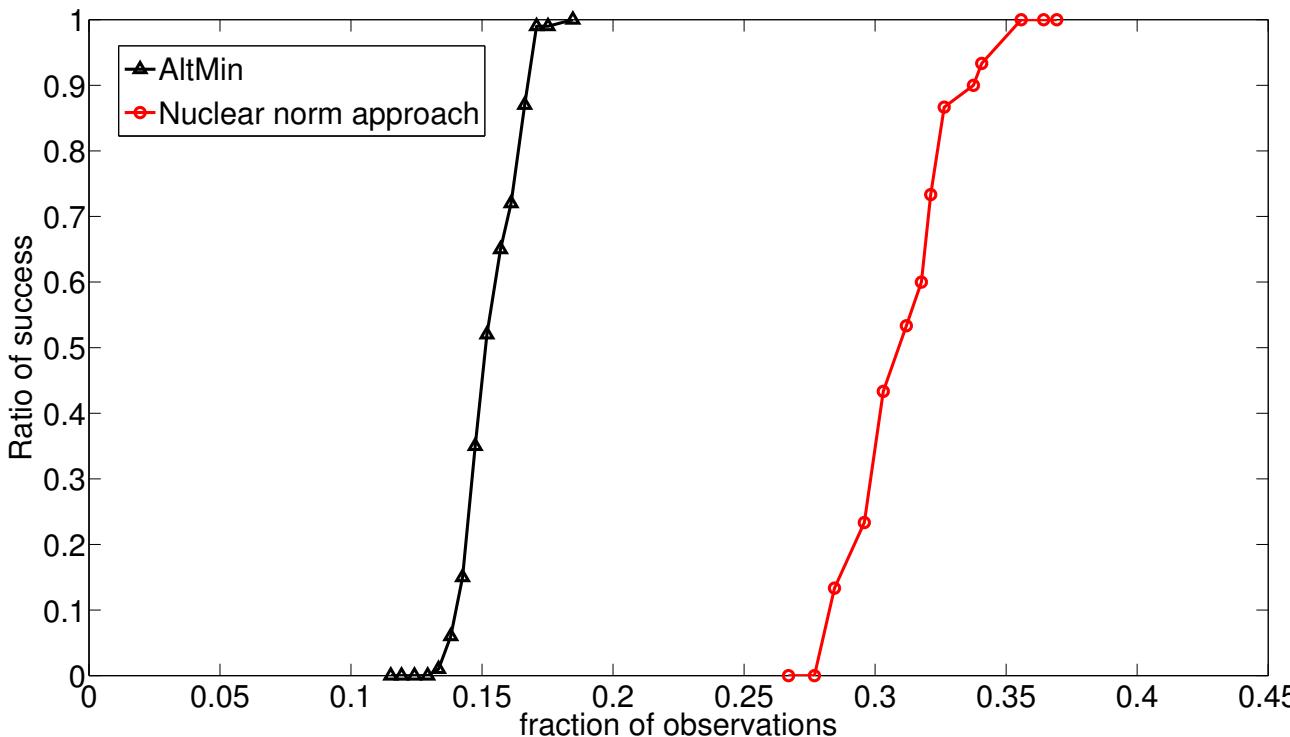
$$u_i \leftarrow \min_u \sum_{j:(i,j) \in \Omega} (m_{ij} - \langle u, v_j \rangle)^2$$

(b) For all j

$$v_j \leftarrow \min_v \sum_{i:(i,j) \in \Omega} (m_{ij} - \langle u_i, v \rangle)^2$$

Closed form, embarrassingly parallel

Matrix Completion



Empirically: AltMin needs **fewer samples** than convex methods like trace-norm minimization

- with memory as small as input and output
- very fast, parallel

Theoretically: both take $O(nr^2 \log n)$

randomly chosen samples for exact recovery of incoherent matrices

- w/ spectral initialization
- close to the lower bound of $\Omega(nr \log n)$

Mixed linear regression

Solve linear equations, except that each is either

$$y_i = \langle x_i, \beta_0^* \rangle \quad \text{or} \quad y_i = \langle x_i, \beta_1^* \rangle$$

Find β_1^*, β_0^* given $\{y_i, x_i\}$

Natural for modeling with latent classes

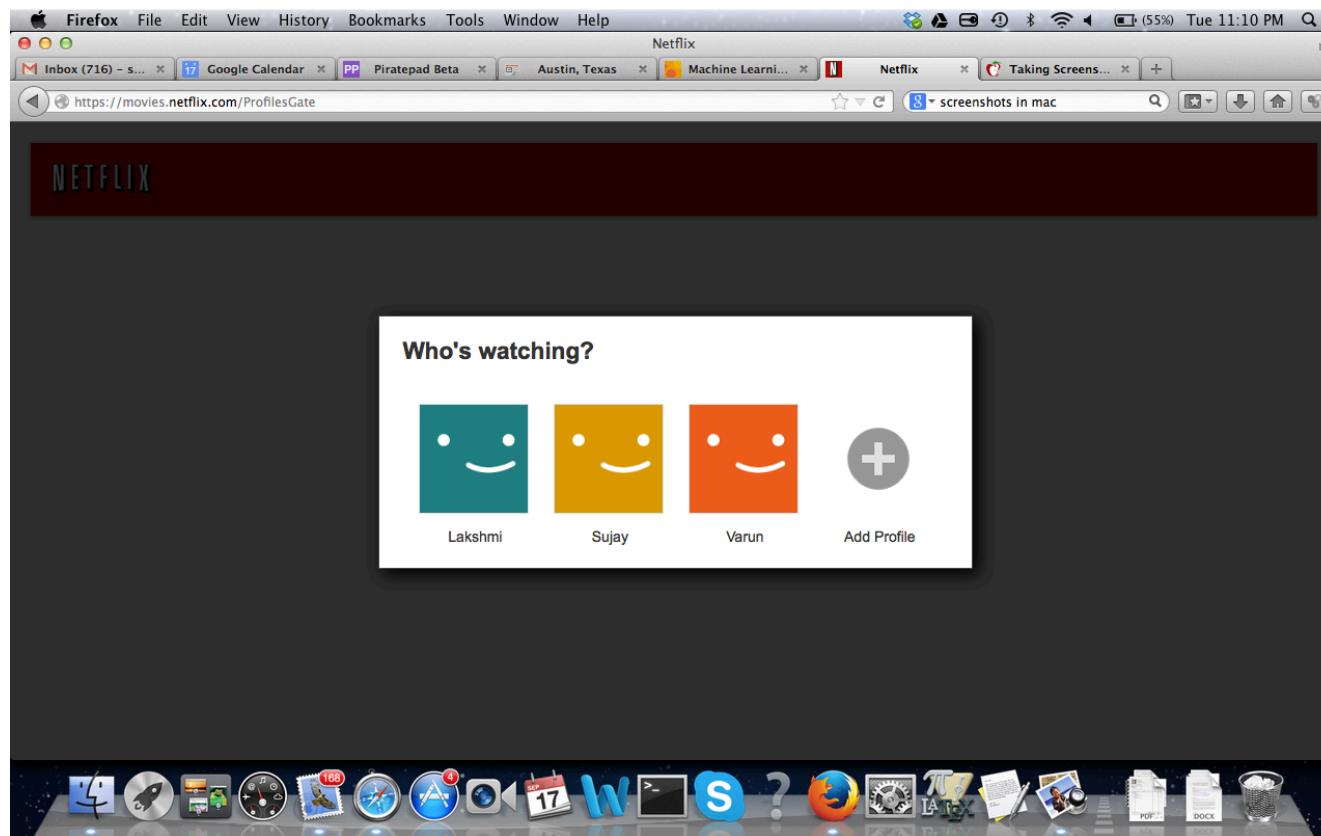
- Evolutionary biology: separating phenotypes
- Quantitative Finance: detecting regime change
- Healthcare: separating patient classes for differential treatment

Several specialized R packages (see [Grun,Leisch] for overview)

- all implement variants / optimizations of EM - **AltMin**

Mixed Linear Regression

... my netflix problem ...



Mixed Linear Regression

AltMin: alternate between the z 's and the β 's

$$\min_{\beta_1, \beta_0} \sum_i \min_{z_i \in \{0,1\}} (y_i - z_i \langle x_i, \beta_1 \rangle + (1 - z_i) \langle x_i, \beta_0 \rangle)^2$$

(a) Assign each sample to the lower current error

$$\hat{z}_i = 1 \iff (y_i - \langle x_i, \hat{\beta}_1 \rangle)^2 < (y_i - \langle x_i, \hat{\beta}_0 \rangle)^2$$

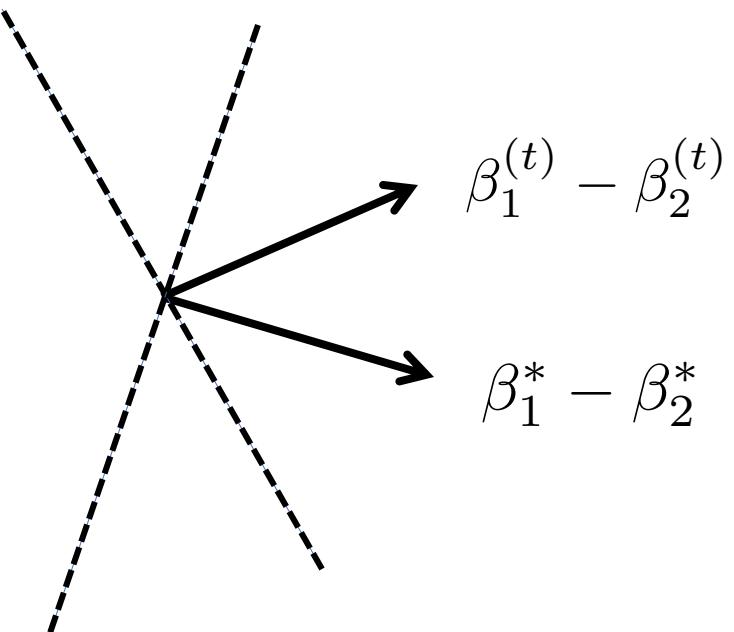
(b) Update each β from its assigned samples

$$\hat{\beta}_1 \leftarrow \arg \min_{\beta} \sum_{i:z_i=1} (y_i - \langle x_i, \beta \rangle)^2$$

Both updates are
Simple closed forms !

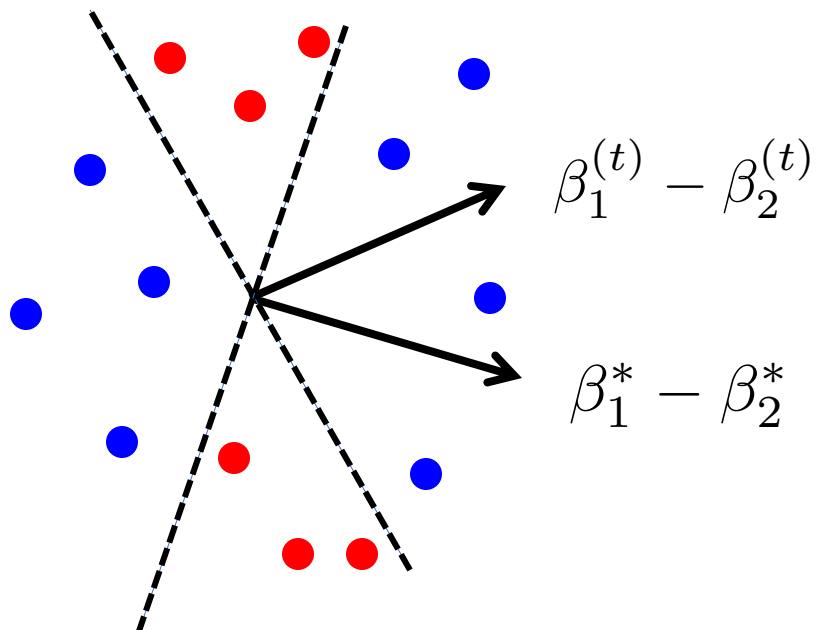
Mixed Linear Regression

Intuition: current iterate $\beta_1^{(t)}, \beta_2^{(t)}$ truth β_1^*, β_2^*



Mixed Linear Regression

Intuition: current iterate $\beta_1^{(t)}, \beta_2^{(t)}$ truth β_1^*, β_2^*



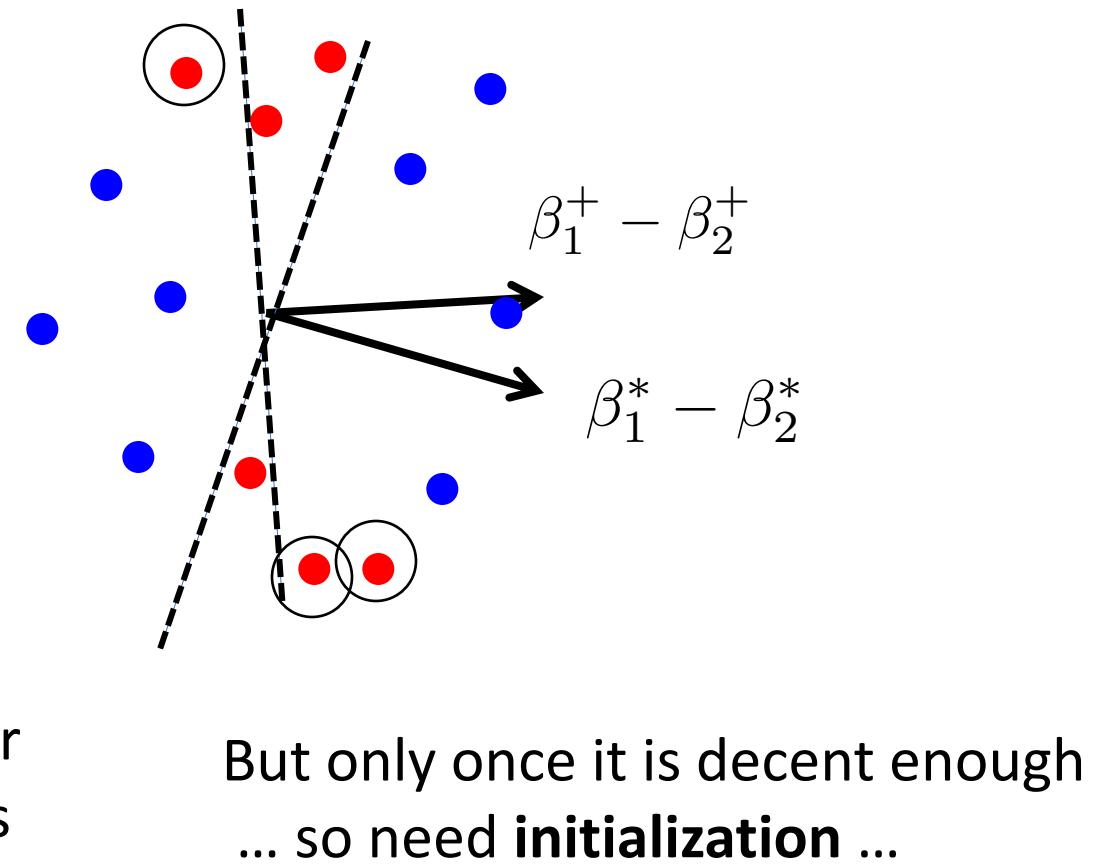
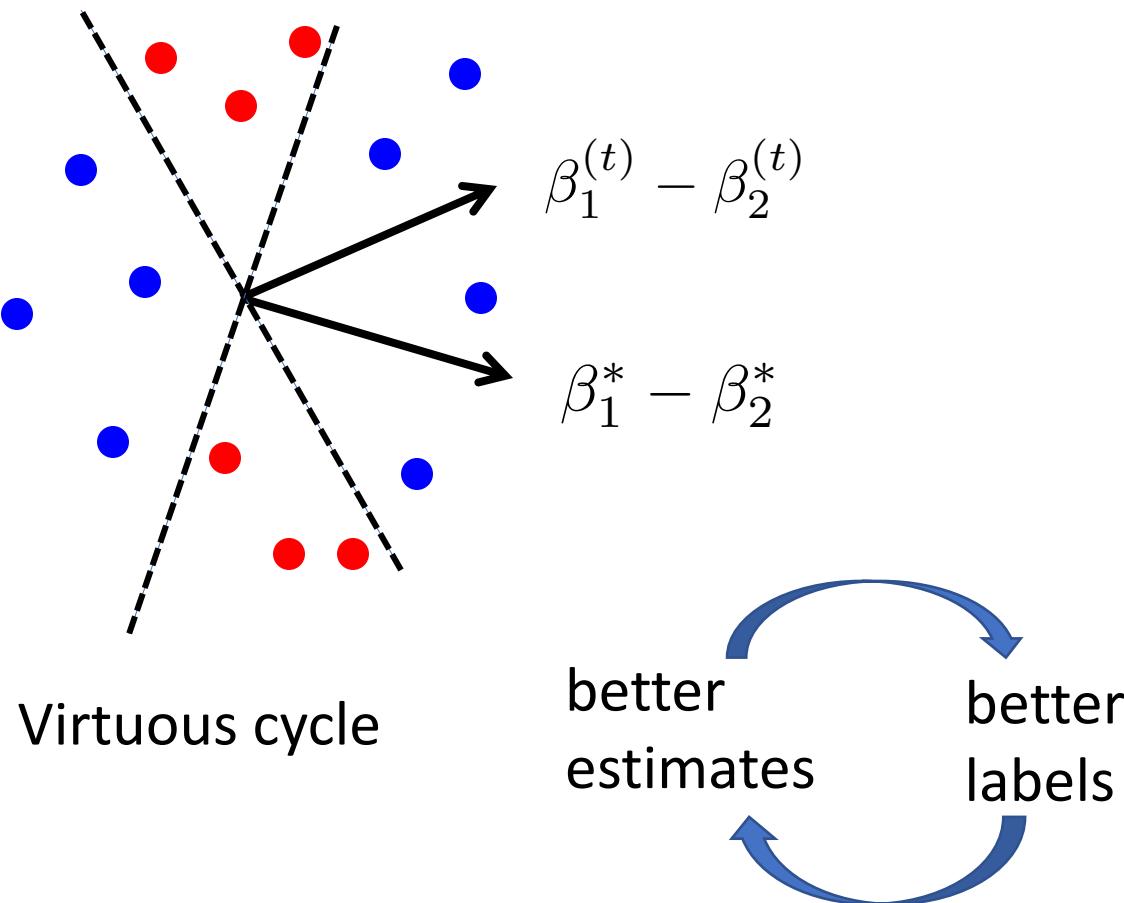
If β_1, β_2 not too far from β_1^*, β_2^*

Then **majority** points will be correctly assigned.

So, running least-squares on these will yield better next iterate.

Mixed Linear Regression

Intuition: current iterate $\beta_1^{(t)}, \beta_2^{(t)}$ truth β_1^*, β_2^*



Spectral Initialization

Matrix Completion

Take top-r eigenvectors of
sparse 0-filled matrix

Mixed linear regression

Take top 2 eigenvectors of

$$M = \sum_{i=1}^n y_i^2 x_i x_i^T$$

AltMin also successful for Phase retrieval, matrix sensing, robust PCA etc.

Open Problem: a more general theory of AltMin and its convergence ...

Open problems

Adaptive Gradient Methods

Condition Number

Recall: for β smooth and α strongly convex function, gradient descent gives

$$f(x_+) - f^* \leq \left(1 - \frac{\alpha}{\beta}\right) (f(x) - f^*)$$

Can be slow for bad condition number, i.e. when $\alpha \approx \beta$
acceleration only partially alleviates problem

Classical fix: Newton method, which needs 2nd order oracle and (big) matrix inversion
(we will talk about Quasi Newton methods in a few slides)

AdaGrad: a fix using only 1st order oracle

Lower bounds do not apply to it because it violates a technical condition there ...

AdaGrad

$$\text{Newton method: } x_+ = x - \eta (\nabla^2 f(x))^{-1} \nabla f(x)$$

Full-matrix adagrad:

Accumulate past gradients: $G_t = \sum_{k \leq t} g_k g_k^T$ where g_k is the (stochastic) (sub) gradient of f at x

$$x_{t+1} = x_t - \eta (G_t)^{-1} \nabla f(x_t)$$

Intuition: large eigenvalues of Hessian will show up in accumulated gradients ...

No 2nd order oracle, but still needs large matrix inversion ...

AdaGrad

Diagonal AdaGrad:

$$G_t = \text{diag}(G_{t,i}), \quad G_{t,i} = \sqrt{\sum_{k=1}^t g_{k,i}^2} + \epsilon$$

$$x_{t+1} = x_t - \eta G_t^{-1} g_t$$

Compare to standard SGD: $x_{t+1} = x_t - \eta_t g_t$

Intuition: Standard SGD treats all coordinates **equally**; AdaGrad uses an **adaptive** learning rate for every coordinate.

Comparison of SGD and AdaGrad

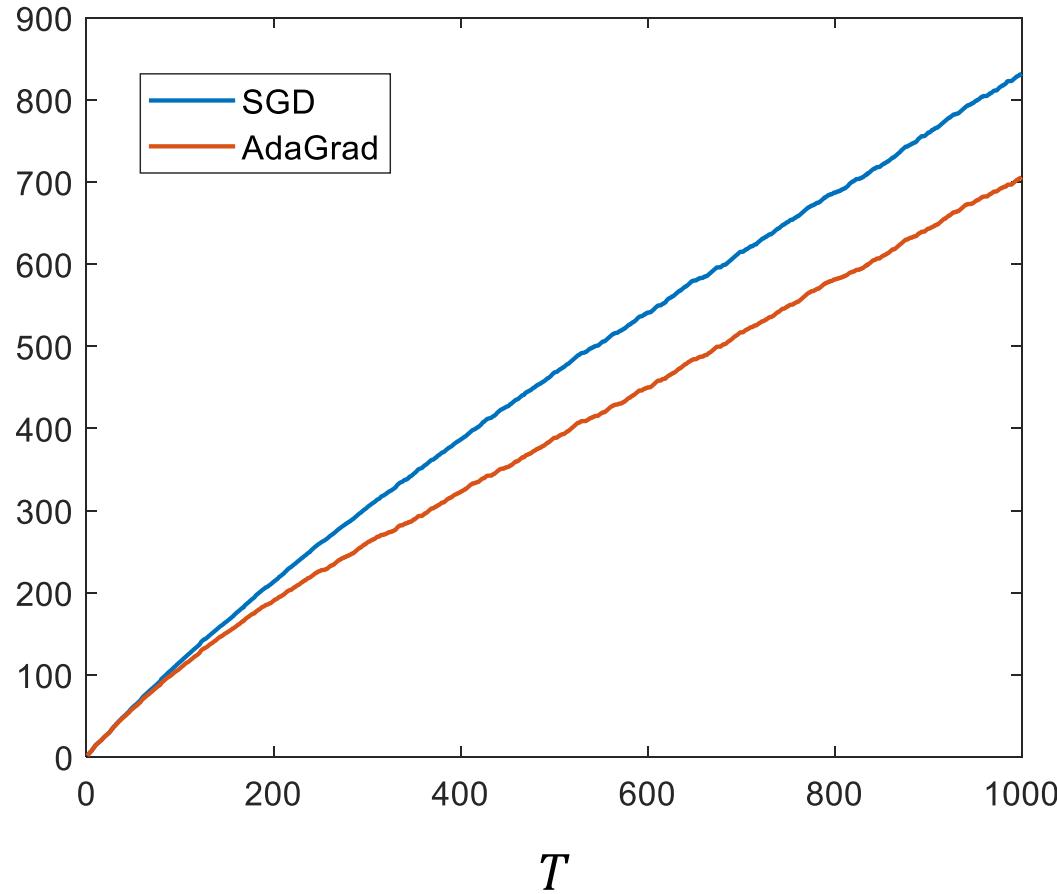
- AdaGrad outperforms SGD when the gradient vectors are **sparse**.
- Consider an **online learning** problem [Duchi et al.'2011]:
 - Given sample (z_t, y_t) , loss $f_t(x) = [1 - y_t \langle z_t, x \rangle]_+$
 - Sample $z_t \in \{-1, 0, 1\}^d$, feature i is non-zero w.p. $p_i \propto i^{-2}$
 - Constraint on x : $\|x\|_\infty \leq 1$
- Regret bounds for this particular example:

Algorithm	Regret
SGD	$O(\sqrt{dT})$
AdaGrad	$O(\log d \sqrt{T})$

Simulation of the previous example

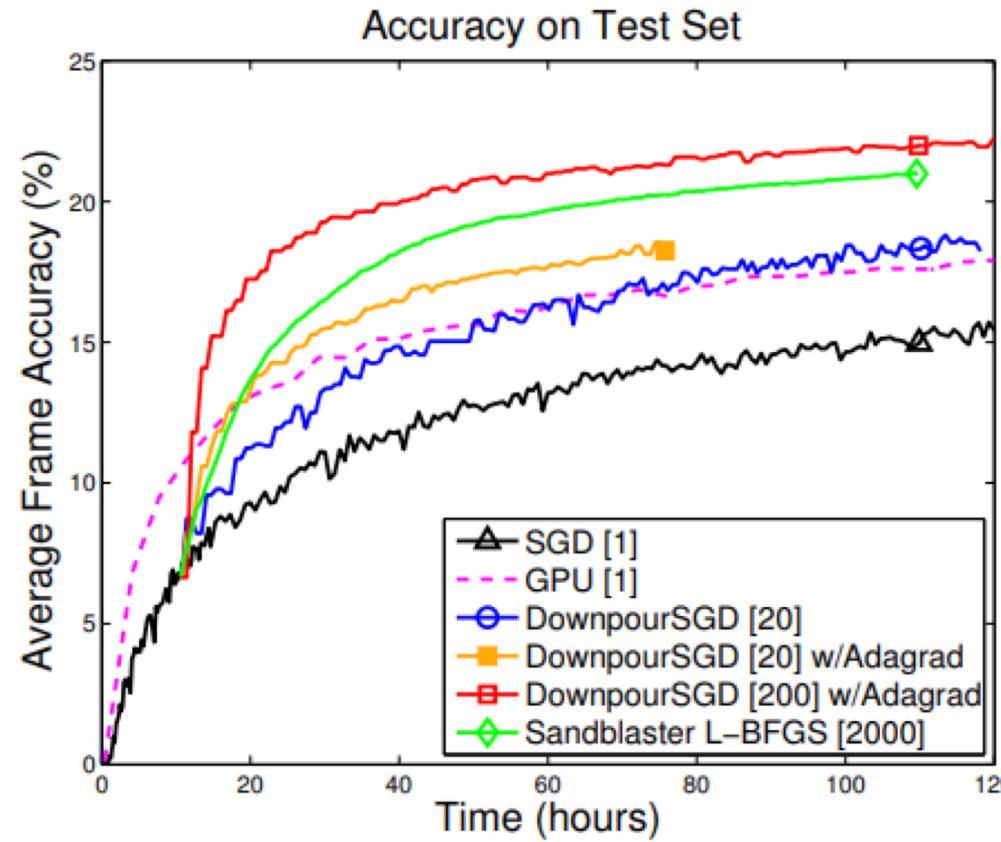
Regret

$$\sum_{t=1}^T f_t(x_t) - f_t(x^*)$$



Sample: $y_t = \text{sign}(\langle z_t, x^* \rangle)$
 x^* : vector of all ones
Dimension: $d = 10$

Real data



A neural network model trained for speech recognition. Figure taken from [Dean et al.'2012].

RMSprop

- Problem with AdaGrad:
 - Learning rate \propto accumulated sum of squared gradients \Rightarrow decreases too fast
- RMSprop:
 - Learning rate \propto exponential moving average of squared gradients
 - Update every coordinate $i \in [d]$ as follows

$$\nu_t = \beta \nu_{t-1} + (1 - \beta) g_t^2$$

$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\nu_t} + \epsilon} g_t$$

Adam

- Can be viewed as RMSprop + momentum + bias correction:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Convergence rates

- Convex online setting:
 - AdaGrad has regret bound $O(\sqrt{T})$ [Duchi et al.' 2011]
 - RMSprop and Adam does not always converge [Reddi et al.'2018]
 - Because v_t may decrease too fast \Rightarrow very large learning rate
 - AMSGrad [Reddi et al.'2018]: $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$
 - AMSGrad has regret bound $O(\sqrt{T})$
- Non-convex setting:
 - AdaGrad, AMSGrad satisfy [Chen et al.'2019]
$$\min_{t \in [T]} \mathbb{E}[\|g_t\|_2^2] = O\left(\frac{\log T}{T}\right)$$

Reddi et al., “On the Convergence of Adam and Beyond”, ICLR 2018.

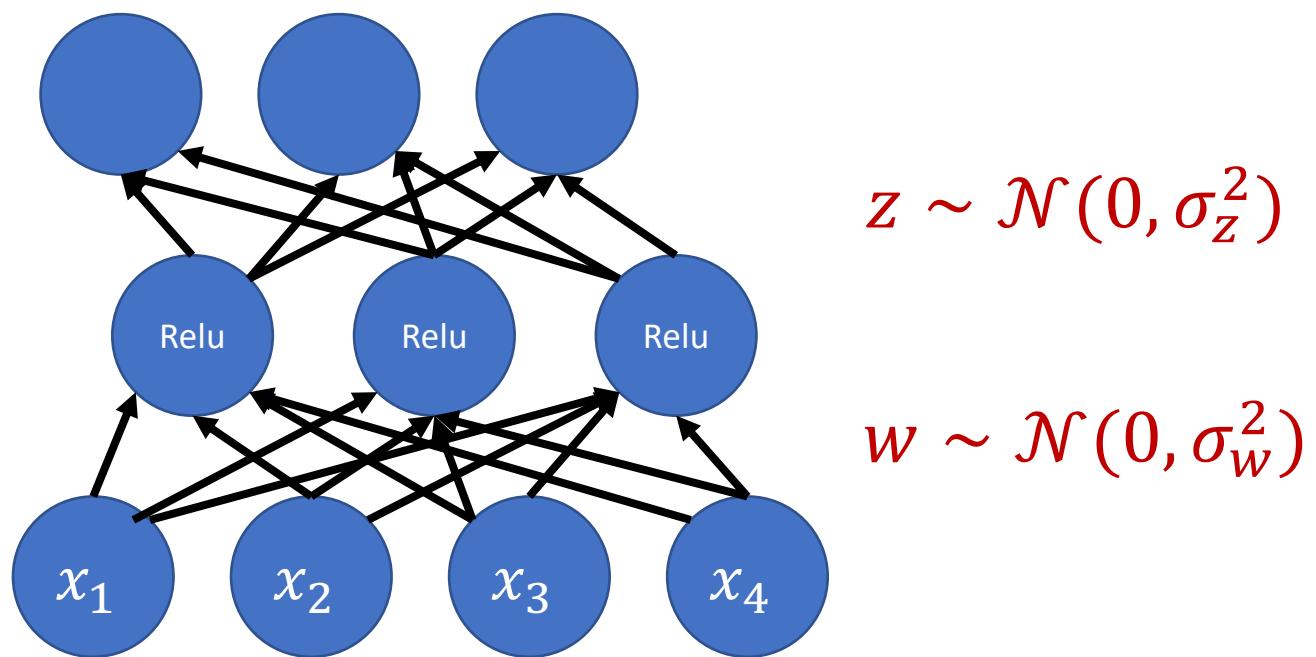
Chen et al., “On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization”, ICLR 2019.

Quasi Newton methods

- First order methods that try to emulate second order methods (such as Newton method) by estimating Hessians (using gradients)
 - E.g., BFGS, L-BFGS $\nabla f(x_{k+1}) - \nabla f(x_k) \sim H(x_{k+1} - x_k)$
 - Well understood theory; widely successful for convex optimization in practice
- Nonconvex optimization
 - **Theory:** Initial results on convergence to FOSP [Wang, Ma, Goldfarb, Liu 2016]
 - **Practice:** Has not been effective so far
 - Need to design better algorithms?

Initialization

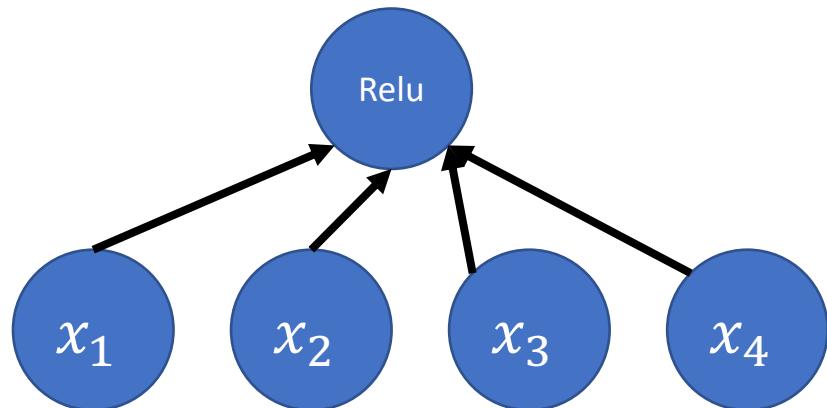
- Refers to choice of x_0 ; affects both optimization speed and quality of the local minimum



Initialization

- Highly influential empirical works: [Glorot, Bengio 2010]; [Sutskever, Martens, Dahl, Hinton 2013]; [He, Zhang, Ren, Sun 2015]
- Main viewpoint: Gradients and activations at initialization

$$y = \text{ReLU}(w^\top x)$$



$$\text{Var}(y) \propto \sigma_w^2 \cdot \|x\|^2$$

- For $y \sim x_i$, choose $\sigma_w^2 \propto \frac{1}{d_{\text{in}}}$
- Training phase: not understood

Nonconvex nonconcave minimax optimization

$$\min_x \max_y f(x, y)$$

- The function to minimize $g(x) \stackrel{\text{def}}{=} \max_y f(x, y)$ defined implicitly
- Basis of generative adversarial networks (GAN) and robust training
- Notions of local optimality not well understood in the general setting
- Gradient descent ascent widely used but its convergence properties not understood

Summary

- Nonconvex optimization is the primary computational work horse in machine learning now
- Main ideas behind these algorithms come from convex optimization
- Research direction 1: Understanding statistical + optimization landscape of important nonconvex problems
 - E.g., SOSPs = global optima in matrix factorization problems
- Research direction 2: Designing faster algorithms