Optimal mini-batch and step sizes for SAGA

Nidham Gazagnadou[1,a]

joint work with Robert M. Gower[1] & Joseph Salmon[2]

[1]Télécom ParisTech, France
[2]Université de Montpellier, France

[a]This work was supported by grants from Région Ile-de-France

Introduction

Why Machine Learning (ML)?

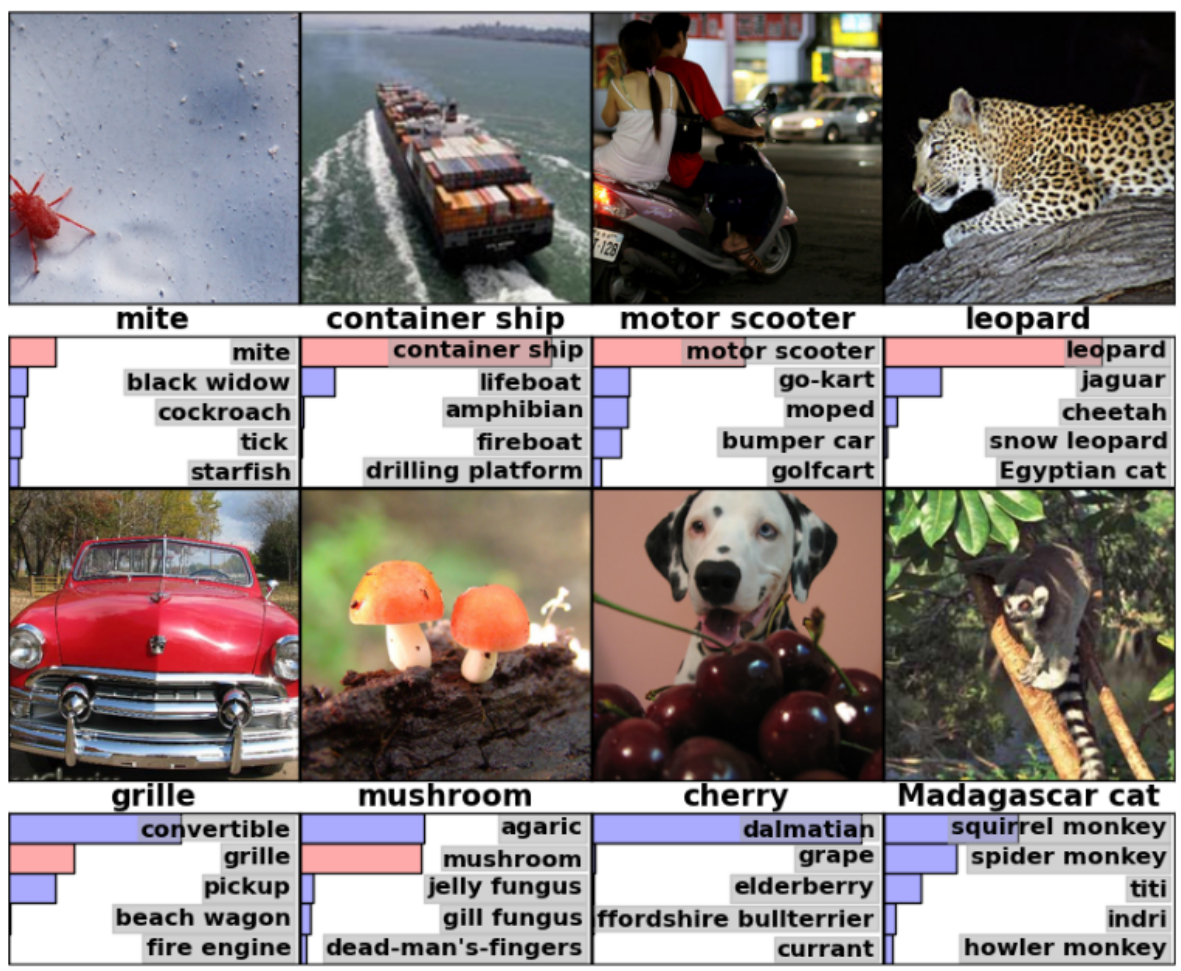
- **"Big" Data framework**
  - **n**: number of observations
  - **d**: dimension of each observation

Why Machine Learning (ML)?

- **"Big" Data framework**
  - **n**: number of observations
  - **d**: dimension of each observation

- Very different data types: numbers, images, text, etc

Why Machine Learning (ML)?



Automatic labelling with AlexNet architecture[1]

[1] Krizhevsky, Sutskever, Hinton, 2012

Why Machine Learning (ML)?

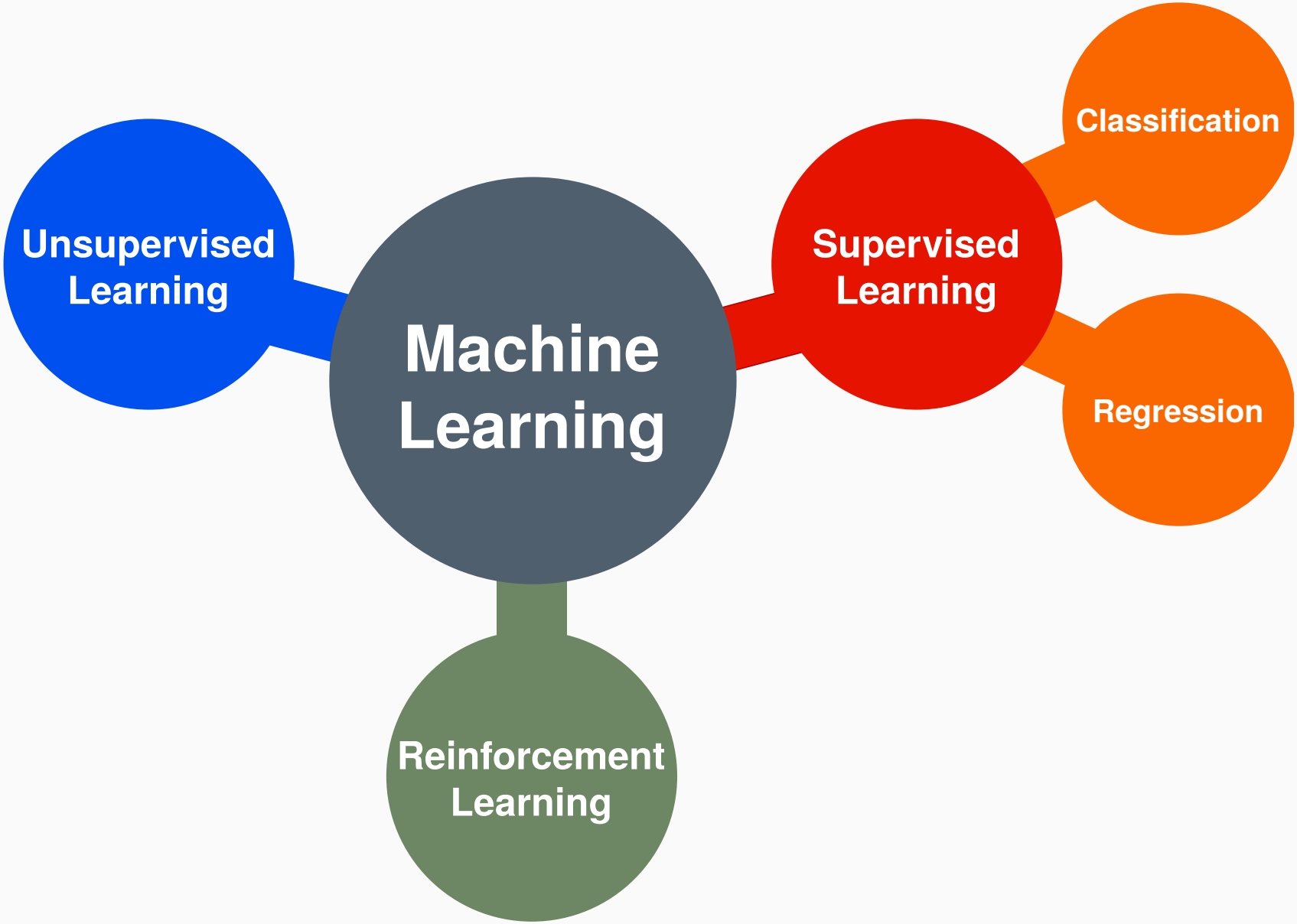
- **"Big" Data framework**
  - **n**: number of observations
  - **d**: dimension of each observation

- Very different data types: numbers, images, text, etc

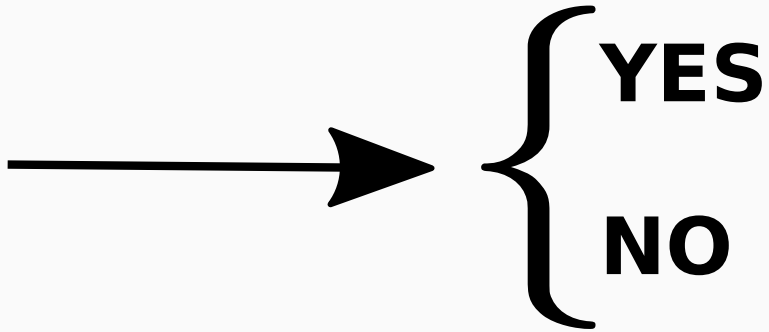
- Main idea of Machine Learning

  Use available data to build models generalizing to unseen one

Different frameworks



Is there a cat in this picture?



Is there a cat in this picture?

YES

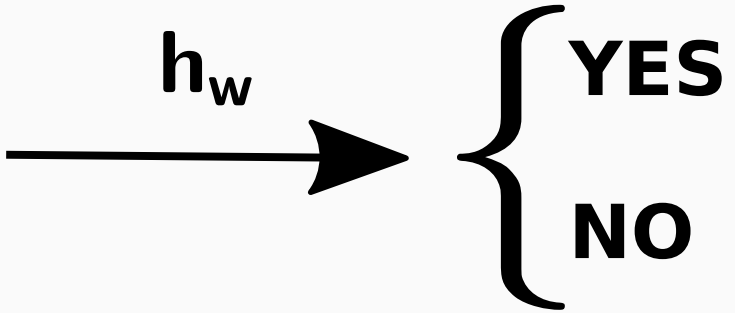
Is there a cat in this picture?

NO

Is there a cat in this picture?

YES

Building a mapping: classification example



Input/Feature
$a \in \mathbb{R}^d$

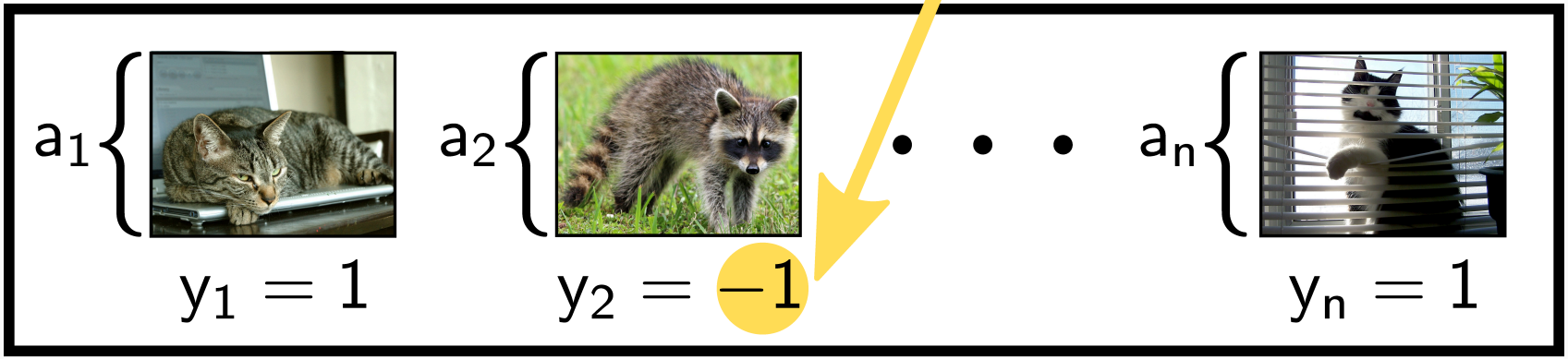
Output/Label
$y \in \{\pm 1\}$

e.g., linear mapping: $h_w(a) = a^\top w$

How to find the **best parameter** $\boldsymbol{w}$ to build the mapping h_w?

Supervised learning: learn from labeled data

$y = -1$ means no/false

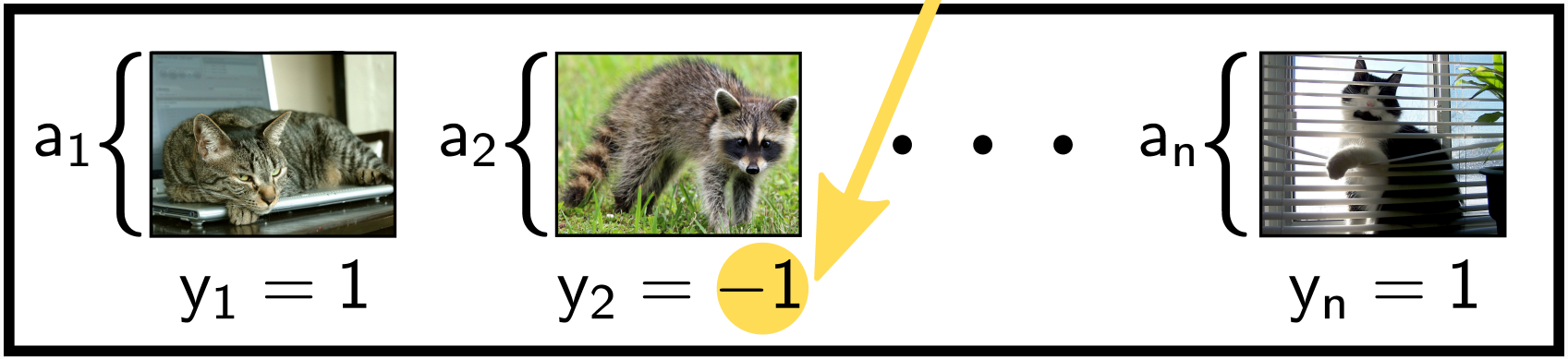


Training set

Learning algorithm → $h_w : a \in \mathbb{R}^d \mapsto y \in \{\pm 1\}$

Supervised learning: learn from labeled data

$y = -1$ means no/false



Training set

Learning algorithm → $h_w : a \in \mathbb{R}^d \mapsto y \in \{\pm 1\}$

Prediction

h_w  $\longmapsto -1$

Loss function & risk minimization

- **Training set**

$$\{(\text{feature}_i, \text{label}_i)_{i=1\ldots n}\} = \{(a_1, y_1), \ldots, (a_n, y_n)\}$$

Loss function & risk minimization

- **Training set**

$$\{(\text{feature}_i, \text{label}_i)_{i=1\ldots n}\} = \{(a_1, y_1), \ldots, (a_n, y_n)\}$$

- **Loss function**

$$\begin{array}{rccc} \ell: & \mathbb{R} \times \mathbb{R} & \longrightarrow & \mathbb{R}_+ \\ & (h_w(a), y) & \longmapsto & \ell(h_w(a), y) \end{array}$$

e.g., Ordinary Least Squares: linear mapping + squared loss
$\ell(h_w(a_i), y_i) = \frac{1}{2}\left(a_i^\top w - y_i\right)^2$, with $y_i \in \mathbb{R}$

Loss function & risk minimization

- **Training set**

$$\{(\text{feature}_i, \text{label}_i)_{i=1\dots n}\} = \{(a_1, y_1), \dots, (a_n, y_n)\}$$

- **Loss function**

$$\begin{array}{rccc} \ell: & \mathbb{R} \times \mathbb{R} & \longrightarrow & \mathbb{R}_+ \\ & (h_w(a), y) & \longmapsto & \ell(h_w(a), y) \end{array}$$

e.g., Ordinary Least Squares: linear mapping + squared loss
$\ell(h_w(a_i), y_i) = \frac{1}{2}\left(a_i^\top w - y_i\right)^2$, with $y_i \in \mathbb{R}$

- **Training problem**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(h_w(a_i), y_i)$$

$\rightarrow$ **Empirical Risk Minimization (ERM)**

Structure of the training problem

- **Regularized Empirical Risk Minimization (R-ERM)**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\phi_i(a_i^\top w) + \frac{\lambda}{2} \|w\|_2^2}_{f_i(w)}$$

where

  - $a_i \in \mathbb{R}^d$: feature vectors (input)
  - $\lambda > 0$: ridge/Tikhonov's regularization parameter
  - $w \in \mathbb{R}^d$: parameter/model

Structure of the training problem

- **Regularized Empirical Risk Minimization (R-ERM)**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\phi_i(a_i^\top w) + \frac{\lambda}{2} \|w\|_2^2}_{f_i(w)}$$

  where
  - $a_i \in \mathbb{R}^d$: feature vectors (input)
  - $\lambda > 0$: ridge/Tikhonov's regularization parameter
  - $w \in \mathbb{R}^d$: parameter/model

- **Covered problems**
  - Ridge regression: $\phi_i(z) = \frac{1}{2}(z - y_i)^2$
  - Logistic regression: $\phi_i(z) = \log(1 + e^{-y_i z})$

  with $y_i \in \mathbb{R}$ or $\{-1, 1\}$ the labels (output)

Talk Overview

Stochastic gradient methods to solve the ERM

Optimal mini-batch and step sizes for SAGA

Numerical experiments

Conclusion

Stochastic gradient methods to solve the ERM

Solving the ERM problem

- **Goal**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

Solving the ERM problem

- **Goal**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

- **Regularity assumptions**

Assumption (f is μ-strongly convex)

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\boldsymbol{\mu}}{2} \|y - x\|_2^2$$

i.e., quadratic lower-bound of f

Assumption (f is L-smooth)

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{\boldsymbol{L}}{2} \|y - x\|_2^2$$

i.e., quadratic upper-bound of f

Gradient Descent

Theorem (Linear convergence of Gradient Descent (GD))

Let f be μ-strongly convex and L-smooth,

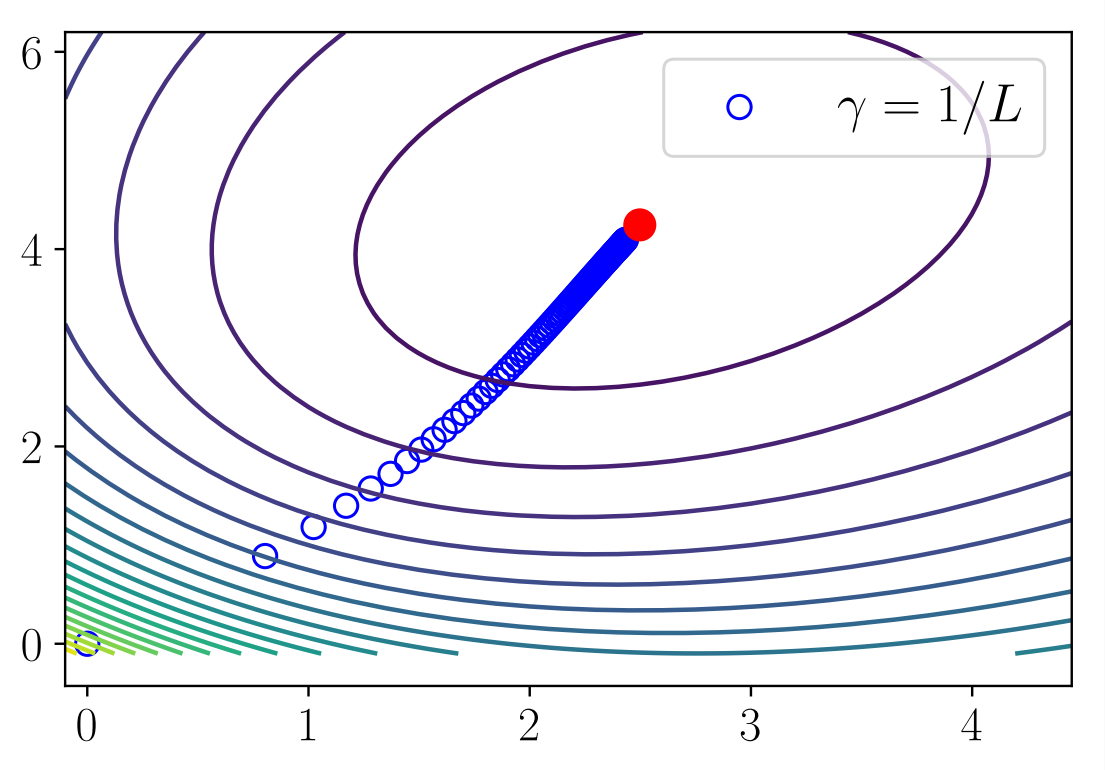
$$||w^k - w^*||_2^2 \leq \left(1 - \frac{\mu}{L}\right)^k ||w^1 - w^*||_2^2$$

where the step at iteration $k+1$ is taken following

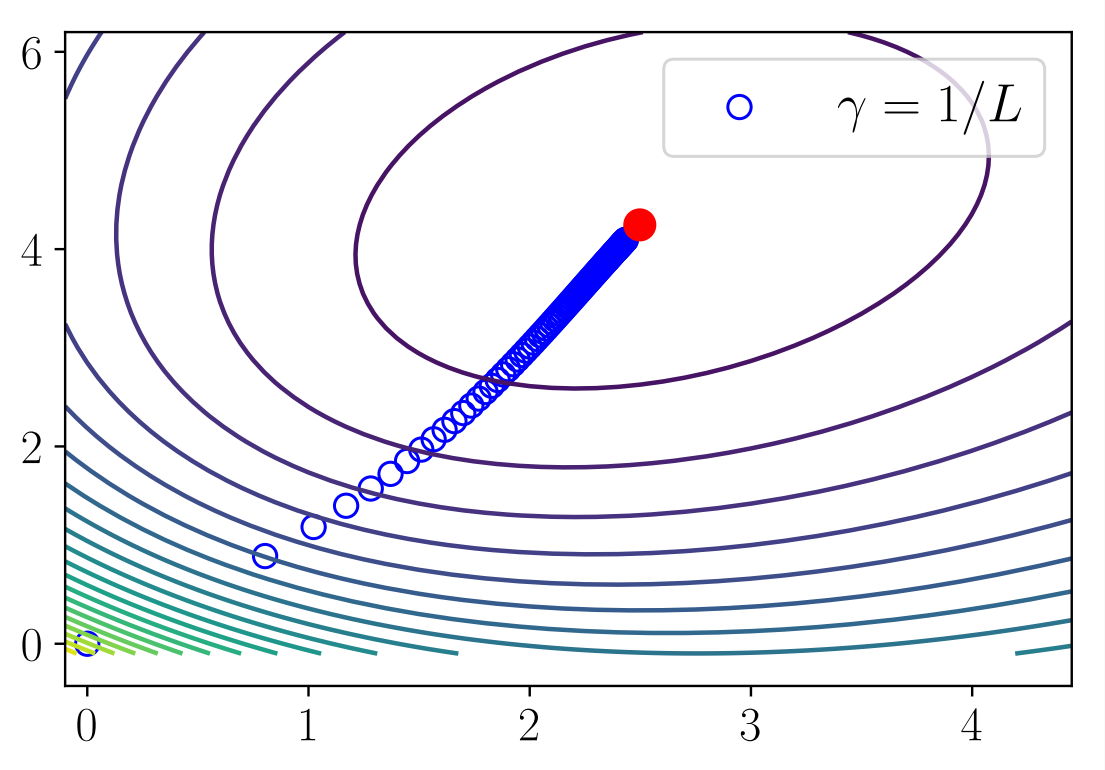
$$w^{k+1} = w^k - \gamma \nabla f(w^k)$$

with the step size $\gamma = \frac{1}{L}$

Example of Gradient Descent steps



Example of Gradient Descent steps

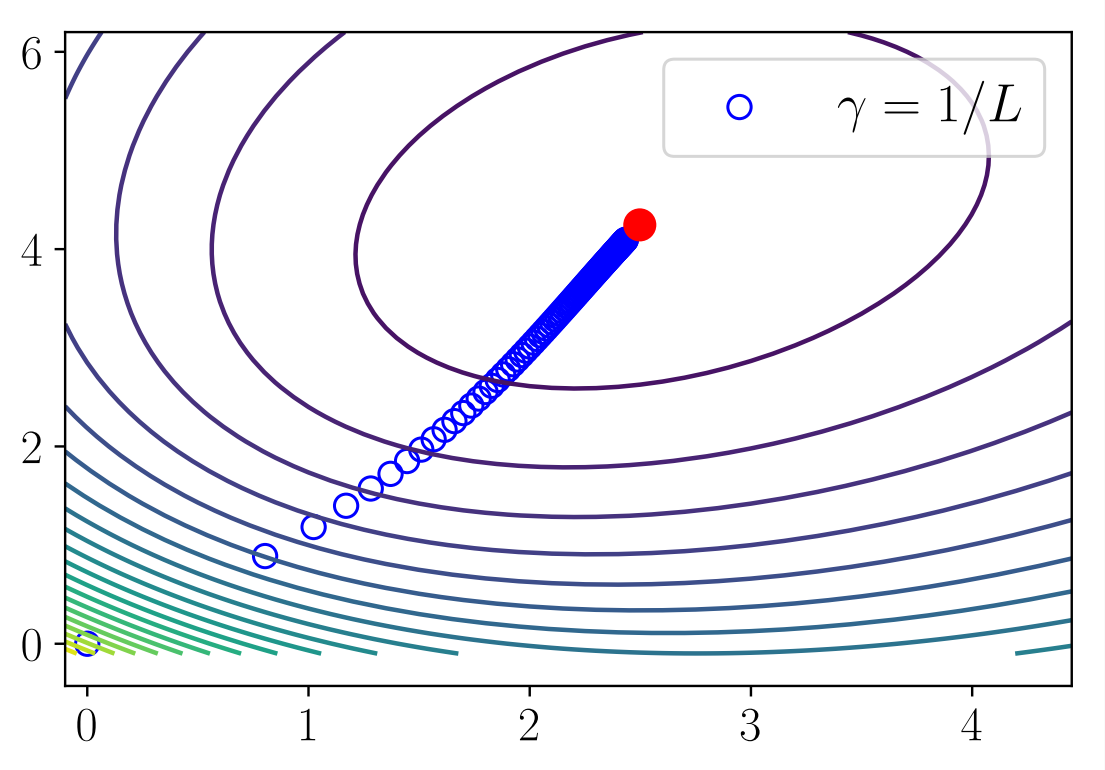


- Here $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$, so a GD step is

$$w^{k+1} = w^k - \frac{\gamma}{n}\left(\sum_{i=1}^{n} \nabla f_i(w^k)\right)$$

with γ the step size

Example of Gradient Descent steps



- Here $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$, so a GD step is

$$w^{k+1} = w^k - \frac{\gamma}{n}\left(\sum_{i=1}^{n} \nabla f_i(w^k)\right)$$

with γ the step size

Problem: $\mathcal{O}(nd)$ computations per iteration

Stochastic Gradient Descent

- **Stochastic Gradient Descent[1] (SGD)**
  - Draw $i \sim \mathcal{U}_{\{1,\dots,n\}}$
  - Take a step

$$w^{k+1} = w^k - \gamma_k \nabla f_i(w^k)$$

    with $(\gamma_k)_k$ a step size sequence

[1] Robbins and Monro, 1951b

Stochastic Gradient Descent

- **Stochastic Gradient Descent[1] (SGD)**
  - Draw $i \sim \mathcal{U}_{\{1,\dots,n\}}$
  - Take a step

$$w^{k+1} = w^k - \gamma_k \nabla f_i(w^k)$$

    with $(\gamma_k)_k$ a step size sequence

- Unbiased estimator of the gradient: $\mathbb{E}\left[\nabla f_i(w^k)\right] = \nabla f(w^k)$

[1] Robbins and Monro, 1951b

Stochastic Gradient Descent

- **Stochastic Gradient Descent[1] (SGD)**
  - Draw $i \sim \mathcal{U}_{\{1,\dots,n\}}$
  - Take a step

$$w^{k+1} = w^k - \gamma_k \nabla f_i(w^k)$$

    with $(\gamma_k)_k$ a step size sequence

- Unbiased estimator of the gradient: $\mathbb{E}\left[\nabla f_i(w^k)\right] = \nabla f(w^k)$

- Converges under assumptions on $(f_i)_{i=1,\dots,n}$ and $(\gamma_k)_k$

[1] Robbins and Monro, 1951b

Stochastic Gradient Descent

- **Stochastic Gradient Descent[1] (SGD)**
  - Draw $i \sim \mathcal{U}_{\{1,\dots,n\}}$
  - Take a step

$$w^{k+1} = w^k - \gamma_k \nabla f_i(w^k)$$

    with $(\gamma_k)_k$ a step size sequence

- Unbiased estimator of the gradient: $\mathbb{E}\left[\nabla f_i(w^k)\right] = \nabla f(w^k)$
- Converges under assumptions on $(f_i)_{i=1,\dots,n}$ and $(\gamma_k)_k$
- In practice people try decreasing step size like $\gamma_k \propto \frac{1}{\sqrt{k}}$ or $\gamma_k \propto \frac{1}{k}$

[1] Robbins and Monro, 1951b

Stochastic Gradient Descent

- **Stochastic Gradient Descent[1] (SGD)**
  - Draw $i \sim \mathcal{U}_{\{1,\dots,n\}}$
  - Take a step

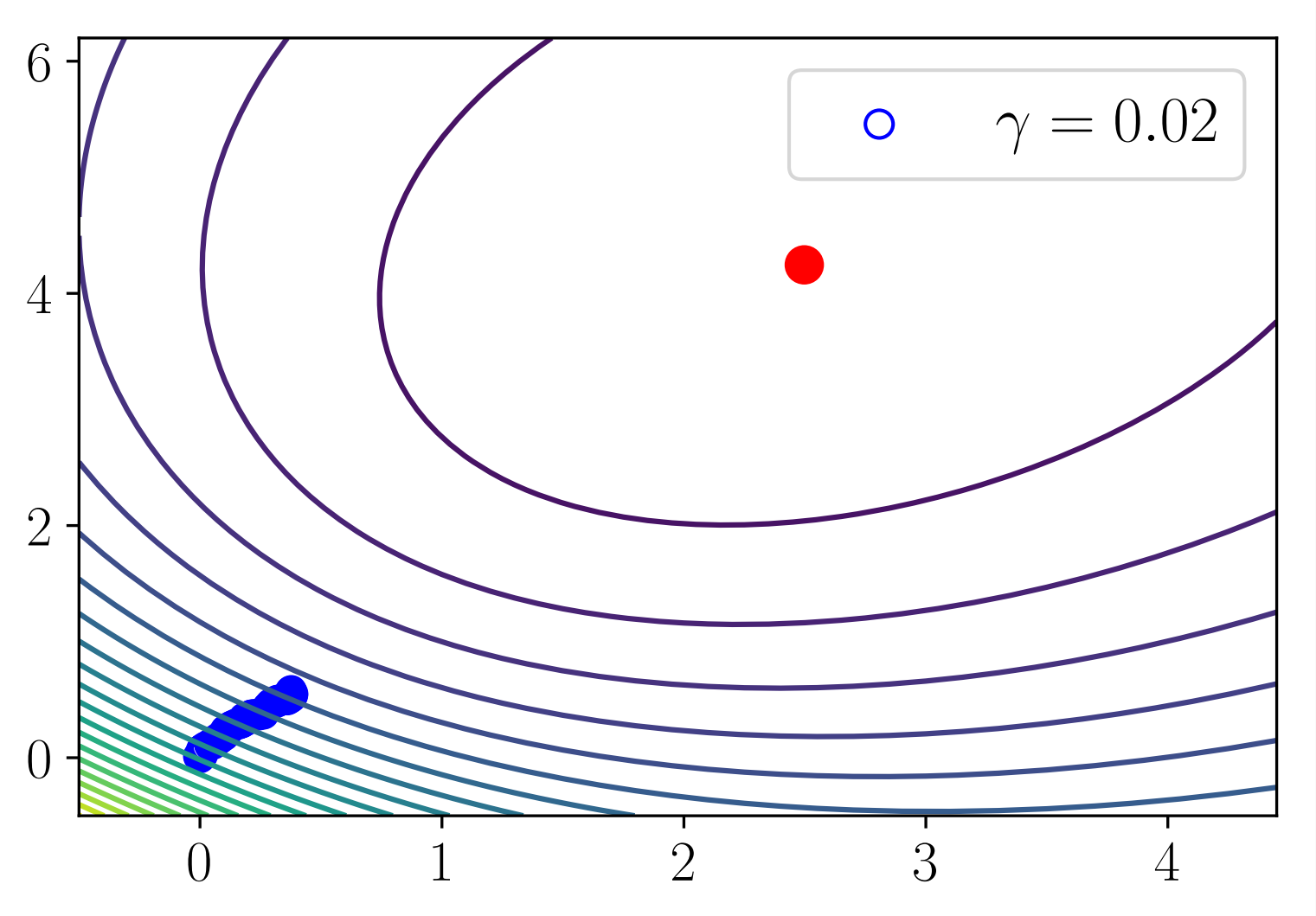
$$w^{k+1} = w^k - \gamma_k \nabla f_i(w^k)$$

    with $(\gamma_k)_k$ a step size sequence

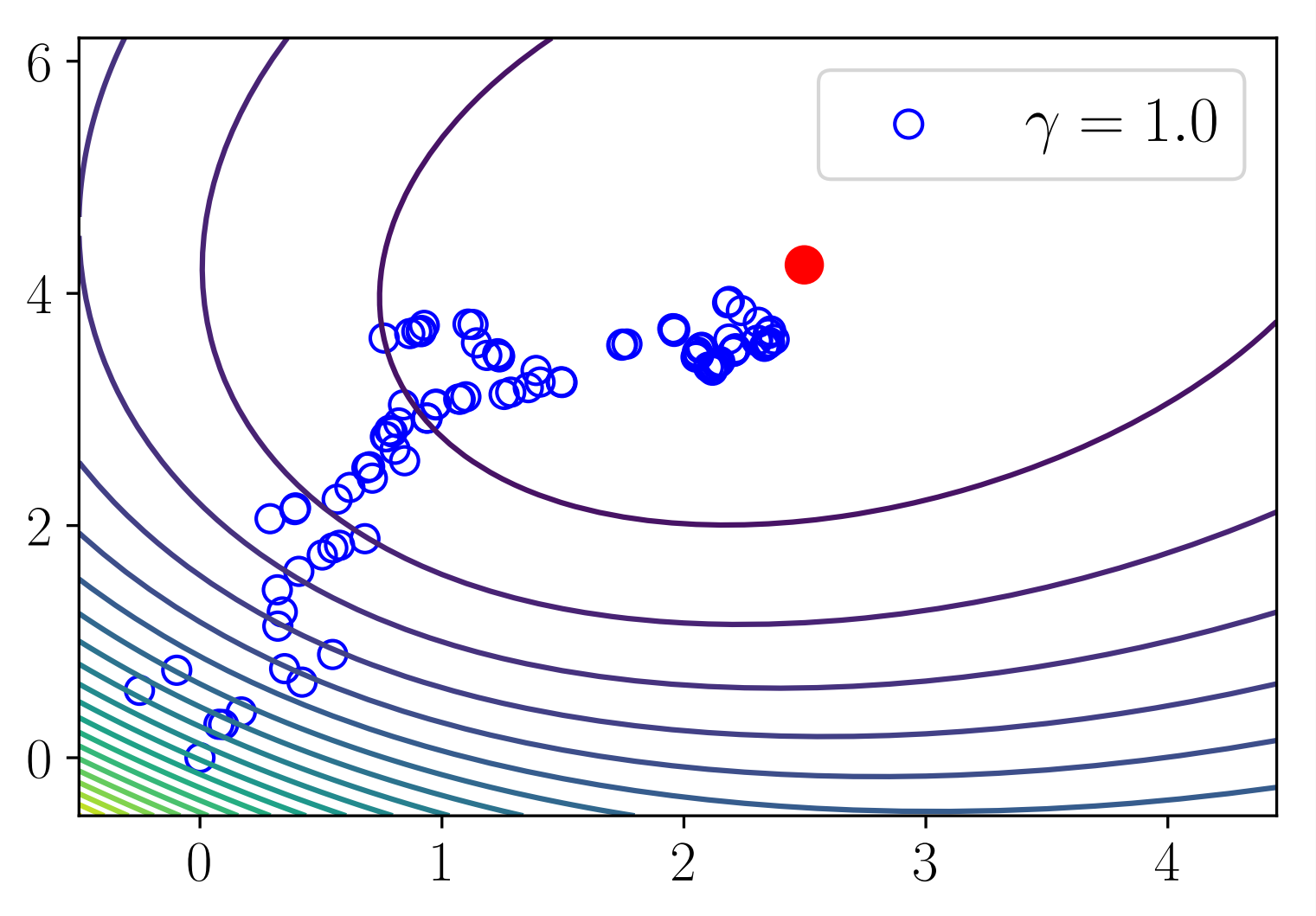
- Unbiased estimator of the gradient: $\mathbb{E}\left[\nabla f_i(w^k)\right] = \nabla f(w^k)$
- Converges under assumptions on $(f_i)_{i=1,\dots,n}$ and $(\gamma_k)_k$
- In practice people try decreasing step size like $\gamma_k \propto \frac{1}{\sqrt{k}}$ or $\gamma_k \propto \frac{1}{k}$
- Assuming each f_i is L_i-smooth, SGD convergence driven by $L_{\max}/\mu$, where $L_{\max} := \max_{i=1,\dots,n} L_i$

[1] Robbins and Monro, 1951b

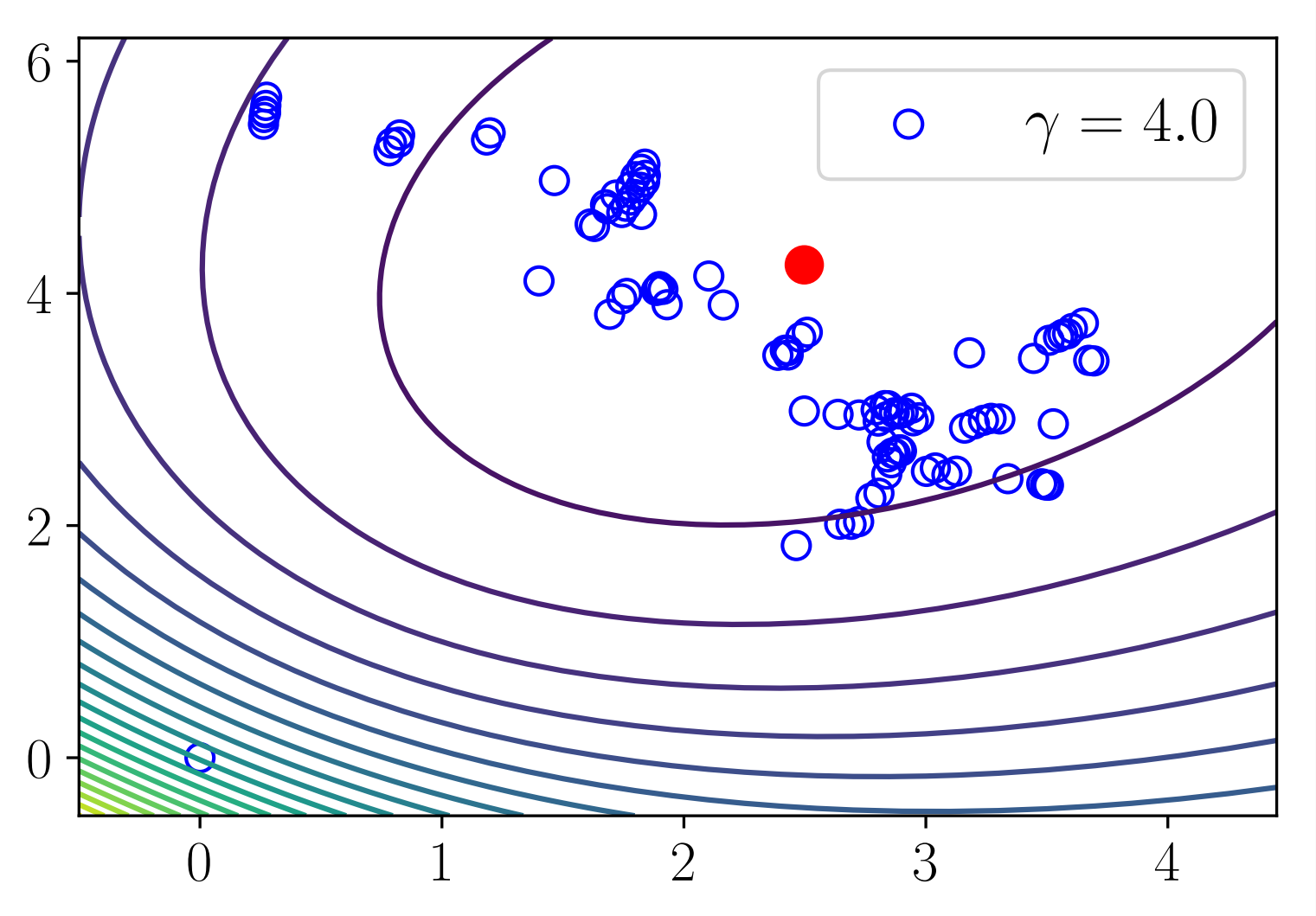
SGD examples with constant step size



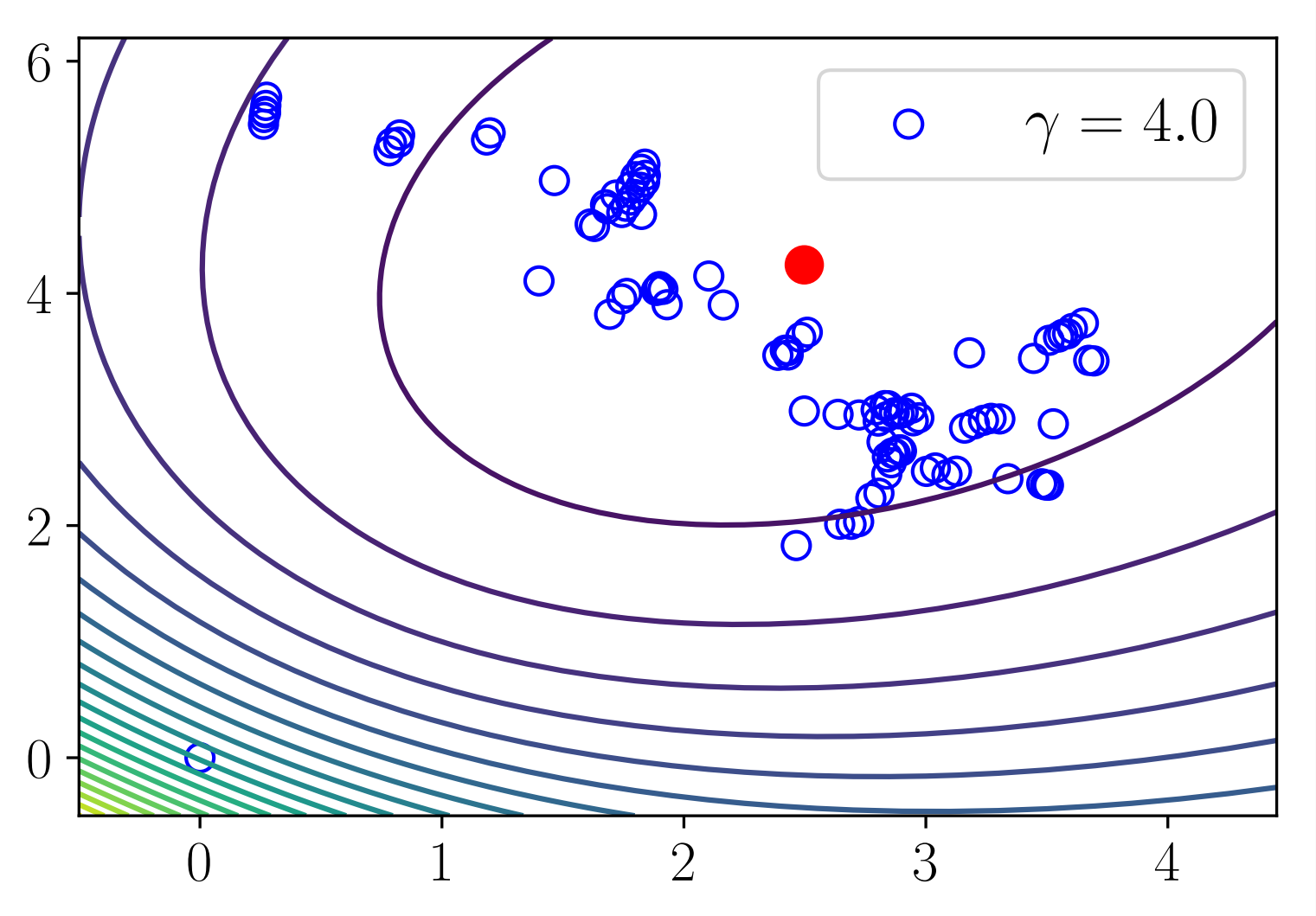
SGD examples with constant step size



SGD examples with constant step size



SGD examples with constant step size



Problem: tuning the step size sequence is painful

Mini-batching

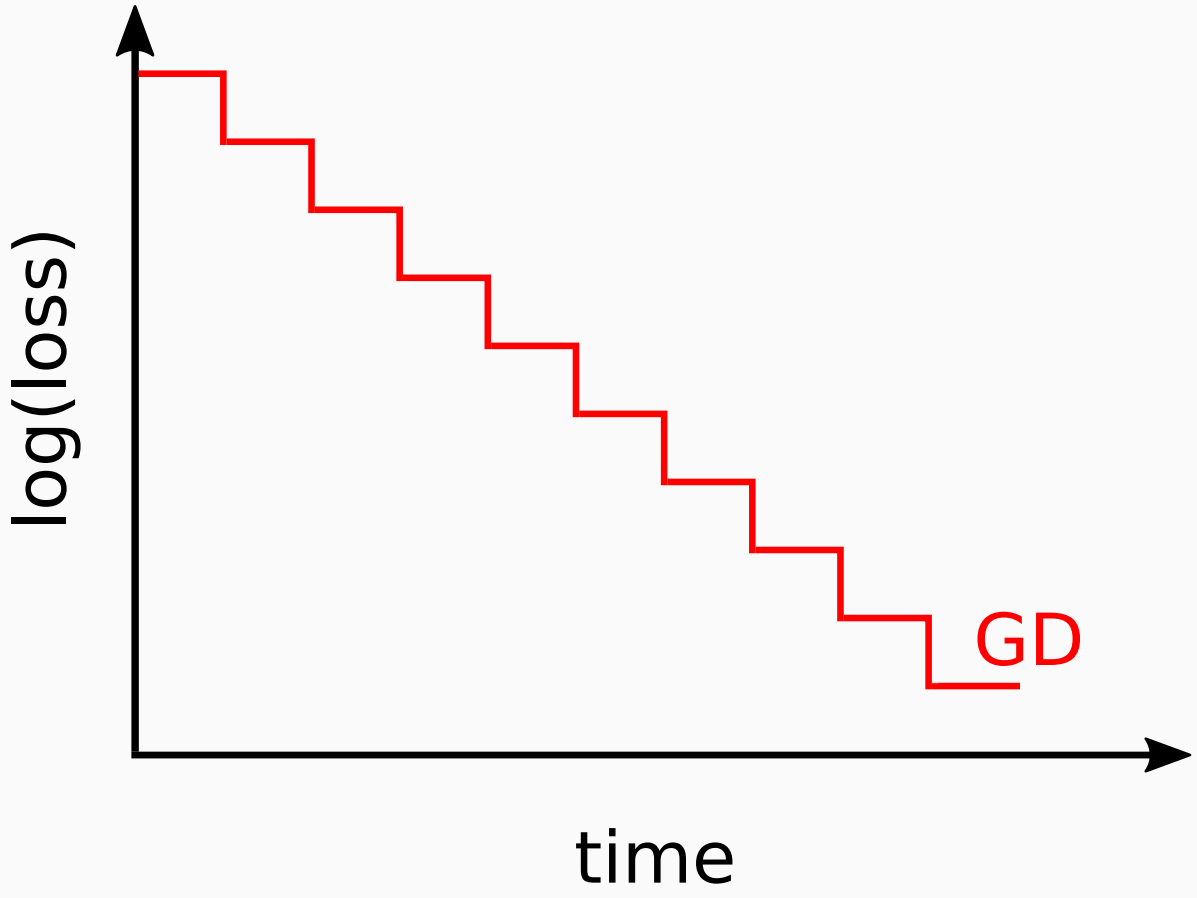
- **Compute several stochastic gradients instead of one**
  - Sample randomly B a mini-batch of b data samples
    i.e., a subset of $\{1, \dots, n\}$
  - Take a step
    $$w^{k+1} = w^k - \gamma_k \sum_{i \in B} \nabla f_i(w^k)$$
    with $(\gamma_k)_k$ a step size sequence.

Mini-batching

- **Compute several stochastic gradients instead of one**
  - Sample randomly B a mini-batch of b data samples
    i.e., a subset of $\{1, \dots, n\}$
  - Take a step
    $$w^{k+1} = w^k - \gamma_k \sum_{i \in B} \nabla f_i(w^k)$$
    with $(\gamma_k)_k$ a step size sequence.

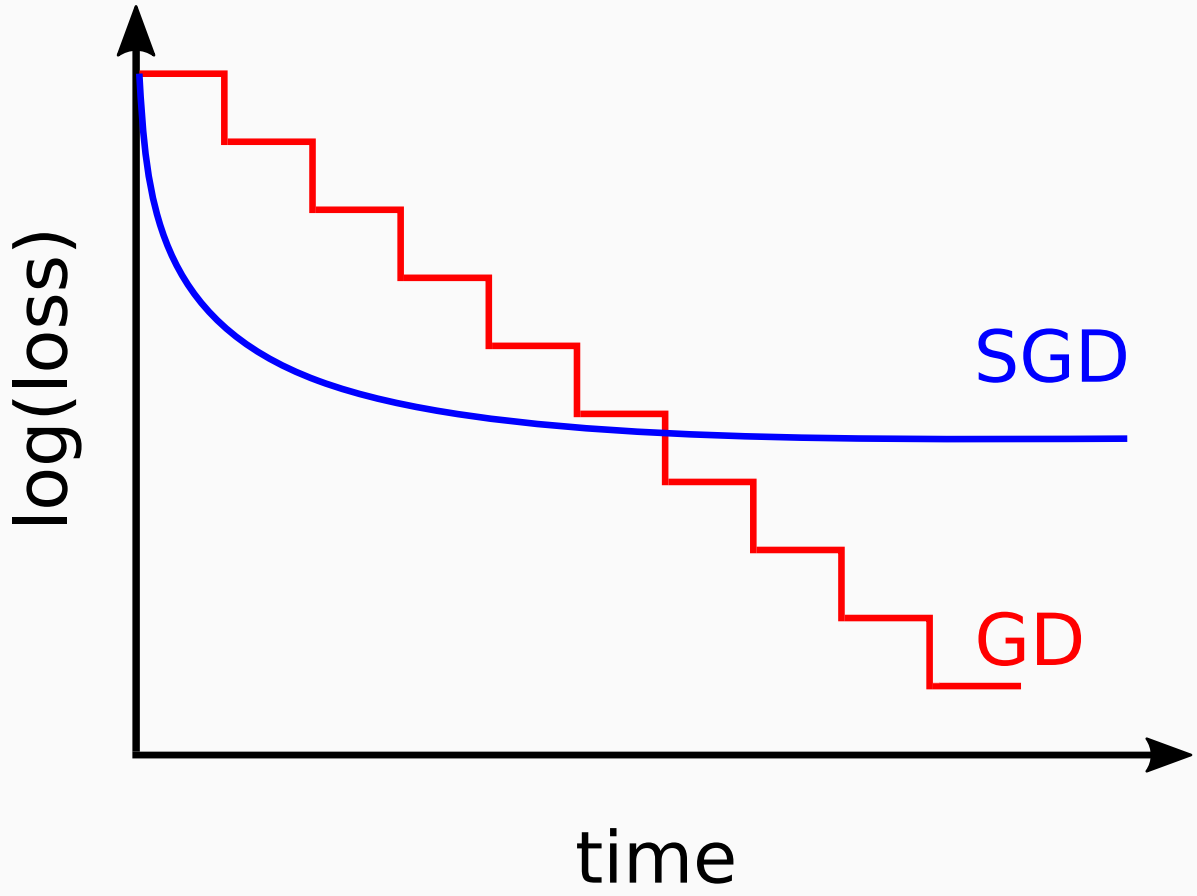
- **Why is it useful?**
  - Works well in practice
  - Calculating several gradients per iteration is cheap

Stochastic versus deterministic methods[2]



[2] *cf.* F. Bach course slides "Statistical machine learning and convex optimization", at Paris-Sud

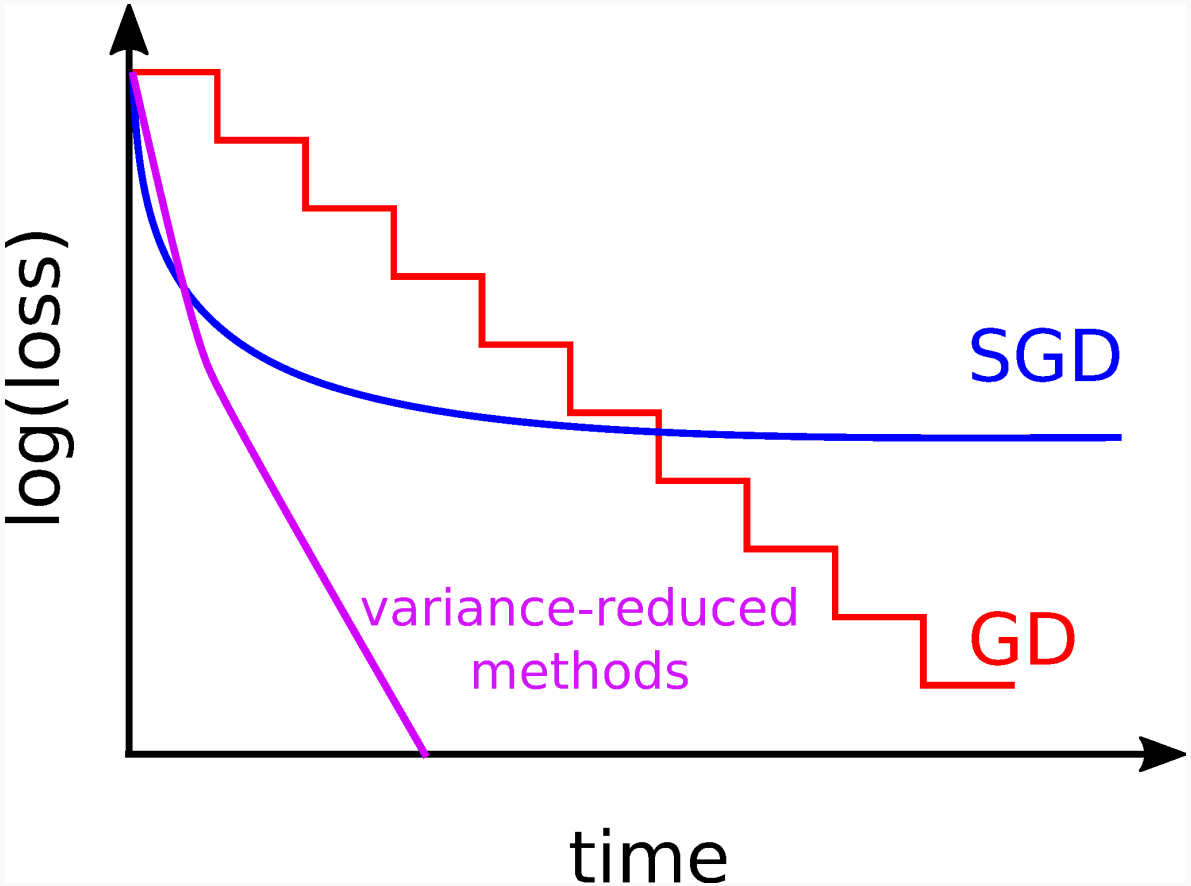
Stochastic versus deterministic methods[2]



[2] *cf.* F. Bach course slides "Statistical machine learning and convex optimization", at Paris-Sud

Stochastic versus deterministic methods[2]

Best of both worlds: $\begin{cases} \text{linear rate with } \mathcal{O}(d) \text{ iteration cost} \\ \text{simple step size} \end{cases}$



[2] *cf.* F. Bach course slides "Statistical machine learning and convex optimization", at Paris-Sud

Reformulation of the ERM

- **Sampling vector**
  Let $v \in \mathbb{R}^n$, with distribution $\mathcal{D}$ s.t.

$$\mathbb{E}_{\mathcal{D}}[v] = e$$

  where e being the all-ones vector

Reformulation of the ERM

- **Sampling vector**

  Let $v \in \mathbb{R}^n$, with distribution $\mathcal{D}$ s.t.

  $$\mathbb{E}_{\mathcal{D}}[v] = e$$

  where e being the all-ones vector
- **Unbiased subsampled function**

  $$f_v(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w)v_i = \frac{1}{n}\langle F(w), v\rangle$$

  where $F(w) := (f_1(w), \ldots, f_n(w))^\top$

  $$\implies \mathbb{E}_{\mathcal{D}}[f_v(w)] = \frac{1}{n}\langle F(w), \mathbb{E}_{\mathcal{D}}[v]\rangle = f(w)$$

Reformulation of the ERM

- **Sampling vector**

  Let $v \in \mathbb{R}^n$, with distribution $\mathcal{D}$ s.t.

  $$\mathbb{E}_{\mathcal{D}}[v] = e$$

  where e being the all-ones vector

- **Unbiased subsampled function**

  $$f_v(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) v_i = \frac{1}{n}\langle F(w), v\rangle$$

  where $F(w) := (f_1(w), \ldots, f_n(w))^\top$

  $$\implies \mathbb{E}_{\mathcal{D}}[f_v(w)] = \frac{1}{n}\langle F(w), \mathbb{E}_{\mathcal{D}}[v]\rangle = f(w)$$

- **ERM reformulation**

  $$\text{Solving ERM} \iff \text{find } w^* \in \arg\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \mathbb{E}_{\mathcal{D}}[f_v(w)]$$

Unbiased gradient estimate

- **Unbiased estimates**

$$\mathbb{E}_{\mathcal{D}}\left[f_v(w)\right] = \frac{1}{n}\langle F(w), \mathbb{E}_{\mathcal{D}}\left[v\right]\rangle = f(w)$$

$$\mathbb{E}_{\mathcal{D}}\left[\nabla f_v(w)\right] = \frac{1}{n}\nabla F(w)\mathbb{E}_{\mathcal{D}}\left[v\right] = \nabla f(w)$$

where $\nabla F(w) := [\nabla f_1(w), \ldots, \nabla f_n(w)] \in \mathbb{R}^{d\times n}$ is the Jacobian

Unbiased gradient estimate

- **Unbiased estimates**

$$\mathbb{E}_{\mathcal{D}}\left[f_v(w)\right] = \frac{1}{n}\langle F(w), \mathbb{E}_{\mathcal{D}}\left[v\right]\rangle = f(w)$$

$$\mathbb{E}_{\mathcal{D}}\left[\nabla f_v(w)\right] = \frac{1}{n}\nabla F(w)\mathbb{E}_{\mathcal{D}}\left[v\right] = \nabla f(w)$$

  where $\nabla F(w) := [\nabla f_1(w), \ldots, \nabla f_n(w)] \in \mathbb{R}^{d\times n}$ is the Jacobian

- **Examples of sampling vector**
  - Let $v = e$ (deterministic)

$$\nabla f_v(w) = \frac{1}{n}\nabla F(w)e = \nabla f(w) \qquad \text{(GD)}$$

Unbiased gradient estimate

- **Unbiased estimates**

$$\mathbb{E}_{\mathcal{D}}\left[f_v(w)\right] = \frac{1}{n}\langle F(w), \mathbb{E}_{\mathcal{D}}\left[v\right]\rangle = f(w)$$

$$\mathbb{E}_{\mathcal{D}}\left[\nabla f_v(w)\right] = \frac{1}{n}\nabla F(w)\mathbb{E}_{\mathcal{D}}\left[v\right] = \nabla f(w)$$

  where $\nabla F(w) := [\nabla f_1(w), \ldots, \nabla f_n(w)] \in \mathbb{R}^{d\times n}$ is the Jacobian

- **Examples of sampling vector**
  - Let $v = e$ (deterministic)

$$\nabla f_v(w) = \frac{1}{n}\nabla F(w)e = \nabla f(w) \qquad \text{(GD)}$$

  - Let $\mathbb{P}\left[v = ne_i\right] = 1/n$ for all $i \in \{1, \ldots, n\}$

$$\nabla f_v(w) = \frac{1}{n}\nabla F(w)ne_i = \nabla f_i(w) \qquad \text{(SGD)}$$

Unbiased gradient estimate

- **Unbiased estimates**

$$\mathbb{E}_{\mathcal{D}}\left[f_v(w)\right] = \frac{1}{n}\langle F(w), \mathbb{E}_{\mathcal{D}}\left[v\right]\rangle = f(w)$$

$$\mathbb{E}_{\mathcal{D}}\left[\nabla f_v(w)\right] = \frac{1}{n}\nabla F(w)\mathbb{E}_{\mathcal{D}}\left[v\right] = \nabla f(w)$$

  where $\nabla F(w) := [\nabla f_1(w), \ldots, \nabla f_n(w)] \in \mathbb{R}^{d\times n}$ is the Jacobian

- **Examples of sampling vector**
  - Let $v = e$ (deterministic)

$$\nabla f_v(w) = \frac{1}{n}\nabla F(w)e = \nabla f(w) \qquad \text{(GD)}$$

  - Let $\mathbb{P}\left[v = ne_i\right] = 1/n$ for all $i \in \{1, \ldots, n\}$

$$\nabla f_v(w) = \frac{1}{n}\nabla F(w)ne_i = \nabla f_i(w) \qquad \text{(SGD)}$$

$\rightarrow$ **Motivates using:** $w^{k+1} = w^k - \gamma_k \boldsymbol{\nabla f_v(w^k)}$

Controlled stochastic reformulation of the ERM

- **New ERM reformulation**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w) = \mathbb{E}_{\mathcal{D}}\Big[f_v(w) \underbrace{-z_v(w) + \mathbb{E}_{\mathcal{D}}\left[z_v(w)\right]}_{\text{unbiased correction term}}\Big]$$

with $z_v(\cdot)$ a random function.

Controlled stochastic reformulation of the ERM

- **New ERM reformulation**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w) = \mathbb{E}_{\mathcal{D}}\left[f_v(w) \underbrace{- z_v(w) + \mathbb{E}_{\mathcal{D}}\left[z_v(w)\right]}_{\text{unbiased correction term}}\right]$$

  with $z_v(\cdot)$ a random function.

- **New gradient estimator**

$$\boldsymbol{g_v(w)} := \nabla f_v(w) - \nabla z_v(w) + \mathbb{E}_{\mathcal{D}}\left[\nabla z_v(w)\right]$$

  Iteration: $w^{k+1} = w^k - \gamma \boldsymbol{g_v(w^k)}$

Controlled stochastic reformulation of the ERM

- **New ERM reformulation**

$$\text{find } w^* \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w) = \mathbb{E}_{\mathcal{D}}\Big[f_v(w) \underbrace{- z_v(w) + \mathbb{E}_{\mathcal{D}}\left[z_v(w)\right]}_{\text{unbiased correction term}}\Big]$$

  with $z_v(\cdot)$ a random function.

- **New gradient estimator**

$$\boldsymbol{g_v(w)} := \nabla f_v(w) - \nabla z_v(w) + \mathbb{E}_{\mathcal{D}}\left[\nabla z_v(w)\right]$$

  Iteration: $w^{k+1} = w^k - \gamma \boldsymbol{g_v(w^k)}$

$\rightarrow$ Stochastic variance-reduced methods

Advantages of stochastic variance-reduced methods

- **Summary**

$$w^{k+1} = w^k - \gamma \mathbf{g_v}(\mathbf{w^k})$$

  Such that
  - Unbiased gradient: $\mathbb{E}\left[g_v(w^k)\right] = \nabla f(w^k)$
  - Decreasing variance: $\mathbb{E}\left[\left\|g_v(w^k) - \nabla f(w^k)\right\|_2^2\right] \xrightarrow[w^k \to w^*]{} 0$

Advantages of stochastic variance-reduced methods

- **Summary**

$$w^{k+1} = w^k - \gamma \mathbf{g_v}(\mathbf{w^k})$$

  Such that
  - Unbiased gradient: $\mathbb{E}\left[g_v(w^k)\right] = \nabla f(w^k)$
  - Decreasing variance: $\mathbb{E}\left[\left\|g_v(w^k) - \nabla f(w^k)\right\|_2^2\right] \xrightarrow[w^k \to w^*]{} 0$

- **Advantages**
  - **No decreasing step sizes $(\gamma_k)_k$ needed**

Advantages of stochastic variance-reduced methods

- **Summary**

$$w^{k+1} = w^k - \gamma \mathbf{g_v}(\mathbf{w^k})$$

  Such that
  - Unbiased gradient: $\mathbb{E}\left[g_v(w^k)\right] = \nabla f(w^k)$
  - Decreasing variance: $\mathbb{E}\left[\left\|g_v(w^k) - \nabla f(w^k)\right\|_2^2\right] \xrightarrow[w^k \to w^*]{} 0$

- **Advantages**
  - **No decreasing step sizes $(\gamma_k)_k$ needed**
  - **$f_v(w)$ determines the smoothness** of the controlled stochastic reformulation ($z_v(w)$ linear in w)

Advantages of stochastic variance-reduced methods

- **Summary**

$$w^{k+1} = w^k - \gamma \mathbf{g_v}(\mathbf{w^k})$$

  Such that
  - Unbiased gradient: $\mathbb{E}\left[g_v(w^k)\right] = \nabla f(w^k)$
  - Decreasing variance: $\mathbb{E}\left[\left\|g_v(w^k) - \nabla f(w^k)\right\|_2^2\right] \xrightarrow[w^k \to w^*]{} 0$

- **Advantages**
  - **No decreasing step sizes $(\gamma_k)_k$ needed**
  - **$f_v(w)$ determines the smoothness** of the controlled stochastic reformulation ($z_v(w)$ linear in w)
  - No "bounded gradient" assumption such as $\mathbb{E}\left[\left\|\nabla f_v(w^k)\right\|_2^2\right] \leq cste$

Recovering SAGA algorithm

- **Parameters for SAGA**[3]

$$\begin{cases} f_v(w) = \frac{1}{n}\langle F(w), v\rangle & \Longrightarrow \nabla f_v(w) = \frac{1}{n}\nabla F(w)v \\ z_v(w) = \frac{1}{n}\langle \underbrace{\boldsymbol{J}^\top w}_{\text{linear estimation of } F(w)}, v\rangle & \Longrightarrow \nabla z_v(w) = \frac{1}{n}Jv \end{cases}$$

with $\boldsymbol{J}$ an **estimate of the Jacobian** in $R^{d\times n}$

[3] Defazio, Bach and Lacoste-Julien, 2014b

Recovering SAGA algorithm

- **Parameters for SAGA**[3]

$$\begin{cases} f_v(w) = \frac{1}{n}\langle F(w), v\rangle & \Longrightarrow \nabla f_v(w) = \frac{1}{n}\nabla F(w)v \\ z_v(w) = \frac{1}{n}\langle \underbrace{\boldsymbol{J}^\top w}_{\text{linear estimation of } F(w)}, v\rangle & \Longrightarrow \nabla z_v(w) = \frac{1}{n}Jv \end{cases}$$

  with $\boldsymbol{J}$ an **estimate of the Jacobian** in $R^{d\times n}$

- If $v = ne_i$, where e_i is the i-th vector of basis

$$\begin{aligned} \boldsymbol{g_v(w)} &:= \nabla f_v(w) - \nabla z_v(w) + \mathbb{E}_{\mathcal{D}}\left[\nabla z_v(w)\right] \\ &= \frac{1}{n}\nabla F(w)ne_i - \frac{1}{n}Jne_i + \frac{1}{n}\mathbb{E}\left[Jne_i\right] \\ &= \nabla f_i(w) - J_{:i} + \frac{1}{n}Je \end{aligned}$$

  where e is the all-ones vector and $J_{:i}$ the $i-$th column of $J \in R^{d\times n}$

[3] Defazio, Bach and Lacoste-Julien, 2014b

Recovering SAGA algorithm

- **Parameters for SAGA**[3]

$$\begin{cases} f_v(w) = \frac{1}{n}\langle F(w), v\rangle & \Longrightarrow \nabla f_v(w) = \frac{1}{n}\nabla F(w)v \\ z_v(w) = \frac{1}{n}\langle \underbrace{J^\top w}_{\text{linear estimation of } F(w)}, v\rangle & \Longrightarrow \nabla z_v(w) = \frac{1}{n}Jv \end{cases}$$

  with J an **estimate of the Jacobian** in $R^{d\times n}$

- If $v = ne_i$, where e_i is the i-th vector of basis

$$\begin{aligned} \boldsymbol{g_v(w)} &:= \nabla f_v(w) - \nabla z_v(w) + \mathbb{E}_{\mathcal{D}}\left[\nabla z_v(w)\right] \\ &= \frac{1}{n}\nabla F(w)ne_i - \frac{1}{n}Jne_i + \frac{1}{n}\mathbb{E}\left[Jne_i\right] \\ &= \nabla f_i(w) - J_{:i} + \frac{1}{n}Je \end{aligned}$$

  where e is the all-ones vector and $J_{:i}$ the $i-$th column of $J \in R^{d\times n}$

- **Variance term**
  Convergence analysis: $\mathbb{E}_{\mathcal{D}}\left[\|g_v(w) - \nabla f(w)\|_2^2\right]$ low for $J \approx \nabla F(w)$

[3] Defazio, Bach and Lacoste-Julien, 2014b

Example: mini-batch SAGA

- **Mini-batch SAGA algorithm**
  - Sample a mini-batch $B \subset [n] := \{1, \ldots, n\}$ s.t. $|B| = b$
  - Build the **gradient estimator** from b stochastic gradients $\nabla f_i(w^k)$, $\forall i \in B$

$$\boldsymbol{g(w^k)} = \frac{1}{n}\left(\frac{1}{b}\sum_{i\in B}\nabla f_i(w^k) - \frac{1}{b}\sum_{i\in B} J^k_{:i} + J^k e\right)$$

    where e is the all-ones vector and $J^k_{:i}$ the $i-$th column of $J^k \in R^{d\times n}$.
  - Take a step

$$w^{k+1} = w^k - \gamma \boldsymbol{g(w^k)}$$

  - Update the **Jacobian estimate $\mathbf{J^k}$**

$$J^k_i = \nabla f_i(w^k), \quad \forall i \in B$$

Example: mini-batch SAGA

- **Mini-batch SAGA algorithm**
  - Sample a mini-batch $B \subset [n] := \{1, \ldots, n\}$ s.t. $|B| = b$
  - Build the **gradient estimator** from b stochastic gradients $\nabla f_i(w^k)$, $\forall i \in B$

$$\boldsymbol{g(w^k)} = \frac{1}{n}\left(\frac{1}{b}\sum_{i\in B}\nabla f_i(w^k) - \frac{1}{b}\sum_{i\in B} J^k_{:i} + J^k e\right)$$

    where e is the all-ones vector and $J^k_{:i}$ the $i-$th column of $J^k \in R^{d\times n}$.
  - Take a step

$$w^{k+1} = w^k - \gamma \boldsymbol{g(w^k)}$$

  - Update the **Jacobian estimate $\mathbf{J^k}$**

$$J^k_i = \nabla f_i(w^k), \quad \forall i \in B$$

- **What is the optimal mini-batch size?**

$$\rightarrow \textbf{Find the "best" } \boldsymbol{b} \textbf{ value}$$

Talk Overview

Stochastic gradient methods to solve the ERM

Optimal mini-batch and step sizes for SAGA

Numerical experiments

Conclusion

Optimal mini-batch and step sizes for SAGA

Convergence theorem

Theorem

Consider the iterates w^k of the mini-batch SAGA algorithm. Let the step size be given by

$$\gamma = \frac{1}{4} \frac{1}{\max\left\{\mathcal{L} + \lambda, \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{\mu}{4}\frac{n}{b}\right\}}$$

Given an $\epsilon > 0$, if $k \geq K_{iter}(b)$ where

$$K_{iter}(b) := \left\{\frac{4(\mathcal{L}+\lambda)}{\mu}, \frac{n}{b} + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{b\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

then $\mathbb{E}\left[\left\|w^k - w^*\right\|^2\right] \leq \epsilon\, C$, where $C > 0$ is a constant. [4]

[4] $C := \left\|w^0 - w^*\right\|^2 + \frac{\gamma}{2L_{\max}} \sum_{i\in[n]} \left\|J^0_{:i} - \nabla f(w^*)\right\|^2$

Methodology

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**

$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b) = bK_{\text{iter}}(b)$$

[5] Gower, Richtárik and Bach, 2018

Methodology

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**

gradients per iteration

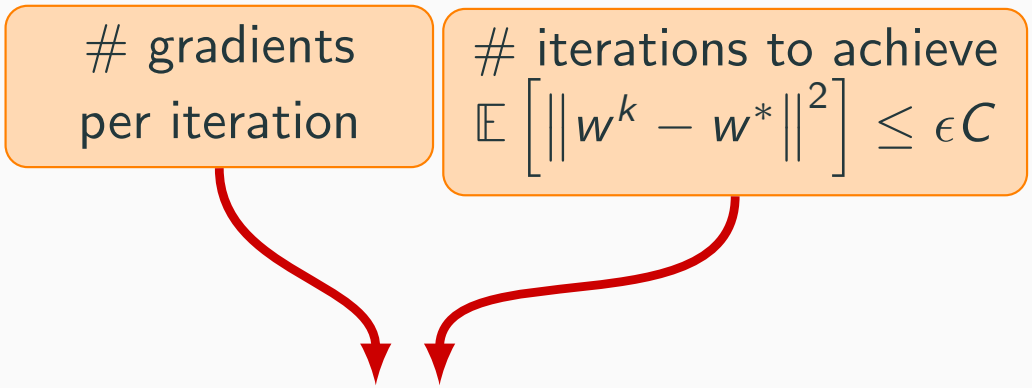
$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b) = bK_{\text{iter}}(b)$$

[5] Gower, Richtárik and Bach, 2018

Methodology

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**

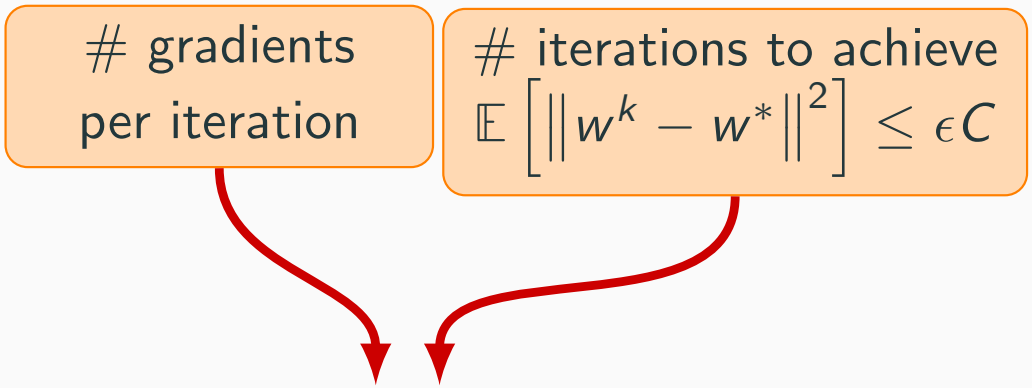


$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b) = bK_{\text{iter}}(b)$$

[5] Gower, Richtárik and Bach, 2018

Methodology

For a desired precision $\epsilon > 0$,



- **Optimal mini-batch size**

$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b) = bK_{\text{iter}}(b)$$

- **Total complexity**[5]

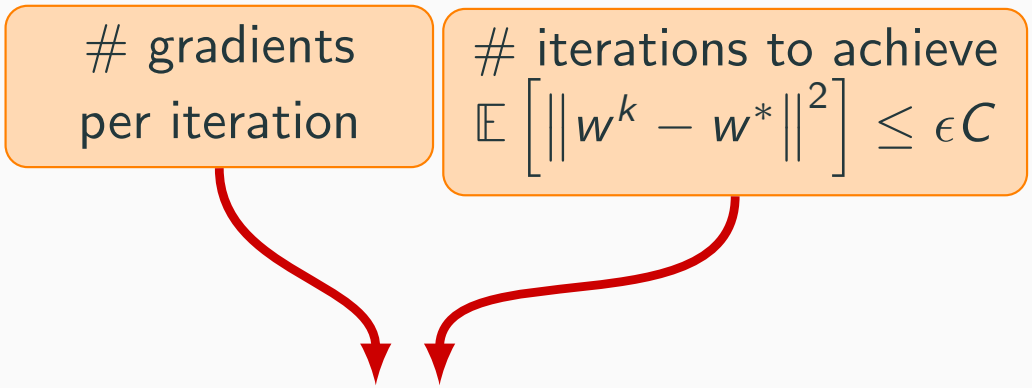
$$K_{\text{total}}(b) = \max\left\{\frac{4b(\mathcal{L}+\lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

[5] Gower, Richtárik and Bach, 2018

Methodology

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**



$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b) = bK_{\text{iter}}(b)$$

- **Total complexity**[5]

$$K_{\text{total}}(b) = \max\left\{\frac{4b(\mathcal{L}+\lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

$\rightarrow$ **Here, what are $\mathcal{L}$, $L_{\max}$, μ ?**

[5] Gower, Richtárik and Bach, 2018

Basic assumptions

Assumption (i.i.d. data)

n **i.i.d.** *observations:* $(a_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ *or* $\mathbb{R}^d \times \{-1, 1\}$

Assumption (f is μ-strongly convex)

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\boldsymbol{\mu}}{2} \|y - x\|_2^2$$

i.e., quadratic lower-bound of f

Assumption (f is L-smooth)

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{\boldsymbol{L}}{2} \|y - x\|_2^2$$

i.e., quadratic upper-bound of f

Additional assumption and notations

Assumption (Bounded second derivatives)

$\exists U \in \mathbb{R}$ *s.t.* $\forall x \in \mathbb{R}, \forall i \in [n]$

$$\phi_i''(x) \leq U$$

e.g., quadratic loss (regularized OLS): $\phi(w) = \frac{1}{2}(z - y_i)^2 \implies U = 1$

Definition (Subsample/batch function)

$$f_B(w) := \frac{1}{b} \sum_{i \in B} \phi_i(a_i^\top w) + \frac{\lambda}{2} \|w\|_2^2 \quad \forall B \subset [n]$$

$$\implies \nabla^2 f_B(w) = \frac{1}{b} \sum_{i \in B} \phi_i''(a_i^\top w) a_i a_i^\top + \lambda \mathbf{I}_d \preceq \frac{U}{b} \mathbf{A}_B \mathbf{A}_B^\top + \lambda \mathbf{I}_d$$

$$\text{with } \mathbf{A}_B = \underbrace{\begin{bmatrix} \vdots & & \vdots \\ a_{i_1} & \dots & a_{i_b} \\ \vdots & & \vdots \end{bmatrix}}_{b} \Bigg\} d$$

The jungle of smoothness constants

Definition (Subsample smoothness constant)

$$L_B := \frac{U}{|B|}\lambda_{\max}\left(\sum_{i\in B} a_i a_i^\top\right) = \frac{U}{|B|}\lambda_{\max}\left(\mathbf{A}_B\mathbf{A}_B^\top\right)$$

f_B is L_B–smooth.

- $B = [n] \implies L$: smoothness constant of f
- $B = \{i\} \implies L_i$: smoothness constant of f_i

Definition (Maximum smoothness constant)

$$L_{\max} := \max_{i\in[n]} L_i$$

Definition (Average smoothness constant)

$$\bar{L} := \frac{1}{n}\sum_{i=1}^{n} L_i$$

Key concept

Definition ($b-$simple random sampling)

S (a random set-valued mapping) is a $b-$simple random sampling if

$$\mathbb{P}\left[S = B\right] = \frac{1}{\binom{n}{b}} \quad \forall B \in \mathrm{supp}(S)$$

where $\mathrm{supp}(S) := \{B \subset [n] \ : \ |B| = b\}$.

Definition (Expected smoothness constant)

For a given sampling S,

$$\mathcal{L} := \frac{1}{c_1} \max_{i=1,\ldots,n} \left\{ \sum_{B \in \mathrm{supp}(S) \,|\, i \in B} L_B \right\}$$

where $c_1 := \left|\{B \in \mathrm{supp}(S) \ : \ i \in B\}\right|$.

Key concept

Definition ($b-$simple random sampling)

S (a random set-valued mapping) is a $b-$simple random sampling if

$$\mathbb{P}\left[S = B\right] = \frac{1}{\binom{n}{b}} \quad \forall B \in \mathrm{supp}(S)$$

where $\mathrm{supp}(S) := \{B \subset [n] \ : \ |B| = b\}$.

Definition (Expected smoothness constant)

For a given sampling S,

$$\mathcal{L} := \frac{1}{c_1} \max_{i=1,\ldots,n} \left\{ \sum_{B \in \mathrm{supp}(S) \,|\, i \in B} L_B \right\}$$

where $c_1 := \left|\{B \in \mathrm{supp}(S) \ : \ i \in B\}\right|$.

Here: $c_1 = \binom{n-1}{b-1}$

Key concept

Definition ($b-$simple random sampling)

S (a random set-valued mapping) is a $b-$simple random sampling if

$$\mathbb{P}\left[S = B\right] = \frac{1}{\binom{n}{b}} \quad \forall B \in \mathrm{supp}(S)$$

where $\mathrm{supp}(S) := \{B \subset [n] \ : \ |B| = b\}$.

Definition (Expected smoothness constant)

For a given sampling S,

$$\mathcal{L} := \frac{1}{c_1} \max_{i=1,\ldots,n} \left\{ \sum_{B \in \mathrm{supp}(S) \,|\, i \in B} L_B \right\}$$

where $c_1 := \left|\{B \in \mathrm{supp}(S) \ : \ i \in B\}\right|$.

Here: $c_1 = \binom{n-1}{b-1}$

Problem: Calculating $\mathcal{L}$ is intractable for large n

Importance of $\mathcal{L}$

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**

$$\text{find} \quad b^* \in \underset{b \in [n]}{\arg\min}\, K_{\text{total}}(b)$$

Importance of $\mathcal{L}$

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**

$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b)$$

- **Total complexity**

$$K_{\text{total}}(b) = \max\left\{\frac{4b(\boldsymbol{\mathcal{L}} + \lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max} + \lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

Importance of $\mathcal{L}$

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**

$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b)$$

- **Total complexity**

$$K_{\text{total}}(b) = \max\left\{\frac{4b(\boldsymbol{\mathcal{L}} + \lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max} + \lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

- **Why is $\mathcal{L}$ important?**
  - **$\mathcal{L}$ embodies the complexity**
  - Gives a **new step size γ**

Importance of $\mathcal{L}$

For a desired precision $\epsilon > 0$,

- **Optimal mini-batch size**

$$\text{find} \quad b^* \in \arg\min_{b \in [n]} K_{\text{total}}(b)$$

- **Total complexity**

$$K_{\text{total}}(b) = \max\left\{\frac{4b(\boldsymbol{\mathcal{L}} + \lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max} + \lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

- **Why is $\mathcal{L}$ important?**
  - **$\mathcal{L}$ embodies the complexity**
  - Gives a **new step size γ**

$\rightarrow$ **Need to upper-bound $\mathcal{L}$**

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \frac{1}{\binom{n-1}{b-1}} \max_{i=1,\ldots,n} \left\{ \sum_{B \in \text{supp}(S) \,|\, i \in B} L_B \right\}$$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \frac{1}{\binom{n-1}{b-1}} \max_{i=1,\ldots,n} \left\{ \sum_{B \in \text{supp}(S) \,|\, i \in B} L_B \right\}$$

- If $\boldsymbol{b = 1}$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \frac{1}{\binom{n-1}{0}} \max_{i=1,\dots,n} \left\{ \sum_{B \in \mathrm{supp}(S) \,|\, i \in B} L_B \right\}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \max_{i=1,\dots,n} \left\{ \sum_{B \in \{\{1\},\dots,\{n\}\} \mid i \in B} L_B \right\}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \dots, \{n\}\}$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \max_{i=1,\ldots,n} L_i$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \ldots, \{n\}\}$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \frac{1}{\binom{n-1}{b-1}} \max_{i=1,\dots,n} \left\{ \sum_{B \in \mathrm{supp}(S) \,|\, i \in B} L_B \right\}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \dots, \{n\}\}$

$$\mathcal{L} = L_{\max}$$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \frac{1}{\binom{n-1}{b-1}} \max_{i=1,\dots,n} \left\{ \sum_{B \in \mathrm{supp}(S) \,|\, i \in B} L_B \right\}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \dots, \{n\}\}$

$$\mathcal{L} = L_{\max}$$

- If $\boldsymbol{b = n}$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \frac{1}{\binom{n-1}{n-1}} \max_{i=1,\dots,n} \left\{ \sum_{B \in \mathrm{supp}(S) \,|\, i \in B} L_B \right\}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \dots, \{n\}\}$

$$\mathcal{L} = L_{\max}$$

- If $\boldsymbol{b = n}$
  - Recovered algorithm: Gradient Descent

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \max_{i=1,\dots,n} \left\{ \sum_{B \in \{[n]\} \mid i \in B} L_B \right\}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \dots, \{n\}\}$

$$\mathcal{L} = L_{\max}$$

- If $\boldsymbol{b = n}$
  - Recovered algorithm: Gradient Descent
  - $\mathrm{supp}(S) = \{[n]\}$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \max_{i=1,\ldots,n} L_{[n]}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \ldots, \{n\}\}$

$$\mathcal{L} = L_{\max}$$

- If $\boldsymbol{b = n}$
  - Recovered algorithm: Gradient Descent
  - $\mathrm{supp}(S) = \{[n]\}$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \max_{i=1,\ldots,n} L_{[n]}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \ldots, \{n\}\}$

$$\mathcal{L} = L_{\max}$$

- If $\boldsymbol{b = n}$
  - Recovered algorithm: Gradient Descent
  - $\mathrm{supp}(S) = \{[n]\}$

$$\mathcal{L} = L$$

Extreme values of $\mathcal{L}$

- **Recall**: $S = b-$simple random sampling

$$\mathcal{L} = \max_{i=1,\dots,n} L_{[n]}$$

- If $\boldsymbol{b = 1}$
  - Recovered algorithm: SAGA
  - $\mathrm{supp}(S) = \{\{1\}, \dots, \{n\}\}$

$$\mathcal{L} = L_{\max}$$

- If $\boldsymbol{b = n}$
  - Recovered algorithm: Gradient Descent
  - $\mathrm{supp}(S) = \{[n]\}$

$$\mathcal{L} = L$$

$\rightarrow$ **$\mathcal{L}$ interpolates between $L_{\max}$ and L**

The "simple" upper-bound

Lemma (Simple bound)

If S is a $b-$simple random sampling, we have

$$\mathcal{L} \leq \mathcal{L}_{simple} := \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{n}{b}\frac{b-1}{n-1}\bar{L}$$

The "simple" upper-bound

Lemma (Simple bound)

If S is a $b-$simple random sampling, we have

$$\mathcal{L} \leq \mathcal{L}_{simple} := \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{n}{b}\frac{b-1}{n-1}\bar{L}$$

- $\mathcal{L}_{\text{simple}}$ interpolates between $L_{\max}$ and $\bar{L}$ ...
- ... but $\mathcal{L}$ interpolates between $L_{\max}$ and L

The "simple" upper-bound

Lemma (Simple bound)

If S is a $b-$simple random sampling, we have

$$\mathcal{L} \leq \mathcal{L}_{simple} := \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{n}{b}\frac{b-1}{n-1}\bar{L}$$

- $\mathcal{L}_{\text{simple}}$ interpolates between $L_{\max}$ and $\bar{L}$ ...
- ... but $\mathcal{L}$ interpolates between $L_{\max}$ and L

Problem: $\bar{L}$ and L can be far from each other

Other estimates

Lemma (Bernstein bound)

If S is a $b-$simple random sampling, we have

$$\mathcal{L} \leq \mathcal{L}_{Bernstein} := \frac{1}{b}\left(\frac{n-b}{n-1} + \frac{4}{3}\log d\right) L_{\max} + 2\frac{b-1}{b}\frac{n}{n-1}L$$

Other estimates

Lemma (Bernstein bound)

If S is a $b-$simple random sampling, we have

$$\mathcal{L} \leq \mathcal{L}_{Bernstein} := \frac{1}{b}\left(\frac{n-b}{n-1} + \frac{4}{3}\log d\right) L_{\max} + 2\frac{b-1}{b}\frac{n}{n-1}L$$

- **Practical approximation**

$$\mathcal{L}_{\text{practical}} := \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{n}{b}\frac{b-1}{n-1}L$$

⚠ NOT an upper-bound of $\mathcal{L}$

Other estimates

Lemma (Bernstein bound)

If S is a $b-$simple random sampling, we have

$$\mathcal{L} \leq \mathcal{L}_{Bernstein} := \frac{1}{b}\left(\frac{n-b}{n-1} + \frac{4}{3}\log d\right) L_{\max} + 2\frac{b-1}{b}\frac{n}{n-1}L$$

- **Practical approximation**

$$\mathcal{L}_{\text{practical}} := \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{n}{b}\frac{b-1}{n-1}L$$

⚠ NOT an upper-bound of $\mathcal{L}$

$\rightarrow$ **Numerically** $\mathcal{L}_{\text{practical}} \approx \mathcal{L}$

Optimal mini-batch from the "simple" bound

For a desired precision $\epsilon > 0$,

- **Total complexity**

$$K_{\text{total}}(b) = \max\left\{\frac{4b(\mathcal{L}+\lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

- **Pessimistic total complexity**

$$K_{\text{total}}(b) \leq K^{\text{simple}}_{\text{total}}(b)$$
$$:= \max\left\{\frac{4b(\mathcal{L}_{\text{simple}}+\lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right)$$

- **Optimal mini-batch**

$$\text{find } b_{\text{simple}} \in \arg\min_{b\in[n]} K^{\text{simple}}_{\text{total}}(b) \implies \boxed{b_{\text{simple}} = \left\lfloor 1 + \frac{\mu(n-1)}{4(\bar{L}+\lambda)} \right\rfloor}$$

Larger step sizes

- **Convergence theorem**

$$\gamma = \frac{1}{4} \frac{1}{\max\left\{ \mathcal{L} + \lambda, \frac{1}{b}\frac{n-b}{n-1} L_{\max} + \frac{\mu}{4}\frac{n}{b} \right\}}$$

Larger step sizes

- **Convergence theorem**

$$\gamma = \frac{1}{4} \frac{1}{\max\left\{ \mathcal{L} + \lambda, \frac{1}{b}\frac{n-b}{n-1} L_{\max} + \frac{\mu}{4}\frac{n}{b} \right\}}$$

- The **smaller** $\mathcal{L}$ (the tighter our upper-bounds), the **larger** γ

Larger step sizes

- **Convergence theorem**

$$\gamma = \frac{1}{4} \frac{1}{\max\left\{ \mathcal{L} + \lambda, \frac{1}{b} \frac{n-b}{n-1} L_{\max} + \frac{\mu}{4} \frac{n}{b} \right\}}$$

- The **smaller** $\mathcal{L}$ (the tighter our upper-bounds), the **larger** γ

- Plugging in $\mathcal{L}_{\text{simple}}$, $\mathcal{L}_{\text{Bernstein}}$ and $\mathcal{L}_{\text{practical}}$ leads to larger step sizes for mini-batch SAGA

Talk Overview

Stochastic gradient methods to solve the ERM

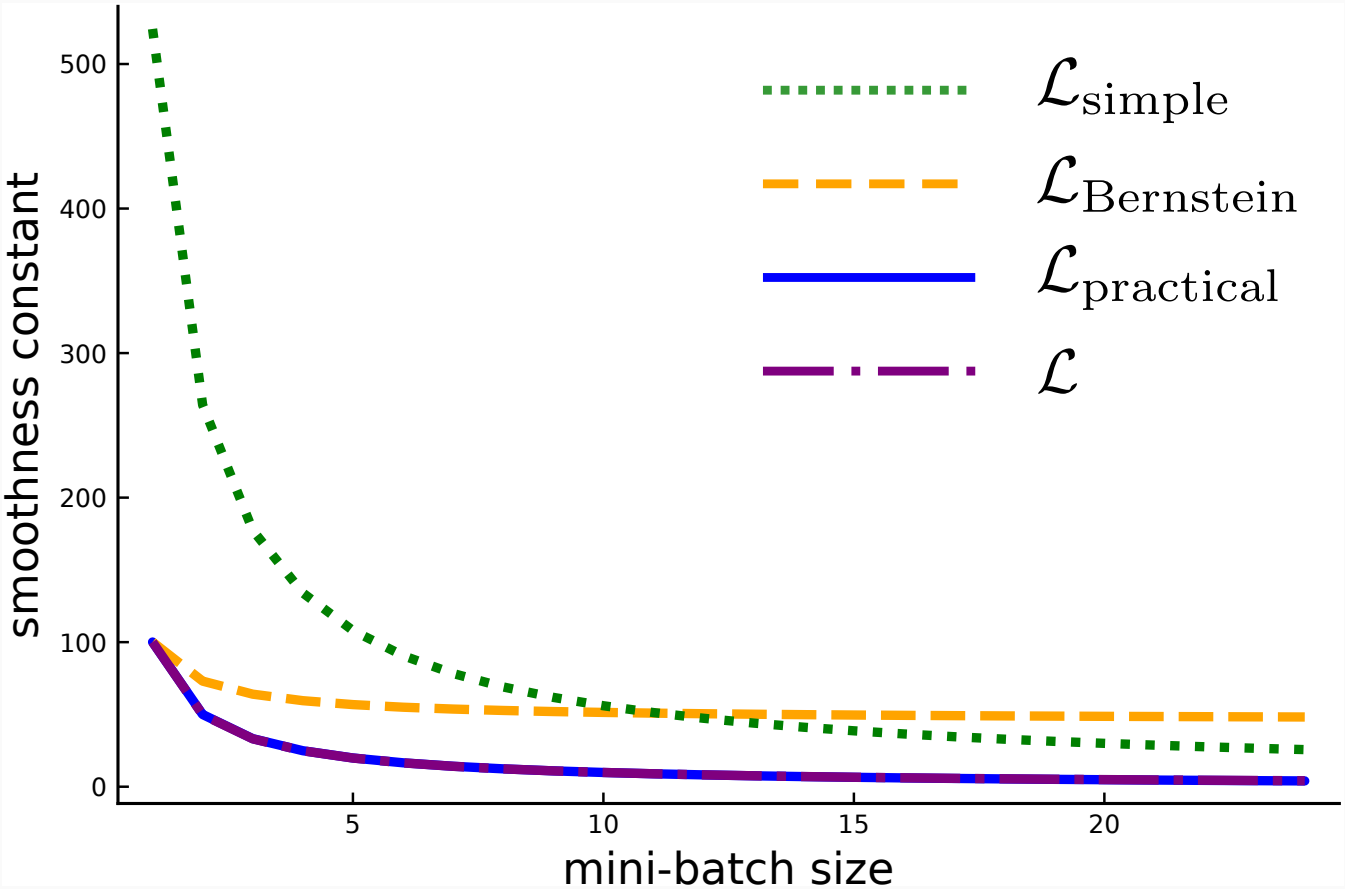
Optimal mini-batch and step sizes for SAGA

Numerical experiments

Conclusion

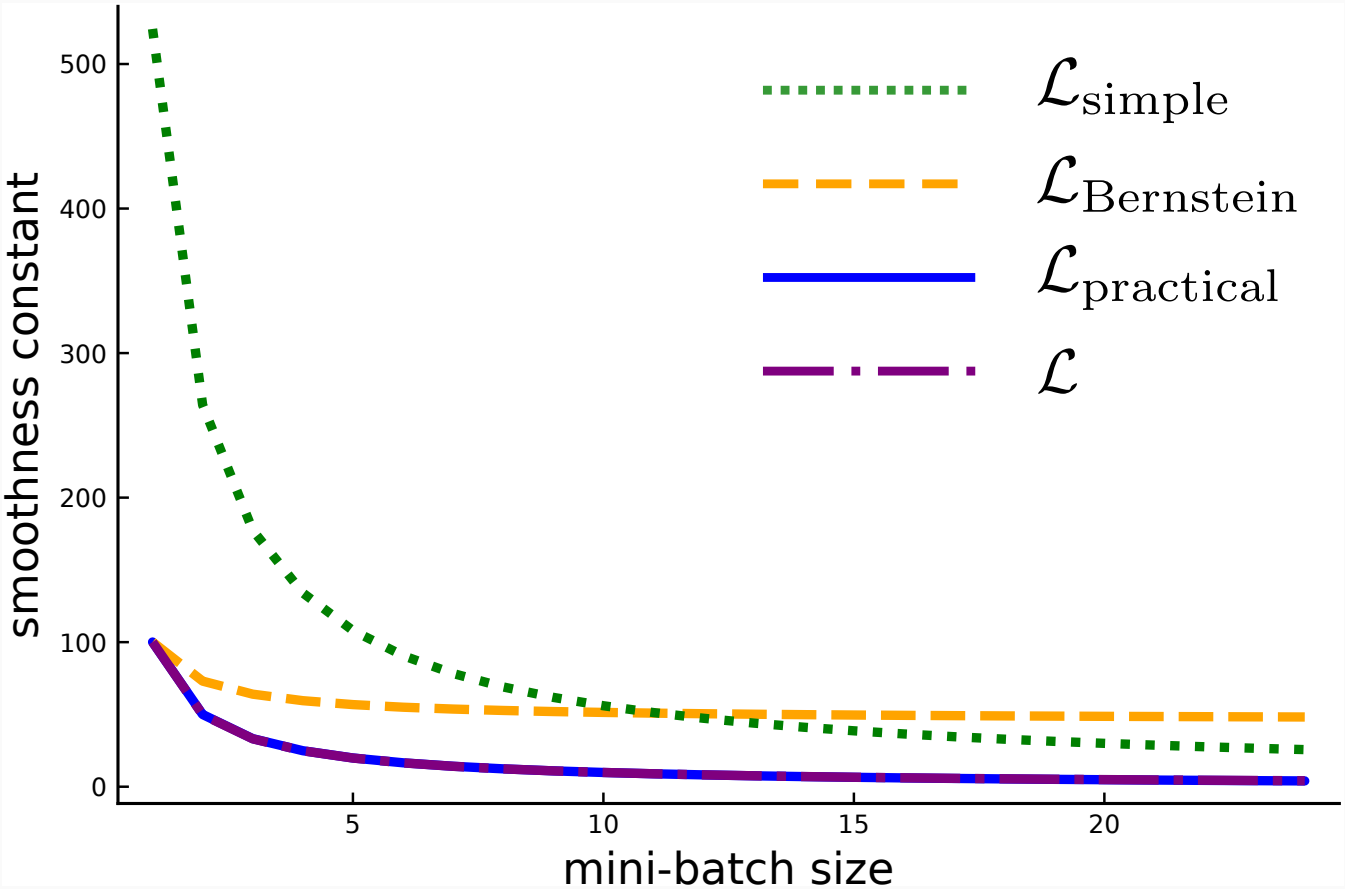
Numerical experiments

Tightness of our upper-bounds



Upper-bounds, practical approximation and $\mathcal{L}$ computed on artificial data ($n = d = 24$, $L_{\max} \approx 102$, $\bar{L} \approx 50$, $L \approx 6$)

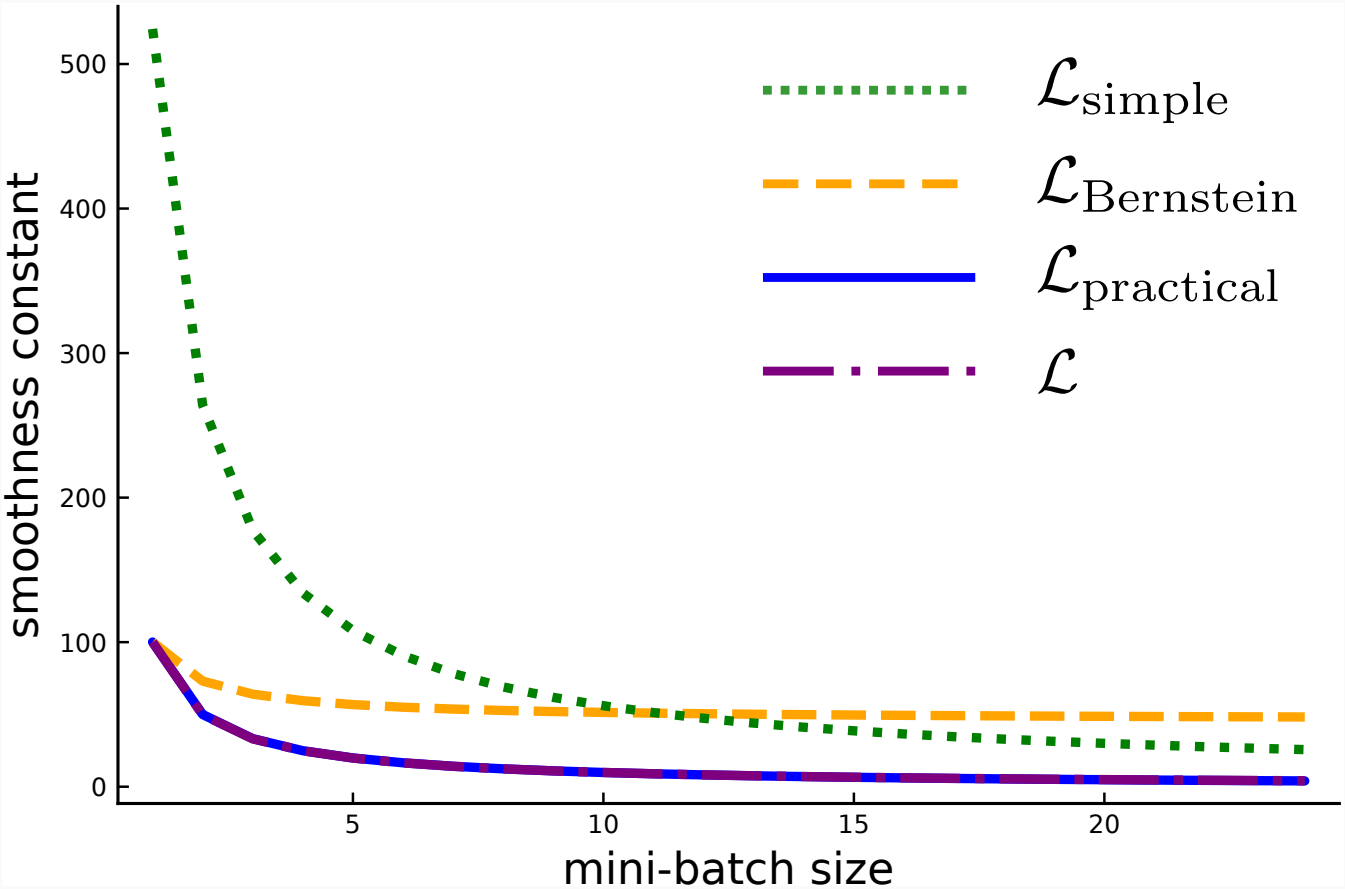
Tightness of our upper-bounds



Upper-bounds, practical approximation and $\mathcal{L}$ computed on artificial data ($n = d = 24$, $L_{\max} \approx 102$, $\bar{L} \approx 50$, $L \approx 6$)

$\rightarrow$ **Validity of our upper-bounds $\mathcal{L}_{\text{simple}}$ and $\mathcal{L}_{\text{Bernstein}}$**

Tightness of our upper-bounds

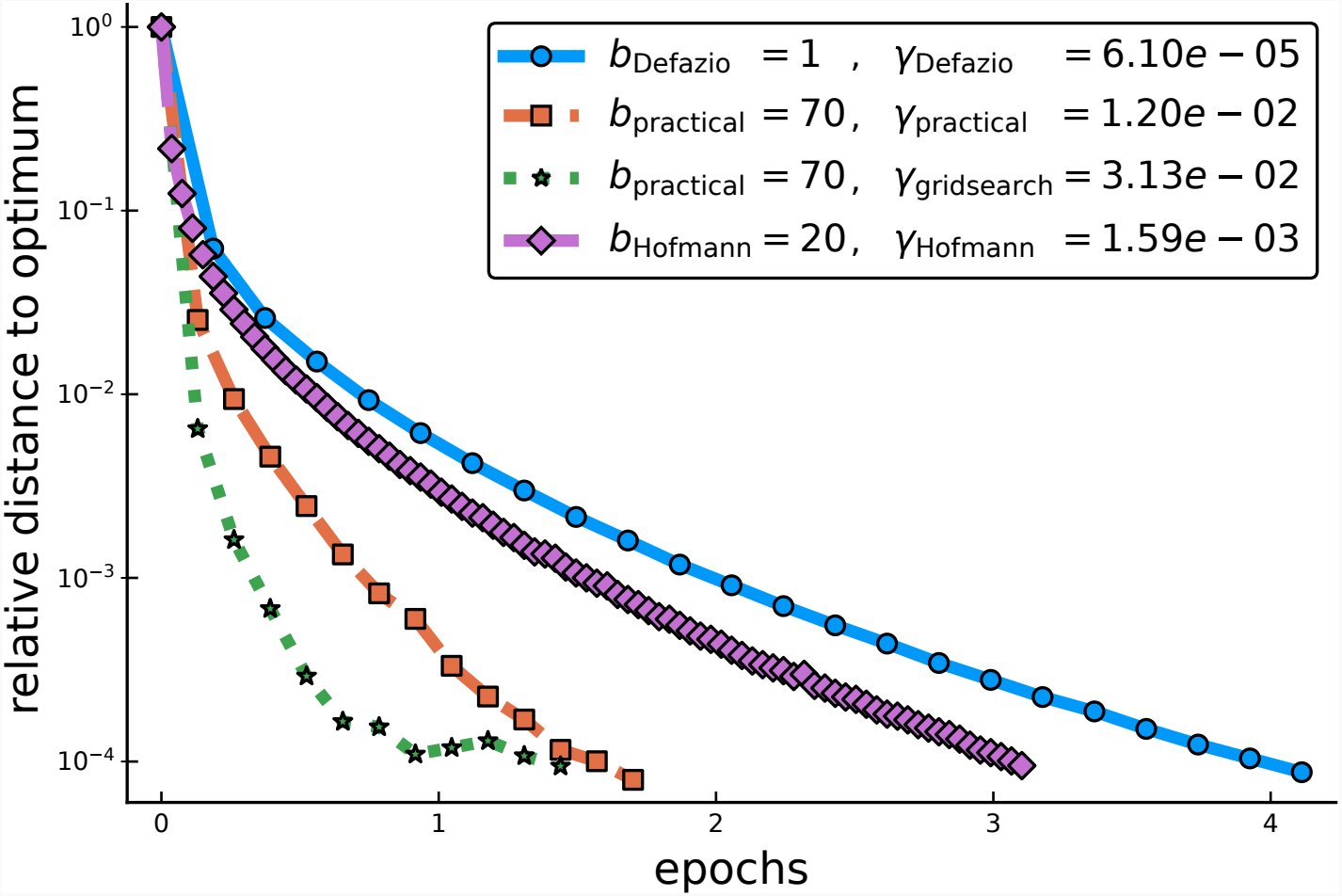


Upper-bounds, practical approximation and $\mathcal{L}$ computed on artificial data ($n = d = 24$, $L_{\max} \approx 102$, $\bar{L} \approx 50$, $L \approx 6$)

$\rightarrow$ **Validity of our upper-bounds $\mathcal{L}_{\text{simple}}$ and $\mathcal{L}_{\text{Bernstein}}$**

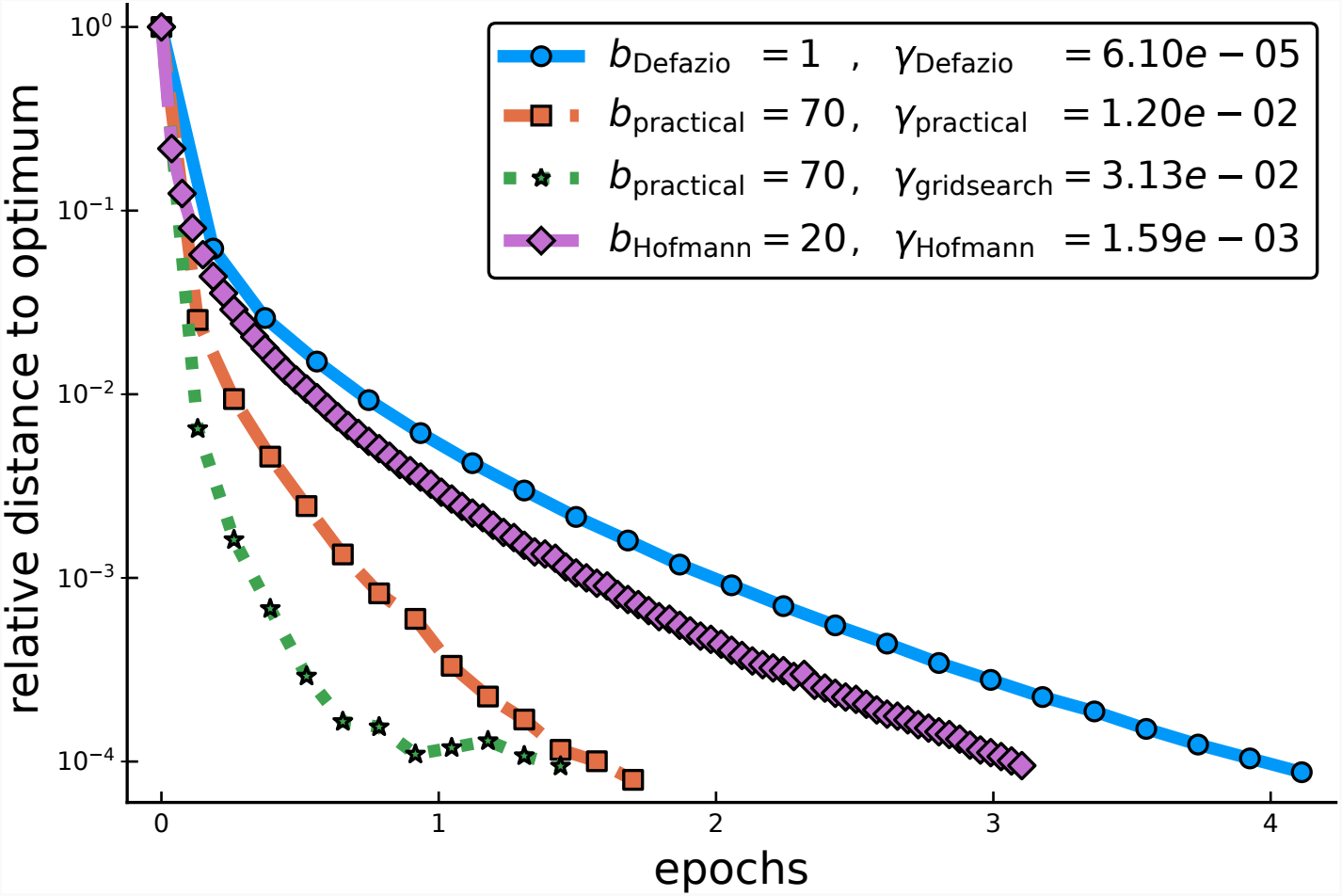
$\rightarrow$ **Practical approximation $\mathcal{L}_{\text{practical}}$ close to $\mathcal{L}$**

Convergence results on real data



Comparison of SAGA settings for the unscaled *slice* data set ($n = 53,500, d = 384$)

Convergence results on real data



Comparison of SAGA settings for the unscaled *slice* data set ($n = 53,500, d = 384$)

→ **Larger mini-batch and step sizes: faster convergence**

Talk Overview

Stochastic gradient methods to solve the ERM

Optimal mini-batch and step sizes for SAGA

Numerical experiments

Conclusion

Conclusion

Summary

What was done

- Build estimates of $\mathcal{L}$
- Give larger mini-batch b and step sizes γ for SAGA

$$\Longrightarrow \text{ Faster convergence of } w^k \xrightarrow[k\to\infty]{} w^*$$

- Provide convincing numerical improvements on real datasets

What is to be done

- Extend the study to other stochastic gradient algorithms, like SVRG[6]

[6] Johnson and Zhang, 2013

References (1/2)

- F. Bach. "Sharp analysis of low-rank kernel matrix approximations". In:ArXiv e-prints (Aug. 2012). arXiv: 1208.2015 [cs.LG].
- C. C. Chang and C. J. Lin. "LIBSVM : A library for support vector machines". In: ACM Transactions on Intelligent Systems and Technology 2.3 (Apr. 2011), pp. 127.
- A. Defazio, F. Bach, and S. Lacoste-julien. "SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives". In: Advances in Neural Information Processing Systems 27. 2014, pp. 16461654.
- R. M. Gower, P. Richtrik, and F. Bach. "Stochastic Quasi-Gradient Methods: Variance Reduction via Jacobian Sketching". In: arXiv preprint arXiv:1805.02632 (2018).
- D. Gross and V. Nesme. "Note on sampling without replacing from a finite collection of matrices". In: arXiv preprint arXiv:1001.2738 (2010)
- W. Hoeffding. "Probability inequalities for sums of bounded random variables". In: Journal of the American statistical association 58.301 (1963), pp. 1330.

References (2/2)

- R. Johnson and T. Zhang. "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction". In: Advances in Neural Information Processing Systems 26. Curran Associates, Inc., 2013, pp. 315323.
- H. Robbins and S. Monro. "A stochastic approximation method". In: Annals of Mathematical Statistics 22 (1951), pp. 400407.
- M. Schmidt, N. Le Roux, and F. Bach. "Minimizing finite sums with the stochastic average gradient". In: Mathematical Programming 162.1 (2017), pp. 83112.
- J. A. Tropp. "An Introduction to Matrix Concentration Inequalities". In: ArXiv e-prints (Jan. 2015). arXiv:1501.01571 [math.PR]
- J. A. Tropp. "Improved analysis of the subsampled randomized Hadamard transform". In: Advances in Adaptive Data Analysis 3.01n02 (2011), pp. 115126.
- J. A. Tropp. "User-Friendly Tail Bounds for Sums of Random Matrices". In: Foundations of Computational Mathematics 12.4 (2012), pp. 389434.

Questions?

JacSketch Lyapunov function

Definition (Stochastic Lyapunov function)

$$\Psi^k := \left\|w^k - w^*\right\|_2^2 + \frac{\gamma}{2bL_{\max}} \left\|\mathbf{J}^k - \nabla \mathbf{F}(w^*)\right\|_{\mathrm{F}}^2$$

- $\|\cdot\|_{\mathrm{F}}$: Frobenius norm
- $\nabla \mathbf{F}(w) = [\nabla f_1(w), \ldots, \nabla f_n(w)] \in \mathbb{R}^{d \times n}$: Jacobian matrix
- $\left\{w^k, \mathbf{J}^k\right\}_{k \geq 0}$ are the points and Jacobian estimate

If $\epsilon > 0$ denotes the desired precision, Theorem 3.6 ensures that, for a step size $\gamma = \min\left\{\frac{1}{4\mathcal{L}}, \frac{1}{\frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{\mu}{4}\frac{n}{b}}\right\}$,

$$\mathbb{E}\left[\Psi^k\right] \leq \epsilon \Psi^0 \quad .$$

(Gower, Richtárik and Bach, 2018)

From sampling without to with replacement

Lemma (Domination of the trace of the mgf of a sample without replacement)

Consider two finite sequences, of same length, $\{\mathbf{X}_k\}$ *and* $\{\mathbf{M}_k\}$ *of Hermitian random matrices of same size sampled respectively with and without replacement from a finite set* $\mathcal{X}$*. Let* $\theta \in \mathbb{R}$*, then*

$$\mathbb{E}\,\mathrm{tr}\,\exp\left(\theta \sum\nolimits_k \mathbf{M}_k\right) \leq \mathbb{E}\,\mathrm{tr}\,\exp\left(\theta \sum\nolimits_k \mathbf{X}_k\right) \quad .$$

(Gross and Nesme, 2010)

Proof sketch of the matrix concentration bound (1/2)

(i) Write $\mathcal{L}$ as an **expectation**

$$\mathcal{L} = \max_{i=1,\dots,n} \mathbb{E}\left[L_{S^i \cup \{i\}}\right]$$

$$= \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j \in S^n \cup \{i\}} a_j a_j^\top\right)\right]$$

$$\leq \frac{1}{b}\frac{n-b}{n-1} L_{\max} + \frac{n}{b}\frac{b-1}{n-1} L$$

$$+ \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\underbrace{\frac{1}{b}\sum_{j \in S^i} a_j a_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j \in [n]\setminus\{i\}} a_j a_j^\top}_{\mathbf{N} = \sum_k \mathbf{M}_k}\right)\right]$$

Proof sketch of the matrix concentration bound (1/2)

(i) Write $\mathcal{L}$ as an **expectation**

$$\mathcal{L} = \max_{i=1,\dots,n} \mathbb{E}\left[L_{S^i \cup \{i\}}\right]$$

$$= \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j \in S^n \cup \{i\}} a_j a_j^\top\right)\right]$$

$$\leq \frac{1}{b}\frac{n-b}{n-1} L_{\max} + \frac{n}{b}\frac{b-1}{n-1} L$$

$$+ \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\underbrace{\frac{1}{b}\sum_{j \in S^i} a_j a_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j \in [n]\setminus\{i\}} a_j a_j^\top}_{\mathbf{N} = \sum_k \mathbf{M}_k}\right)\right]$$

Practical approximation

Proof sketch of the matrix concentration bound (2/2)

(ii) Write $\mathbf{N}$ as a sum of random matrices and apply

Theorem (Matrix Bernstein Inequality Without Replacement)

Let $\mathcal{X}$ be a finite set of Hermitian matrices with dimension d s.t.

$$\lambda_{\max}(\mathbf{X}) \leq L \quad \forall \mathbf{X} \in \mathcal{X} \ .$$

Sample $\{\mathbf{X}_k\}$ and $\{\mathbf{M}_k\}$ uniformly at random from $\mathcal{X}$ resp. with and without replacement s.t.

$$\mathbb{E}\,\mathbf{X}_k = 0 \quad \forall k \ .$$

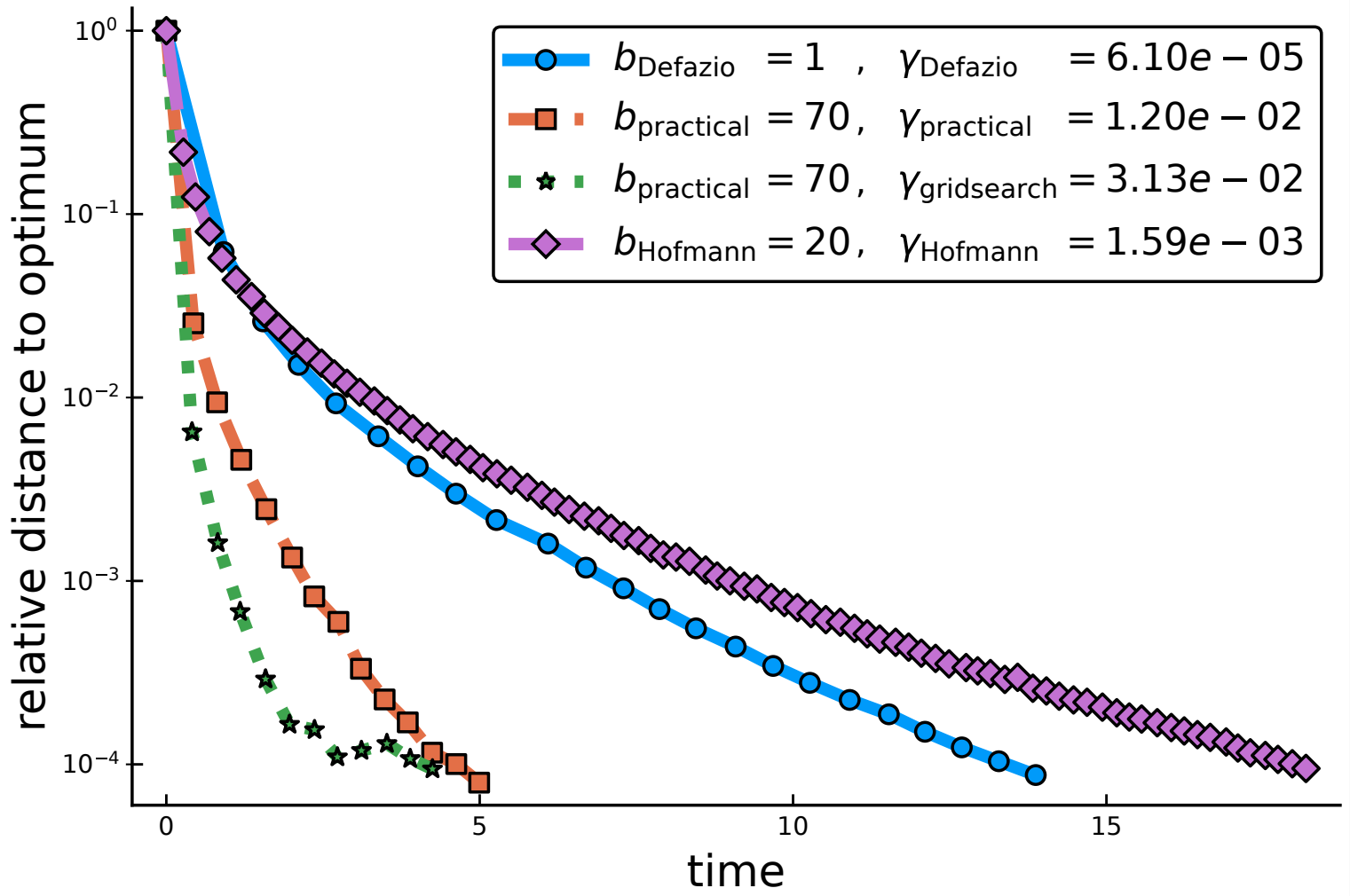
Let $\mathbf{Y} := \sum_k \mathbf{X}_k$ and $\mathbf{N} := \sum_k \mathbf{M}_k$. Then

$$\mathbb{E}\,\lambda_{\max}(\mathbf{N}) \leq \sqrt{2v(\mathbf{Y})\log d} + \frac{1}{3}L\log d \ .$$

where

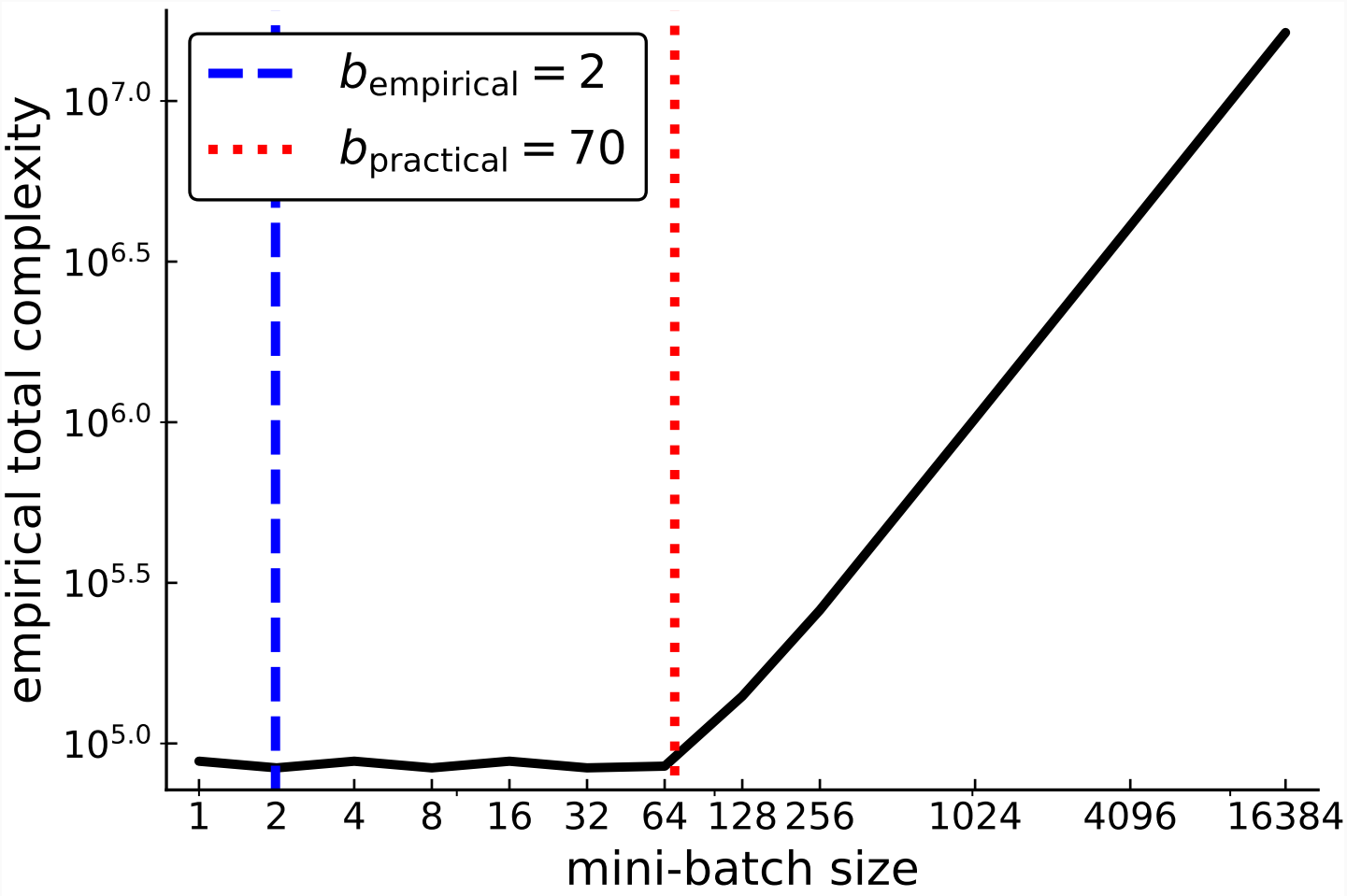
$$v(\mathbf{Y}) := \left\|\mathbb{E}\,\mathbf{Y}^2\right\| = \left\|\sum\nolimits_k \mathbb{E}\,\mathbf{X}_k^2\right\| = \lambda_{\max}\left(\sum\nolimits_k \mathbb{E}\,\mathbf{X}_k^2\right).$$

Convergence results on real data in time



Comparison of SAGA settings for the unscaled *slice* data set

Optimality of our mini-batch size



Complexity explosion for the unscaled *slice* dataset ($\lambda = 10^{-1}$)