COMP 414/514: Optimization – Algorithms, Complexity and Approximations

Lecture7

Overview

- In the last lecture, we:
  - Talked about how acceleration leads to a better convergence rate
  - Worked in practice and theory with accelerated gradient descent variants
  - Discussed the limits and convergence rates of accelerated gradient descent
- Often, gradient descent is not sufficient in practice. In this lecture, we will:
  - Discuss **alternatives to batch gradient descent: stochastic gradient descent**
  - Discuss **alternatives to batch gradient descent: coordinate descent**
  - Discuss recent advances on these topics

Acceleration #2: Cut-off complexity per iteration

- Common situation in machine learning/signal processing

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

(Empirical risk minimization)

where each $f_i(x)$ depends on, let's say, different part of input data.

- Examples:
  - Least squares: $f_i(x) = \frac{1}{2}(y_i - \alpha_i^\top x)^2$
  - Logistic regression: $f_i(x) = \log(1 + \exp(-y_i \alpha_i^\top x))$
- Dimensions to worry about: $x \in \mathbb{R}^p$, number of samples n

Acceleration #2: Cut−off complexity per iteration

- **Stochastic gradient descent (SGD)**

$$x_{t+1} = x_t - \eta_t \nabla f(x_t) \longrightarrow x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t), \quad i_t \in [n]$$

  where per iteration we select randomly $i_t \in [n]$.

- "Why do we want to do this?"

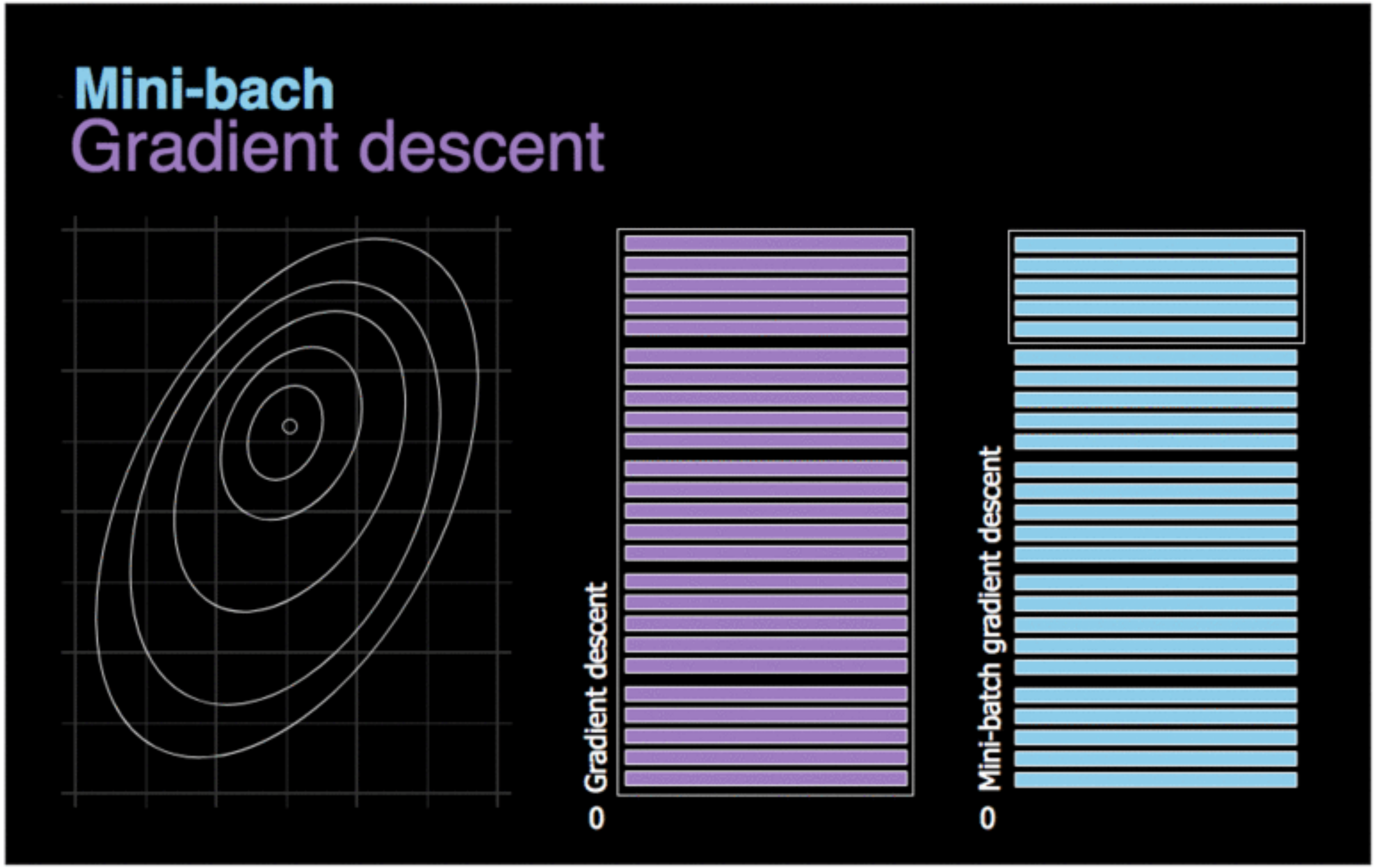
  What is the complexity of computing full gradient?

  What is the complexity of computing a single gradient?

  $O(np)$ (Assume least−squares objective) $O(p)$

- When is $n \gg p$? Big−data regime!
  - There is redundancy in data
  - Far from the optimal, exact gradients might have small returns

Acceleration #2: Cut-off complexity per iteration



Tribute to
Piotr Skalski

Acceleration #2: Cut−off complexity per iteration

- Some notes on SGD (before we proceed)

1. It is a stochastic process that depends on a random sequence $i_t \in [n]$

   (This means you will see some probability involved in theory)

2. While $-\nabla f(x_t)$ is a descent direction, $-\nabla f_{i_t}(x_t)$ might not be

   (This means that some tools from deterministic optimization do not hold here)

3. The above lead to the intuition that if we have a descent direction in **expectation,** we probably will perform just fine

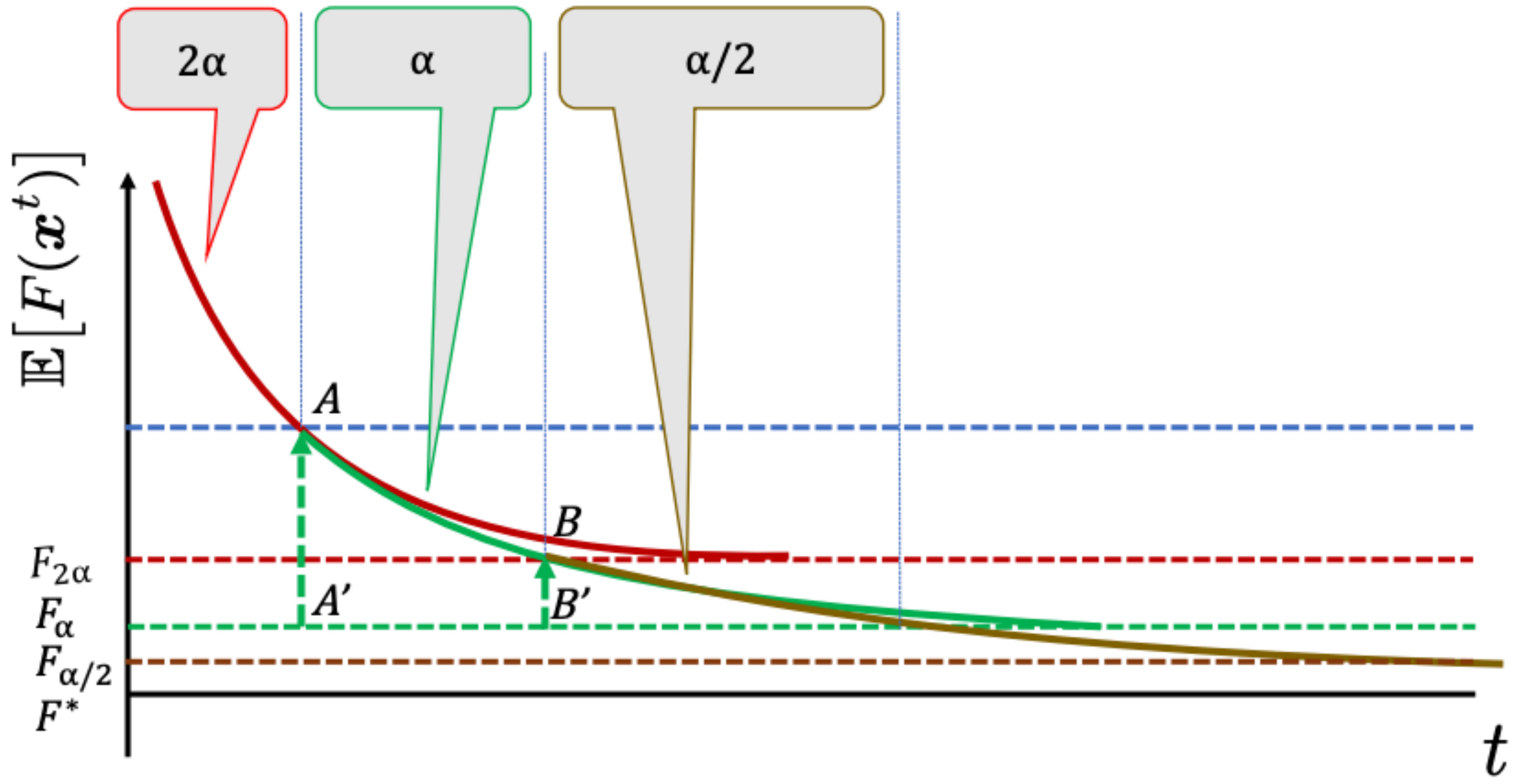
   (This means that we will work with expectations w.r.t. the random sequence)

Guarantees of SGD

Whiteboard

Guarantees of SGD

Pic. from: "Optimization Methods for Large-Scale Machine Learning"



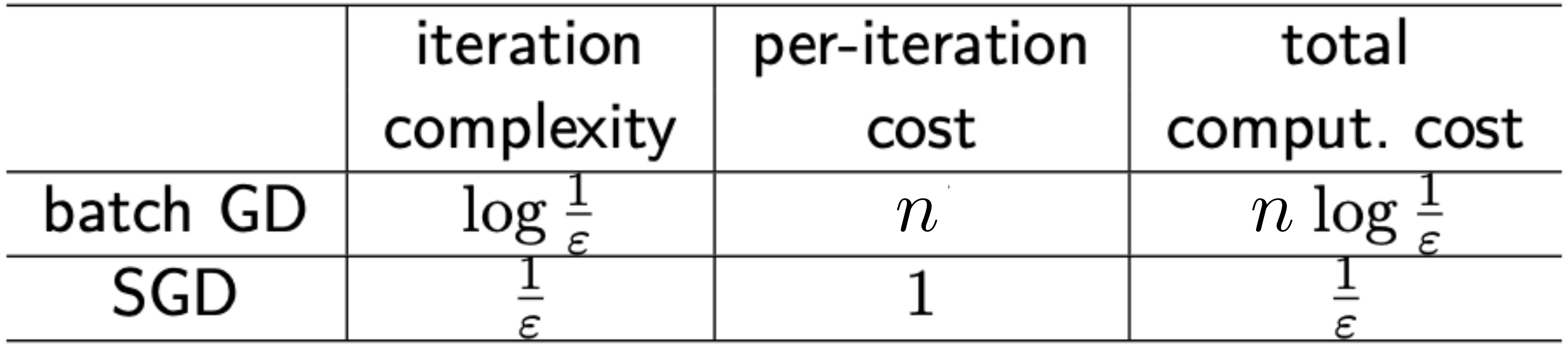
- Start with large step size; decrease it when SGD "stalls"

Guarantees of SGD

Whiteboard

Intuition behind preference for SGD

- Overall, for strongly convex and smooth functions

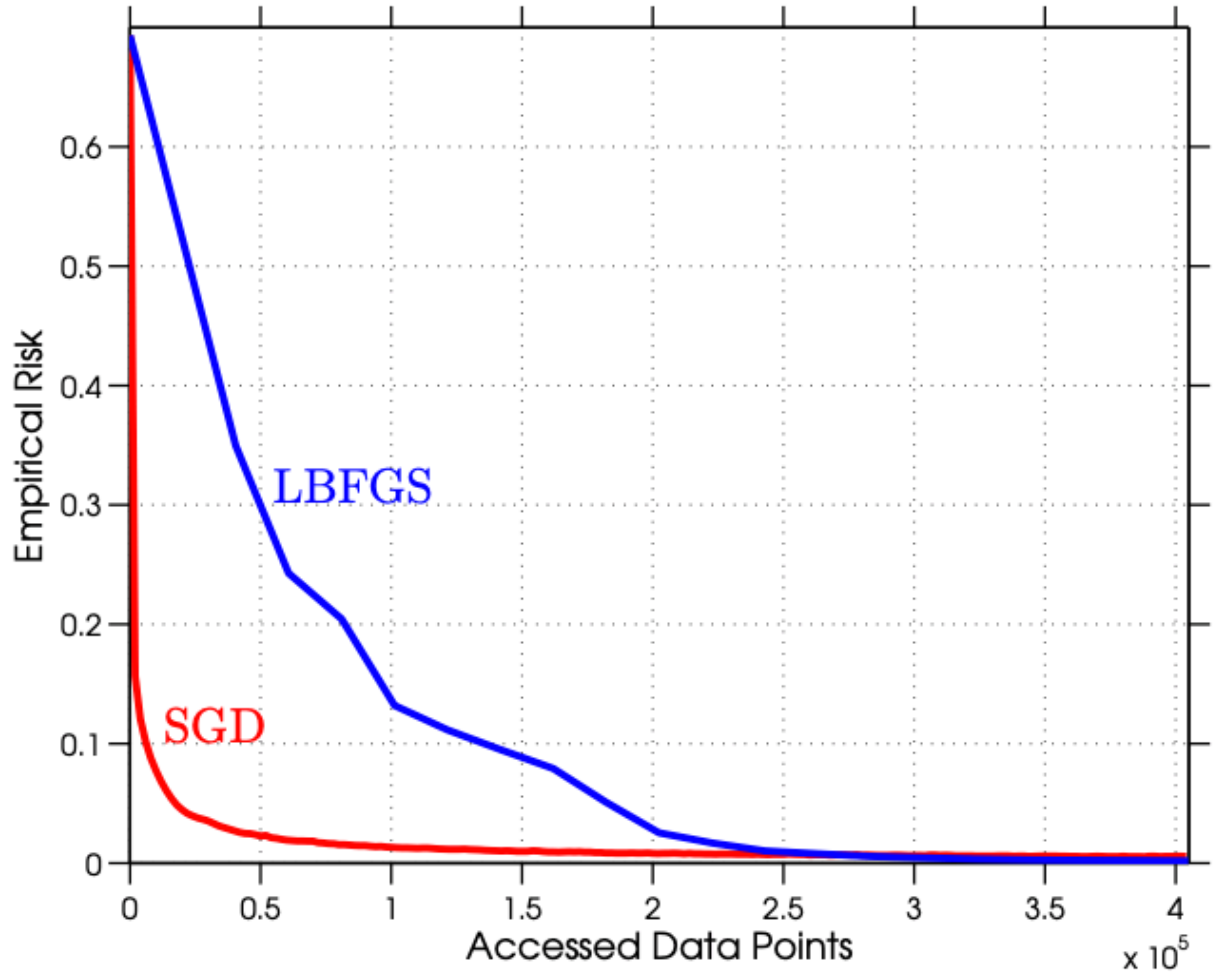


	iteration complexity	per-iteration cost	total comput. cost
batch GD	$\log \frac{1}{\varepsilon}$	n	$n \log \frac{1}{\varepsilon}$
SGD	$\frac{1}{\varepsilon}$	1	$\frac{1}{\varepsilon}$

- The real comparison is between $n \log \frac{1}{\varepsilon} \ ? \ \frac{1}{\varepsilon}$
- In the big data regime, n can be huge!
- Gradient descent uses full dataset per iteration; there might be **redundancies**
- It actually works great in practice!

Intuition behind preference for SGD

Pic. from: “Optimization Methods for Large-Scale Machine Learning“



Variants of SGD

- Mini-batch SGD: instead of picking one sample, pick multiple

$$x_{t+1} = x_t - \eta_t \nabla f_{\mathcal{I}_t}(x_t) = x_t - \eta_t \cdot \sum_{j=1}^{|\mathcal{I}_t|} \nabla f_j(x_t)$$

  - Still less time than computing the full gradient
  - Converges to a smaller ball around optimum: trade-off

- SGD with importance sampling: select ``carefully`` the next sample

  - Select $i_t \in [n]$ according to distribution $p \in [0,1]^n, \ \sum_i p_i = 1$
  - Main question: can we compute a good probability distribution without too much effort?

Variants of SGD

- Stochastic variance-reduced gradient (SVRG)

$$x_{t+1} = x_t - \eta_t \left(\nabla f_{i_t}(x_t) - \left(\nabla f_{i_t}(\widetilde{x}_q) - \nabla f(\widetilde{x}_q)\right)\right)$$

Bias in gradient estimate
Correction term

- Observe that: $\mathbb{E}[\nabla f_{i_t}(\cdot)] = \nabla f(\cdot)$; then

$$\mathbb{E}\left[\nabla f_{i_t}(x_t) - \nabla f_{i_t}(\widetilde{x}_q) + \nabla f(\widetilde{x}_q)\right] = \nabla f(x_t)$$

Unbiased estimator!
We expect smaller variance

- Theoretical guarantees:

$$\mathbb{E}\left[f(x_{t+1}) - f(x^\star)\right] \leq \rho \cdot \mathbb{E}\left[f(x_t) - f(x^\star)\right], \quad \rho = O\left(\tfrac{1}{1-2\eta L} \cdot \left(\tfrac{1}{m\eta} + 2L\eta\right)\right) < 1$$

- Main drawback: Full gradient, but overall complexity $O\left((n+\kappa)\log\frac{1}{\varepsilon}\right)$

Performance of SGD

Demo

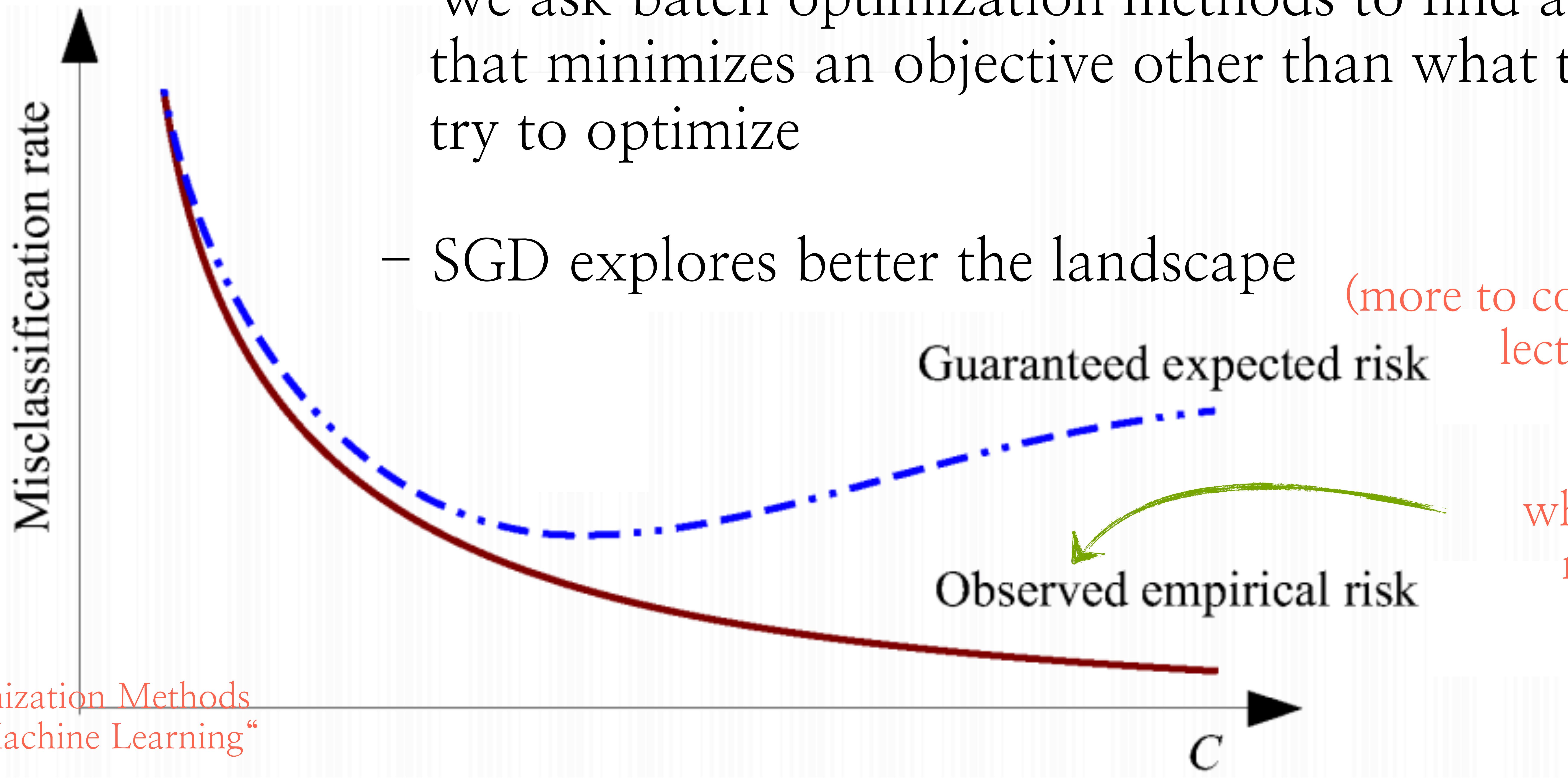
Why SGD is so important in machine learning?

(or some of the reasons)

- We ask batch optimization methods to find a model that minimizes an objective other than what they try to optimize

- SGD explores better the landscape

(more to come in future lectures)



where batch methods thrive

Pic. from: "Optimization Methods for Large-Scale Machine Learning"

Acceleration #3: Coordinate descent methods

- Stochastic gradient descent (SGD) selects mini batches of training data; what if we subselect variables to update per iteration?

$$x_{t+1} = x_t - \eta_t \nabla f(x_t) \quad \longrightarrow \quad (x_{t+1})_{i_t} = (x_t)_{i_t} - \eta_t \nabla_{i_t} f(x_t),$$

  where per iteration we select randomly $i_t \in [p]$.

- "Why do we want to do this?"

  What is the complexity of computing full gradient?

  What is the complexity of computing a gradient for a single variable?

  $O(np)$ (Assume least-squares objective) $O(n)$

- When is $n \ll p$? High-dimensional case
  - There is not enough data
  - We will see that it provides solutions to distributed systems