COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Automated Inference with Adaptive Batches

Soham De

Joint work with Abhay Yadav, David Jacobs, Tom Goldstein

University of Maryland

OVERVIEW

Most machine learning models use SGD for training

BUT…**noisy gradients** & **large variance**

↓

Hard to automate stepsize selection & stopping conditions

OVERVIEW

Most machine learning models use SGD for training

BUT…**noisy gradients** & **large variance**

↓

Hard to automate stepsize selection & stopping conditions

In this work: **Big Batch SGD**

Adaptively grow batch size based on amount of noise in the gradients

Easy automated stepsize selection

MOST MODEL FITTING PROBLEMS LOOK LIKE THIS

$$\min \ell(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, z_i)$$

MOST MODEL FITTING PROBLEMS LOOK LIKE THIS

$$\min \ell(x) := \frac{1}{N}\sum_{i=1}^{N} f(x, z_i)$$

We also consider the more general problem:

$$\min \ell(x) := \mathbb{E}_{z\sim p}[f(x, z)]$$

Applications

SVM

logistic regression

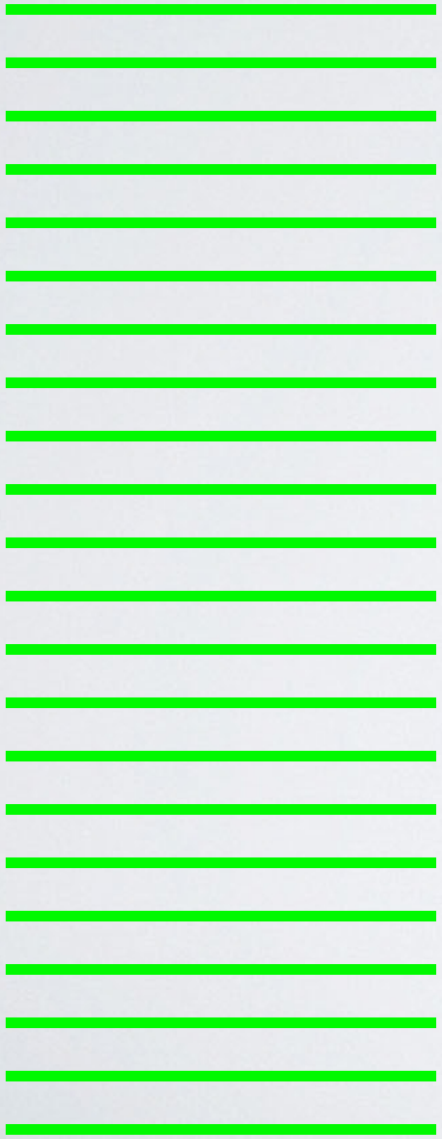
neural nets

matrix factorization

blah blah blah

SGD

select data

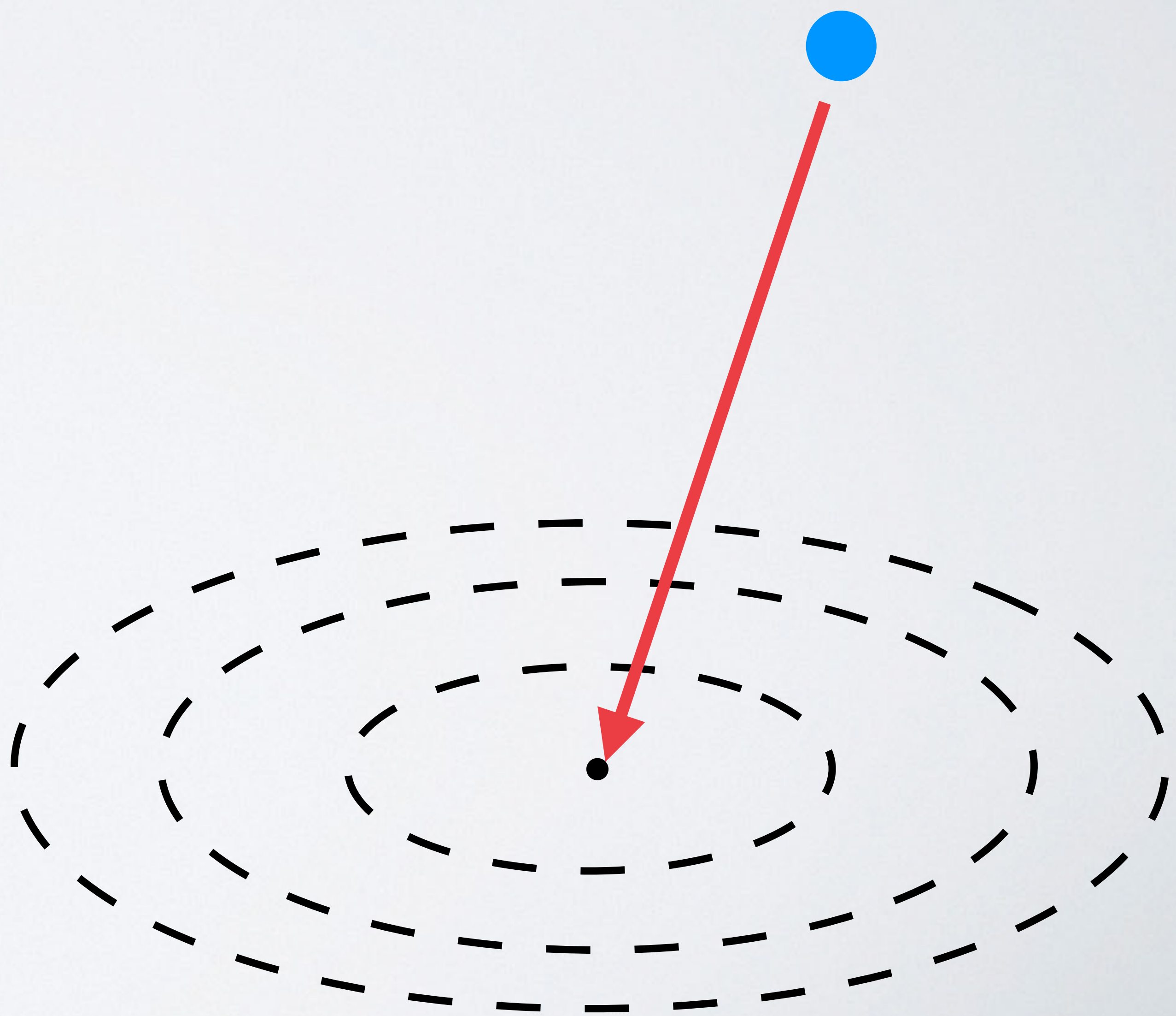


compute gradient

$$g_t = \frac{1}{N}\sum_{i=1}^{N} \nabla f(x_t, z_i)$$

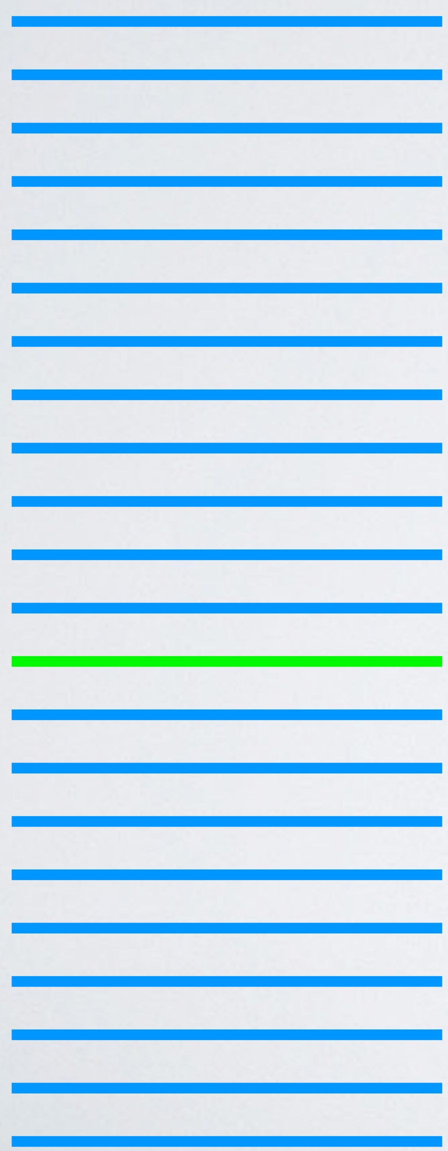
update

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

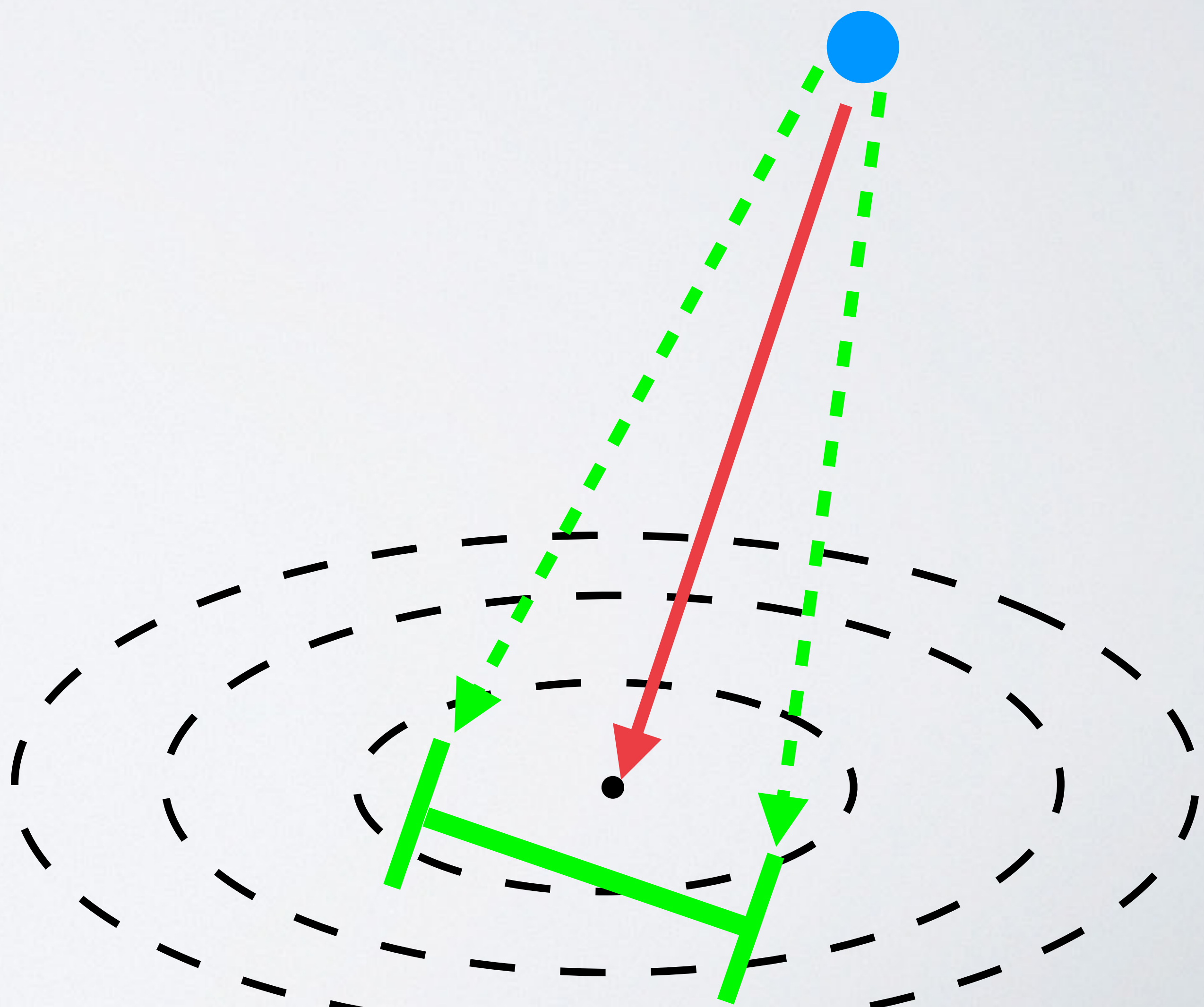


SGD

select data **compute gradient** **update**



$$g_t \approx \nabla f(x_t, z_{12}) \rightarrow x_{t+1} = x_t - \alpha_t g_t$$



SGD

select data

compute gradient

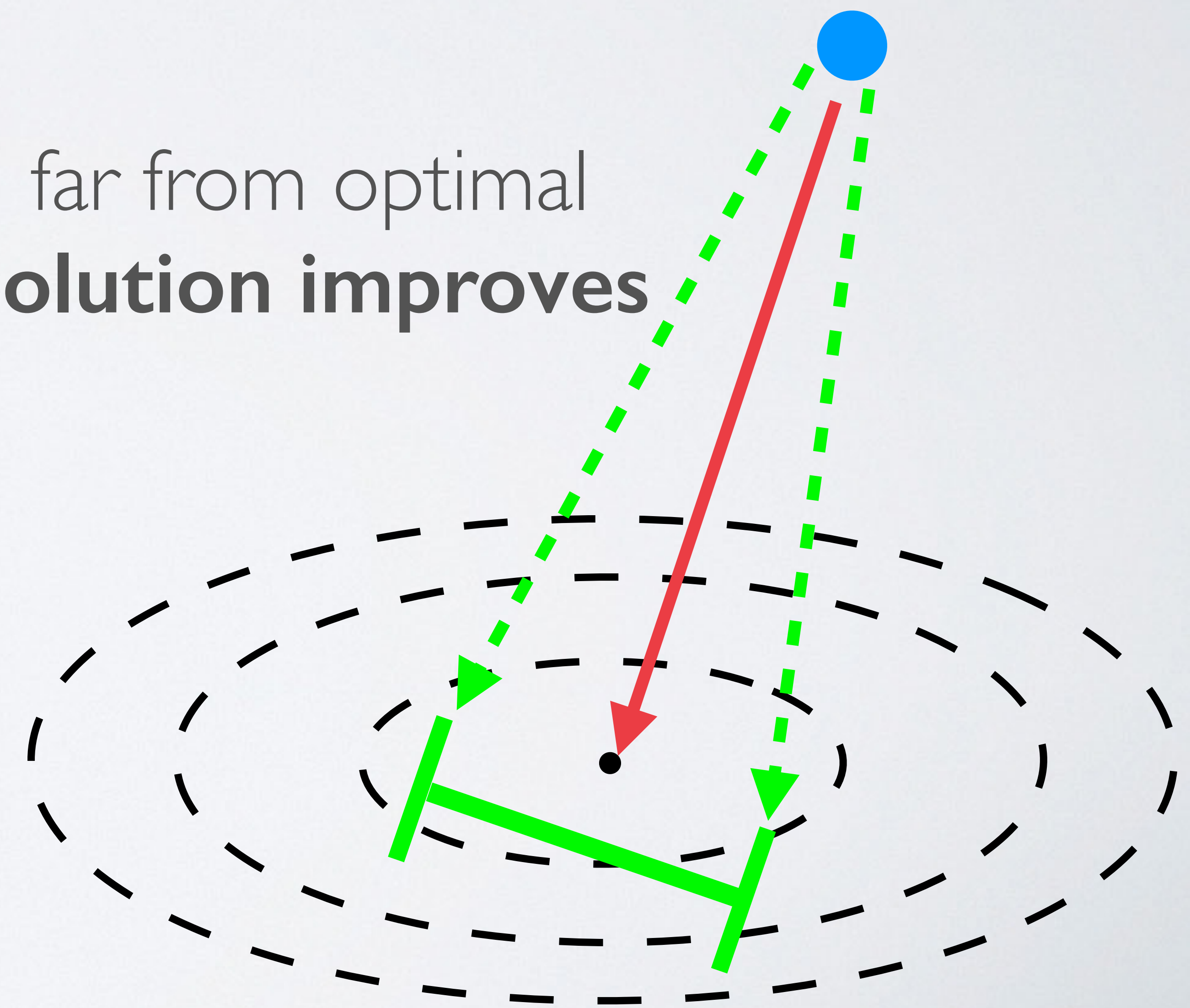
$$g_t \approx \nabla f(x_t, z_8)$$

update

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

far from optimal

solution improves



SGD

select data

compute gradient

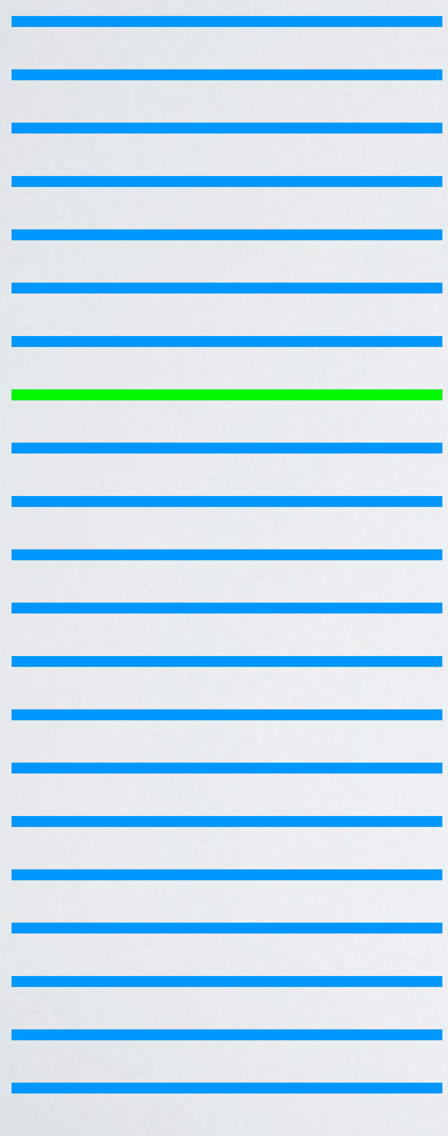
$$g_t \approx \nabla f(x_t, z_8)$$

update

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

SGD

select data



compute gradient

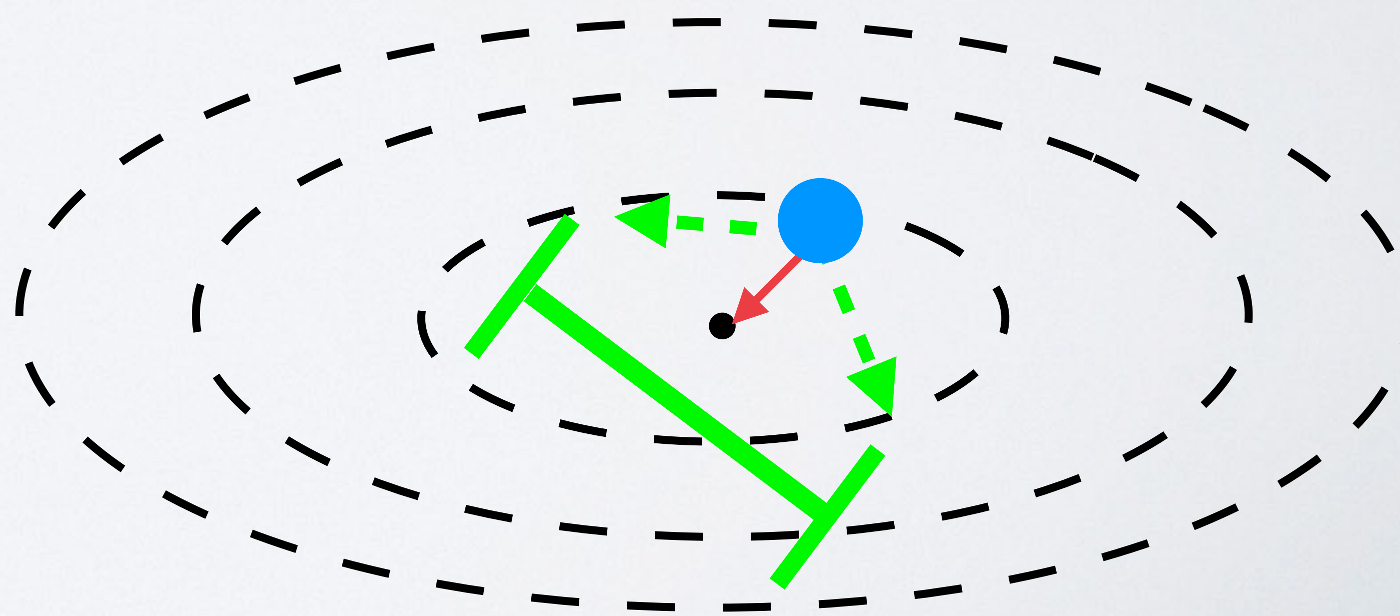
$$g_t \approx \nabla f(x_t, z_8)$$

update

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

close to optimal

solution gets worse



SGD

select data

compute gradient

$$g_t \approx \nabla f(x_t, z_8)$$

update

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

Error must **decrease**
as we approach solution

SGD

select data

compute gradient

$$g_t \approx \nabla f(x_t, z_8)$$

update

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

Error must **decrease**
as we approach solution

classical solution

shrink stepsize

$$\lim_{t \to \infty} \alpha_t = 0$$

SGD

select data

compute gradient

$$g_t \approx \nabla f(x_t, z_8)$$

update

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

Error must **decrease**
as we approach solution

classical solution

shrink stepsize

$$\lim_{t \to \infty} \alpha_t = 0$$

hard to pick
stepsize schedule

USE BIGGER BATCHES?

select data

compute gradient

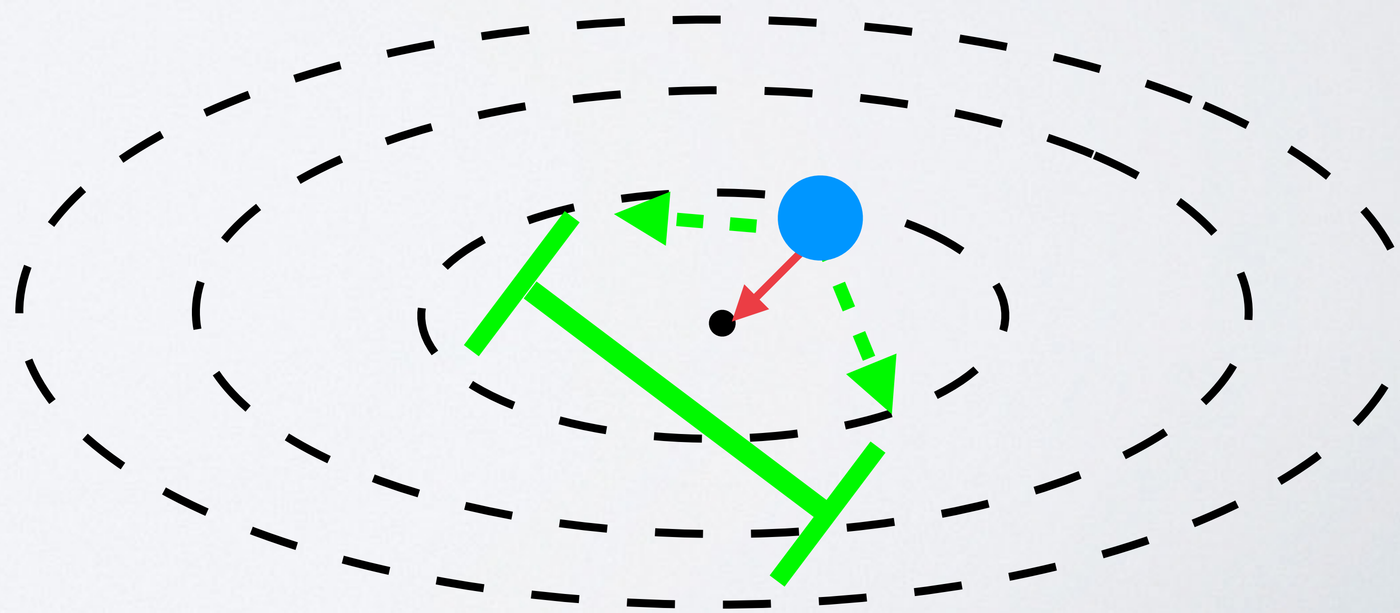
$$g_t \approx \nabla f(x_t, z_8)$$

update

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

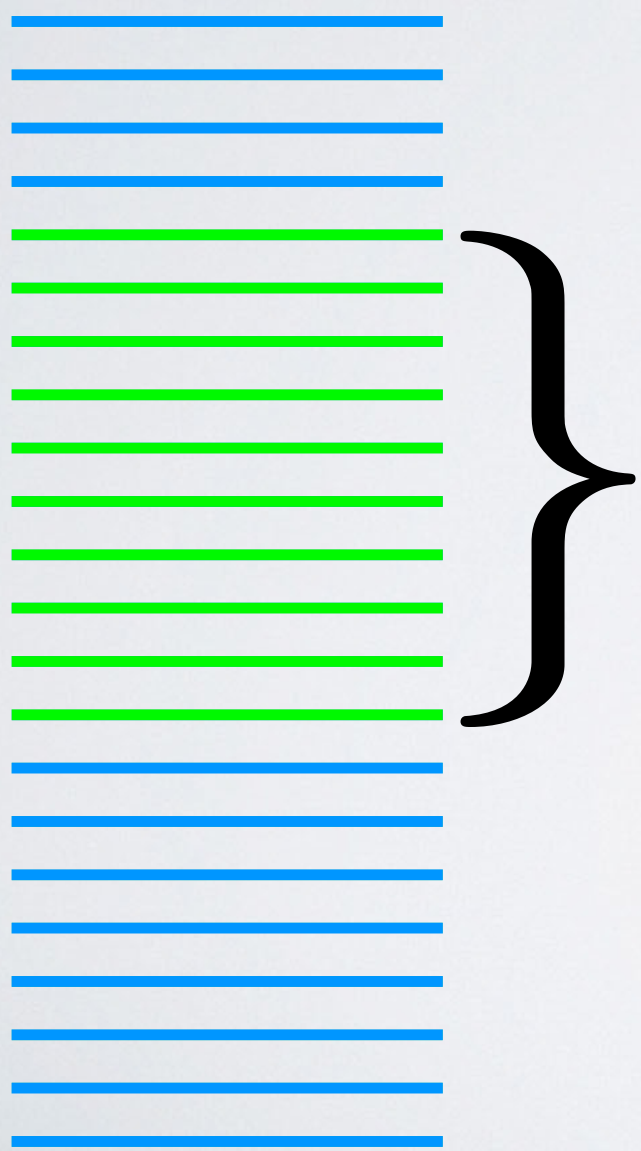
close to optimal

solution gets worse



USE BIGGER BATCHES?

select data



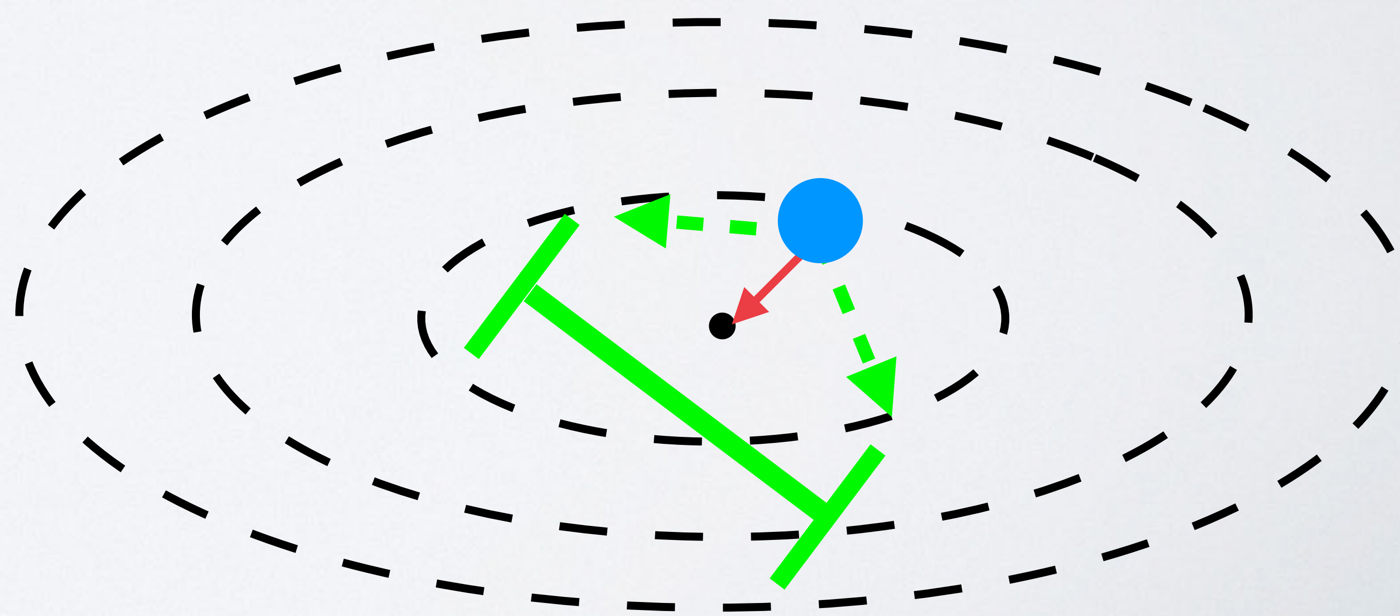
compute gradient

update

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

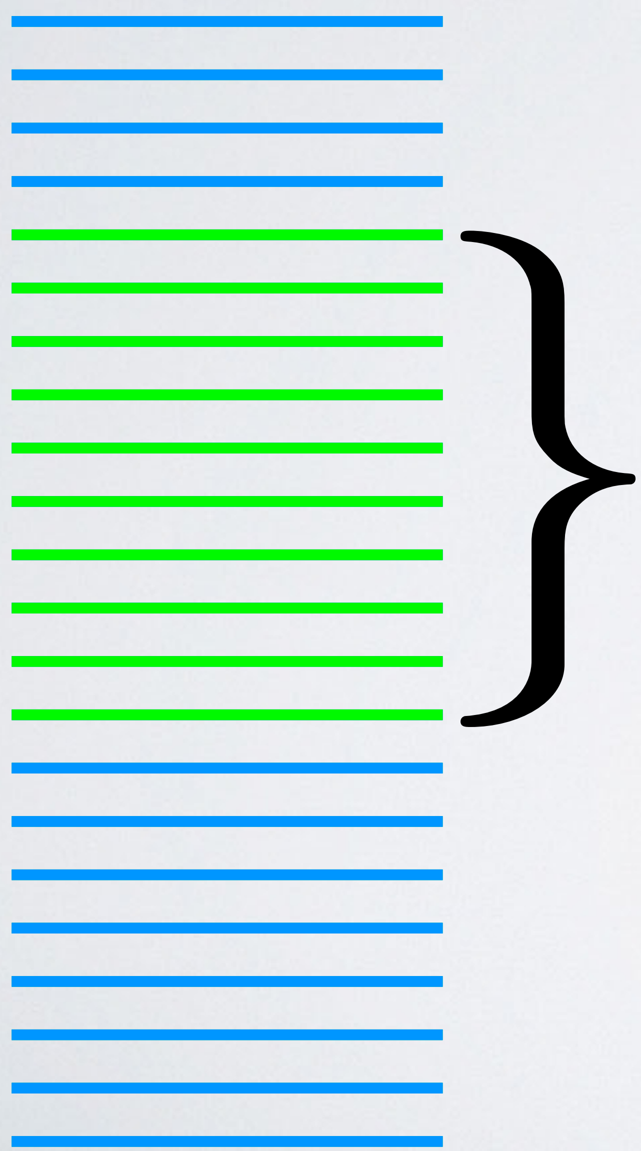
close to optimal

solution gets worse



USE BIGGER BATCHES?

select data



compute gradient

$$g_t \approx \underline{\nabla \ell_{\mathcal{B}}(x_t)}$$

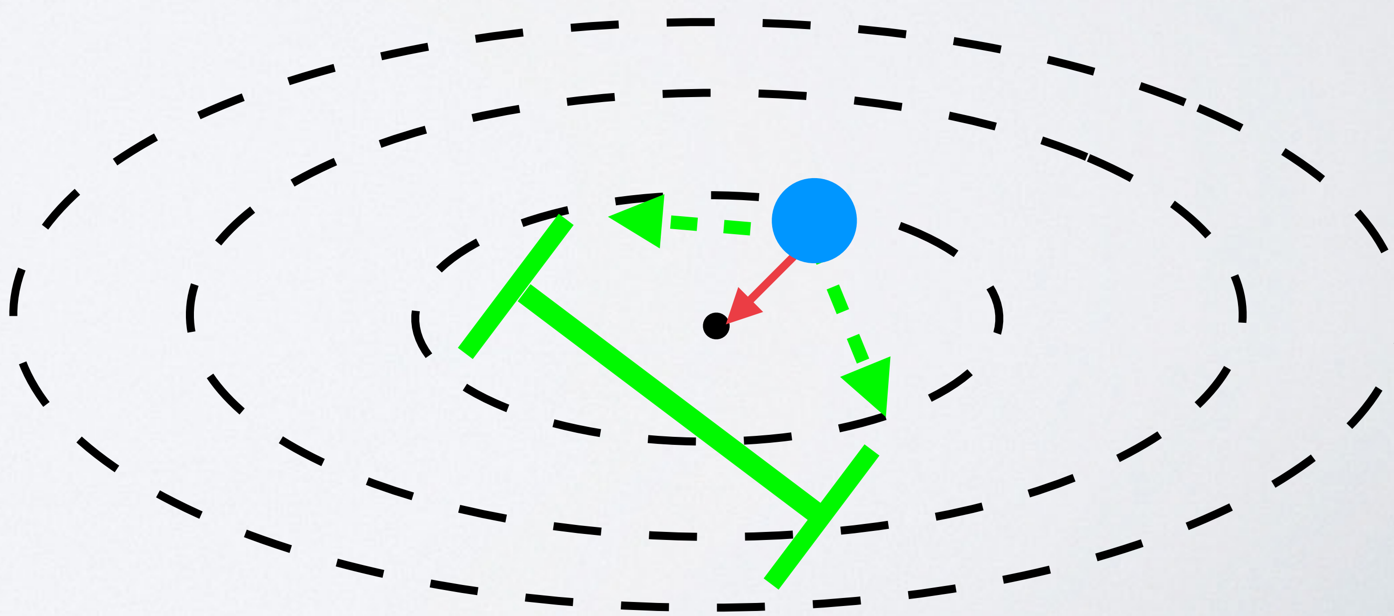
gradient on batch *B*

update

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

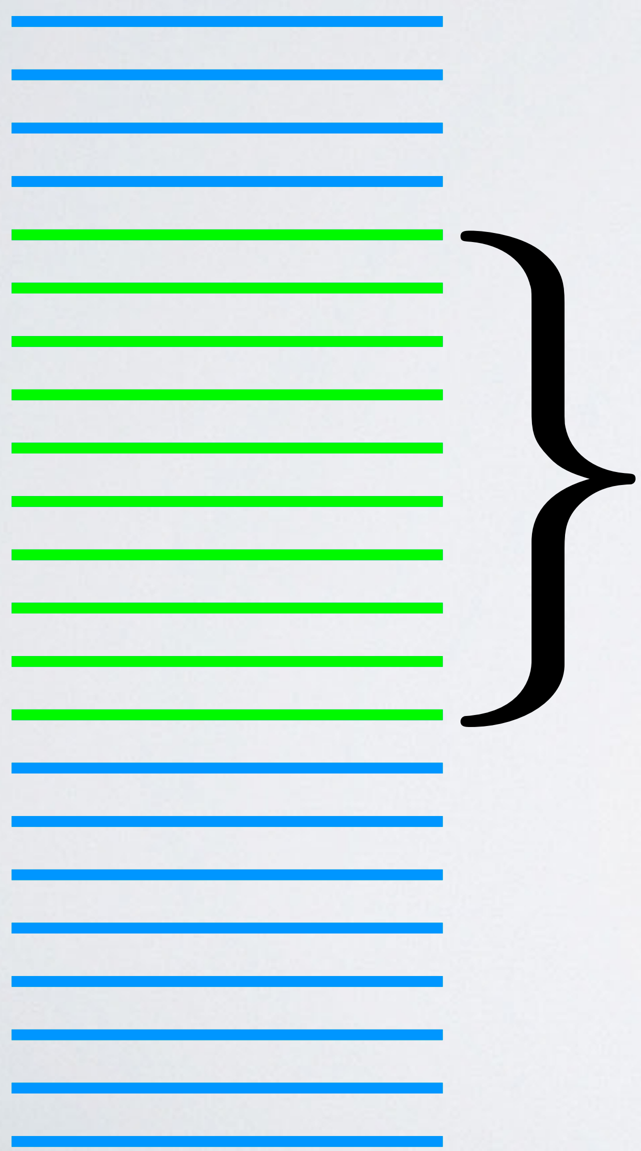
close to optimal

solution gets worse



USE BIGGER BATCHES?

select data



compute gradient

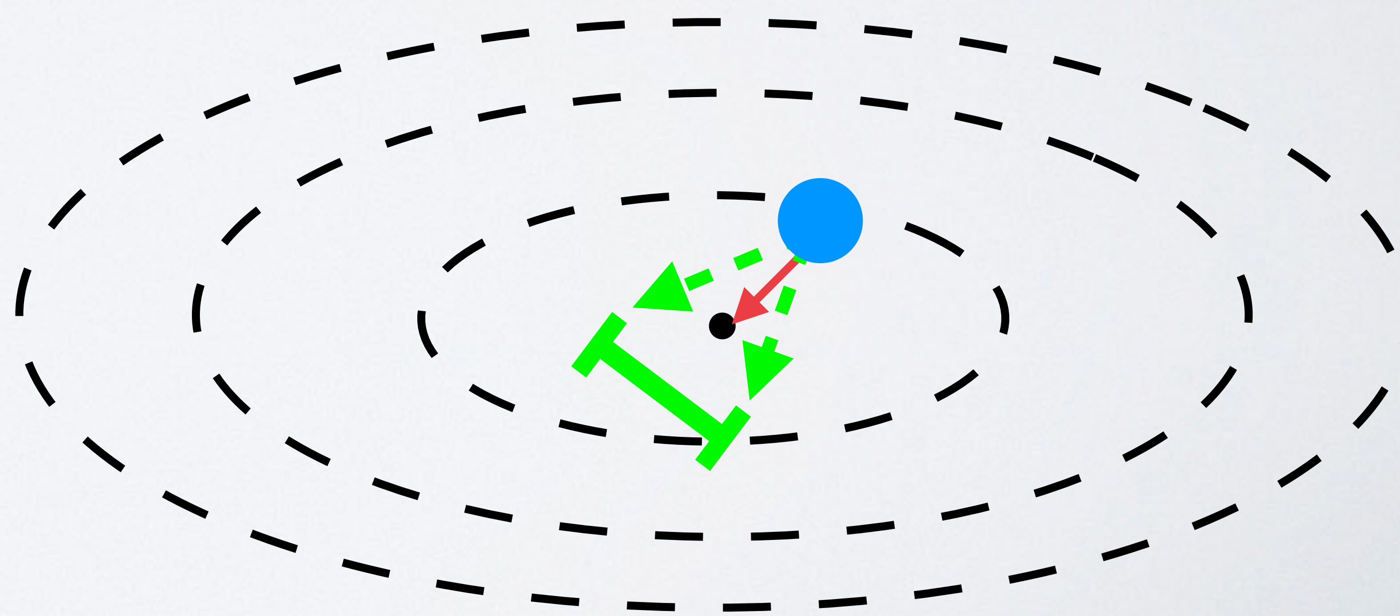
$$g_t \approx \underline{\nabla \ell_{\mathcal{B}}(x_t)} \longrightarrow x_{t+1} = x_t - \alpha_t g_t$$

gradient on batch B

update

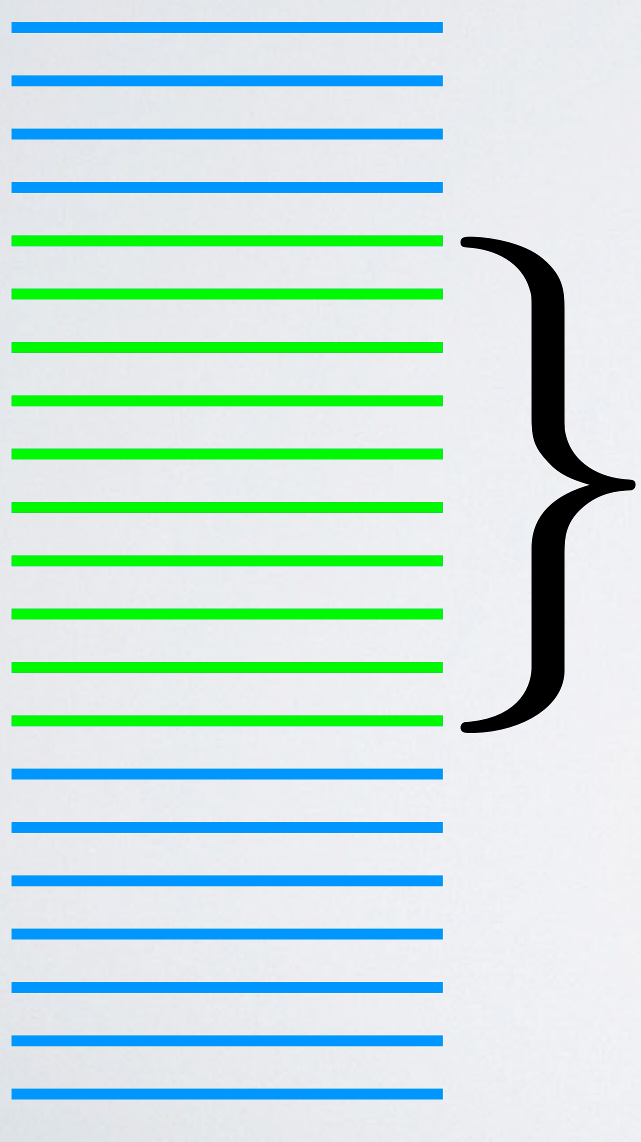
close to optimal

solution gets worse



USE BIGGER BATCHES?

select data



compute gradient

$$g_t \approx \underline{\nabla \ell_{\mathcal{B}}(x_t)}$$

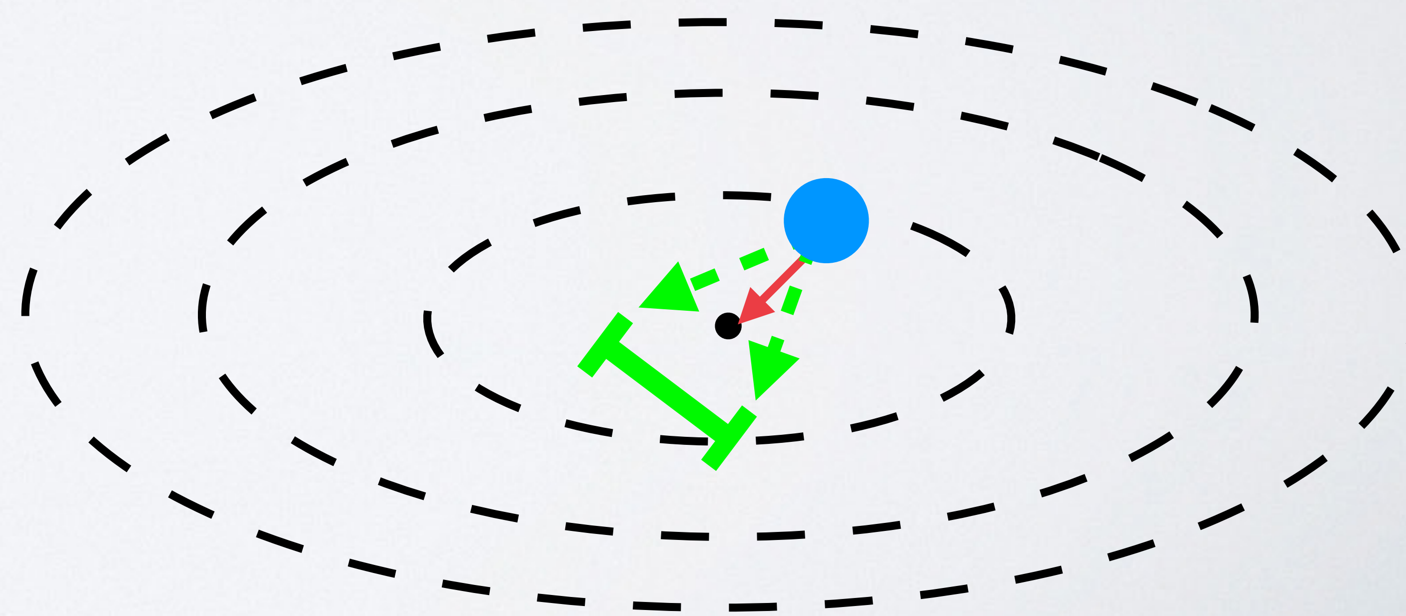
gradient on batch *B*

update

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

lower variance

solution gets better



GROWING BATCHES

regime 1: far from optimal

regime 2: close to optimal

GROWING BATCHES

regime 1: far from optimal

Noisy gradients
improve solution

↓

Large batches do
unnecessary work
+
Get stuck in local minima

regime 2: close to optimal

GROWING BATCHES

regime 1: far from optimal

Noisy gradients
improve solution

↓

Large batches do
unnecessary work
+
Get stuck in local minima

small batches
work well!

regime 2: close to optimal

GROWING BATCHES

regime 1: far from optimal

Noisy gradients improve solution

↓

Large batches do unnecessary work
+
Get stuck in local minima

small batches work well!

regime 2: close to optimal

Noisy gradients with high variance worsen solution

↓

Small batches require stepsize decay (hard to tune)

GROWING BATCHES

regime 1: far from optimal

Noisy gradients improve solution

↓

Large batches do unnecessary work
+
Get stuck in local minima

small batches work well!

regime 2: close to optimal

Noisy gradients with high variance worsen solution

↓

Small batches require stepsize decay (hard to tune)

large batches work well!

GROWING BATCHES

regime 1: far from optimal

Noisy gradients
improve solution

Large b
unneces

Get stuck in local minima

small batches
work well!

regime 2: close to optimal

Noisy gradients with high
variance worsen solution

require
rd to tune)

large batches
work well!

Adaptively Grow
Batches Over Time!

PRELIMINARIES

Standard result in stochastic optimization:

Lemma
A sufficient condition for $-\nabla \ell_{\mathcal{B}}(x)$ to be a descent direction is:

$$\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2 < \|\nabla \ell_{\mathcal{B}}(x)\|^2.$$

PRELIMINARIES

Standard result in stochastic optimization:

Lemma

A sufficient condition for $-\nabla \ell_{\mathcal{B}}(x)$ to be a descent direction is:

$$\underbrace{\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2}_{\text{error}} < \underbrace{\|\nabla \ell_{\mathcal{B}}(x)\|^2}_{\text{approximate gradient}}.$$

PRELIMINARIES

Standard result in stochastic optimization:

Lemma

A sufficient condition for $-\nabla \ell_{\mathcal{B}}(x)$ to be a descent direction is:

$$\underbrace{\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2}_{\textbf{error}} < \underbrace{\|\nabla \ell_{\mathcal{B}}(x)\|^2}_{\textbf{approximate gradient}}.$$

If error is **small** relative to gradient : descent direction

PRELIMINARIES

Standard result in stochastic optimization:

Lemma
A sufficient condition for $-\nabla \ell_{\mathcal{B}}(x)$ to be a descent direction is:

$$\underbrace{\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2}_{\text{error}} < \underbrace{\|\nabla \ell_{\mathcal{B}}(x)\|^2}_{\text{approximate gradient}}.$$

How big is this error?

How large does the batch need to be to guarantee this?

ERROR BOUND

Theorem

Assume f has L_z - Lipschitz dependence on data z.

Then, expected error is uniformly bounded by:

$$\begin{aligned} \mathbb{E}_{\mathcal{B}}\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2 &= \operatorname{Tr}\operatorname{Var}_{\mathcal{B}}(\nabla \ell_{\mathcal{B}}(x)) \\ &\leq \frac{1}{|\mathcal{B}|} \cdot 4L_z^2 \operatorname{Tr}\operatorname{Var}_z(z) \end{aligned}$$

ERROR BOUND

Theorem

Assume f has L_z - Lipschitz dependence on data z.

Then, expected error is uniformly bounded by:

$$\underbrace{\mathbb{E}_{\mathcal{B}} \|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2}_{\textbf{expected error}} = \underbrace{\operatorname{Tr} \operatorname{Var}_{\mathcal{B}}(\nabla \ell_{\mathcal{B}}(x))}_{\textbf{batch gradient variance}}$$

$$\leq \frac{1}{|\mathcal{B}|} \cdot 4 L_z^2 \operatorname{Tr} \operatorname{Var}_z(z)$$

ERROR BOUND

Theorem

Assume f has L_z - Lipschitz dependence on data z.

Then, expected error is uniformly bounded by:

$$\underbrace{\mathbb{E}_{\mathcal{B}} \|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2}_{\text{expected error}} = \underbrace{\mathrm{Tr}\,\mathrm{Var}_{\mathcal{B}}(\nabla \ell_{\mathcal{B}}(x))}_{\text{batch gradient variance}}$$

$$\leq \underbrace{\frac{1}{|\mathcal{B}|}}_{\boxed{\text{batch size}}} \cdot 4L_z^2 \, \underbrace{\mathrm{Tr}\,\mathrm{Var}_z(z)}_{\text{data variance}}$$

ERROR BOUND

Theorem

Expected Error (Gradient Variance) < Data Variance

Higher Batch Size → Lower Gradient Variance

data variance

ERROR BOUND

From the previous Lemma and Theorem:
We expect $-\nabla \ell_{\mathcal{B}}(x)$ to be a descent direction reasonably often provided:

$$\theta^2 \mathbb{E}\|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 \geq \frac{1}{|\mathcal{B}_t|} \operatorname{Tr} \operatorname{Var}_z \nabla f(x_t, z)$$

where $\theta \in (0,1)$

ERROR BOUND

From the previous Lemma and Theorem:
We expect $-\nabla \ell_{\mathcal{B}}(x)$ to be a descent direction reasonably often provided:

$$\theta^2 \mathbb{E}\|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 \geq \frac{1}{|\mathcal{B}_t|} \mathrm{Tr}\, \mathrm{Var}_z \nabla f(x_t, z)$$

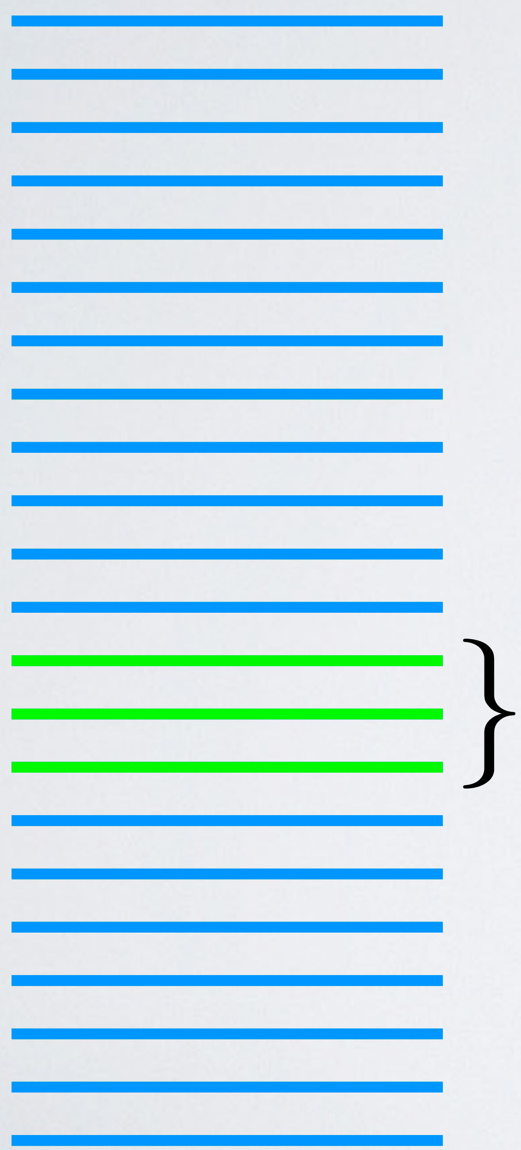
where $\theta \in (0, 1)$

We use this observation in Big Batch SGD

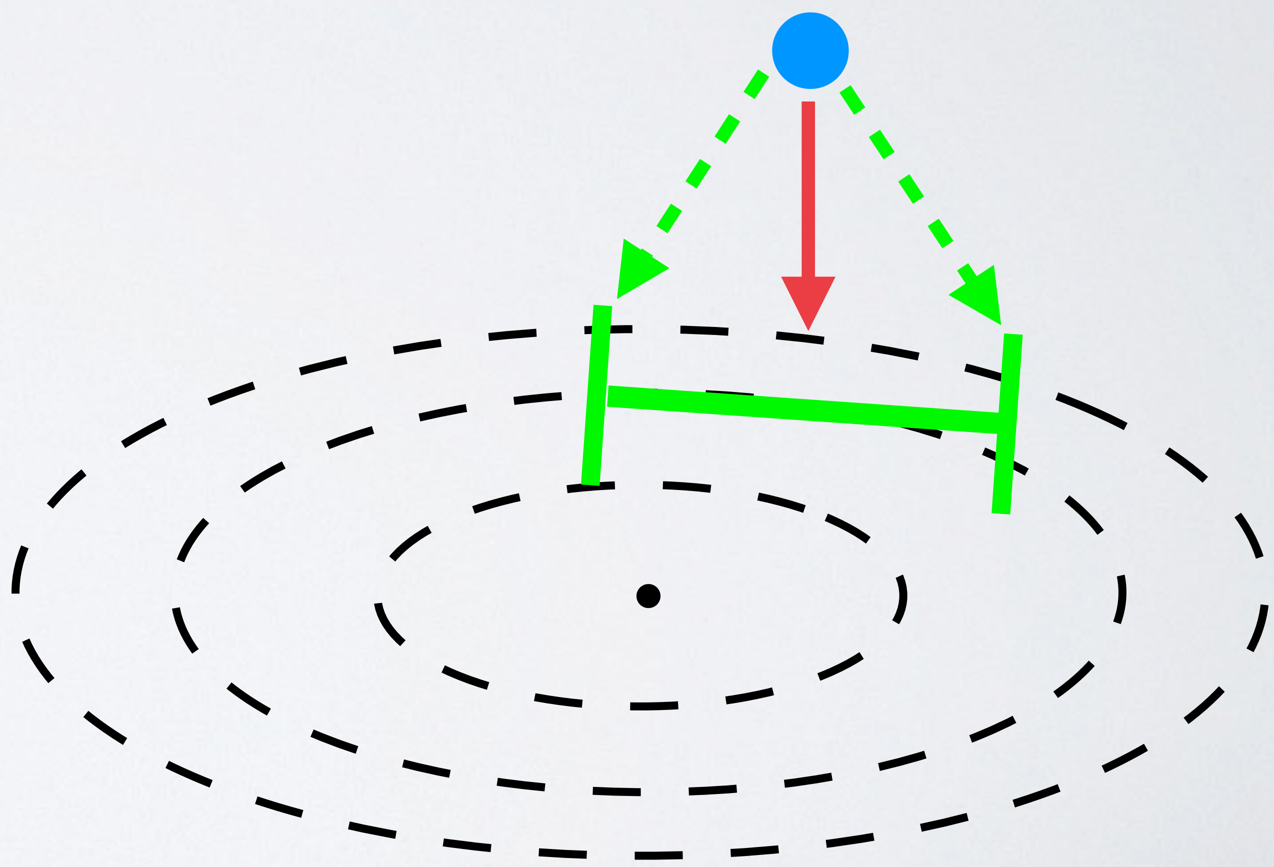
BIG BATCH SGD

BIG BATCH SGD

estimate size of error using
variance of batch gradients



pick batch *B*



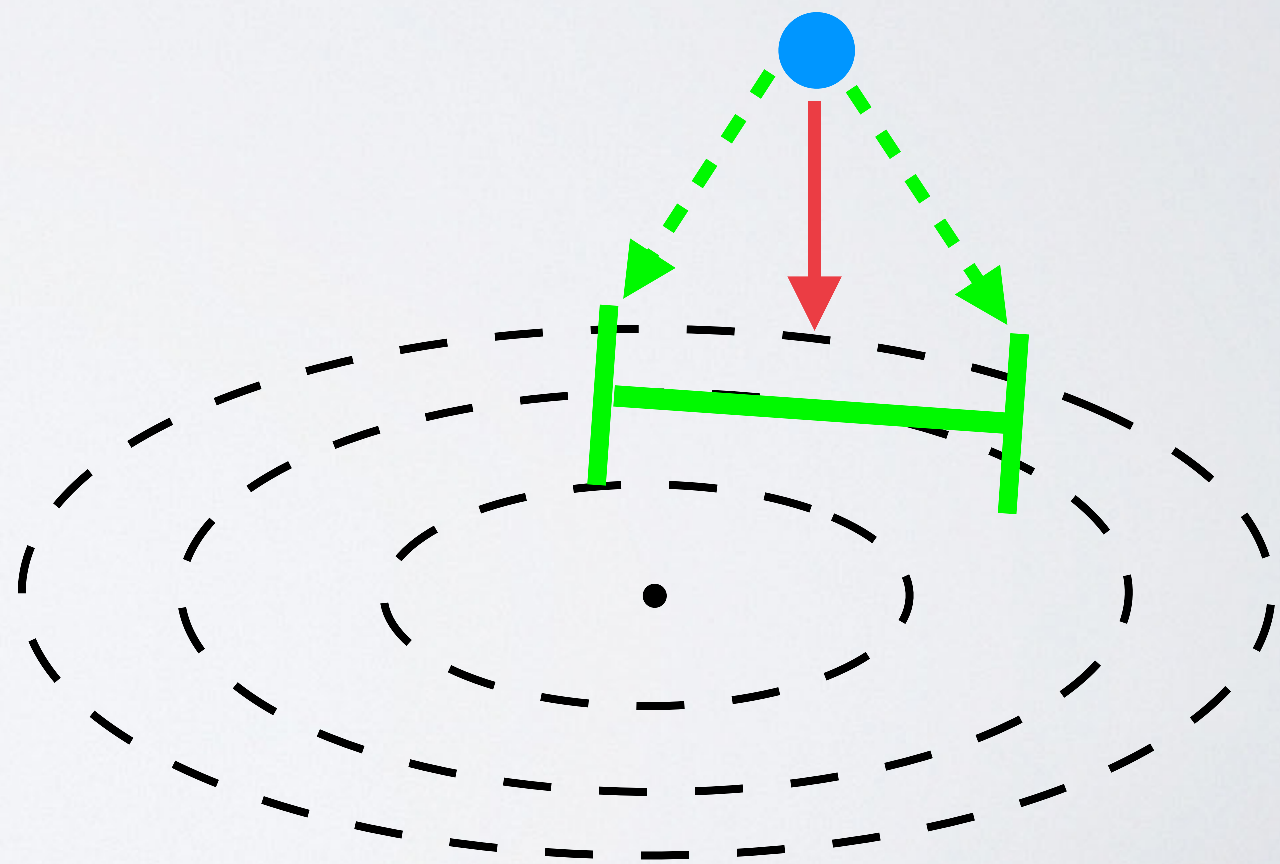
BIG BATCH SGD

if signal > noise

(gradient) (variance)

update

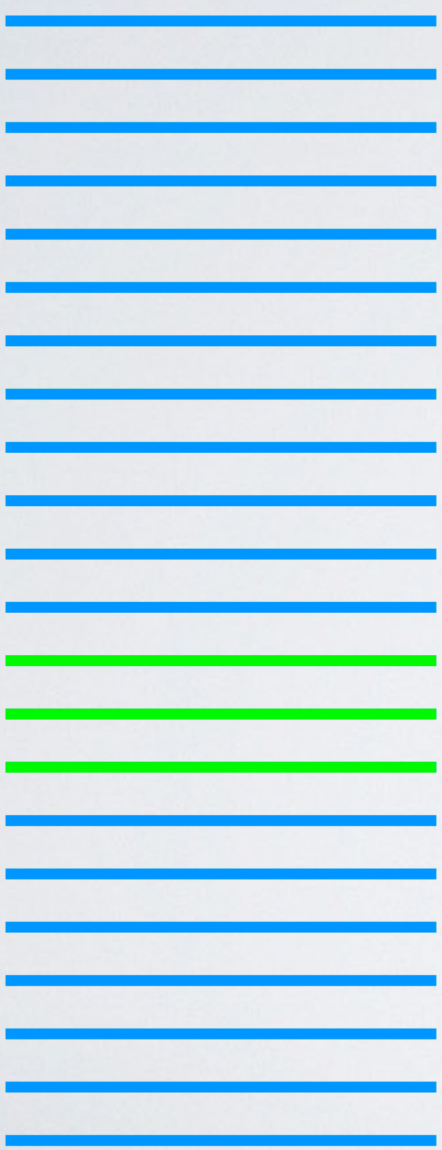
pick batch *B*



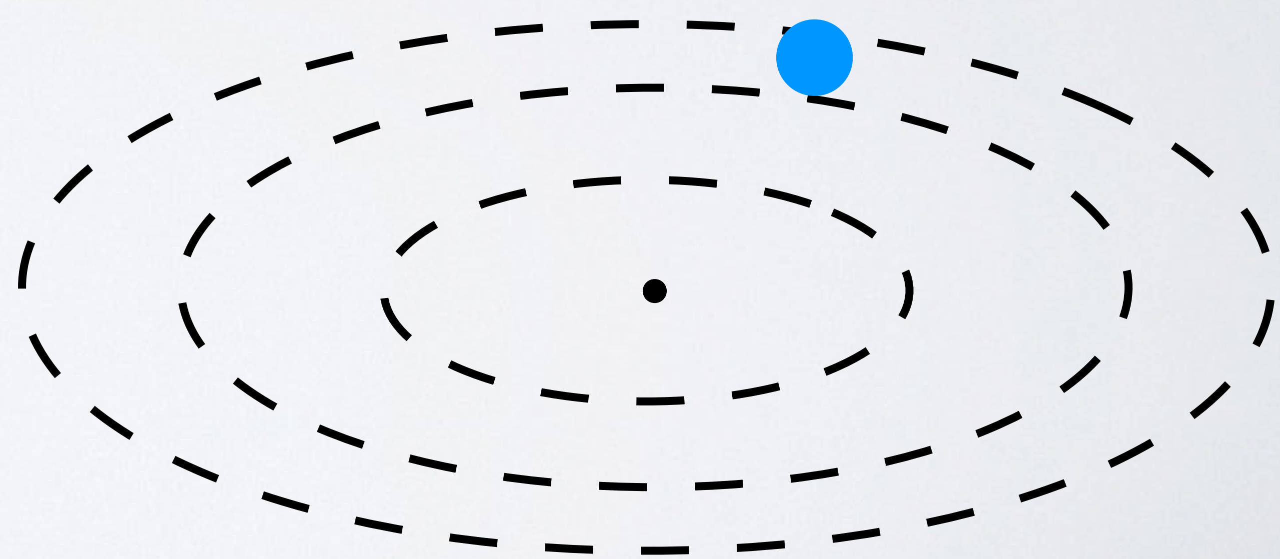
BIG BATCH SGD

if signal > noise
(gradient) (variance)

update



} pick batch *B*

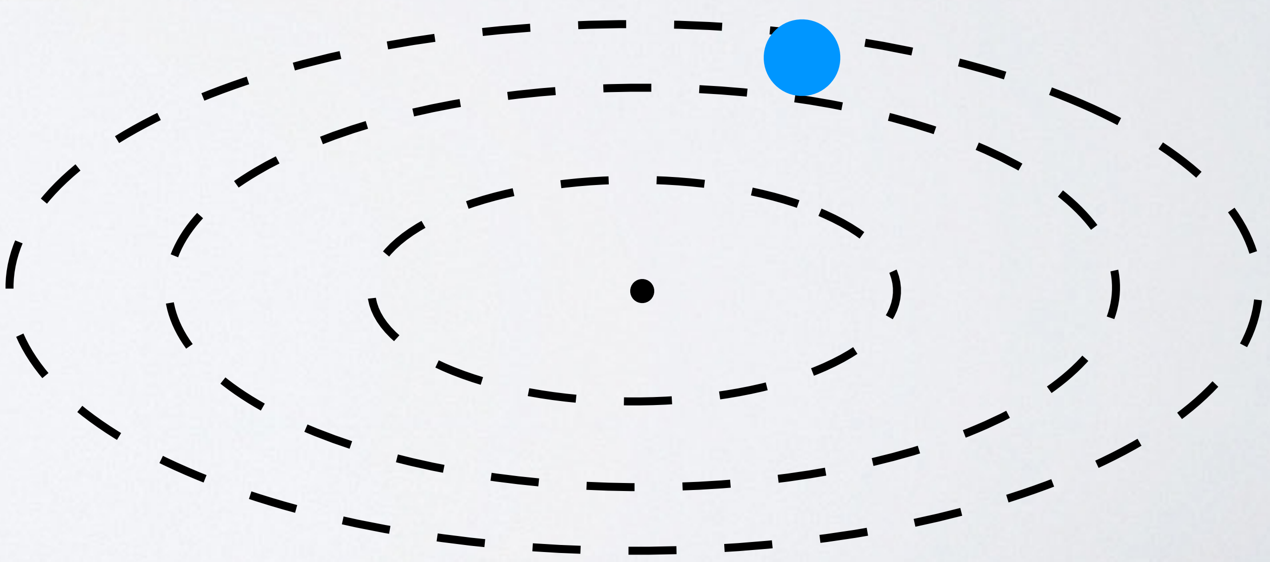


BIG BATCH SGD

if signal (gradient) > noise (variance)

update

} pick batch B

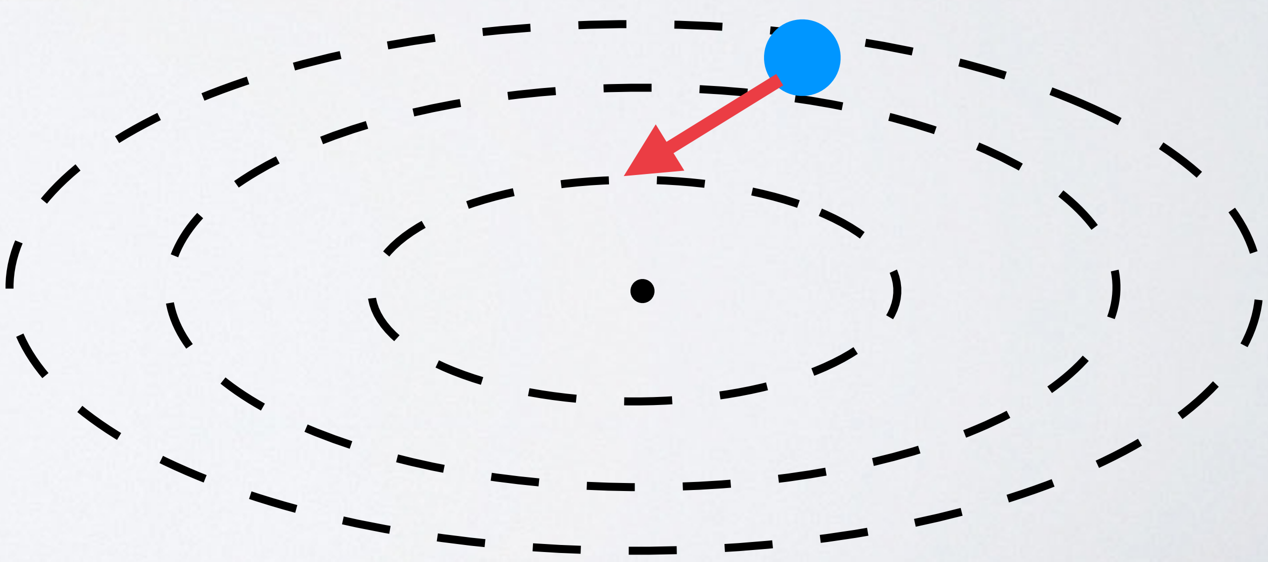


BIG BATCH SGD

if signal > noise
(gradient) (variance)

update

} pick batch *B*

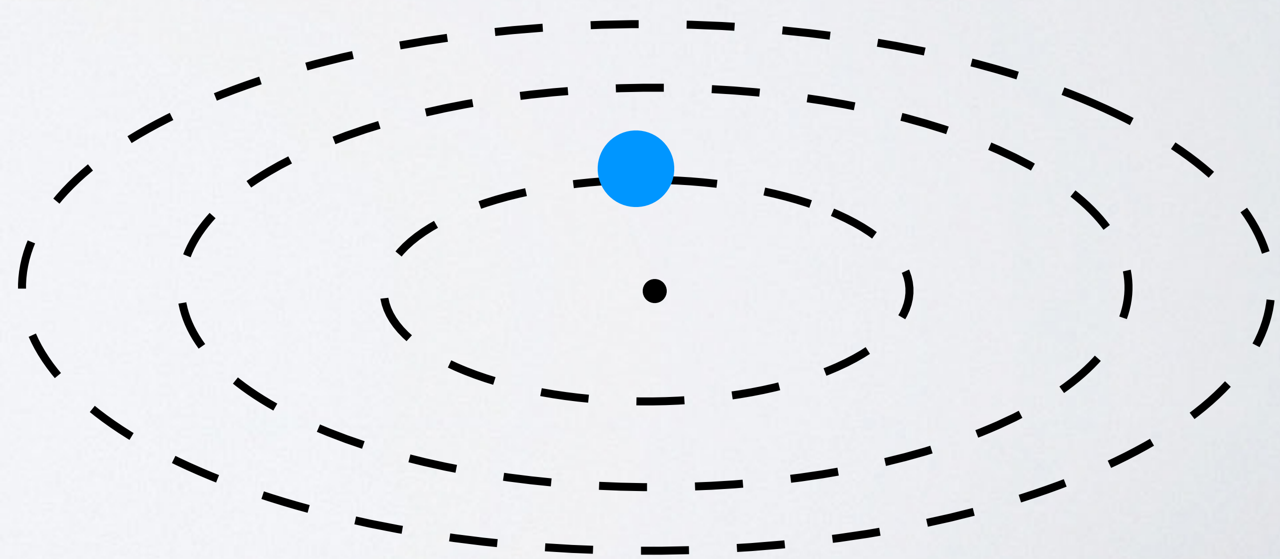


BIG BATCH SGD

if signal > noise
(gradient) (variance)

update

} pick batch *B*

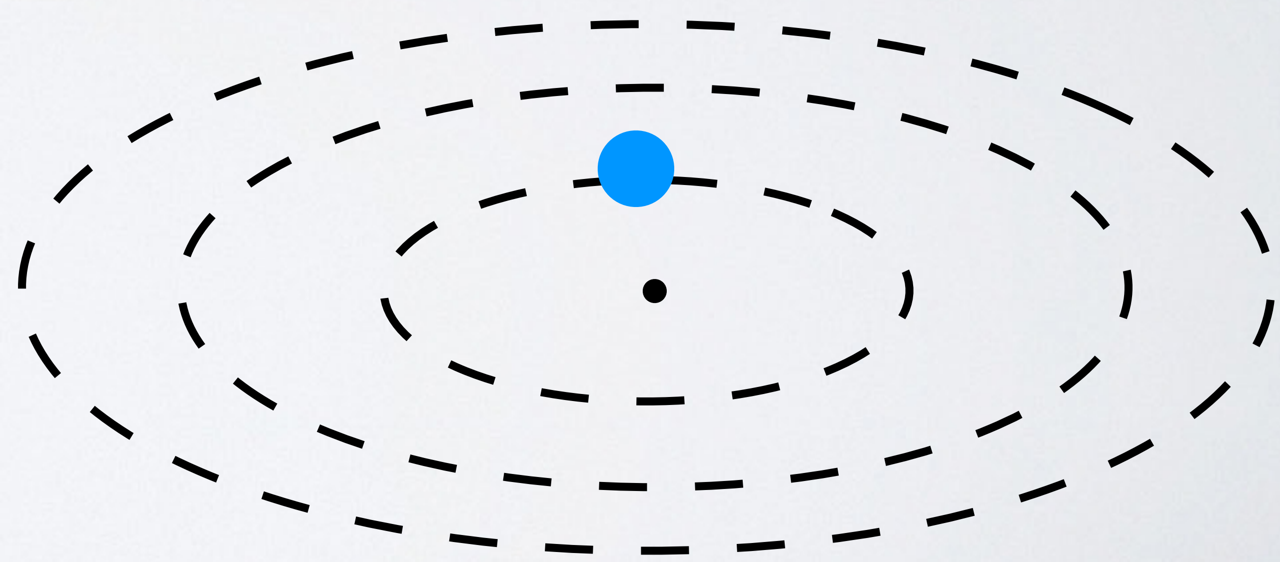


BIG BATCH SGD

if signal > noise
(gradient) (variance)

update

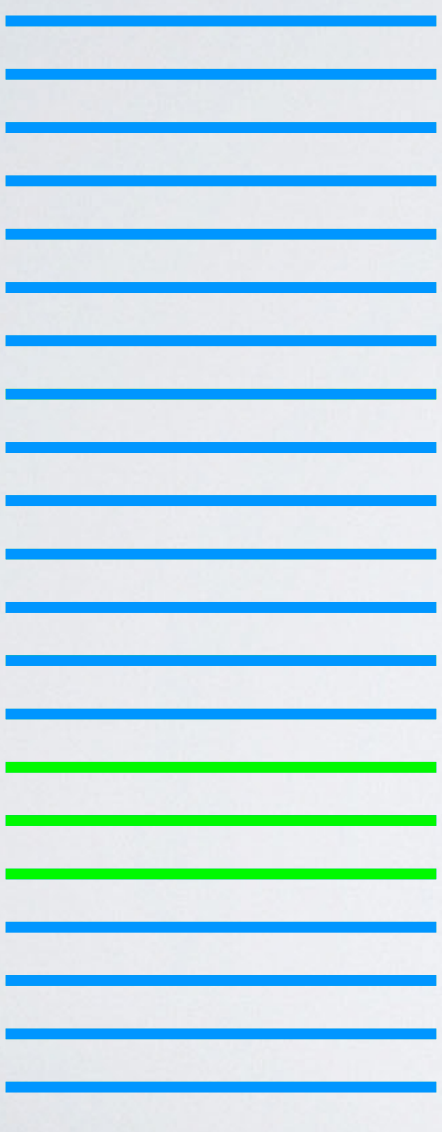
} pick batch *B*



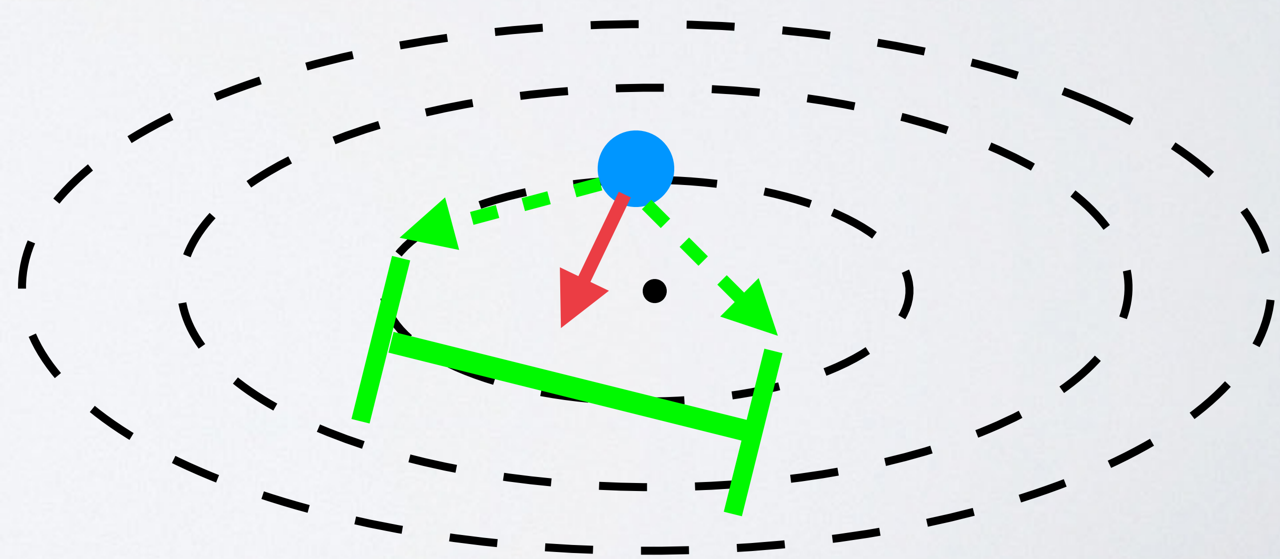
BIG BATCH SGD

if signal > noise
(gradient) (variance)

update



} pick batch *B*



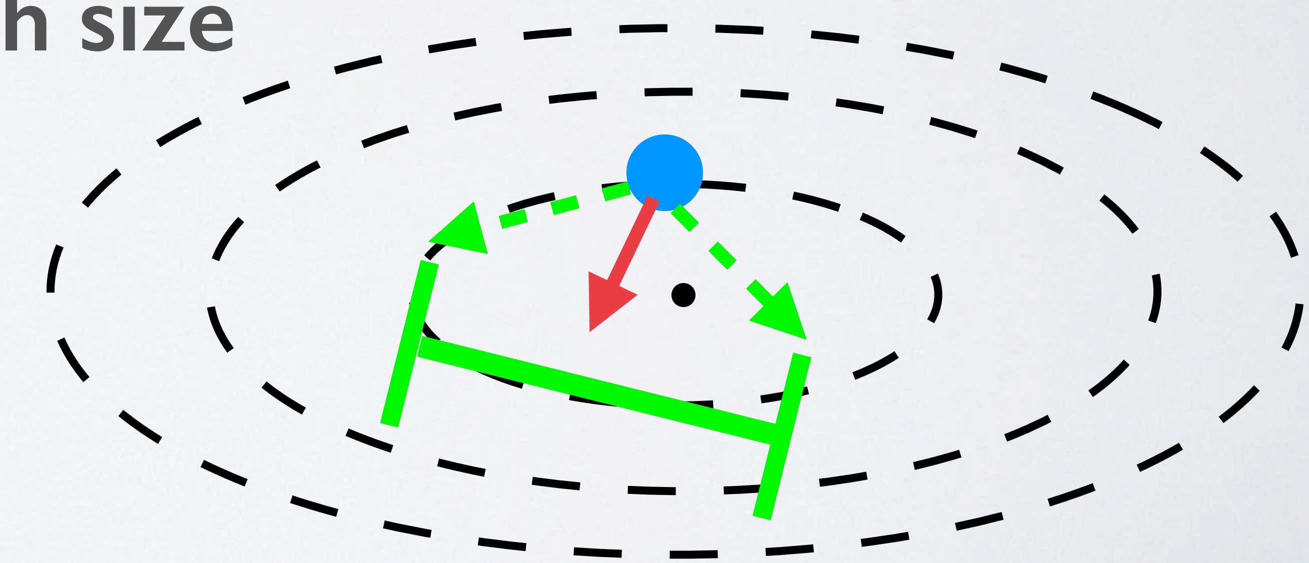
BIG BATCH SGD

if signal > noise
(gradient) (variance)

update

otherwise
increase batch size

} pick batch *B*

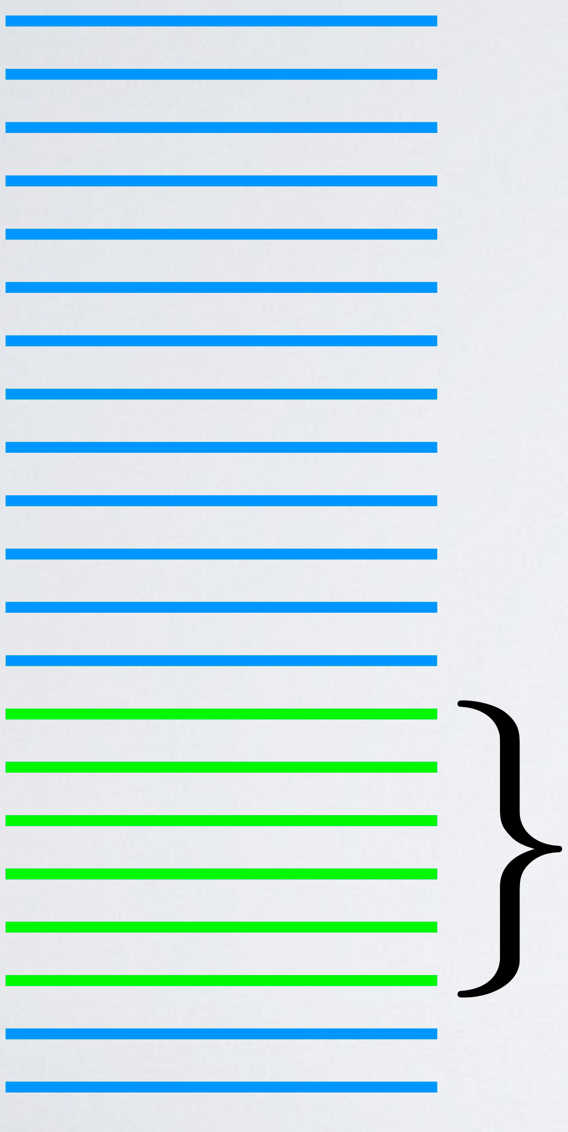


BIG BATCH SGD

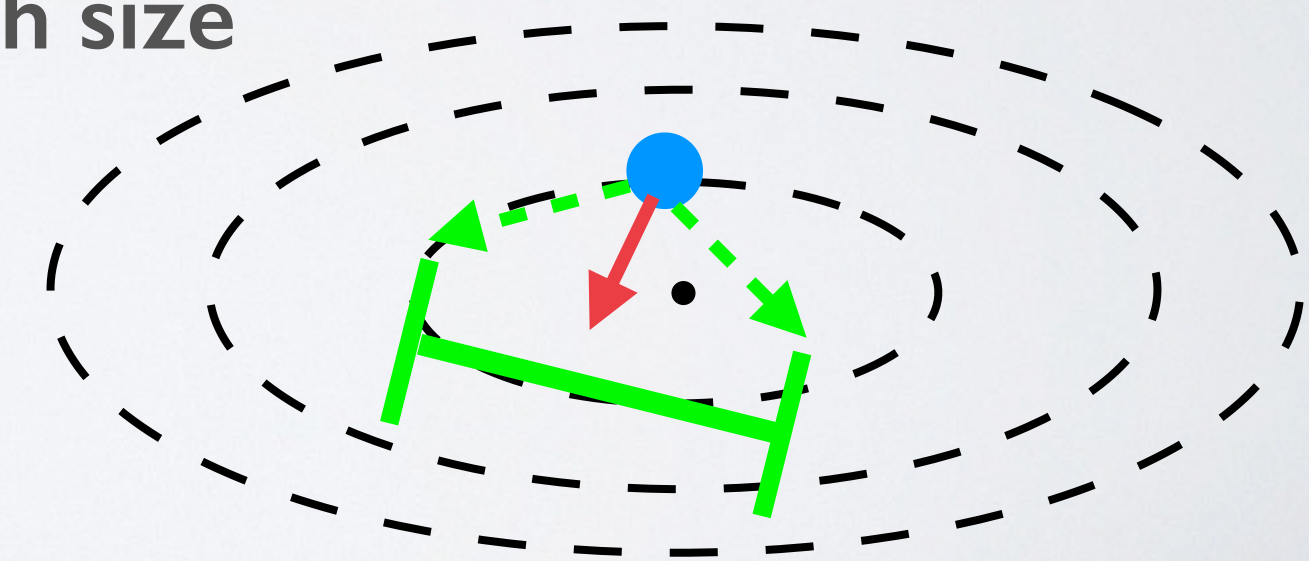
if signal > noise
(gradient) (variance)

update

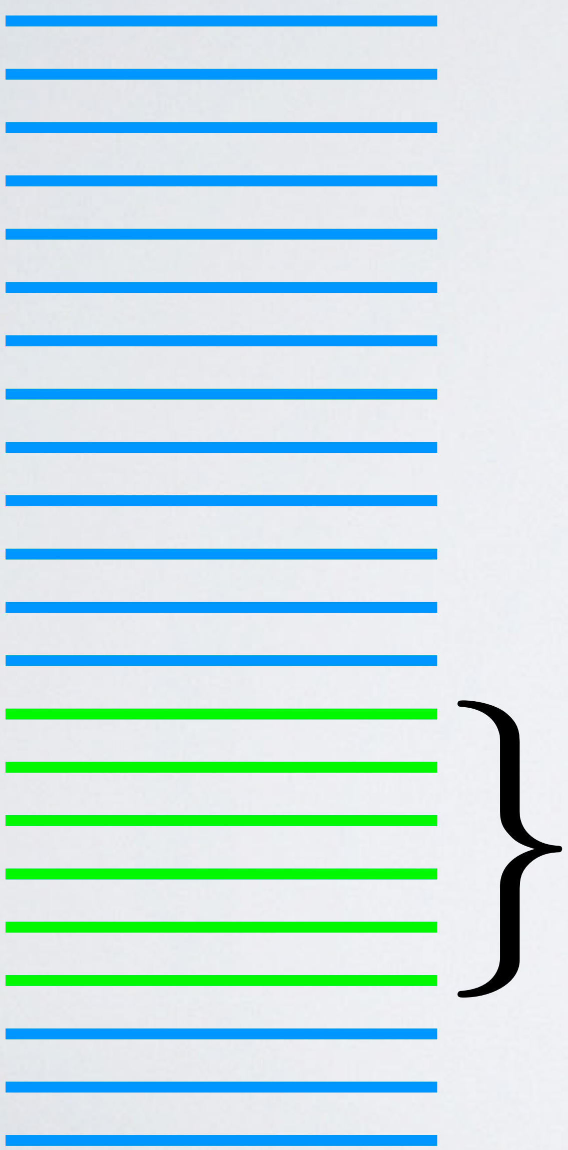
otherwise
increase batch size



pick batch *B*



BIG BATCH SGD

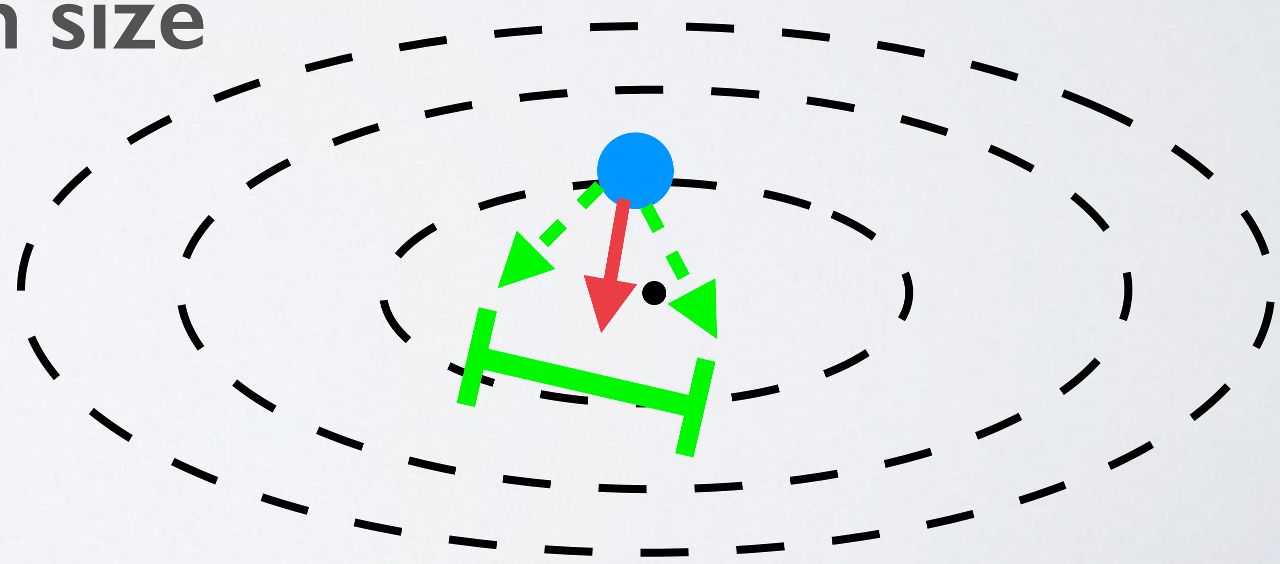


if signal > noise
(gradient) (variance)

update

otherwise
increase batch size

pick batch *B*



BIG BATCH SGD

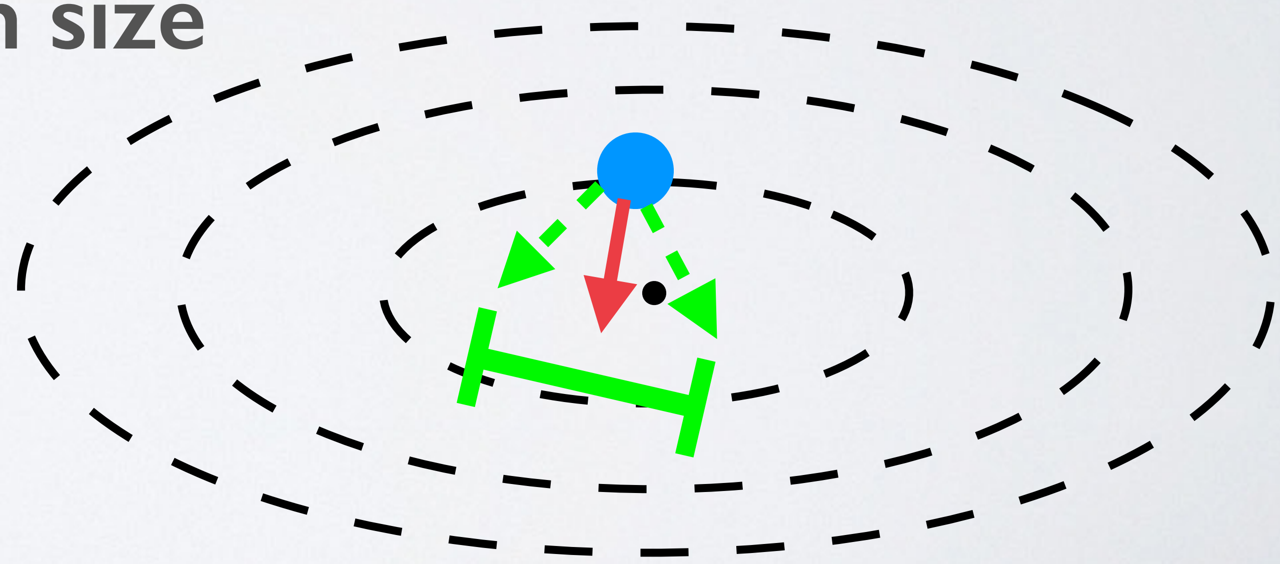
if signal (gradient) > noise (variance)

update

otherwise

increase batch size

pick batch B

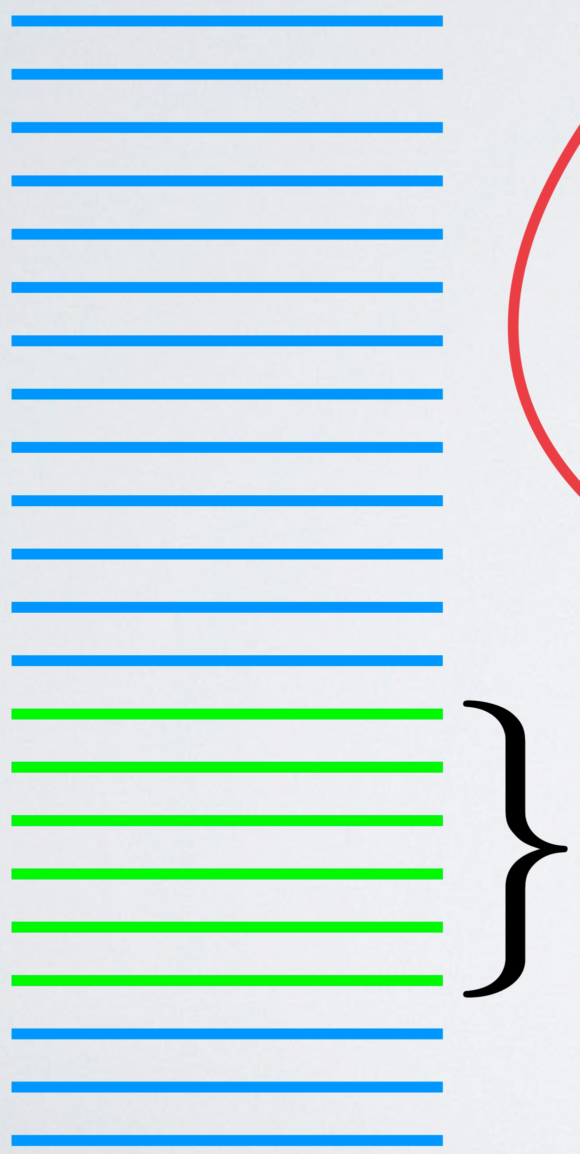


BIG BATCH SGD

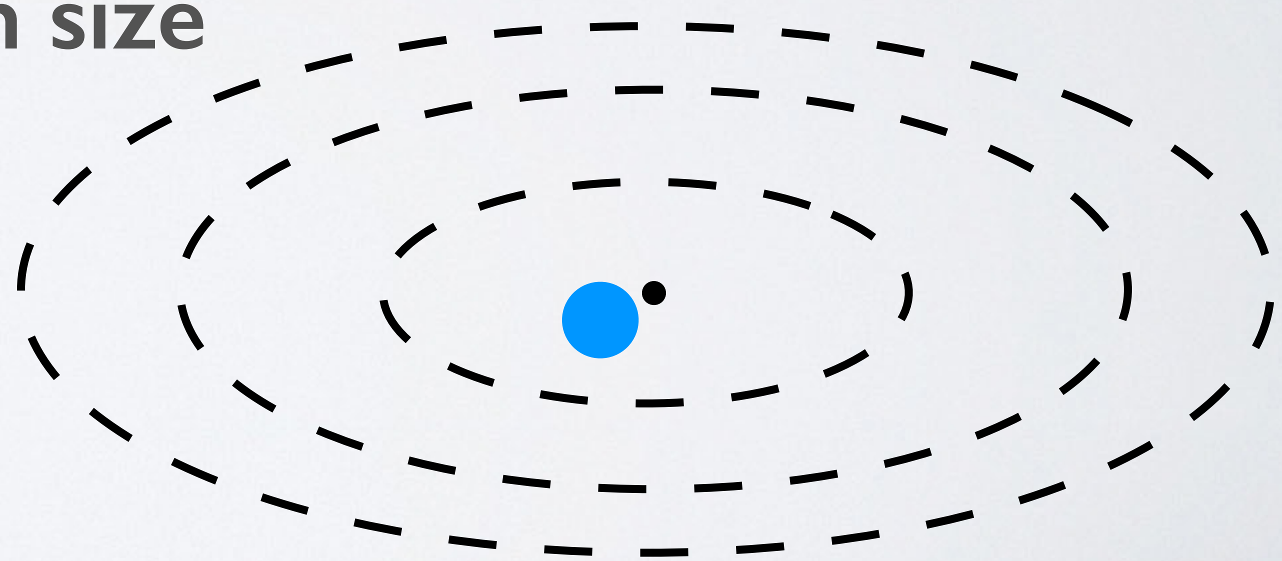
if signal > noise
(gradient) (variance)

update

otherwise
increase batch size



pick batch B



BIG BATCH SGD

On each iteration t

BIG BATCH SGD

On each iteration t

- Estimate size of gradient error by computing **variance**

BIG BATCH SGD

On each iteration t

- Estimate size of gradient error by computing **variance**
- Pick batch *B* large enough such that

$$\theta^2 \mathbb{E}\|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 \geq \frac{1}{|\mathcal{B}_t|} \operatorname{Tr} \operatorname{Var}_z \nabla f(x_t, z)$$

where $\theta \in (0,1)$

BIG BATCH SGD

On each iteration t

- Estimate size of gradient error by computing **variance**
- Pick batch *B* large enough such that

$$\theta^2 \mathbb{E}\|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 \geq \frac{1}{|\mathcal{B}_t|} \mathrm{Tr}\, \mathrm{Var}_z \nabla f(x_t, z)$$

where $\theta \in (0, 1)$

- Choose stepsize α_t

BIG BATCH SGD

On each iteration t

- Estimate size of gradient error by computing **variance**
- Pick batch *B* large enough such that

$$\theta^2 \mathbb{E}\|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 \geq \frac{1}{|\mathcal{B}_t|} \mathrm{Tr}\,\mathrm{Var}_z \nabla f(x_t, z)$$

where $\theta \in (0,1)$

- Choose stepsize α_t
- Update

$$x_{t+1} = x_t - \alpha_t \nabla \ell_{\mathcal{B}_t}(x_t)$$

BIG BATCH SGD

On each iteration t

- Estimate size of gradient error by computing **variance**
- Pick batch *B* large enough such that

$$\boxed{\theta^2 \mathbb{E}\|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 \geq \frac{1}{|\mathcal{B}_t|} \mathrm{Tr}\,\mathrm{Var}_z \nabla f(x_t, z)}$$

where $\theta \in (0, 1)$

extra step to SGD

- Choose stepsize α_t
- Update

$$x_{t+1} = x_t - \alpha_t \nabla \ell_{\mathcal{B}_t}(x_t)$$

BIG BATCH SGD

On each iteration t

- Estimate size of gradient error by computing **variance**
- Pick batch *B* large enough such that

$$\theta^2 \mathbb{E}\|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 \geq \boxed{\frac{1}{|\mathcal{B}_t|} \mathrm{Tr}\,\mathrm{Var}_z \nabla f(x_t, z)}$$

where $\theta \in (0,1)$

Can be estimated using the batch *B*

- Choose stepsize α_t
- Update

$$x_{t+1} = x_t - \alpha_t \nabla \ell_{\mathcal{B}_t}(x_t)$$

CONVERGENCE

Assumption: ℓ has L-Lipschitz gradients

$$\ell(x) \leq \ell(y) + \nabla \ell(y)^T (x - y) + \frac{L}{2}\|x - y\|^2$$

Assumption: ℓ satisfies the Polyak-Lojasiewicz (PL) Inequality

$$\|\nabla \ell(x)\|^2 \geq 2\mu(\ell(x) - \ell(x^\star))$$

CONVERGENCE

Assumption: ℓ has L-Lipschitz gradients

$$\ell(x) \leq \ell(y) + \nabla \ell(y)^T (x - y) + \frac{L}{2}\|x - y\|^2$$

Assumption: ℓ satisfies the Polyak-Lojasiewicz (PL) Inequality

$$\|\nabla \ell(x)\|^2 \geq 2\mu(\ell(x) - \ell(x^\star))$$

Theorem

Big Batch SGD converges linearly:

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \left(1 - \frac{\mu}{\beta L}\right) \cdot \mathbb{E}[\ell(x_t) - \ell(x^\star)]$$

with optimal step size $\alpha = \dfrac{1}{\beta L}$ where $\beta = \dfrac{\theta^2 + (1-\theta)^2}{(1-\theta)^2}$

CONVERGENCE RESULTS

Theorem

Big Batch SGD converges linearly:

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \big(1 - \frac{\mu}{\beta L}\big) \cdot \mathbb{E}[\ell(x_t) - \ell(x^\star)]$$

with optimal step size $\alpha = \dfrac{1}{\beta L}$ where $\beta = \dfrac{\theta^2 + (1-\theta)^2}{(1-\theta)^2}$

Per-iteration convergence rate

CONVERGENCE RESULTS

Theorem

Big Batch SGD converges linearly:

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \left(1 - \frac{\mu}{\beta L}\right) \cdot \mathbb{E}[\ell(x_t) - \ell(x^\star)]$$

with optimal step size $\alpha = \dfrac{1}{\beta L}$ where $\beta = \dfrac{\theta^2 + (1-\theta)^2}{(1-\theta)^2}$

$\downarrow$

Per-iteration convergence rate

We show: $\boxed{|\mathcal{B}| > \mathcal{O}(1/\epsilon) \longrightarrow \text{Error} < \epsilon}$

optimal convergence in the infinite data case

ADVANTAGES

ADVANTAGES

- Controlling noise enables **automated stepsize selection**

ADVANTAGES

- Controlling noise enables **automated stepsize selection**
  - Backtracking line search works well!

ADVANTAGES

- Controlling noise enables **automated stepsize selection**
  - Backtracking line search works well!
  - Stepsize schemes using curvature estimates also work!

ADVANTAGES

- Controlling noise enables **automated stepsize selection**
  - Backtracking line search works well!
  - Stepsize schemes using curvature estimates also work!
- More accurate gradients enable **automatic stopping conditions**

ADVANTAGES

- Controlling noise enables **automated stepsize selection**
  - Backtracking line search works well!
  - Stepsize schemes using curvature estimates also work!
- More accurate gradients enable **automatic stopping conditions**
- Bigger batches are **better in parallel/distributed settings**

BACKTRACKING LINE SEARCH

Also referred to as **Armijo Line Search**

Measures **sufficient decrease condition** of objective function

For regular (deterministic) gradient descent:

$$\ell(x_{t+1}) \quad \leq \quad \ell(x_t) \quad - \quad c\alpha_t \|\nabla \ell(x_t)\|^2$$

BACKTRACKING LINE SEARCH

Also referred to as **Armijo Line Search**

Measures **sufficient decrease condition** of objective function

For regular (deterministic) gradient descent:

$$\underbrace{\ell(x_{t+1})}_{\text{new objective}} \leq \underbrace{\ell(x_t)}_{\text{current objective}} - \underbrace{c\alpha_t \|\nabla \ell(x_t)\|^2}_{\text{sufficient decrease}}$$

BACKTRACKING LINE SEARCH

Also referred to as **Armijo Line Search**

Measures **sufficient decrease condition** of objective function

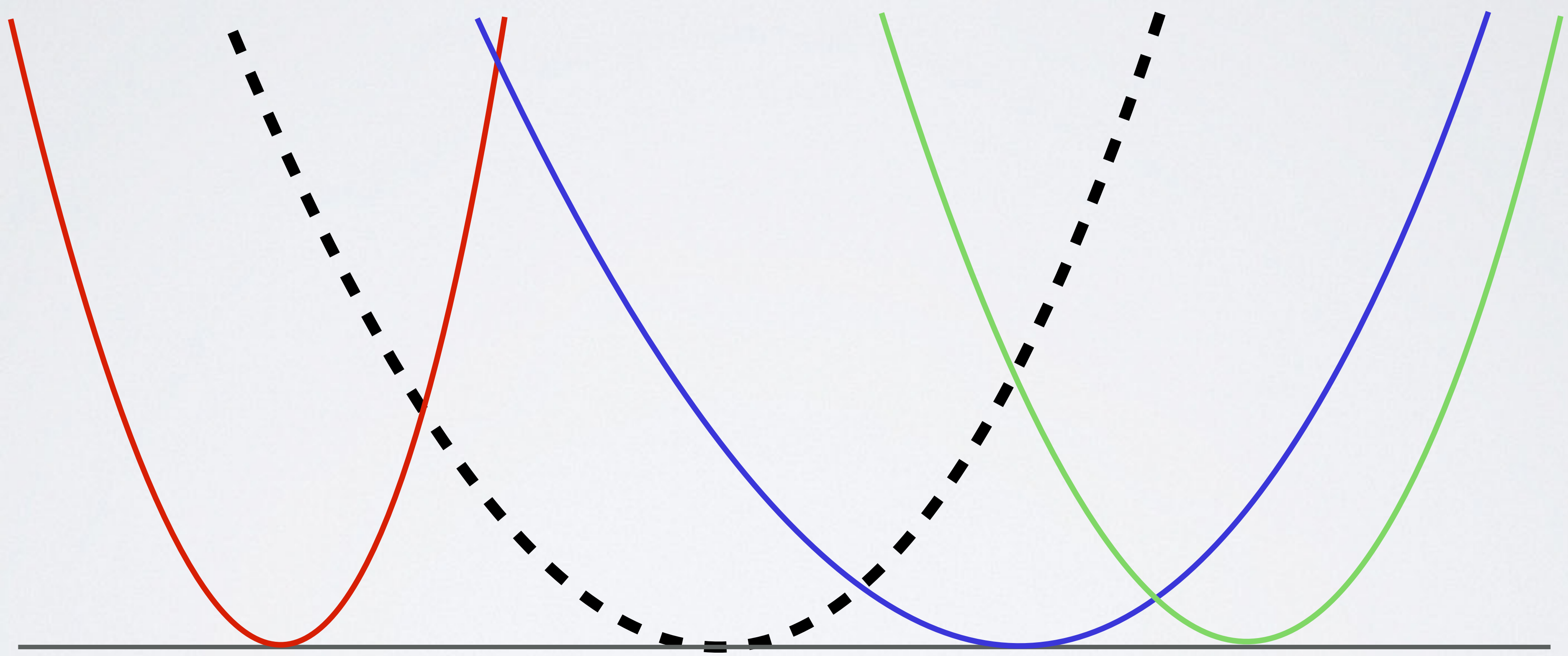
For regular (deterministic) gradient descent:

$$\underbrace{\ell(x_{t+1})}_{\text{new objective}} \leq \underbrace{\ell(x_t)}_{\text{current objective}} - \underbrace{c\alpha_t \|\nabla \ell(x_t)\|^2}_{\text{sufficient decrease}}$$

If this fails, **decrease stepsize** and check again

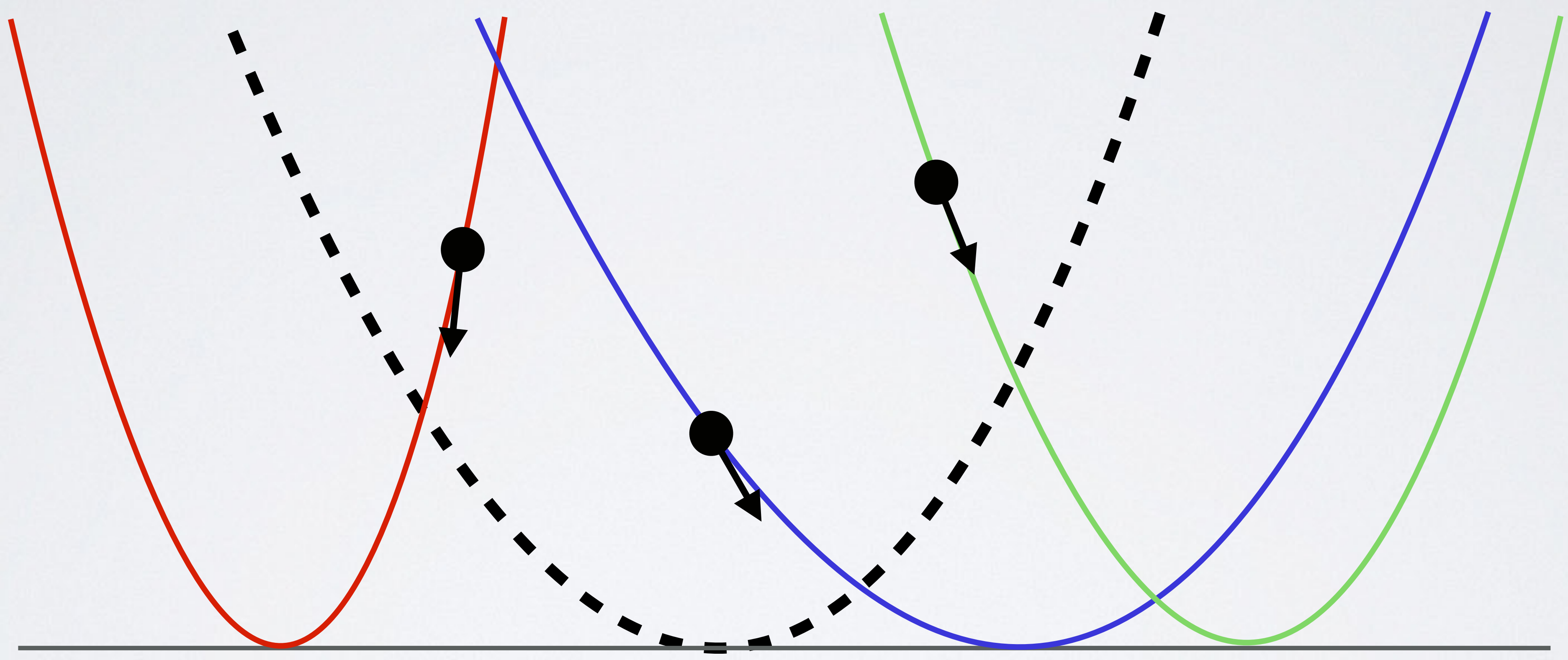
BACKTRACKING WITH SGD

Measures sufficient decrease condition on **individual functions**



BACKTRACKING WITH SGD

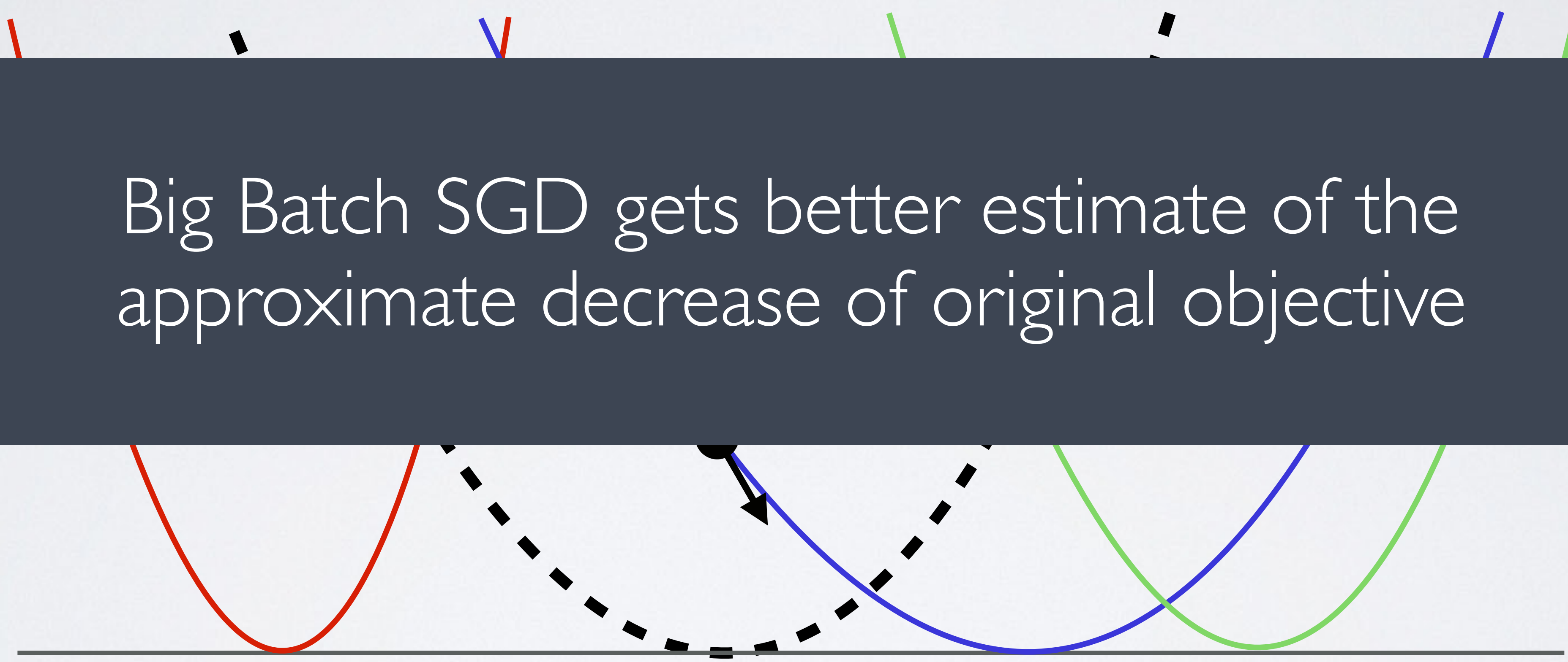
Measures sufficient decrease condition on **individual functions**



Moves to the optimum of individual functions, not the global average

BACKTRACKING WITH SGD

Measures sufficient decrease condition on **individual functions**



Big Batch SGD gets better estimate of the approximate decrease of original objective

Moves to the optimum of individual functions, not the global average

BACKTRACKING WORKS WITH BIG BATCH SGD

Decrease stepsize until:

$$\ell_{\mathcal{B}}(x_{t+1}) \leq \ell_{\mathcal{B}}(x_t) - c\alpha_t \|\nabla \ell_{\mathcal{B}}(x_t)\|^2$$

$\downarrow$

Measures a condition of **sufficient decrease** using batch *B*

BACKTRACKING WORKS WITH BIG BATCH SGD

Decrease stepsize until:

$$\ell_{\mathcal{B}}(x_{t+1}) \leq \ell_{\mathcal{B}}(x_t) - c\alpha_t \|\nabla \ell_{\mathcal{B}}(x_t)\|^2$$

$$\Big\downarrow$$

Measures a condition of **sufficient decrease** using batch *B*

Theorem

Big Batch SGD with backtracking line search converges linearly:

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \left(1 - \frac{c\mu}{\beta L}\right)\mathbb{E}[\ell(x_t) - \ell(x^\star)]$$

with initial step size set large enough s.t. $\alpha_0 \geq \dfrac{1}{2\beta L}$

BACKTRACKING WORKS WITH BIG BATCH SGD

Decrease stepsize until:

$$\ell_{\mathcal{B}}(x_{t+1}) \leq \ell_{\mathcal{B}}(x_t) - c\alpha_t \|\nabla \ell_{\mathcal{B}}(x_t)\|^2$$

We also derive optimal stepsizes for Big Batch SGD using Barzilai-Borwein (BB) curvature estimates, with provable guarantees. Check paper for details.

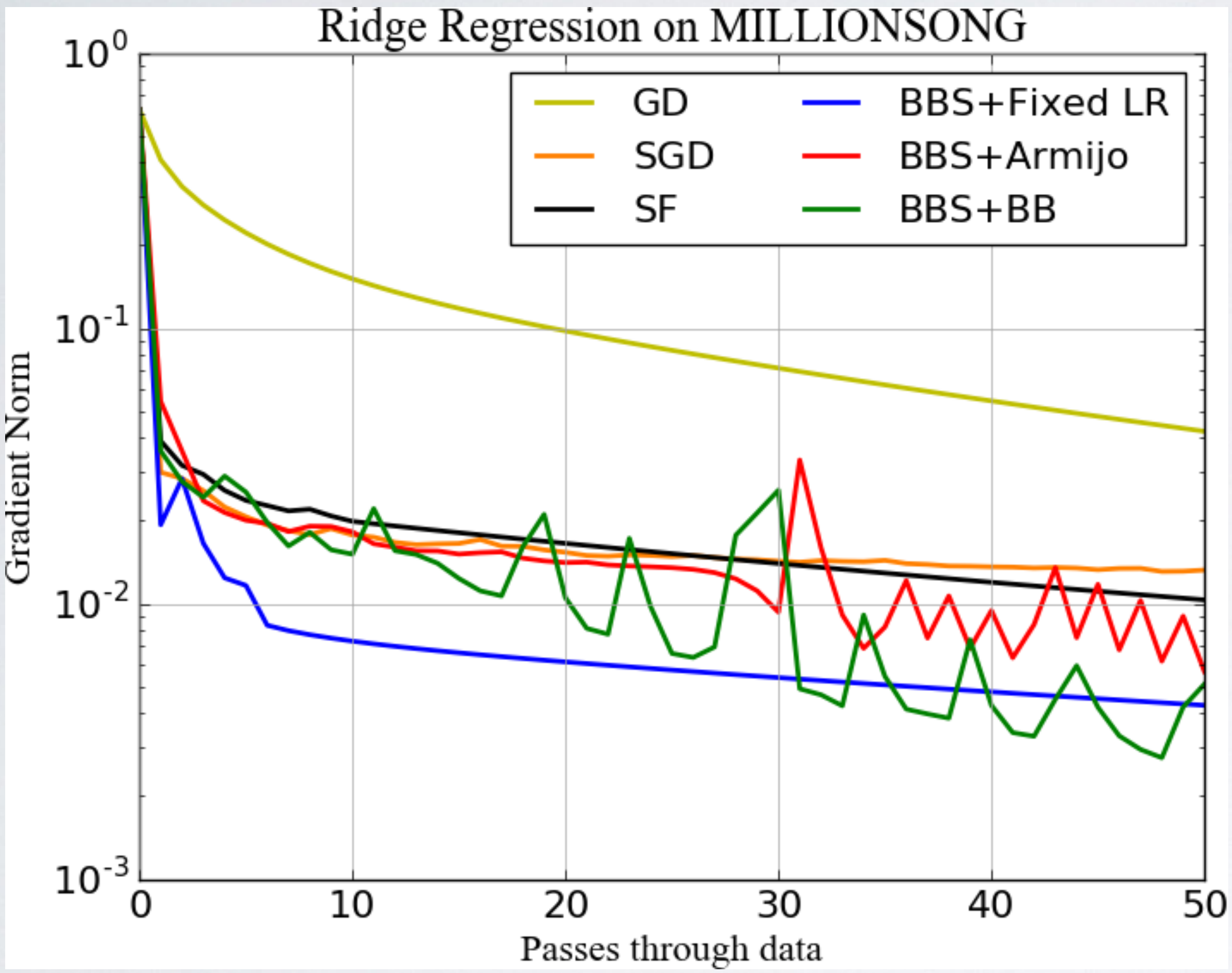
Big Batch SGD with backtracking line search converges linearly:

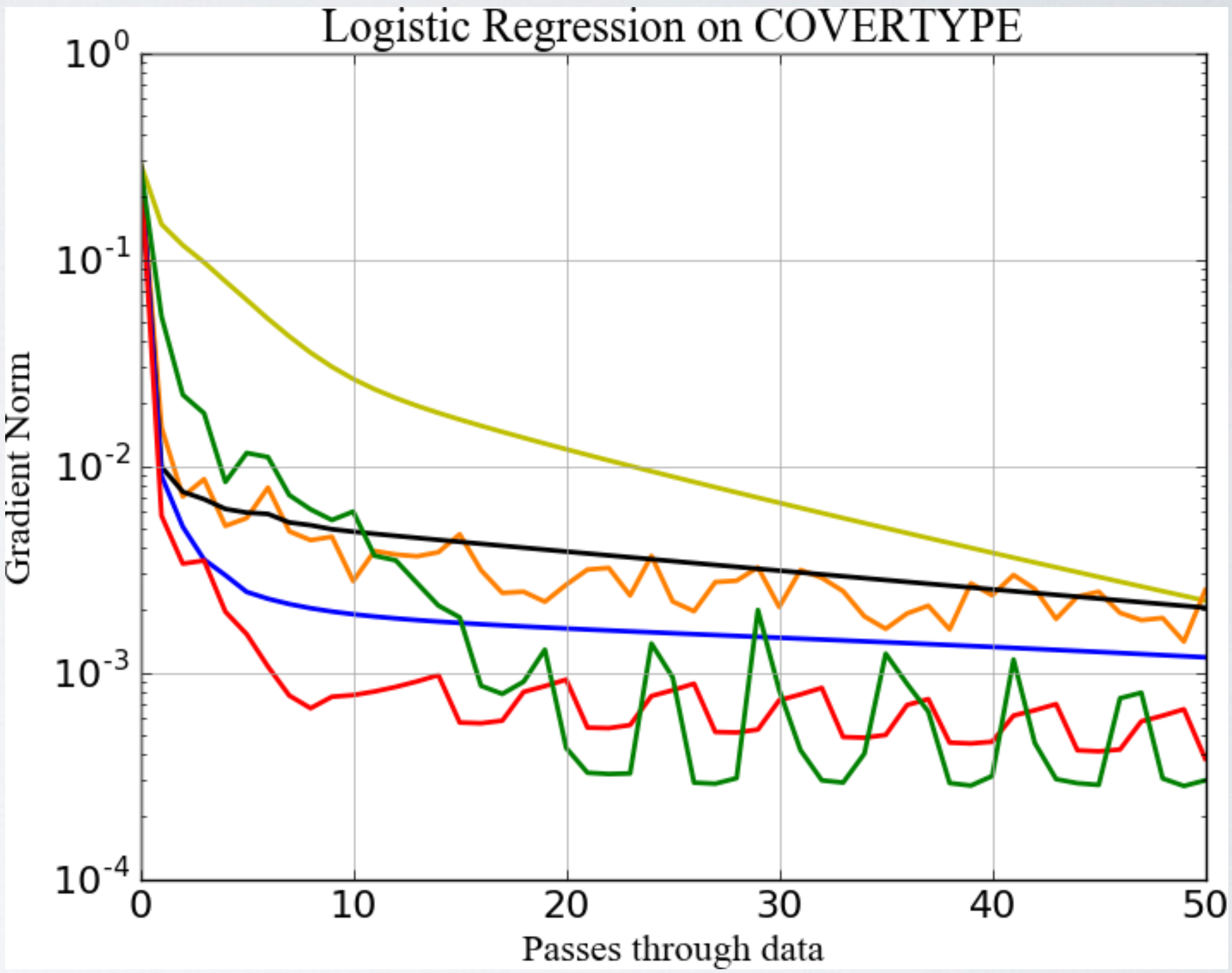
$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \left(1 - \frac{c\mu}{\beta L}\right)\mathbb{E}[\ell(x_t) - \ell(x^\star)]$$

with initial step size set large enough s.t. $\alpha_0 \geq \dfrac{1}{2\beta L}$

CONVEX EXPERIMENTS

Model: Ridge Regression & Logistic Regression



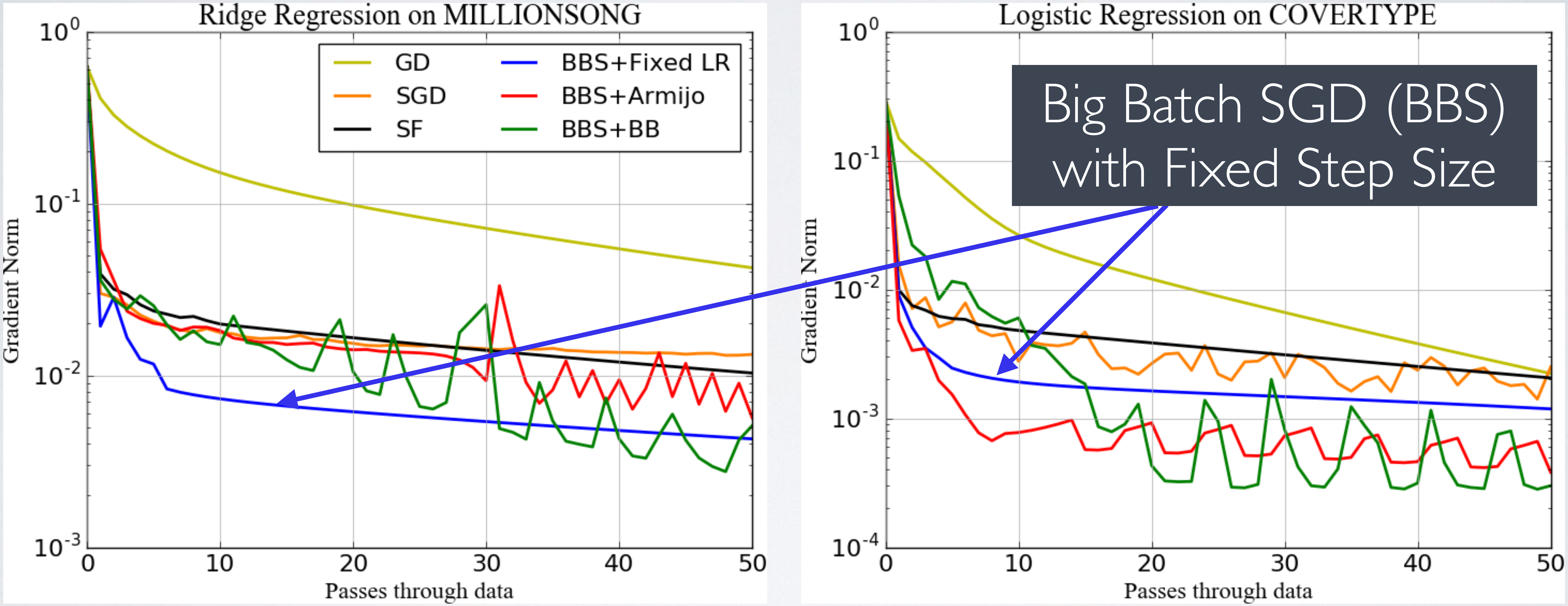


BBS: Big Batch SGD

Proposed methods: **Blue** (Fixed stepsize), **Red** (Backtracking), **Green** (Optimal stepsizes using curvature estimates) curves

CONVEX EXPERIMENTS

Model: Ridge Regression & Logistic Regression

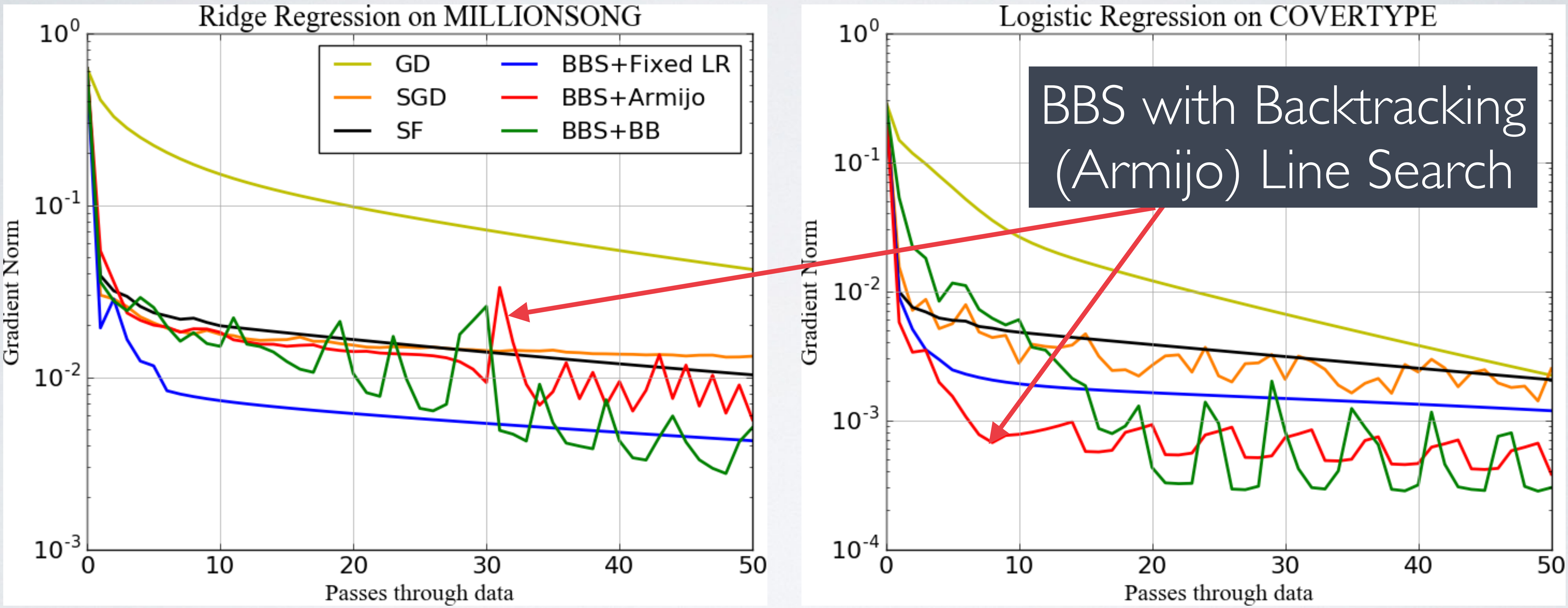


BBS: Big Batch SGD

Proposed methods: **Blue** (Fixed stepsize), **Red** (Backtracking), **Green** (Optimal stepsizes using curvature estimates) curves

CONVEX EXPERIMENTS

Model: Ridge Regression & Logistic Regression

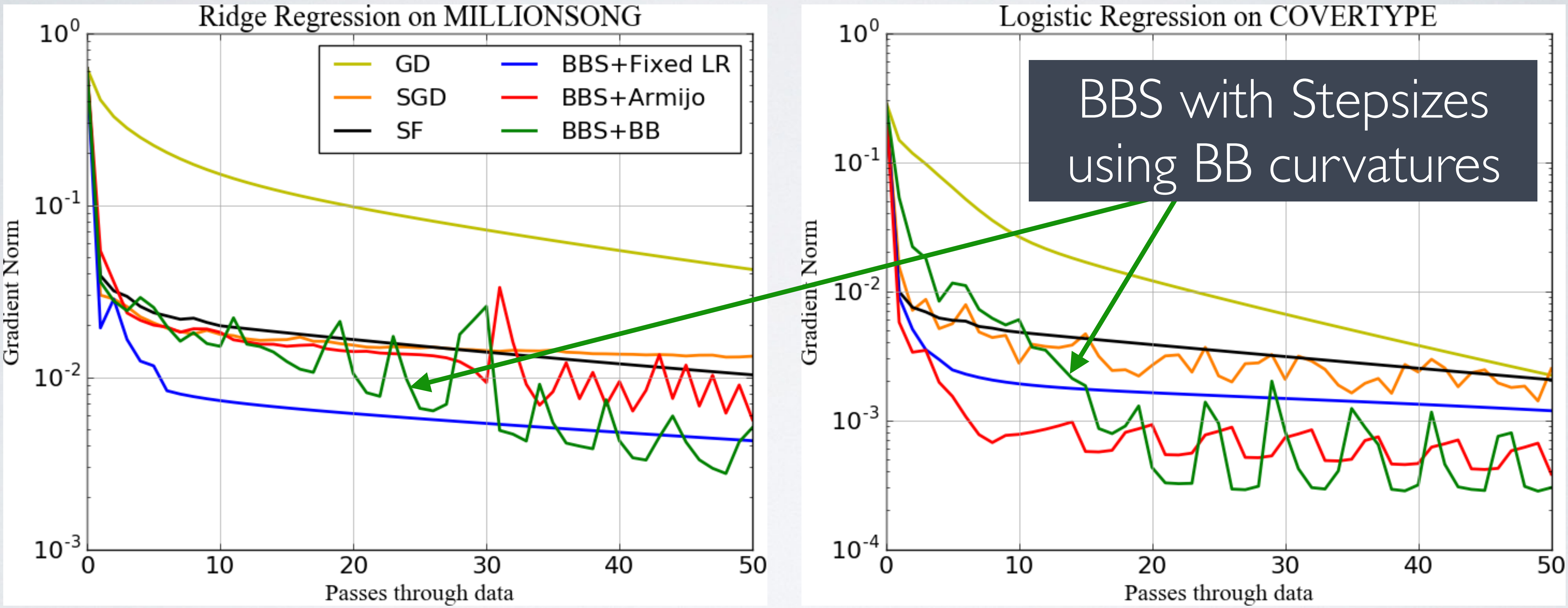


BBS: Big Batch SGD

Proposed methods: **Blue** (Fixed stepsize), **Red** (Backtracking), **Green** (Optimal stepsizes using curvature estimates) curves

CONVEX EXPERIMENTS

Model: Ridge Regression & Logistic Regression

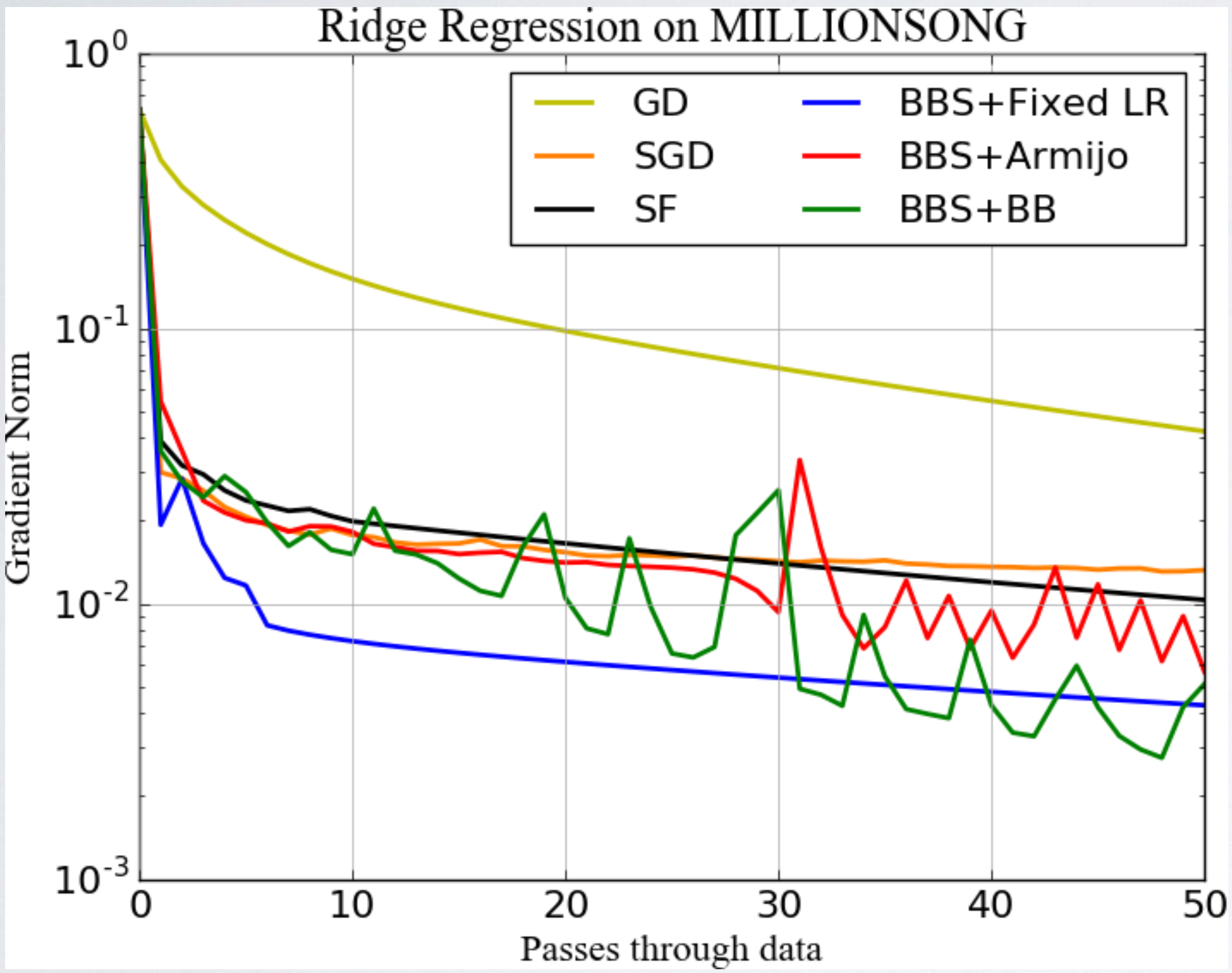


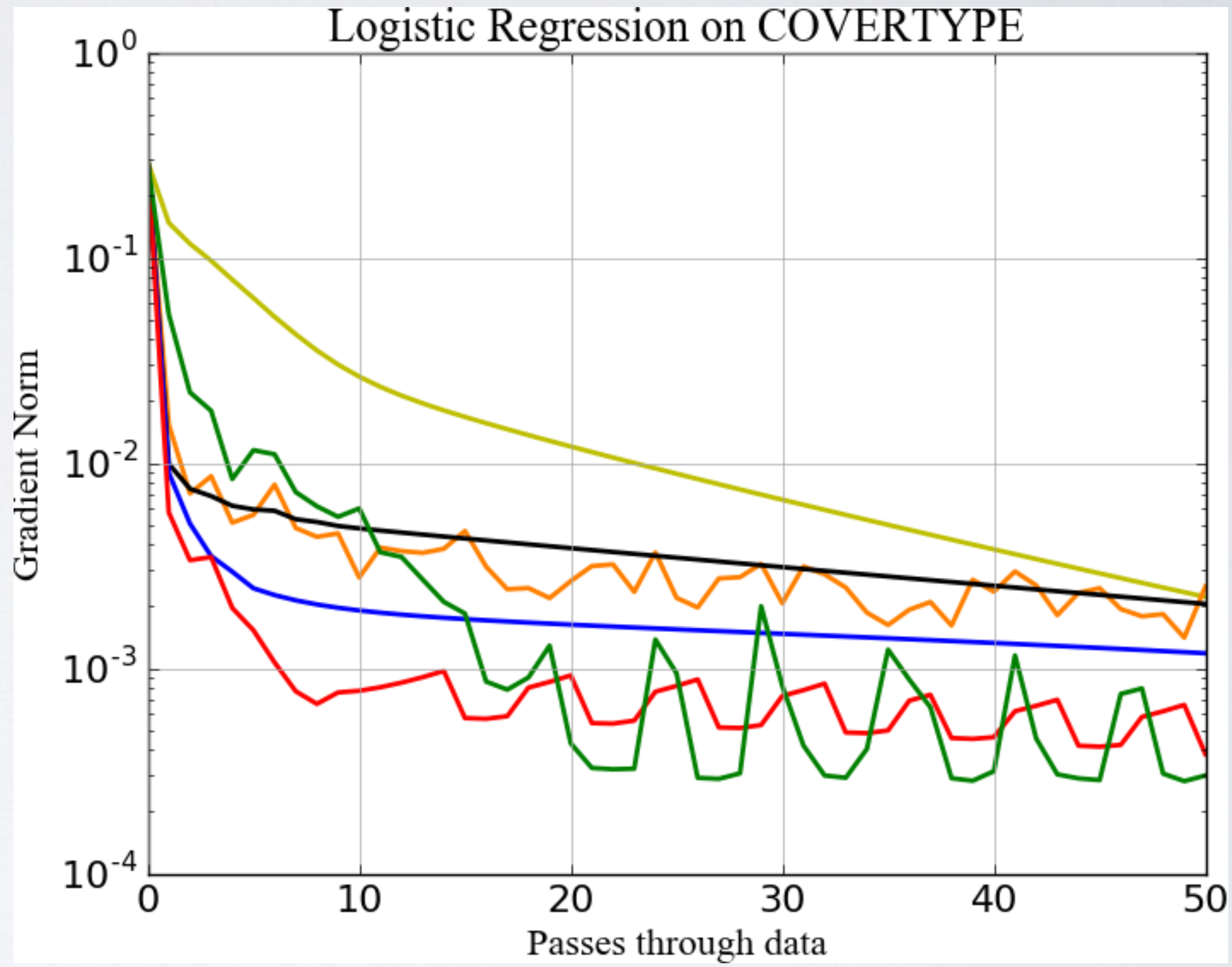
BBS: Big Batch SGD

Proposed methods: **Blue** (Fixed stepsize), **Red** (Backtracking), **Green** (Optimal stepsizes using curvature estimates) curves

CONVEX EXPERIMENTS

Model: Ridge Regression & Logistic Regression



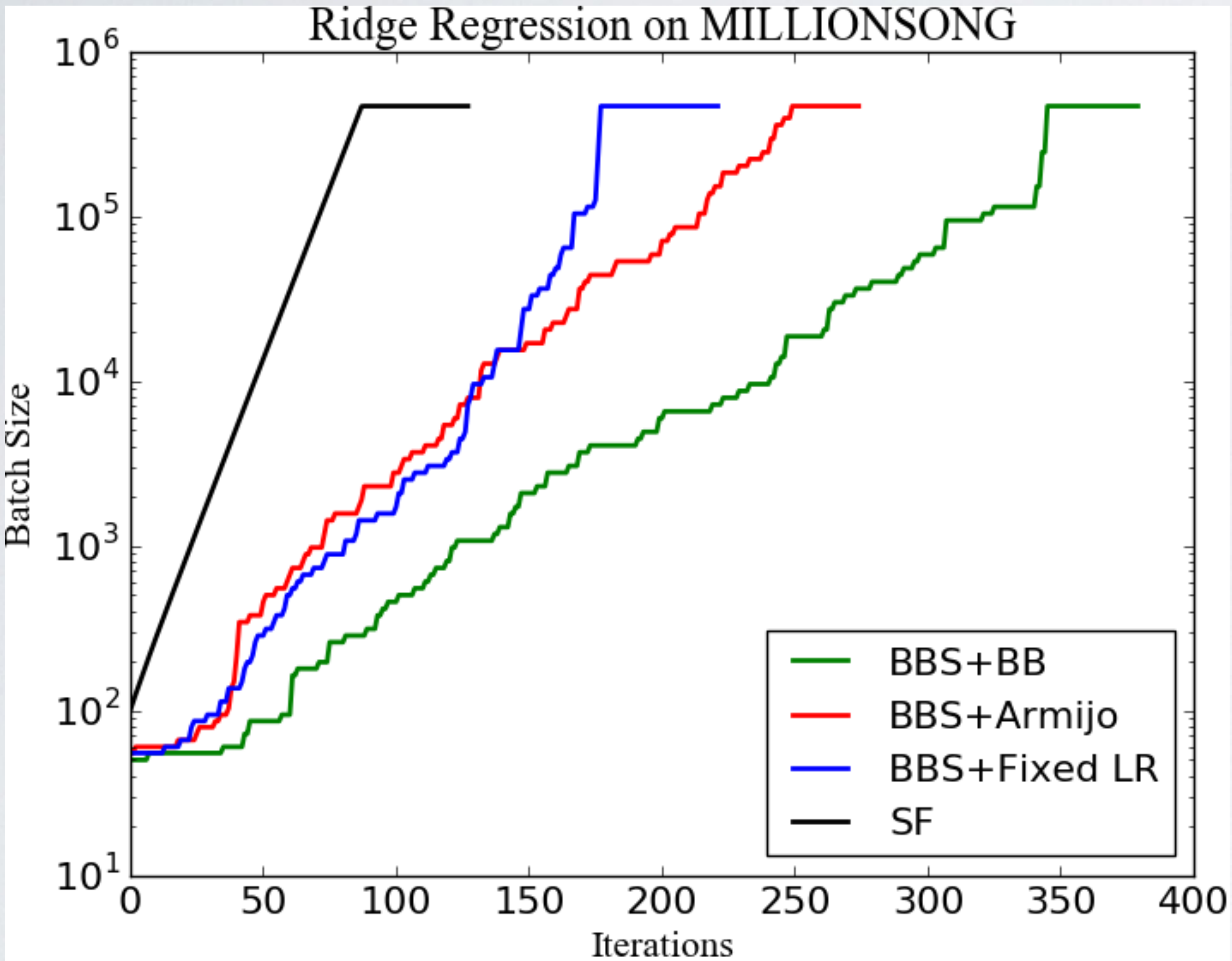


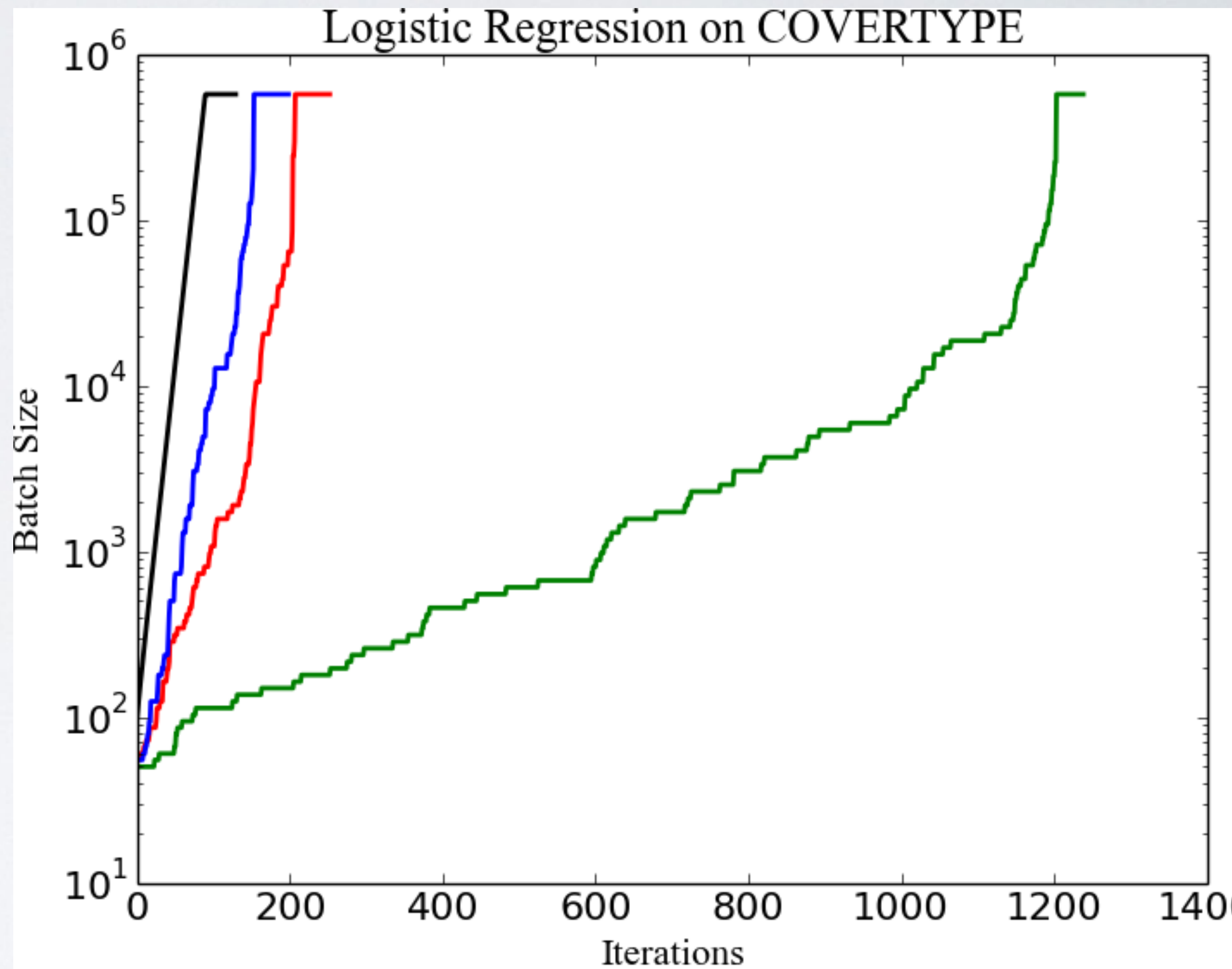
BBS: Big Batch SGD

Proposed methods: **Blue** (Fixed stepsize), **Red** (Backtracking), **Green** (Optimal stepsizes using curvature estimates) curves

CONVEX EXPERIMENTS

Batch Size Increase



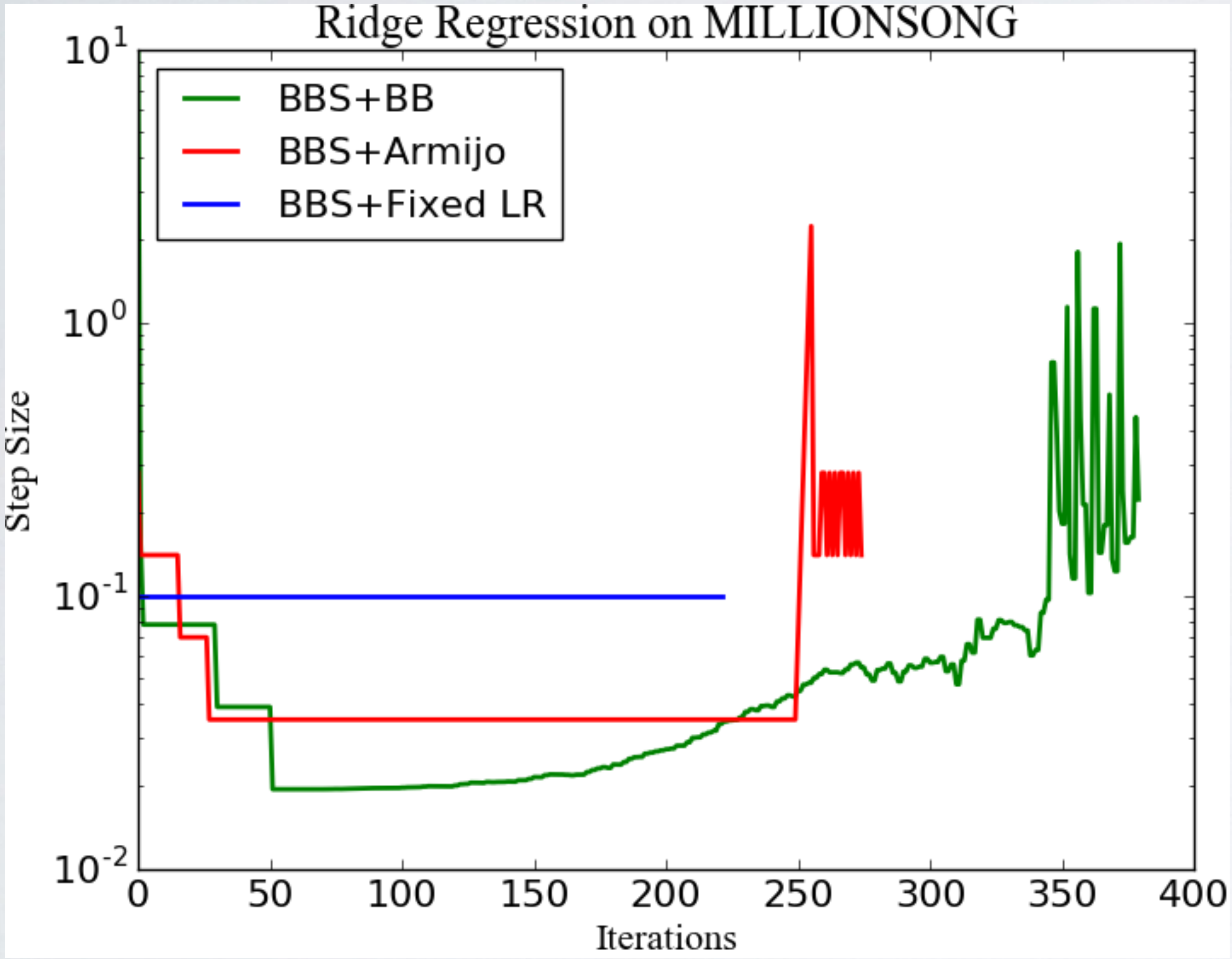


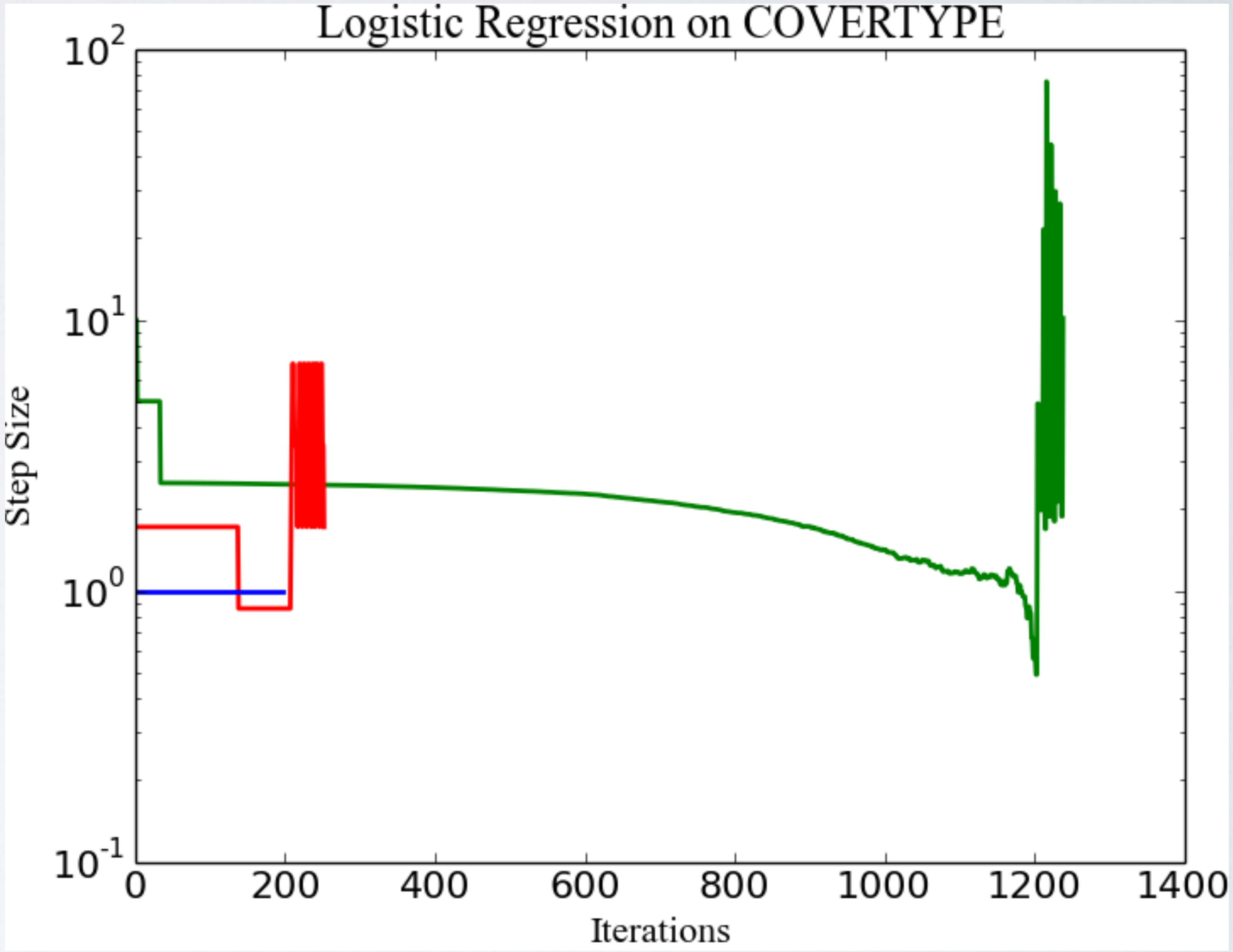
BBS: Big Batch SGD

Proposed methods: **Blue** (Fixed stepsize), **Red** (Backtracking), **Green** (Optimal stepsizes using curvature estamates) curves

CONVEX EXPERIMENTS

Stepsizes Used



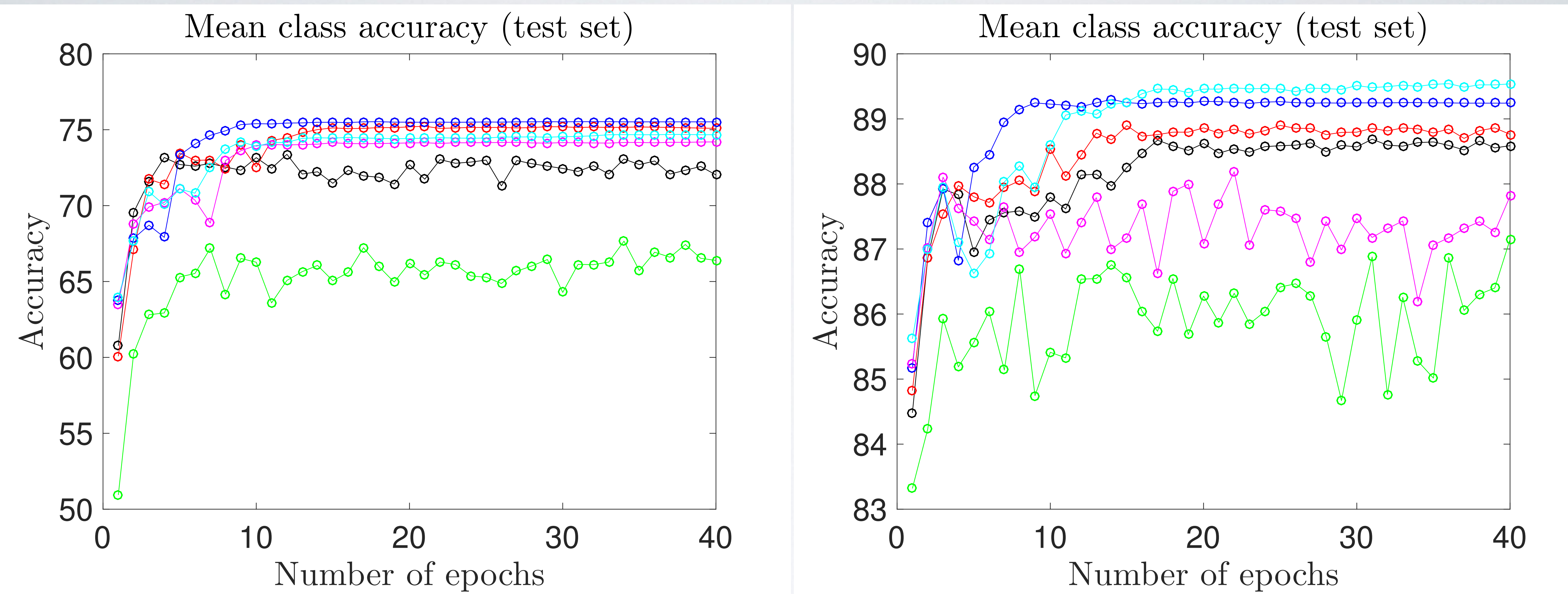


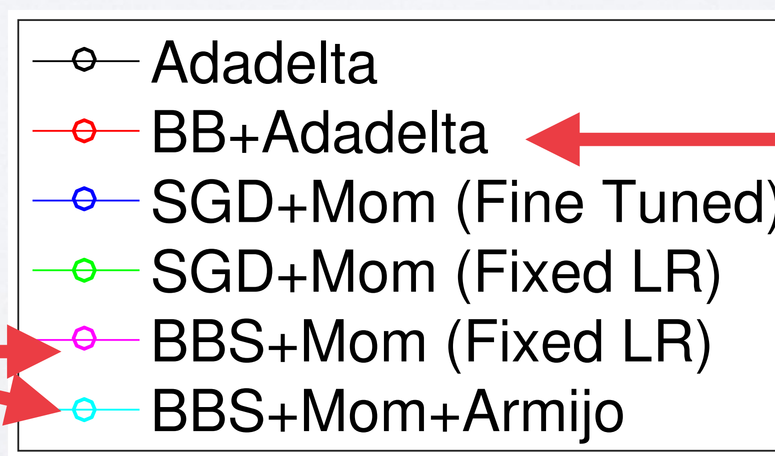
BBS: Big Batch SGD

Proposed methods: **Blue** (Fixed stepsize), **Red** (Backtracking), **Green** (Optimal stepsizes using curvature estamates) curves

4-LAYER CNN

CIFAR-10 (left) & SVHN (right)





Big Batch SGD can also be used with AdaGrad, AdaDelta…

Proposed Methods

TAKEAWAYS

We introduce: **Big Batch SGD**

Adaptively grows batch size over time to maintain a nearly constant signal-to-noise ratio in the gradients

Better control of the noise makes it **easy to automate**

Adaptive stepsize methods work well with this method

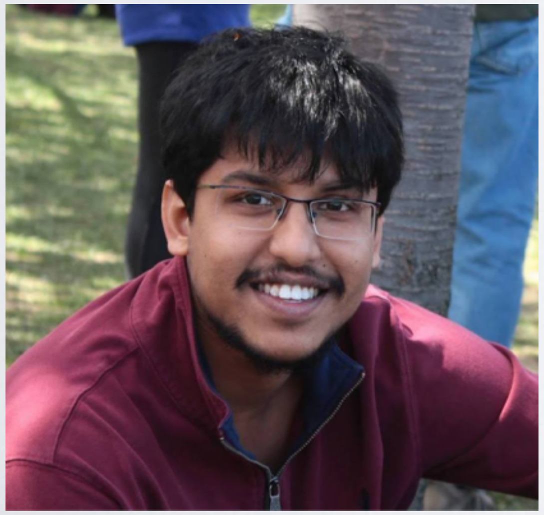
Better for **parallel/distributed settings**

THANKS!

Feel free to get in touch!

Extended version on arXiv:
"Big Batch SGD: Automated Inference using Adaptive Batch Sizes"
https://arxiv.org/abs/1610.05792

email: sohamde@cs.umd.edu
website: https://cs.umd.edu/~sohamde/



Soham De

Abhay Yadav

David Jacobs

Tom Goldstein