SGD and Friends

How to solve large-scale optimization problems?

Ketan Rajawat
February 24, 2020

Indian Institute of Technology Kanpur
1 Context

2 Background

3 Vanilla Stochastic Gradient Descent: Large $N$

4 Variance-Reduced SGD: Moderate $N$

5 High-dimensional problems: large $d$

6 Conclusion
Context
Outline

1 Context
   Problem Formulation: Online and Finite Sum
   Examples
   State-of-the-art and Oracle Complexity

2 Background

3 Vanilla Stochastic Gradient Descent: Large $N$

4 Variance-Reduced SGD: Moderate $N$

5 High-dimensional problems: large $d$

6 Conclusion
Consider the optimization problem:

$$\min_{x \in X} F(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i)$$

\( (P) \)
Consider the optimization problem:

$$\min_{x \in \mathcal{X}} F(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i)$$

(\mathcal{P})

- \( \mathcal{X} \subseteq \mathbb{R}^d \) where \( d \) is problem dimension
Consider the optimization problem:

$$\min_{x \in \mathcal{X}} F(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i)$$

(\mathcal{P})

- $\mathcal{X} \subseteq \mathbb{R}^d$ where $d$ is problem dimension
- $\xi_i$ indexes the data points/observations/samples
Consider the optimization problem:

\[
\min_{x \in X} F(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i)
\]  

\( \mathcal{P} \)

- \( X \subseteq \mathbb{R}^d \) where \( d \) is problem dimension
- \( \xi_i \) indexes the data points/observations/samples
- \( N \) is the size of data set
Variants

• Online optimization or $N \to \infty$

$$\min_{x \in X} F(x) := \mathbb{E}_\xi [f(x, \xi)]$$
Variants

- Online optimization or $N \to \infty$

\[
\min_{x \in X} F(x) := \mathbb{E}_\xi [f(x, \xi)]
\]

- Use a regularizer $h$

\[
\min_{x \in X} R(x) := F(x) + h(x)
\]
Variants

- Online optimization or $N \to \infty$
  \[
  \min_{x \in \mathcal{X}} F(x) := \mathbb{E}_\xi [f(x, \xi)]
  \]

- Use a regularizer $h$
  \[
  \min_{x \in \mathcal{X}} R(x) := F(x) + h(x)
  \]

- Distributed/decentralized setting with $K$ nodes
  \[
  \min_{x \in \mathcal{X}} \sum_{k=1}^{K} R_k(x)
  \]
Challenges of Big Data

- Large dimension $d$
  - Hessian inverse $[\nabla^2 F(x)]^{-1}$ requires $\mathcal{O}(d^3)$ computations
  - Approximate Hessian inverse still requires $\mathcal{O}(d^2)$ computations, e.g., BFGS
  - Very large $d$: must store $x$ on the disk instead of RAM, write operation is bottleneck
Challenges of Big Data

- **Large dimension** $d$
  - Hessian inverse $[\nabla^2 F(x)]^{-1}$ requires $O(d^3)$ computations
  - Approximate Hessian inverse still requires $O(d^2)$ computations, e.g., BFGS
  - Very large $d$: must store $x$ on the disk instead of RAM, write operation is bottleneck

- **Large dataset size** $N$
  - Even calculating the gradient $\nabla F(x)$ at every iteration impractical
  - Cannot store entire data on a single machine
  - Read/write operations become the bottleneck
Challenges of Big Data

- Large dimension $d$
  - Hessian inverse $[\nabla^2 F(x)]^{-1}$ requires $O(d^3)$ computations
  - Approximate Hessian inverse still requires $O(d^2)$ computations, e.g., BFGS
  - Very large $d$: must store $x$ on the disk instead of RAM, write operation is bottleneck

- Large dataset size $N$
  - Even calculating the gradient $\nabla F(x)$ at every iteration impractical
  - Cannot store entire data on a single machine
  - Read/write operations become the bottleneck

- Ideally complexity should be $O(dN)$
Outline

1 Context

   Problem Formulation: Online and Finite Sum
   Examples
   State-of-the-art and Oracle Complexity

2 Background

3 Vanilla Stochastic Gradient Descent: Large $N$

4 Variance-Reduced SGD: Moderate $N$

5 High-dimensional problems: large $d$

6 Conclusion
Example: Lasso Regression

Predictors for breast cancer selected via LASSO regression [Wang et al., 2016]

<table>
<thead>
<tr>
<th>Variables</th>
<th>Coefficient</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Premenopausal</td>
<td>Postmenopausal</td>
</tr>
<tr>
<td>Age</td>
<td>0.367</td>
<td>0.346</td>
</tr>
<tr>
<td>Body mass index</td>
<td>0.935</td>
<td></td>
</tr>
<tr>
<td>Age at menarche</td>
<td></td>
<td>-0.075</td>
</tr>
<tr>
<td>Age at 1st give birth</td>
<td>0.141</td>
<td></td>
</tr>
<tr>
<td>Number of parity</td>
<td>0.137</td>
<td>-0.184</td>
</tr>
<tr>
<td>Breast feeding</td>
<td></td>
<td>-0.110</td>
</tr>
<tr>
<td>Oral contraceptive</td>
<td></td>
<td>-0.090</td>
</tr>
<tr>
<td>hormone replace treatment</td>
<td></td>
<td>-0.710</td>
</tr>
<tr>
<td>Case number of BCFDR</td>
<td>0.855</td>
<td>0.844</td>
</tr>
<tr>
<td>Benign breast diseases</td>
<td></td>
<td>0.296</td>
</tr>
<tr>
<td>Alcohol drinking</td>
<td>0.631</td>
<td></td>
</tr>
<tr>
<td>LAN</td>
<td>0.264</td>
<td>0.238</td>
</tr>
<tr>
<td>Sleep quality</td>
<td>-0.256</td>
<td>-0.122</td>
</tr>
</tbody>
</table>

Age (20, 30, 40, 50, 60, 70, and >70 years old); body mass index (<18.5, 18.5–24, 24–27, and >27); age at menarche (<12, 12, 13, 14, 15, and 16–years old); age at 1st give birth (<20, 20–25, and 25–years old); number of parity (0, 1, 2, and >2); breast feeding duration (no, <1, 1–3 and, >3 years); LAN (1, dark; 2, few light; and 3, little bright); sleep quality (1, good; 2, common; 3; poor; and 4, poor with sleep pill). BCFDR = breast cancer in first degree-relatives, LAN = light at night, LASSO = least absolute shrinkage and selection operator, SD = standard deviation.
Example: Lasso Regression

- Given feature-label pairs \((a_i, b_i)\) for each patient \(i \in \{1, \ldots, N\}\)
Example: Lasso Regression

- Given feature-label pairs \((a_i, b_i)\) for each patient \(i \in \{1, \ldots, N\}\)
- Optimization problem formulated as

\[
\min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \ell(a_i^\top x, b_i) + \lambda \|x\|_1
\]

- Loss function \(\ell\) could be least-squares, logistic, hinge loss, etc.
- Non-zero entries of \(x\) correspond to features that explain \(b_i\)
- \(\ell_1\)-norm penalty "encourages" sparsity
Example: Lasso Regression

- Given feature-label pairs \((a_i, b_i)\) for each patient \(i \in \{1, \ldots, N\}\)
- Optimization problem formulated as

\[
\min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \ell(a_i^\top x, b_i) + \lambda \|x\|_1
\]

- Loss function \(\ell\) could be least-squares, logistic, hinge loss, etc.
Example: Lasso Regression

• Given feature-label pairs \((a_i, b_i)\) for each patient \(i \in \{1, \ldots, N\}\)

• Optimization problem formulated as

\[
\min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \ell(a_i^\top x, b_i) + \lambda \|x\|_1
\]

• Loss function \(\ell\) could be least-squares, logistic, hinge loss, etc.

• Non-zero entries of \(x\) correspond to features that explain \(b_i\)
Example: Lasso Regression

- Given feature-label pairs \((a_i, b_i)\) for each patient \(i \in \{1, \ldots, N\}\)
- Optimization problem formulated as

\[
\min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \ell(a_i^\top x, b_i) + \lambda \|x\|_1
\]

- Loss function \(\ell\) could be least-squares, logistic, hinge loss, etc.
- Non-zero entries of \(x\) correspond to features that explain \(b_i\)
- \(\ell_1\)-norm penalty “encourages” sparsity
Example: Visual Object Recognition

CIFAR-10 dataset contains 60000 labeled images of 10 objects [Krizhevsky, 2009]
Example: Neural Networks

- Given feature-label pairs \((a_i, b_i)\), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]
• Given feature-label pairs \((a_i, b_i)\), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]

• Objective \(f\) is non-convex and may take the form

\[
f(x, (a_i, b_i)) = \ell(\text{NN}(a_i, x), b_i)
\]
Example: Neural Networks

• Given feature-label pairs \((a_i, b_i)\), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]

• Objective \(f\) is non-convex and may take the form

\[
f(x, (a_i, b_i)) = \ell(\text{NN}(a_i, x), b_i)
\]

• Here, \(\text{NN}(a_i, x)\) is a non-linear function of \(x\), and
Example: Neural Networks

• Given feature-label pairs \( (a_i, b_i) \), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]

• Objective \( f \) is non-convex and may take the form

\[
f(x, (a_i, b_i)) = \ell(\text{NN}(a_i, x), b_i)
\]

• Here, \( \text{NN}(a_i, x) \) is a non-linear function of \( x \), and
  • structure of \( \text{NN()} \) is defined by the neural network
Example: Neural Networks

• Given feature-label pairs \((a_i, b_i)\), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]

• Objective \(f\) is non-convex and may take the form

\[
f(x, (a_i, b_i)) = \ell(\text{NN}(a_i, x), b_i)
\]

• Here, \(\text{NN}(a_i, x)\) is a non-linear function of \(x\), and
  • structure of \(\text{NN}()\) is defined by the neural network
  • elements of \(x\) are weights/parameters of the network
Given feature-label pairs \((a_i, b_i)\), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]

Objective \(f\) is non-convex and may take the form

\[
f(x, (a_i, b_i)) = \ell(\text{NN}(a_i, x), b_i)
\]

Here, \(\text{NN}(a_i, x)\) is a non-linear function of \(x\), and
- structure of \(\text{NN}()\) is defined by the neural network
- elements of \(x\) are weights/parameters of the network

\(\nabla_x \text{NN}(a_i, x)\) can be efficiently calculated via \textit{back-propagation}
Example: Neural Networks

• Given feature-label pairs \((a_i, b_i)\), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]

• Objective \(f\) is non-convex and may take the form

\[
f(x, (a_i, b_i)) = \ell(\text{NN}(a_i, x), b_i)
\]

• Here, \(\text{NN}(a_i, x)\) is a non-linear function of \(x\), and
  • structure of \(\text{NN}()\) is defined by the neural network
  • elements of \(x\) are weights/parameters of the network

• \(\nabla_x \text{NN}(a_i, x)\) can be efficiently calculated via back-propagation

• Deep Learning community focuses on designing \(\text{NN}\)
Example: Neural Networks

• Given feature-label pairs \((a_i, b_i)\), optimization problem is

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, (a_i, b_i))
\]

• Objective \(f\) is non-convex and may take the form

\[
f(x, (a_i, b_i)) = \ell(\text{NN}(a_i, x), b_i)
\]

• Here, \(\text{NN}(a_i, x)\) is a non-linear function of \(x\), and
  • structure of \(\text{NN}()\) is defined by the neural network
  • elements of \(x\) are weights/parameters of the network

• \(\nabla_x \text{NN}(a_i, x)\) can be efficiently calculated via back-propagation

• Deep Learning community focuses on designing \(\text{NN}\)

• Optimization community focuses on solving (GD) for general \(f\)
Example: Recommender Systems
Example: Non-negative Matrix Completion

- Given ratings matrix $M \in \mathbb{R}^{m_1 \times m_2}$ with observed entries $\{M_{ij}\}_{(i,j) \in \Omega}$
Example: Non-negative Matrix Completion

- Given ratings matrix $M \in \mathbb{R}^{m_1 \times m_2}$ with observed entries $\{M_{ij}\}_{(i,j) \in \Omega}$
- Find the complete matrix $X$
Example: Non-negative Matrix Completion

- Given ratings matrix $\mathbf{M} \in \mathbb{R}^{m_1 \times m_2}$ with observed entries $\{M_{ij}\}_{(i,j) \in \Omega}$
- Find the complete matrix $\mathbf{X}$
- If $\mathbf{X}$ is suspected to be low-rank, solve [Recht et al., 2011]

$$\min_{\mathbf{X} \in \mathbb{R}_{+}^{m_1 \times m_2}} \frac{1}{|\Omega|} \sum_{(i,j) \in \Omega} (M_{i,j} - X_{i,j})^2 + \lambda \|\mathbf{X}\|_*$$
Example: Non-negative Matrix Completion

- Given ratings matrix $\mathbf{M} \in \mathbb{R}^{m_1 \times m_2}$ with observed entries $\{M_{i,j}\}_{(i,j) \in \Omega}$
- Find the complete matrix $\mathbf{X}$
- If $\mathbf{X}$ is suspected to be low-rank, solve [Recht et al., 2011]

$$
\min_{\mathbf{X} \in \mathbb{R}_+^{m_1 \times m_2}} \frac{1}{|\Omega|} \sum_{(i,j) \in \Omega} (M_{i,j} - X_{i,j})^2 + \lambda \|\mathbf{X}\|_*
$$

- Here, $\|\mathbf{X}\|_*$ encourages $\mathbf{X}$ to be low-rank
Example: Non-negative Matrix Completion

• Given ratings matrix $M \in \mathbb{R}^{m_1 \times m_2}$ with observed entries $\{M_{ij}\}_{(i,j) \in \Omega}$

• Find the complete matrix $X$

• If $X$ is suspected to be low-rank, solve [Recht et al., 2011]

$$\min_{X \in \mathbb{R}_{+}^{m_1 \times m_2}} \frac{1}{|\Omega|} \sum_{(i,j) \in \Omega} (M_{i,j} - X_{i,j})^2 + \lambda \|X\|_*$$

• Here, $\|X\|_*$ encourages $X$ to be low-rank

• High-dimensional problem: since $d = m_1 m_2 \gg |\Omega| = N$
Outline

1 Context

Problem Formulation: Online and Finite Sum
Examples
State-of-the-art and Oracle Complexity

2 Background

3 Vanilla Stochastic Gradient Descent: Large $N$

4 Variance-Reduced SGD: Moderate $N$

5 High-dimensional problems: large $d$

6 Conclusion
How to compare?

- Which is better: GD or SGD?
How to compare?

• Which is better: GD or SGD?
• Which variant of SGD should I use for a given problem?
Which is better: GD or SGD?
Which variant of SGD should I use for a given problem?
Such questions arise in any field
How to compare?

- Which is better: GD or SGD?
- Which variant of SGD should I use for a given problem?
- Such questions arise in any field
- Sometimes left unanswered, e.g. in, Deep Learning
How to compare?

• Which is better: GD or SGD?
• Which variant of SGD should I use for a given problem?
• Such questions arise in any field
• Sometimes left unanswered, e.g. in, Deep Learning
• But, the landscape of SGD is much more structured
Oracle Complexity

• Given $x$, an oracle provides us $\nabla f(x, \xi_i)$

Call to an oracle costs 1 unit

So an algorithm that makes fewer calls to the oracle is better!

Oracle complexity is the cost required to obtain a desired accuracy

Oracle complexity of SGD: convex objectives

For general convex objective functions, SGD requires $O(Ld \epsilon^2)$ calls to oracle in order to achieve an optimality gap of $\epsilon$.

Terms within $O$ may be initialization dependent

Notation hides away many complexities

Gap measured by $\|x - x^\star\|^2$, $\|\nabla F(x)\|^2$, or $F(x) - F(x^\star)$.
Oracle Complexity

• Given $x$, an oracle provides us $\nabla f(x, \xi_i)$
• Call to an oracle costs 1 unit
• Given $x$, an oracle provides us $\nabla f(x, \xi_i)$
• Call to an oracle costs 1 unit
• So an algorithm that makes fewer calls to the oracle is better!
Oracle Complexity

- Given $x$, an oracle provides us $\nabla f(x, \xi_i)$
- Call to an oracle costs 1 unit
- So an algorithm that makes fewer calls to the oracle is better!
- Oracle complexity is the cost required to obtain a desired accuracy
Oracle Complexity

- Given $x$, an oracle provides us $\nabla f(x, \xi_i)$
- Call to an oracle costs 1 unit
- So an algorithm that makes fewer calls to the oracle is better!
- Oracle complexity is the cost required to obtain a desired accuracy

**Oracle complexity of SGD: convex objectives**

For general convex objective functions, SGD requires $O\left(\frac{Ld}{\epsilon^2}\right)$ calls to oracle in order to achieve an optimality gap of $\epsilon$. 
Oracle Complexity

- Given \( x \), an oracle provides us \( \nabla f(x, \xi_i) \)
- Call to an oracle costs 1 unit
- So an algorithm that makes fewer calls to the oracle is better!
- Oracle complexity is the cost required to obtain a desired accuracy

Oracle complexity of SGD: convex objectives

For general convex objective functions, SGD requires \( \mathcal{O}(\frac{Ld}{\epsilon^2}) \) calls to oracle in order to achieve an optimality gap of \( \epsilon \).

- Terms within \( \mathcal{O} \) may be initialization dependent
- Notation hides away many complexities
Oracle Complexity

• Given $x$, an oracle provides us $\nabla f(x, \xi_i)$
• Call to an oracle costs 1 unit
• So an algorithm that makes fewer calls to the oracle is better!
• Oracle complexity is the cost required to obtain a desired accuracy

Oracle complexity of SGD: convex objectives

For general convex objective functions, SGD requires $O\left(\frac{Ld}{\epsilon^2}\right)$ calls to oracle in order to achieve an optimality gap of $\epsilon$.

• Terms within $O$ may be initialization dependent
• Notation hides away many complexities
• Gap measured by $\|x - x^*\|^2$, $\|\nabla F(x)\|^2$, or $F(x) - F(x^*)$
State-of-the-art in SGD

• New avenues for applying SGD emerge every year
State-of-the-art in SGD

- New avenues for applying SGD emerge every year
- Several variants of SGD are proposed every month
State-of-the-art in SGD

- New avenues for applying SGD emerge every year
- Several variants of SGD are proposed every month
- Papers analyzing performance of these variants come up everyday
State-of-the-art in SGD

- New avenues for applying SGD emerge every year
- Several variants of SGD are proposed every month
- Papers analyzing performance of these variants come up everyday
- Difficult to consolidate and maintain perspective
This Tutorial

- Unified view of many SGD variants
• Unified view of many SGD variants
• Based on recent results, but readily accessible: “easy” math
• Unified view of many SGD variants
• Based on recent results, but readily accessible: “easy” math
• First timers: do not try to understand it all, but do ask questions
• Unified view of many SGD variants
• Based on recent results, but readily accessible: “easy” math
• First timers: do not try to understand it all, but do ask questions
• Up-and-comers: identify gaps and target them, also keep asking questions
This Tutorial

- Unified view of many SGD variants
- Based on recent results, but readily accessible: “easy” math
- **First timers:** do not try to understand it all, but do ask questions
- **Up-and-comers:** identify gaps and target them, also keep asking questions
- **Experts:** what new result am I unaware of?

Later: get slides from my website
• Unified view of many SGD variants
• Based on recent results, but readily accessible: “easy” math
• First timers: do not try to understand it all, but do ask questions
• Up-and-comers: identify gaps and target them, also keep asking questions
• Experts: what new result am I unaware of?
• Later: get slides from my website
• Key reference text: [Beck, 2017]
• Introductory (deterministic): [Vandenberghe, 2019]
• [Bubeck et al., 2015] is good introduction to the topic
• Related course lecture notes: [Saunders, 2019, Chen, 2019]
• Sebastien Bubeck’s blog: [Bubeck, 2019]
• This tutorial is an amalgamation of [Gorbunov et al., 2019], [Bottou et al., 2018], and [Recht et al., 2011]
• Inspired from the tutorial: https://www.youtube.com/watch?v=a05S0kL5u30
Outline

1. Context

2. Background

   Convexity
   
   Smoothness
   
   Subgradients, projection, and proximal operators

3. Vanilla Stochastic Gradient Descent: Large $N$

4. Variance-Reduced SGD: Moderate $N$

5. High-dimensional problems: large $d$

6. Conclusion
Convex Functions: Zeroth Order Condition

**Definition**

A function $f$ is convex if (a) its domain is a convex set; and (b) it satisfies

$$f(\theta x + (1 - \theta) y) \leq \theta f(x) + (1 - \theta) f(y)$$
Definition
A function $f$ is convex if (a) its domain is a convex set; and (b) it satisfies

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$$

Alternatively: eigenvalues of $(\nabla^2 F(x)) \geq 0$
Strongly Convex Functions
Definition
A function $F$ is $\mu$-strongly convex if (a) its domain is a convex set; and (b) it satisfies

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \| y - x \|^2$$

where $\mu > 0$. Alternatively, eigenvalues of $(\nabla^2 F(x)) \geq \mu$
Strongly Convex Functions: Quadratic Lower Bound

**Definition**
A function $F$ is $\mu$-strongly convex if (a) its domain is a convex set; and (b) it satisfies

$$
f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2
$$

where $\mu > 0$. Alternatively, eigenvalues of $(\nabla^2 F(x)) \geq \mu$

**$\ell_2$-norm square example**
The function $f(x) = \frac{1}{2} \|x\|^2$ is 1-strongly convex
Strongly Convex Functions: Quadratic Lower Bound

Definition
A function $F$ is $\mu$-strongly convex if (a) its domain is a convex set; and (b) it satisfies

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2$$

where $\mu > 0$. Alternatively, eigenvalues of $(\nabla^2 F(x)) \geq \mu$

$\ell_2$-norm square example
The function $f(x) = \frac{1}{2} \|x\|^2$ is 1-strongly convex

Least-squares example
Is the lasso regression objective strongly convex? Recall

$$R(x) = \frac{1}{N} \sum_{i=1}^{N} (a_i^\top x - b_i)^2 + \lambda \|x\|_1.$$
Strongly Convex Functions: Quadratic Lower Bound

Definition
A function $F$ is $\mu$-strongly convex if (a) its domain is a convex set; and (b) it satisfies

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2$$

where $\mu > 0$. Alternatively, eigenvalues of $(\nabla^2 F(x)) \geq \mu$.

$\ell_2$-norm square example
The function $f(x) = \frac{1}{2} \|x\|^2$ is 1-strongly convex.

Least-squares example
Is the lasso regression objective strongly convex? Recall

$$R(x) = \frac{1}{N} \sum_{i=1}^{N} (a_i^\top x - b_i)^2 + \lambda \|x\|_1.$$ 

Show that for this case $\mu = \text{smallest eigenvalue of } \frac{1}{N} \sum_{i=1}^{N} a_i a_i^\top$. 


1 Context

2 Background
   Convexity
   Smoothness
      Subgradients, projection, and proximal operators

3 Vanilla Stochastic Gradient Descent: Large $N$

4 Variance-Reduced SGD: Moderate $N$

5 High-dimensional problems: large $d$

6 Conclusion
Smooth Functions

Smooth Functions

Smooth

Smooth
Smooth Functions: Quadratic Upper Bound

**Definition**
A function $F$ is $L$-smooth

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|x - y\|^2$$

Alternatively: eigenvalues of $(\nabla^2 F(x)) \leq L$
Bregman divergence over a function $F$ is defined as

$$D_F(x, y) = F(y) - F(x) - \langle \nabla F(x), y - x \rangle$$
Bregman Divergence

• Bregman divergence over a function $F$ is defined as

$$D_F(x, y) = F(y) - F(x) - \langle \nabla F(x), y - x \rangle$$

• Bregman divergence is not symmetric (and not a metric) but satisfies

$$\frac{\mu}{2} \|x - y\|^2 \leq D_F(x, y) \leq \frac{L}{2} \|x - y\|^2$$
Bregman Divergence

• Bregman divergence over a function $F$ is defined as

$$D_F(x, y) = F(y) - F(x) - \langle \nabla F(x), y - x \rangle$$

• Bregman divergence is not symmetric (and not a metric) but satisfies

$$\frac{\mu}{2} \| x - y \|^2 \leq D_F(x, y) \leq \frac{L}{2} \| x - y \|^2$$

$$\frac{1}{2L} \| \nabla F(x) - \nabla F(y) \|^2 \leq D_F(x, y) \leq \frac{1}{2\mu} \| \nabla F(x) - \nabla F(y) \|^2$$
1. Context

2. Background
   - Convexity
   - Smoothness
   - Subgradients, projection, and proximal operators

3. Vanilla Stochastic Gradient Descent: Large $N$

4. Variance-Reduced SGD: Moderate $N$

5. High-dimensional problems: large $d$

6. Conclusion
Non-smooth convex functions

• If \( h \) is non-smooth convex, may still define subgradient \( v(x) \in \partial h(x) \)

f(y) \geq f(x) + \langle v(x), y-x \rangle

Optimality condition for \( x^\star = \arg \min_x f(x) \): 

\( v(x^\star) = 0 \in \partial h(x^\star) \)
If $h$ is non-smooth convex, may still define subgradient $v(x) \in \partial h(x)$

Satisfies first order convexity condition as usual

$$f(y) \geq f(x) + \langle v(x), y - x \rangle$$
Non-smooth convex functions

• If $h$ is non-smooth convex, may still define subgradient $v(x) \in \partial h(x)$

• Satisfies first order convexity condition as usual

$$f(y) \geq f(x) + \langle v(x), y - x \rangle$$

• Optimality condition for $x^* = \arg\min_x f(x)$:

$$v(x^*) = 0 \in \partial h(x^*)$$
• Define the projection over a set $\mathcal{X}$ as

$$
\mathcal{P}_\mathcal{X}(x) = \arg \min_{y \in \mathcal{X}} \frac{1}{2} \|y - x\|^2
$$

Projection Operator

• Define the projection over a set $\mathcal{X}$ as

$$\mathcal{P}_\mathcal{X}(x) = \arg\min_{y \in \mathcal{X}} \frac{1}{2} \|y - x\|^2$$

• Equivalent formulation

$$\mathcal{P}_\mathcal{X}(x) = \arg\min_y \frac{1}{2} \|y - x\|^2 + 1_{\mathcal{X}}(x)$$

where the indicator function is defined as

$$1_{\mathcal{X}}(x) = \begin{cases} 0 & x \in \mathcal{X} \\ \infty & x \notin \mathcal{X} \end{cases}$$
• Proximal operator generalizes projection

\[ \text{prox}_h(x) = y^* = \arg \min_y \frac{1}{2} \|y - x\|^2 + h(x) \]
• Proximal operator generalizes projection

\[ \text{prox}_h(x) = y^* = \arg \min_y \frac{1}{2} \|y - x\|^2 + h(x) \]

• Useful property: differentiate and equate to zero

\[ y^* - x + v(y^*) = 0 \]

where \( y^* = \text{prox}_h(x) \) and \( v(y^*) \in \partial h(y^*) \)
Vanilla Stochastic Gradient Descent: Large $N$
Outline

1. Context
2. Background
3. Vanilla Stochastic Gradient Descent: Large $N$
   - Gradient Descent vs. Stochastic Gradient Descent
   - Performance of Stochastic Gradient Descent
4. Variance-Reduced SGD: Moderate $N$
5. High-dimensional problems: large $d$
6. Conclusion
Gradient Descent vs. Stochastic Gradient Descent

- Gradient descent for solving \( \mathcal{P} \)

\[
x_{t+1} = \mathcal{P} \chi \left( x_t - \frac{\eta}{N} \sum_{i=1}^{N} \nabla f(x_t, \xi_i) \right)
\]

- \( N \) oracle calls per iteration
Gradient Descent vs. Stochastic Gradient Descent

• Gradient descent for solving \((P)\)

\[
x_{t+1} = P \chi \left( x_t - \frac{\eta}{N} \sum_{i=1}^{N} \nabla f(x_t, \xi_i) \right)
\]

• \(N\) oracle calls per iteration

• Stochastic gradient descent for solving \((P)\)

\[
x_{t+1} = P \chi (x_t - \eta \nabla f(x_t, \xi_{it}))
\]

where \(i_t \in \{1, \ldots, N\}\) is a random number.
Gradient Descent vs. Stochastic Gradient Descent

- Gradient descent for solving \((\mathcal{P})\)

\[
x_{t+1} = \mathcal{P}_\chi \left( x_t - \frac{\eta}{N} \sum_{i=1}^{N} \nabla f(x_t, \xi_i) \right)
\]

- \(N\) oracle calls per iteration

- Stochastic gradient descent for solving \((\mathcal{P})\)

\[
x_{t+1} = \mathcal{P}_\chi (x_t - \eta \nabla f(x_t, \xi_{i_t}))
\]

where \(i_t \in \{1, \ldots, N\}\) is a random number.

- Descent direction on average: expectation w.r.t. \(i_t\)

\[
\mathbb{E}_{i_t} [\nabla f(x_t, \xi_{i_t})] = \frac{1}{N} \sum_{i=1}^{N} f(x_t, \xi_i) = \nabla F(x_t)
\]
Intuition

- SGD more efficient at accessing data
Intuition

• SGD more efficient at accessing data
• handles redundancy in dataset better
Intuition

- SGD more efficient at accessing data
- handles redundancy in dataset better
- consider lasso example: features $a_i \in \text{span}(a^{(1)}, a^{(2)}, a^{(3)})$
- Given \((X, Y)\) observations, let \(\Phi(X)\) be a transformation
- SGD has been applied to specific problems

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Loss</th>
<th>Gradient/Subgradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS (Widrow-Hoff’60)</td>
<td>(\frac{1}{2}(Y - \Phi(X)^T x)^2)</td>
<td>((\Phi(X)^T x - Y)\Phi(X))</td>
</tr>
</tbody>
</table>
History of SGD

• Given \((X, Y)\) observations, let \(\Phi(X)\) be a transformation

• SGD has been applied to specific problems

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Loss</th>
<th>Gradient/Subgradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS (Widrow-Hoff’60)</td>
<td>(\frac{1}{2}(Y - \Phi(X)^\top x)^2)</td>
<td>((\Phi(X)^\top x - Y)\Phi(X))</td>
</tr>
<tr>
<td>Perceptron (Rosenblatt'57)</td>
<td>([-Y\langle\Phi(X), x\rangle]_+)</td>
<td>(-Y\Phi(X)1_{Y\langle\Phi(X), x\rangle \leq 0})</td>
</tr>
</tbody>
</table>
History of SGD

- Given \((X, Y)\) observations, let \(\Phi(X)\) be a transformation
- SGD has been applied to specific problems

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Loss</th>
<th>Gradient/Subgradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS (Widrow-Hoff'60)</td>
<td>(\frac{1}{2}(Y - \Phi(X)^\top x)^2)</td>
<td>((\Phi(X)^\top x - Y)\Phi(X))</td>
</tr>
<tr>
<td>Perceptron (Rosenblatt'57)</td>
<td>([-Y\langle\Phi(X), x\rangle]_+)</td>
<td>(-Y\Phi(X)1_{Y\langle\Phi(X), x\rangle \leq 0})</td>
</tr>
<tr>
<td>SVM (Cortes-Vapnik'95)</td>
<td>(\frac{\lambda}{2}|x|^2 + [1 - Y\langle\Phi(X), x\rangle]_+)</td>
<td>(\lambda x - Y\Phi(X)1_{Y\langle\Phi(X), x\rangle \leq 1})</td>
</tr>
</tbody>
</table>
Outline

1. Context
2. Background
3. Vanilla Stochastic Gradient Descent: Large \( N \)
   - Gradient Descent vs. Stochastic Gradient Descent
   - Performance of Stochastic Gradient Descent
4. Variance-Reduced SGD: Moderate \( N \)
5. High-dimensional problems: large \( d \)
6. Conclusion
Assumptions

$L$-smoothness

\[ D_F(x, y) \leq \frac{L}{2} \|x - y\|^2 \]
Assumptions

**L-smoothness**

\[ D_F(x, y) \leq \frac{L}{2} \|x - y\|^2 \]

**μ-convexity**

\[ D_F(x, y) \geq \frac{\mu}{2} \|x - y\|^2 \]
Assumptions

$L$-smoothness

\[ D_F(x, y) \leq \frac{L}{2} \| x - y \|^2 \]

$\mu$-convexity

\[ D_F(x, y) \geq \frac{\mu}{2} \| x - y \|^2 \]

Bounded Variance

\[ \mathbb{E}_{i_t} \left[ \| \nabla f(x, \xi_{i_t}) \|^2 \right] \leq \sigma^2 + c \| \nabla F(x) \|^2 \]

\[ \Rightarrow \mathbb{E}_{i_t} \left[ \| \nabla f(x^*, \xi_{i_t}) \|^2 \right] \leq \sigma^2 \]

provided $\nabla F(x^*) = 0$ and $c \geq 1$.

$\sigma^2$ is the inherent data variance
Strong Convexity and Smoothness: Condition Number

\[(\text{small } \kappa = L/\mu)\]  
\[(\text{large } \kappa = L/\mu)\]
Lemma (SGD: Strongly Convex + Smooth [Bottou et al., 2018])

For $L$-smooth, $\mu$-convex functions, SGD incurs oracle complexity of $O\left(\frac{L}{\mu \epsilon}\right)$. 
Lemma (SGD: Strongly Convex + Smooth [Bottou et al., 2018])

For $L$-smooth, $\mu$-convex functions, SGD incurs oracle complexity of $\mathcal{O}\left(\frac{L}{\mu \epsilon}\right)$.

For simplicity, consider unconstrained version: $x_{t+1} - x_t = \eta \nabla f(x_t, \xi_t)$

Proof: Step 1. Quadratic upper bound ($L$-smoothness):

\[
F(x_{t+1}) \leq F(x_t) + \langle \nabla F(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \|x_{t+1} - x_t\|^2
\]
Oracle Complexity for SGD: Strongly Convex + Smooth

Lemma (SGD: Strongly Convex + Smooth [Bottou et al., 2018])

For $L$-smooth, $\mu$-convex functions, SGD incurs oracle complexity of $O\left(\frac{L}{\mu \epsilon}\right)$.

For simplicity, consider unconstrained version: $x_{t+1} - x_t = \eta \nabla f(x_t, \xi_{it})$

Proof: Step 1. Quadratic upper bound ($L$-smoothness):

$$F(x_{t+1}) \leq F(x_t) + \langle \nabla F(x_t), x_{t+1} - x_t \rangle + \frac{L}{2} \| x_{t+1} - x_t \|^2$$

$$= F(x_t) - \eta \langle \nabla F(x_t), \nabla f(x_t, \xi_{it}) \rangle + \frac{\eta^2 L}{2} \| \nabla f(x_t, \xi_{it}) \|^2$$

Update Equation

$$x_{t+1} - x_t = \eta \nabla f(x_t, \xi_{it})$$
SGD: Strongly Convex + Smooth

Step 2. Take expectation

$$\mathbb{E}_{i_t}[F(x_{t+1})] \leq F(x_t) - \eta \langle \nabla F(x_t), \mathbb{E}_{i_t} [\nabla f(x_t, \xi_{i_t})] \rangle + \frac{\eta^2 L}{2} \mathbb{E}_{i_t} \left[ \| \nabla f(x_t, \xi_{i_t}) \|^2 \right]$$
Step 2. Take expectation, use $\mathbb{E}_{i_t} [\nabla f(x_t, \xi_{i_t})] = \nabla F(x_t)$ 

$$
\mathbb{E}_{i_t} [F(x_{t+1})] \leq F(x_t) - \eta \langle \nabla F(x_t), \mathbb{E}_{i_t}[\nabla f(x_t, \xi_{i_t})] \rangle + \frac{\eta^2 L}{2} \mathbb{E}_{i_t} \left[ \| \nabla f(x_t, \xi_{i_t}) \|^2 \right]
$$

$$
= F(x_t) - \eta \langle \nabla F(x_t), \nabla F(x_t) \rangle + \frac{\eta^2 L}{2} \mathbb{E}_{i_t} \left[ \| \nabla f(x_t, \xi_{i_t}) \|^2 \right]
$$
SGD: Strongly Convex + Smooth

Step 2. Take expectation, use $E_{i_t}[\nabla f(x_t, \xi_{i_t})] = \nabla F(x_t)$

$$E_{i_t}[F(x_{t+1})] \leq F(x_t) - \eta \langle \nabla F(x_t), E_{i_t}[\nabla f(x_t, \xi_{i_t})] \rangle + \frac{\eta^2 L}{2} E_{i_t} \left[ \| \nabla f(x_t, \xi_{i_t}) \|^2 \right]$$

$$= F(x_t) - \eta \langle \nabla F(x_t), \nabla F(x_t) \rangle + \frac{\eta^2 L}{2} E_{i_t} \left[ \| \nabla f(x_t, \xi_{i_t}) \|^2 \right]$$

$$\leq F(x_t) - \eta \left( 1 - \frac{\eta L c}{2} \right) \| \nabla F(x_t) \|^2 + \frac{\eta^2 \sigma^2 L}{2}$$

$$E_{i_t} \left[ \| \nabla f(x, \xi_{i_t}) \|^2 \right] \leq \sigma^2 + c \| \nabla F(x) \|^2$$
SGD: Strongly Convex + Smooth

Step 2. Take expectation, use $\mathbb{E}_{i_t} [\nabla f(x_t, \xi_{i_t})] = \nabla F(x_t)$

$$
\mathbb{E}_{i_t} [F(x_{t+1})] \leq F(x_t) - \eta \langle \nabla F(x_t), \mathbb{E}_{i_t} [\nabla f(x_t, \xi_{i_t})] \rangle + \frac{\eta^2 L}{2} \mathbb{E}_{i_t} \left[ \| \nabla f(x_t, \xi_{i_t}) \|^2 \right]
$$

$$
= F(x_t) - \eta \langle \nabla F(x_t), \nabla F(x_t) \rangle + \frac{\eta^2 L}{2} \mathbb{E}_{i_t} \left[ \| \nabla f(x_t, \xi_{i_t}) \|^2 \right]
$$

$$
\leq F(x_t) - \eta \left( 1 - \frac{\eta L c}{2} \right) \| \nabla F(x_t) \|^2 + \frac{\eta^2 \sigma^2 L}{2}
$$

$$
\leq F(x_t) - \frac{\eta}{2} \| \nabla F(x_t) \|^2 + \frac{\eta^2 \sigma^2 L}{2}
$$

$\eta L c < 1$
Step 2. Take expectation, use $E_{it} [\nabla f(x_t, \xi_{it})] = \nabla F(x_t)$

$$E_{it} [F(x_{t+1})] \leq F(x_t) - \eta \langle \nabla F(x_t), E_{it} [\nabla f(x_t, \xi_{it})] \rangle + \frac{\eta^2 L}{2} E_{it} \left[ \| \nabla f(x_t, \xi_{it}) \|^2 \right]$$

$$= F(x_t) - \eta \langle \nabla F(x_t), \nabla F(x_t) \rangle + \frac{\eta^2 L}{2} E_{it} \left[ \| \nabla f(x_t, \xi_{it}) \|^2 \right]$$

$$\leq F(x_t) - \eta \left( 1 - \frac{\eta L c}{2} \right) \| \nabla F(x_t) \|^2 + \frac{\eta^2 \sigma^2 L}{2}$$

$$\leq F(x_t) - \frac{\eta}{2} \| \nabla F(x_t) \|^2 + \frac{\eta^2 \sigma^2 L}{2}$$

**Function decrement in SGD**

Function value decreases (on average) only when the gradient is large!
Step 3. Relate $\|\nabla F(x_t)\|^2$ with optimality gap: subtract $F(x^*)$, and use strong convexity

$$\mathbb{E}_{i_t} [F(x_{t+1})] - F(x^*) \leq F(x_t) - F(x^*) - \frac{\eta}{2} \|\nabla F(x_t)\|^2 + \frac{\eta^2 \sigma^2 L}{2}$$
Step 3. Relate $\|\nabla F(x_t)\|^2$ with optimality gap: subtract $F(x^*)$, and use strong convexity

$$
\mathbb{E}_{i_t} [F(x_{t+1})] - F(x^*) \leq F(x_t) - F(x^*) - \frac{\eta}{2} \|\nabla F(x_t)\|^2 + \frac{\eta^2 \sigma^2 L}{2} \\
\leq (1 - \mu \eta)(F(x_t) - F(x^*)) + \frac{\eta^2 \sigma^2 L}{2}
$$

$$
\frac{1}{2} \|\nabla F(x_t)\|^2 \geq \mu (F(x_t) - F(x^*))
$$
SGD: Strongly Convex + Smooth

Step 3. Relate $\|\nabla F(x_t)\|^2$ with optimality gap: subtract $F(x^*)$, and use strong convexity

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

$$\leq (1-\mu\eta)(F(x_t) - F(x^*)) + \frac{\eta^2 \sigma^2 L}{2}$$

Set $\Delta_t = \mathbb{E}[F(x_{t+1}) - F(x^*)]$
Step 3. Relate $\|\nabla F(x_t)\|^2$ with optimality gap: subtract $F(x^*)$, and use strong convexity.

\[
\mathbb{E}_{i_t} [F(x_{t+1})] - F(x^*) \leq F(x_t) - F(x^*) - \frac{\eta}{2} \|\nabla F(x_t)\|^2 + \frac{\eta^2 \sigma^2 L}{2} \\
\leq (1 - \mu \eta)(F(x_t) - F(x^*)) + \frac{\eta^2 \sigma^2 L}{2}
\]

Set $\Delta_t = \mathbb{E}[F(x_{t+1}) - F(x^*)]$.

One-step inequality

\[
\Delta_{t+1} \leq (1 - \mu \eta) \Delta_t + \frac{\eta^2 \sigma^2 L}{2}
\]
One-step inequality

\[ \Delta_{t+1} \leq (1 - \mu \eta) \Delta_t + \frac{\eta^2 \sigma^2 L}{2} \]

Step 4. Obtain final inequality:
One-step inequality

\[ \Delta_{t+1} \leq (1 - \mu \eta) \Delta_t + \frac{\eta^2 \sigma^2 L}{2} \]

Step 4. Obtain final inequality:
Apply recursively over \( t = 1, \ldots, T \):

\[ \Delta_{T+1} \leq (1 - \mu \eta)^T \Delta_1 + \frac{\eta^2 \sigma^2 L}{2} \frac{1}{\mu \eta} \]
Final inequality

$$\Delta_{T+1} \leq (1 - \mu \eta)^T \Delta_1 + \frac{\eta \sigma^2 L}{2\mu}$$

Step 5. Pick $\eta$: 

- Equate each term to $\epsilon$ to obtain $\eta = O\left(\frac{\sigma^2 L}{\mu \epsilon}\right)$ (ignore unimportant constants).
- Solve for $T$: $(1 - \mu \eta)^T = \epsilon$ and use $\log(1 - \mu \eta) \approx -\mu \eta$ to obtain $T = O\left(\frac{\sigma^2 L}{\mu \epsilon \log\left(\frac{1}{\epsilon}\right)}\right) \approx O\left(\frac{\sigma^2 L}{\mu \epsilon}\right)$. 

Final inequality

\[ \Delta_{T+1} \leq (1 - \mu \eta)^T \Delta_1 + \frac{\eta \sigma^2 L}{2\mu} \]

Step 5. Pick \( \eta \):

- Equate each term to \( \epsilon \Rightarrow \eta = O\left(\frac{\mu \epsilon}{\sigma^2 L}\right) \) (ignore unimportant constants)
Final inequality

\[ \Delta_{T+1} \leq (1 - \mu \eta)^T \Delta_1 + \frac{\eta \sigma^2 L}{2\mu} \]

**Step 5.** Pick \( \eta \):

- Equate each term to \( \epsilon \Rightarrow \eta = \mathcal{O}\left(\frac{\mu \epsilon}{\sigma^2 L}\right) \) (ignore unimportant constants).
- Solve for \( T \): \((1 - \mu \eta)^T = \epsilon\) and use \( \log(1 - \mu \eta) \approx -\mu \eta \) to obtain

\[ T = \mathcal{O}\left(\frac{\sigma^2 L}{\mu \epsilon \log \left(\frac{1}{\epsilon}\right)}\right) \approx \mathcal{O}\left(\frac{\sigma^2 L}{\mu \epsilon}\right) \]
Practical Considerations

- With fixed $\eta$, SGD converges fast, but slows when optimality gap is $O(\eta)$

- Can select a diminishing step-size to obtain slight improvement

- Other approach: half the step-size when progress stalls [Bottou et al., 2018]
Practical Considerations

• With fixed $\eta$, SGD converges fast, but slows when optimality gap is $\mathcal{O}(\eta)$
• Can select a diminishing step-size to obtain slight improvement
With fixed $\eta$, SGD converges fast, but slows when optimality gap is $O(\eta)$

Can select a diminishing step-size to obtain slight improvement

Other approach: half the step-size when progress stalls [Bottou et al., 2018]
Lemma (SGD: smooth)

For $L$-smooth functions, SGD incurs oracle complexity of $O\left(\frac{L}{\epsilon^2}\right)$. 
Lemma (SGD: smooth)

For $L$-smooth functions, SGD incurs oracle complexity of $O\left(\frac{L}{\epsilon^2}\right)$.

**Proof for unconstrained version:** $\mathbf{x}_{t+1} - \mathbf{x}_t = \eta \nabla f(\mathbf{x}_t, \xi_{it})$.

Recall from $L$-smoothness and $\eta L c < 1$ (here: $\Delta_t = \mathbb{E}[F(\mathbf{x}_t)] - F(\mathbf{x}^*) \geq 0$):

$$
\Delta_{t+1} \leq \Delta_t - \frac{\eta}{2} \left\| \nabla F(\mathbf{x}_t) \right\|^2 + \frac{\eta^2 \sigma^2 L}{2}
$$

$$
\leq \Delta_1 - \frac{\eta}{2} \sum_{t=1}^{T} \left\| \nabla F(\mathbf{x}_t) \right\|^2 + \frac{T \eta^2 \sigma^2 L}{2}
$$
• Rearrange to obtain:

\[
\min_{1 \leq t \leq T} \mathbb{E}[\|\nabla F(x_t)\|_2^2] \leq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\|\nabla F(x_t)\|_2^2] \leq \eta \sigma^2 L + \frac{2\Delta_1}{\eta T}
\]
• Rearrange to obtain:
\[
\min_{1 \leq t \leq T} \mathbb{E}[\|\nabla F(x_t)\|_2^2] \leq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\|\nabla F(x_t)\|_2^2] \leq \eta \sigma^2 L + \frac{2\Delta_1}{\eta T}
\]

• Equate each term to $\epsilon$ to obtain $\eta = \frac{\epsilon}{\sigma^2 L}$ and
\[
T = \mathcal{O}\left(\frac{\sigma^2 L}{\epsilon^2}\right)
\]

oracle calls required to reach close to a first order stationary point
Variance-Reduced SGD: Moderate $N$
Gradient Descent or Stochastic Gradient Descent?

- Standard gradient descent requires $O\left(L\mu \log\left(\frac{1}{\epsilon}\right)\right)$ iterations.
- But each iteration requires $N$ oracle calls: so oracle complexity is $O\left(LN\mu \log\left(\frac{1}{\epsilon}\right)\right)$.
- In contrast, SGD requires $O\left(L\mu \epsilon\right)$ oracle calls: independent of $N$.

**Figure 1**: Gradient Descent

**Figure 2**: Stochastic Gradient Descent
Gradient Descent or Stochastic Gradient Descent?

• Standard gradient descent requires $O \left( \frac{L}{\mu} \log \left( \frac{1}{\epsilon} \right) \right)$ iterations
Gradient Descent or Stochastic Gradient Descent?

Figure 1: Gradient Descent

• Standard gradient descent requires \( \mathcal{O} \left( \frac{L}{\mu} \log \left( \frac{1}{\epsilon} \right) \right) \) iterations

• But each iteration requires \( N \) oracle calls: so oracle complexity is \( \mathcal{O} \left( \frac{LN}{\mu} \log \left( \frac{1}{\epsilon} \right) \right) \)

Figure 2: Stochastic Gradient Descent
Gradient Descent or Stochastic Gradient Descent?

Figure 1: Gradient Descent

Figure 2: Stochastic Gradient Descent

• Standard gradient descent requires $O \left( \frac{L}{\mu} \log \left( \frac{1}{\epsilon} \right) \right)$ iterations
• But each iteration requires $N$ oracle calls: so oracle complexity is $O \left( \frac{LN}{\mu} \log \left( \frac{1}{\epsilon} \right) \right)$
• In contrast, SGD requires $O \left( \frac{L}{\mu \epsilon} \right)$ oracle calls: independent of $N$
Speeding up SGD?

\[ \log(\text{excess loss}) \]

\# oracle calls

SGD

GD
Speeding up SGD?

\[
\log(\text{excess loss}) \quad \# \text{oracle calls}
\]

- **GD**
- **SGD**

???
• We consider the generic SGD algorithm:

\[ x_{t+1} = x_t - \eta g_t \]

where \( g_t \) is an unbiased gradient approximation
Variance Reduction

• We consider the generic SGD algorithm:

\[ x_{t+1} = x_t - \eta g_t \]

where \( g_t \) is an unbiased gradient approximation

• Example:

\[ g_t = \frac{1}{N} \sum_{i=1}^{N} \nabla f(x_t, \xi_i) \] \hspace{1cm} (GD)

\[ g_t = \nabla f(x_t, \xi_{it}) \] \hspace{1cm} (SGD)

\[ g_t = \frac{1}{|B|} \sum_{i \in B} \nabla f(x_t, \xi_i) \] \hspace{1cm} (mini-batch)
• We consider the generic SGD algorithm:

\[ x_{t+1} = x_t - \eta g_t \]

where \( g_t \) is an unbiased gradient approximation

• Example:

\[ g_t = \frac{1}{N} \sum_{i=1}^{N} \nabla f(x_t, \xi_i) \quad \text{(GD)} \]

\[ g_t = \nabla f(x_t, \xi_{it}) \quad \text{(SGD)} \]

\[ g_t = \frac{1}{|B|} \sum_{i \in B} \nabla f(x_t, \xi_i) \quad \text{(mini-batch)} \]
Effect of Mini Batching

- Consider $b$ random variables $\{X_i\}_{i=1}^b$ such that $\nabla_i(X_i) = \sigma^2$
Effect of Mini Batching

-Consider $b$ random variables $\{X_i\}_{i=1}^b$ such that $\nabla_i(X_i) = \sigma^2$

-Then it holds that $\nabla_i (\frac{1}{b} \sum_i X_i) = \frac{\sigma^2}{b}$
Effect of Mini Batching

- Consider $b$ random variables $\{X_i\}_{i=1}^{b}$ such that $\nabla_i(X_i) = \sigma^2$
- Then it holds that $\nabla_i(\frac{1}{b} \sum_i X_i) = \frac{\sigma^2}{b}$
- So

\[
\text{# of iterations} = \mathcal{O}\left(\frac{L}{\mu b} \log \left(\frac{1}{\epsilon}\right)\right)
\]
Effect of Mini Batching

- Consider $b$ random variables $\{X_i\}_{i=1}^b$ such that $\nabla_i X_i = \sigma^2$
- Then it holds that $\nabla_i (\frac{1}{b} \sum_i X_i) = \frac{\sigma^2}{b}$
- So

$$\text{# of iterations} = \mathcal{O}\left(\frac{L \mu b}{\log \left(\frac{1}{\epsilon}\right)}\right)$$

- But each iteration requires $b$ oracle calls: oracle complexity still same
Effect of Mini Batching

• Consider $b$ random variables $\{X_i\}_{i=1}^b$ such that $\mathbb{V}_i(X_i) = \sigma^2$

• Then it holds that $\mathbb{V}_i\left(\frac{1}{b} \sum_i X_i\right) = \frac{\sigma^2}{b}$

• So

\[
\#	ext{ of iterations} = \mathcal{O}\left(\frac{L}{\mu b} \log \left(\frac{1}{\epsilon}\right)\right)
\]

• But each iteration requires $b$ oracle calls: oracle complexity still same

• In practice: lesser wall-clock time if gradients can be calculated in parallel
• Consider the loss functions

\[ \phi(x, \xi_i) = f(x, \xi_i) - a_i^T x \]

so that the overall objective remains the same, i.e.,

\[ \Phi(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i) - a_i^T x = F(x) \]

provided that \( \sum_i a_i = 0 \).
Intuition: Shifted SGD

• Consider the loss functions

\[ \phi(x, \xi_i) = f(x, \xi_i) - a_i^\top x \]

so that the overall objective remains the same, i.e.,

\[ \Phi(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i) - a_i^\top x = F(x) \]

provided that \( \sum_i a_i = 0 \).

• Note that \( \nabla \phi(x, \xi_i) = \nabla f(x, \xi_i) - a_i \)
Intuition: Shifted SGD

• Consider the loss functions

\[ \phi(x, \xi_i) = f(x, \xi_i) - a_i^T x \]

so that the overall objective remains the same, i.e.,

\[ \Phi(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i) - a_i^T x = F(x) \]

provided that \( \sum_i a_i = 0 \).

• Note that \( \nabla \phi(x, \xi_i) = \nabla f(x, \xi_i) - a_i \)

• Recall that SGD performance depends on variance at \( x^* \)

\[ \nabla_{i_t} [\| \nabla f(x^*, \xi_{i_t}) \|] \leq \sigma^2 \]
Intuition: Shifted SGD

**Shifted gradient**

\[ \nabla \phi(x, \xi_i) = \nabla f(x, \xi_i) - a_i \]

- Goal: select \( a_i \) so that \( \nabla_{it} [\nabla \phi(x^*, \xi_{it})] \) is small
Intuition: Shifted SGD

**Shifted gradient**

\[ \nabla \phi(x, \xi_i) = \nabla f(x, \xi_i) - a_i \]

- Goal: select \( a_i \) so that \( \nabla_{i_t} [\nabla \phi(x^*, \xi_{i_t})] \) is small
- Hypothetically, \( \nabla_{i_t} [\nabla \phi(x^*, \xi_{i_t})] = 0 \) requires

\[ a_i = \nabla f(x^*, \xi_i) \]
Intuition: Shifted SGD

**Shifted gradient**

\[ \nabla \phi(x, \xi_i) = \nabla f(x, \xi_i) - a_i \]

- Goal: select \( a_i \) so that \( \nabla_{i_t} \left[ \nabla \phi(x^*, \xi_{i_t}) \right] \) is small
- Hypothetically, \( \nabla_{i_t} \left[ \nabla \phi(x^*, \xi_{i_t}) \right] = 0 \) requires
  \[ a_i = \nabla f(x^*, \xi_i) \]
- Not practical as \( x^* \) unknown
Intuition: Shifted SGD

**Shifted gradient**

\[ \nabla \phi(x, \xi_i) = \nabla f(x, \xi_i) - a_i \]

- **Goal:** select \( a_i \) so that \( \nabla_{i_t} [\nabla \phi(x^*, \xi_{i_t})] \) is small
- **Hypothetically,** \( \nabla_{i_t} [\nabla \phi(x^*, \xi_{i_t})] = 0 \) requires
  \[ a_i = \nabla f(x^*, \xi_i) \]
  - **Not practical** as \( x^* \) unknown
- **Clue:** availability of estimates of \( \nabla f(x^*, \xi_i) \) can help!
A unified approach to approximating gradients [Gorbunov et al., 2019]

Suppose the unbiased gradient approximation $g_t$ satisfies:

$$E_t[\|g_t\|^2] \leq 2AD_F(x_t, x^\star) + B\sigma^2_t$$

$$E_t[\sigma^2_t + 1] \leq (1 - \rho)\sigma^2_t + 2CD_F(x_t, x^\star)$$

where $A, B, C, \sigma^2_t,$ and $\rho > 0$ are some constants (depend on $L, \mu, N$) and $E_t[\cdot]$ is expectation with respect to the random data index at iteration $t$.

Lemma (Simplified version of [Gorbunov et al., 2019])

The following rate result holds:

$$E[\|x_T - x^\star\|^2] \leq (1 - \rho^2\min\{2\mu A\rho + 2BC, 1\})TB_0$$

where $B_0$ depends only on the initialization.
Unified Theory of Gradient Approximation

- A unified approach to approximating gradients [Gorbunov et al., 2019]
- Suppose the unbiased gradient approximation $g_t$ satisfies:

$$
\mathbb{E}_t[\|g_t\|^2] \leq 2AD_F(x_t, x^*) + B\sigma_t^2
$$

$$
\mathbb{E}_t[\sigma_{t+1}^2] \leq (1 - \rho)\sigma_t^2 + 2CD_F(x_t, x^*)
$$

where $A$, $B$, $C$, $\sigma_t^2$, and $\rho > 0$ are some constants (depend on $L$, $\mu$, $N$) and $\mathbb{E}_t[\cdot]$ is expectation with respect to the random data index at iteration $t$
A unified approach to approximating gradients [Gorbunov et al., 2019]

Suppose the unbiased gradient approximation $g_t$ satisfies:

$$
\mathbb{E}_t[\|g_t\|^2] \leq 2AD_F(x_t, x^*) + B\sigma_t^2 \\
\mathbb{E}_t[\sigma_{t+1}^2] \leq (1 - \rho)\sigma_t^2 + 2CD_F(x_t, x^*)
$$

where $A, B, C, \sigma_t^2$, and $\rho > 0$ are some constants (depend on $L, \mu, N$) and $\mathbb{E}_t[\cdot]$ is expectation with respect to the random data index at iteration $t$.

**Lemma (Simplified version of [Gorbunov et al., 2019])**

The following rate result holds:

$$
\mathbb{E}[\|x_T - x^*\|^2] \leq (1 - \frac{\rho}{2} \min\{\frac{2\mu}{A\rho + 2BC}, 1\})^T B_0
$$

where $B_0$ depends only on the initialization.
Lemma (General result, [Gorbunov et al., 2019])

The following rate result holds:

\[ \mathbb{E}[\|x_T - x^*\|^2] \leq (1 - \rho \min\{\frac{2\mu}{A\rho + 2BC}, 1\})^TB_0 \]

where \( B_0 \) depends only on the initialization.
Lemma (General result, [Gorbunov et al., 2019])

The following rate result holds:

$$\mathbb{E}[\norm{x_T - x^*}^2] \leq (1 - \rho^2 \min\{\frac{2\mu}{A\rho + 2BC}, 1\})^T B_0$$

where $B_0$ depends only on the initialization.

Proof: Step 1: Expand the squares

$$\norm{x_{t+1} - x^*}^2 = \norm{x_t - x^* - \eta g_t}^2$$
$$= \norm{x_t - x^*}^2 - 2\eta \langle x_t - x^*, g_t \rangle + \eta^2 \norm{g_t}^2$$
Lemma (General result, [Gorbunov et al., 2019])

The following rate result holds:

\[ \mathbb{E}[\|x_T - x^*\|^2] \leq (1 - \frac{\rho}{2} \min\{\frac{2\mu}{A\rho + 2BC}, 1\})TB_0 \]

where \( B_0 \) depends only on the initialization.

Proof: Step 1: Expand the squares and use unbiased property \( \mathbb{E}_t[g_t] = \nabla F(x_t) \):

\[
\|x_{t+1} - x^*\|^2 = \|x_t - x^* - \eta g_t\|^2
\]
\[
= \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, g_t \rangle + \eta^2 \|g_t\|^2
\]

\[
\Rightarrow \mathbb{E}_t[\|x_{t+1} - x^*\|^2] = \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, \nabla F(x_t) \rangle + \eta^2 \mathbb{E}_t[\|g_t\|^2]
\]
$\mathbb{E}_t[\|x_{t+1} - x^*\|^2] = \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, \nabla F(x_t) \rangle + \eta^2 \mathbb{E}_t[\|g_t\|^2]$
\[ \mathbb{E}_t[\|x_{t+1} - x^*\|^2] = \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, \nabla F(x_t) \rangle + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]

\[ \leq (1 - \eta \mu) \|x_t - x^*\|^2 - 2\eta D_F(x_t, x^*) + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]

**Step 2: Use Strong Convexity**

\[ D_F(x_t, x^*) + D_F(x^*, x_t) = \]

\[ \langle x_t - x^*, \nabla F(x_t) \rangle \geq \mu \|x - y\|^2 \]
Variance Reduced SGD: Proof

\[
\mathbb{E}_t[\|x_{t+1} - x^*\|^2] = \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, \nabla F(x_t) \rangle + \eta^2 \mathbb{E}_t[\|g_t\|^2]
\]
\[
\leq (1 - \eta \mu) \|x_t - x^*\|^2 - 2\eta D_F(x_t, x^*) + \eta^2 \mathbb{E}_t[\|g_t\|^2]
\]

**Step 3:** Use assumed bounds \(\mathbb{E}_t[\|g_t\|^2] \leq 2AD_F(x_t, x^*) + B\sigma_t^2\)

\[
\mathbb{E}_t[\|x_{t+1} - x^*\|^2] \leq (1 - \eta \mu) \|x_t - x^*\|^2 + 2\eta (A\eta - 1) D_F(x_t, x^*) + B\eta^2 \sigma_t^2
\]
\[\mathbb{E}_t[\|x_{t+1} - x^*\|^2] = \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, \nabla F(x_t) \rangle + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]

\[\leq (1 - \eta \mu) \|x_t - x^*\|^2 - 2\eta D_F(x_t, x^*) + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]

**Step 3:** Use assumed bounds \( \mathbb{E}_t[\|g_t\|^2] \leq 2AD_F(x_t, x^*) + B\sigma_t^2 \)

\[\mathbb{E}_t[\|x_{t+1} - x^*\|^2] \leq (1 - \eta \mu) \|x_t - x^*\|^2 + 2\eta (A\eta - 1) D_F(x_t, x^*) + B\eta^2 \sigma_t^2 \]

\[\frac{2B\eta^2}{\rho} \mathbb{E}_t[\sigma_{t+1}^2] \leq \frac{2B\eta^2}{\rho} (1 - \rho) \sigma_t^2 + \frac{2B\eta^2}{\rho} 2CD_F(x_t, x^*) \]
\[ \mathbb{E}_t[\|x_{t+1} - x^*\|^2] = \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, \nabla F(x_t) \rangle + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]
\[ \leq (1 - \eta \mu) \|x_t - x^*\|^2 - 2\eta D_F(x_t, x^*) + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]

**Step 3:** Use assumed bounds \( \mathbb{E}_t[\|g_t\|^2] \leq 2AD_F(x_t, x^*) + B\sigma_t^2 \)

\[ \mathbb{E}_t[\|x_{t+1} - x^*\|^2] \leq (1 - \eta \mu) \|x_t - x^*\|^2 + 2\eta (A\eta - 1) D_F(x_t, x^*) + B\eta^2 \sigma_t^2 \]
\[ + \frac{2B\eta^2}{\rho} \mathbb{E}_t[\sigma_{t+1}^2] \leq \frac{2B\eta^2}{\rho} (1 - \rho) \sigma_t^2 + \frac{2B\eta^2}{\rho} 2CD_F(x_t, x^*) \]
Variance Reduced SGD: Proof

\[ \mathbb{E}_t[\|x_{t+1} - x^*\|^2] = \|x_t - x^*\|^2 - 2\eta \langle x_t - x^*, \nabla F(x_t) \rangle + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]
\[ \leq (1 - \eta \mu) \|x_t - x^*\|^2 - 2\eta D_F(x_t, x^*) + \eta^2 \mathbb{E}_t[\|g_t\|^2] \]

**Step 3:** Use assumed bounds \( \mathbb{E}_t[\|g_t\|^2] \leq 2AD_F(x_t, x^*) + B\sigma_t^2 \)
\[ \mathbb{E}_t[\|x_{t+1} - x^*\|^2] \leq (1 - \eta \mu) \|x_t - x^*\|^2 + 2\eta (A\eta - 1) D_F(x_t, x^*) + B\eta^2 \sigma_t^2 \]
\[ + \frac{2B\eta^2}{\rho} \mathbb{E}_t[\sigma_{t+1}^2] \leq \frac{2B\eta^2}{\rho} (1 - \rho) \sigma_t^2 + \frac{2B\eta^2}{\rho} 2CD_F(x_t, x^*) \]

\[ \mathbb{E}_t[\|x_{t+1} - x^*\|^2 + \frac{2B\eta^2}{\rho} \sigma_{t+1}^2] \leq (1 - \mu \eta) \|x_t - x^*\|^2 + \left( 1 - \frac{\rho}{2} \right) \frac{2B\eta^2}{\rho} \sigma_t^2 + 2\eta^2 \left( \frac{A\rho + 2BC}{\rho} - \frac{1}{\eta} \right) D_F(x_t, x^*) \]

\[ \eta = \frac{\rho}{A\rho + 2BC} \]
Take full expectation

$$\mathbb{E}[\|x_{t+1} - x^*\|^2 + \frac{2B\eta^2}{\rho} \sigma_{t+1}^2] \leq \left(1 - \min\left\{\frac{\mu \rho}{A\rho + 2BC}, \frac{\rho}{2}\right\}\right) \mathbb{E}[\|x_t - x^*\|^2 + \frac{2B\eta^2}{\rho} \sigma_t^2]$$
Take full expectation and apply recursively

\[
\mathbb{E}[\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 + \frac{2B\eta^2}{\rho} \sigma_{t+1}^2] \leq \left(1 - \min\left\{\frac{\mu\rho}{A\rho + 2BC}, \frac{\rho}{2}\right\}\right) \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}^*\|^2 + \frac{2B\eta^2}{\rho} \sigma_t^2]
\]

\[
\leq \left(1 - \min\left\{\frac{\mu\rho}{A\rho + 2BC}, \frac{\rho}{2}\right\}\right)^t \mathbb{E}[\|\mathbf{x}_0 - \mathbf{x}^*\|^2 + \frac{2B\eta^2}{\rho} \sigma_0^2]
\]
Variance Reduced SGD: Proof

Take full expectation and apply recursively

$$
\mathbb{E}[\|x_{t+1} - x^*\|^2 + \frac{2B\eta^2}{\rho} \sigma^2_{t+1}] \leq \left(1 - \min\{\frac{\mu\rho}{A\rho+2BC}, \frac{\rho}{2}\}\right) \mathbb{E}[\|x_t - x^*\|^2 + \frac{2B\eta^2}{\rho} \sigma^2_t]
$$

$$
\leq \left(1 - \min\{\frac{\mu\rho}{A\rho+2BC}, \frac{\rho}{2}\}\right)^t \mathbb{E}[\|x_0 - x^*\|^2 + \frac{2B\eta^2}{\rho} \sigma^2_0]
$$

Equivalently, to get $\mathbb{E}[\|x_{T+1} - x^*\|^2] \leq \epsilon$ needs

$$
T = \frac{\log\left(\frac{1}{\epsilon}\right)}{-\log\left(1 - \min\{\frac{\mu\rho}{A\rho+2BC}, \frac{\rho}{2}\}\right)} \approx \frac{\log\left(\frac{1}{\epsilon}\right)}{\min\{\frac{\mu\rho}{A\rho+2BC}, \frac{\rho}{2}\}}
$$
Outline

1. Context
2. Background
3. Vanilla Stochastic Gradient Descent: Large $N$
4. Variance-Reduced SGD: Moderate $N$
   - SAGA and SVRG
   - State-of-the-art and Open Problems
5. High-dimensional problems: large $d$
6. Conclusion
SAGA

Pick \( i_t \) at random from \( \{1, 2, \ldots, N\} \)

\[
\begin{align*}
\mathbf{h}_{t+1}^j &= \begin{cases} 
\mathbf{h}_t^j & j \neq i_t \\
\nabla f(x_t, \xi_{i_t}) & j = i_t
\end{cases}
\end{align*}
\]
Pick $i_t$ at random from \(\{1, 2, \ldots, N\}\)

\[
h_{t+1}^j = \begin{cases} 
  h_t^j & j \neq i_t \\
  \nabla f(x_t, \xi_{i_t}) & j = i_t
\end{cases}
\]

\[
g_t = h_{t+1}^{i_t} - h_t^{i_t} + \frac{1}{N} \sum_{i=1}^{N} h_t^i
\]

\[
\frac{1}{N} \sum_{i=1}^{N} h_t^i
\]
SAGA Approximation is Unbiased

Unbiased?

$$E_{i_t} [g_t] = E_{i_t} \left[ h_{t+1}^{i_t} \right] - E_{i_t} \left[ h_t^{i_t} \right] + \frac{1}{N} \sum_{i=1}^{N} h_t^i$$
SAGA Approximation is Unbiased

\[
\mathbb{E}_{it} [g_t] = \mathbb{E}_{it} \left[ h_{t+1}^i \right] - \mathbb{E}_{it} \left[ h_t^i \right] + \frac{1}{N} \sum_{i=1}^{N} h_t^i \\
= \nabla F(x_t)
\]

\[
\mathbb{E}_{it} [\nabla f(x_t, \xi_{it})] = \nabla F(x_t)
\]
SAGA Approximation is Unbiased

\[
\mathbb{E}_{i_t} [g_t] = \mathbb{E}_{i_t} [h_{t+1}^{i_t}] - \mathbb{E}_{i_t} [h_t^{i_t}] + \frac{1}{N} \sum_{i=1}^{N} h_t^i
\]

\[
= \nabla F(x_t) - \frac{1}{N} \sum_{i=1}^{N} h_t^i + \frac{1}{N} \sum_{i=1}^{N} h_t^i
\]
SAGA Approximation is Unbiased

\[ \mathbb{E}_{i_t} [g_t] = \mathbb{E}_{i_t} [h_{i_t}^{i+t}] - \mathbb{E}_{i_t} [h_{i_t}^{i}] + \frac{1}{N} \sum_{i=1}^{N} h_{i_t}^{i} \]

\[ = \nabla F(x_t) \]
Since $\nabla F(x^*) = 0$, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - h^i_t - \mathbb{E}_{it} \left[ \nabla f(x^*, \xi_{it}) - h^i_t \right]$$

$$= \mathbf{X} + \mathbf{Y} - \mathbb{E}_{it} \left[ \mathbf{Y} \right]$$
SAGA Approximation: Variance

Since $\nabla F(x^*) = 0$, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - h^i_t - \mathbb{E}_{it}[\nabla f(x^*, \xi_{it}) - h^i_t]$$

$$= X + Y - \mathbb{E}_{it}[Y]$$

$$\mathbb{E}_{it}[\|g_t\|^2] \leq 2\mathbb{E}_{it}[\|\nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2] + 2\mathbb{E}_{it}[\|h^i_t - \nabla f(x^*, \xi_{it})\|^2]$$

$$\mathbb{E}[\|X + Y - \mathbb{E}[Y]\|^2] \leq 2\mathbb{E}[\|X\|^2] + 2\mathbb{E}[\|Y\|^2]$$
Since \( \nabla F(x^*) = 0 \), add and subtract \( \nabla f(x^*, \xi_{it}) \) to write

\[
g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - h^i_t - \mathbb{E}_{it} \left[ \nabla f(x^*, \xi_{it}) - h^i_t \right]
\]

\[
e_{it} \left[ \|g_t\|^2 \right] \leq 2\mathbb{E}_{it} \left[ \|\nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2 \right] + 2\mathbb{E}_{it} \left[ \|h^i_t - \nabla f(x^*, \xi_{it})\|^2 \right]
\]

\[
= \frac{2}{N} \sum_{i=1}^{N} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 + \frac{2}{N} \sum_{i=1}^{N} \|h^i_t - \nabla f(x^*, \xi_i)\|^2
\]
Since $\nabla F(x^*) = 0$, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - h_{it} - \mathbb{E}_{it} \left[ \nabla f(x^*, \xi_{it}) - h_{it} \right]$$

$$= X + Y - \mathbb{E}_{it} [Y]$$

$$\mathbb{E}_{it} \left[ \|g_t\|^2 \right] \leq 2\mathbb{E}_{it} \left[ \|\nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2 \right] + 2\mathbb{E}_{it} \left[ \|h_{it} - \nabla f(x^*, \xi_{it})\|^2 \right]$$

$$= \frac{2}{N} \sum_{i=1}^{N} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 + \frac{2}{N} \sum_{i=1}^{N} \|h_{it} - \nabla f(x^*, \xi_i)\|^2$$

$$\leq 4LD_F(x_t, x^*) + 2\sigma_t^2$$

$L$-smoothness

$$\frac{1}{2L} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 \leq f(x, \xi_i) - f(x^*, \xi_i) - \langle \nabla f(x^*, \xi_i), x - x^* \rangle$$
Since $\nabla F(x^*) = 0$, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - h^i_t - \mathbb{E}_{it} \left[ \nabla f(x^*, \xi_{it}) - h^i_t \right]$$

$$= X + Y - \mathbb{E}_{it} [Y]$$

$$\mathbb{E}_{it} \left[ \|g_t\|^2 \right] \leq 2 \mathbb{E}_{it} \left[ \|\nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2 \right] + 2 \mathbb{E}_{it} \left[ \|h^i_t - \nabla f(x^*, \xi_{it})\|^2 \right]$$

$$= \frac{2}{N} \sum_{i=1}^{N} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 + \frac{2}{N} \sum_{i=1}^{N} \|h^i_t - \nabla f(x^*, \xi_i)\|^2$$

$$\leq 4L D_F(x_t, x^*) + 2\sigma^2_t$$

$A = 2L, B = 2$
Recall that

\[ h_{t+1}^j = \begin{cases} 
  h_t^j & j \neq i_t \text{ with prob. } (1 - \frac{1}{N}) \\
  \nabla f(x_t, \xi_{i_t}) & j = i_t \text{ with prob. } \frac{1}{N}
\end{cases} \]
SAGA Approximation: $\sigma_t^2$

Recall that

$$h_{t+1}^j = \begin{cases} h_t^j & j \neq i_t \text{ with prob. } (1 - \frac{1}{N}) \\ \nabla f(x_t, \xi_{it}) & j = i_t \text{ with prob. } \frac{1}{N} \end{cases}$$

$$\mathbb{E}_{i_t} [\sigma_{t+1}^2] = \frac{1}{N} \sum_{j=1}^{N} \mathbb{E}_{i_t} \left[ \|h_{t+1}^j - \nabla f(x^*, \xi_j)\|^2 \right]$$

$$= \frac{1}{N} \sum_{j=1}^{N} \left[ (1 - \frac{1}{N}) \|h_t^j - \nabla f(x^*, \xi_j)\|^2 + \frac{1}{N} \|\nabla f(x_t, \xi_j) - \nabla f(x^*, \xi_j)\|^2 \right]$$

$$\leq \quad \left(1 - \frac{1}{N}\right) \sigma_t^2 + \frac{2L}{N} D_F(x_t, x^*)$$

$L$-smoothness

$$\frac{1}{2L} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 \leq f(x, \xi_i) - f(x^*, \xi_i) - \langle \nabla f(x^*, \xi_i), x - x^* \rangle$$
Recall that

\[ h_{t+1}^j = \begin{cases} 
    h_t^j & j \neq i_t \text{ with prob. } (1 - \frac{1}{N}) \\
    \nabla f(x_t, \xi_{i_t}) & j = i_t \text{ with prob. } \frac{1}{N}
\end{cases} \]

\[
\mathbb{E}_{i_t} [\sigma_{t+1}^2] = \frac{1}{N} \sum_{j=1}^{N} \mathbb{E}_{i_t} \left[ \left\| h_{t+1}^j - \nabla f(x^*, \xi_j) \right\|^2 \right]
\]

\[
= \frac{1}{N} \sum_{j=1}^{N} \left[ (1 - \frac{1}{N}) \left\| h_t^j - \nabla f(x^*, \xi_j) \right\|^2 + \frac{1}{N} \left\| \nabla f(x_t, \xi_j) - \nabla f(x^*, \xi_j) \right\|^2 \right]
\]

\[
\leq (1 - \frac{1}{N}) \sigma_t^2 + \frac{2L}{N} D_F(x_t, x^*)
\]

\[ \rho = \frac{1}{N}, \quad C = \frac{2L}{N} \]
Plugging in $A = 2L$, $B = 2$, $C = \frac{2L}{N}$, and $\rho = \frac{1}{N}$ (ignoring constants)

$$O \left( \max \left\{ N, \frac{L}{\mu} \right\} \log \left( \frac{1}{\epsilon} \right) \right)$$
Plugging in $A = 2L$, $B = 2$, $C = \frac{2L}{N}$, and $\rho = \frac{1}{N}$ (ignoring constants)

$$O\left(\max\left\{N, \frac{L}{\mu}\right\} \log \left(\frac{1}{\epsilon}\right)\right)$$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \frac{L}{\mu} \times \log \left(\frac{1}{\epsilon}\right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \frac{L}{\mu} \times \frac{1}{\epsilon}$</td>
<td>$d$</td>
</tr>
<tr>
<td>SAGA</td>
<td>max $\left{N, \frac{L}{\mu}\right} \times \log \left(\frac{1}{\epsilon}\right)$</td>
<td>$dN$</td>
</tr>
</tbody>
</table>
Plugging in $A = 2L$, $B = 2$, $C = \frac{2L}{N}$, and $\rho = \frac{1}{N}$ (ignoring constants)

$$O \left( \max \left\{ N, \frac{L}{\mu} \right\} \log \left( \frac{1}{\epsilon} \right) \right)$$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \frac{L}{\mu} \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \frac{L}{\mu} \times \frac{1}{\epsilon}$</td>
<td>$d$</td>
</tr>
<tr>
<td>SAGA</td>
<td>$\max \left{ N, \frac{L}{\mu} \right} \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>$dN$</td>
</tr>
</tbody>
</table>

Improves over SGD when $N$ is *not too large* but high storage
Loopless SVRG

• Consider the loopless SVRG proposed in [Kovalev et al., 2019]
Loopless SVRG

- Consider the loopless SVRG proposed in [Kovalev et al., 2019]
- A “loopless” modification of SVRG [Johnson and Zhang, 2013]
Loopless SVRG

- Consider the loopless SVRG proposed in [Kovalev et al., 2019]
- A “loopless” modification of SVRG [Johnson and Zhang, 2013]
- Pick $i_t$ at random from $\{1, 2, \ldots, N\}$ and set

$$g_t = \nabla f(x_t, \xi_{i_t}) - \nabla f(y_t, \xi_{i_t}) + \nabla F(y_t)$$

$$y_{t+1} = \begin{cases} x_t & \text{with prob. } \frac{1}{N} \text{ and calculate } \nabla F(x_t) \\ y_t & \text{with prob. } 1 - \frac{1}{N} \end{cases}$$
Loopless SVRG

- Consider the loopless SVRG proposed in [Kovalev et al., 2019]
- A “loopless” modification of SVRG [Johnson and Zhang, 2013]
- Pick $i_t$ at random from $\{1, 2, \ldots, N\}$ and set

$$g_t = \nabla f(x_t, \xi_{i_t}) - \nabla f(y_t, \xi_{i_t}) + \nabla F(y_t)$$

$$y_{t+1} = \begin{cases} 
  x_t & \text{with prob. } \frac{1}{N} \text{ and calculate } \nabla F(x_t) \\
  y_t & \text{with prob. } 1 - \frac{1}{N}
\end{cases}$$

- On average, 3 gradients evaluated per iteration
• Consider the loopless SVRG proposed in [Kovalev et al., 2019]
• A “loopless” modification of SVRG [Johnson and Zhang, 2013]
• Pick $i_t$ at random from $\{1, 2, \ldots, N\}$ and set

\[
g_t = \nabla f(x_t, \xi_{i_t}) - \nabla f(y_t, \xi_{i_t}) + \nabla F(y_t)
\]

\[
y_{t+1} = \begin{cases} 
  x_t & \text{with prob. } \frac{1}{N} \\
  y_t & \text{with prob. } 1 - \frac{1}{N}
\end{cases}
\]

• On average, 3 gradients evaluated per iteration
• Unbiased gradient

\[
\mathbb{E}_{i_t} [g_t] = \mathbb{E}_{i_t} [\nabla f(x_t, \xi_{i_t})] - \mathbb{E}_{i_t} [\nabla f(y_t, \xi_{i_t})] + \nabla F(y_t)
\]
• Consider the loopless SVRG proposed in [Kovalev et al., 2019]
• A “loopless” modification of SVRG [Johnson and Zhang, 2013]
• Pick $i_t$ at random from $\{1, 2, \ldots, N\}$ and set

$$g_t = \nabla f(x_t, \xi_{i_t}) - \nabla f(y_t, \xi_{i_t}) + \nabla F(y_t)$$

$$y_{t+1} = \begin{cases} x_t \text{ with prob. } \frac{1}{N} \text{ and calculate } \nabla F(x_t) \\ y_t \text{ with prob. } 1 - \frac{1}{N} \end{cases}$$

• On average, 3 gradients evaluated per iteration
• Unbiased gradient

$$\mathbb{E}_{i_t} [g_t] = \mathbb{E}_{i_t} [\nabla f(x_t, \xi_{i_t})] - \mathbb{E}_{i_t} [\nabla f(y_t, \xi_{i_t})] + \nabla F(y_t)$$

$$= \nabla F(x_t) - \nabla F(y_t) + \nabla F(y_t)$$
Loopless SVRG

- Consider the loopless SVRG proposed in [Kovalev et al., 2019]
- A “loopless” modification of SVRG [Johnson and Zhang, 2013]
- Pick $i_t$ at random from $\{1, 2, \ldots, N\}$ and set

$$g_t = \nabla f(x_t, \xi_{i_t}) - \nabla f(y_t, \xi_{i_t}) + \nabla F(y_t)$$

$$y_{t+1} = \begin{cases} 
  x_t & \text{with prob. } \frac{1}{N} \text{ and calculate } \nabla F(x_t) \\
  y_t & \text{with prob. } 1 - \frac{1}{N}
\end{cases}$$

- On average, 3 gradients evaluated per iteration
- Unbiased gradient

$$E_{i_t}[g_t] = E_{i_t}[\nabla f(x_t, \xi_{i_t})] - E_{i_t}[\nabla f(y_t, \xi_{i_t})] + \nabla F(y_t)$$

$$= \nabla F(x_t)$$
As in SAGA, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it}) - \mathbb{E}_{it} [\nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it})]$$

$$= X + Y - \mathbb{E}_{it} [Y]$$
As in SAGA, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$
g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it}) - \mathbb{E}_{it} [\nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it})]
$$

$$
= X + Y - \mathbb{E}_{it} [Y]
$$

$$
\mathbb{E}_{it} [\|g_t\|^2] \leq 2\mathbb{E}_{it} [\|\nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2] + 2\mathbb{E}_{it} [\|\nabla f(y_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2]
$$

$$
\mathbb{E}[\|X + Y - \mathbb{E}[Y]\|^2] \leq 2\mathbb{E}[\|X\|^2] + 2\mathbb{E}[\|Y\|^2]
$$
Loopless SVRG: Approximation Properties

As in SAGA, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it}) - \mathbb{E}_{it} [\nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it})]$$

$$= X + Y - \mathbb{E}_{it} [Y]$$

$$\mathbb{E}_{it} [\|g_t\|^2] \leq 2\mathbb{E}_{it} [\|\nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2] + 2\mathbb{E}_{it} [\|\nabla f(y_t, \xi_{it}) - \nabla f(x^*, \xi_{it})\|^2]$$

$$= \frac{2}{N} \sum_{i=1}^{N} \|\nabla f(x_t, \xi_{i}) - \nabla f(x^*, \xi_{i})\|^2 + \frac{2}{N} \sum_{i=1}^{N} \|\nabla f(y_t, \xi_{i}) - \nabla f(x^*, \xi_{i})\|^2$$
As in SAGA, add and subtract $\nabla f(x^*, \xi_{i_t})$ to write

$$g_t = \nabla f(x_t, \xi_{i_t}) - \nabla f(x^*, \xi_{i_t}) + \nabla f(x^*, \xi_{i_t}) - \nabla f(y_t, \xi_{i_t}) - E_{i_t} [\nabla f(x^*, \xi_{i_t}) - \nabla f(y_t, \xi_{i_t})]$$

$$= X + Y - E_{i_t} [Y]$$

$$E_{i_t} \left[ \|g_t\|^2 \right] \leq 2E_{i_t} \left[ \|\nabla f(x_t, \xi_{i_t}) - \nabla f(x^*, \xi_{i_t})\|^2 \right] + 2E_{i_t} \left[ \|\nabla f(y_t, \xi_{i_t}) - \nabla f(x^*, \xi_{i_t})\|^2 \right]$$

$$= \frac{2}{N} \sum_{i=1}^{N} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 + \frac{2}{N} \sum_{i=1}^{N} \|\nabla f(y_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2$$

$$\leq 4LD_F(x_t, x^*) + 2\sigma_t^2$$

$L$-smoothness

$$\frac{1}{2L} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 \leq f(x, \xi_i) - f(x^*, \xi_i) - \langle \nabla f(x^*, \xi_i), x - x^* \rangle$$
As in SAGA, add and subtract $\nabla f(x^*, \xi_{it})$ to write

$$g_t = \nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it}) + \nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it}) - \mathbb{E}_{it} \left[ \nabla f(x^*, \xi_{it}) - \nabla f(y_t, \xi_{it}) \right]$$

$$= X + Y - \mathbb{E}_{it} \left[ Y \right]$$

$$\mathbb{E}_{it} \left[ ||g_t||^2 \right] \leq 2\mathbb{E}_{it} \left[ ||\nabla f(x_t, \xi_{it}) - \nabla f(x^*, \xi_{it})||^2 \right] + 2\mathbb{E}_{it} \left[ ||\nabla f(y_t, \xi_{it}) - \nabla f(x^*, \xi_{it})||^2 \right]$$

$$= \frac{2}{N} \sum_{i=1}^{N} ||\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)||^2 + \frac{2}{N} \sum_{i=1}^{N} ||\nabla f(y_t, \xi_i) - \nabla f(x^*, \xi_i)||^2$$

$$\leq 4LD_F(x_t, x^*) + 2\sigma_t^2$$

$$A = 2L, \ B = 2$$
Recall that

\[ y_{t+1} = \begin{cases} 
  y_t & \text{with prob. } (1 - \frac{1}{N}) \\
  x_t & \text{with prob. } \frac{1}{N} \text{ (calculate } \nabla F(x_t))
\end{cases} \]
Loopless SVRG: $\sigma^2_t$

Recall that

$$y_{t+1} = \begin{cases} y_t & \text{with prob. } (1 - \frac{1}{N}) \\ x_t & \text{with prob. } \frac{1}{N} \text{ (calculate } \nabla F(x_t)\text{)} \end{cases}$$

$$\mathbb{E}_{i_t} [\sigma^2_{t+1}] = \frac{1}{N} \sum_{j=1}^{N} \mathbb{E}[\|\nabla f(y_{t+1}, \xi_j) - \nabla f(x^*, \xi_j)\|^2]$$

$$= \frac{1}{N} \sum_{j=1}^{N} \left[ (1 - \frac{1}{N}) \|\nabla f(y_t, \xi_j) - \nabla f(x^*, \xi_j)\|^2 + \frac{1}{N} \|\nabla f(x_t, \xi_j) - \nabla f(x^*, \xi_j)\|^2 \right]$$

$$\leq \left( 1 - \frac{1}{N} \right) \sigma^2_t + \frac{2L}{N} D_F(x_t, x^*)$$

$L$-smoothness

$$\frac{1}{2L} \|\nabla f(x_t, \xi_i) - \nabla f(x^*, \xi_i)\|^2 \leq f(x, \xi_i) - f(x^*, \xi_i) - \langle \nabla f(x^*, \xi_i), x - x^* \rangle$$
Recall that

\[ y_{t+1} = \begin{cases} y_t & \text{with prob. } (1 - \frac{1}{N}) \\ x_t & \text{with prob. } \frac{1}{N} \end{cases} \text{ (calculate } \nabla F(x_t)) \]

\[
\mathbb{E}_{i_t} \left[ \sigma^2_{t+1} \right] = \frac{1}{N} \sum_{j=1}^{N} \mathbb{E}\left[ \| \nabla f(y_{t+1}, \xi_j) - \nabla f(x^*, \xi_j) \|^2 \right]
\]

\[
= \frac{1}{N} \sum_{j=1}^{N} \left[ (1 - \frac{1}{N}) \| \nabla f(y_t, \xi_j) - \nabla f(x^*, \xi_j) \|^2 + \frac{1}{N} \| \nabla f(x_t, \xi_j) - \nabla f(x^*, \xi_j) \|^2 \right]
\]

\[
\leq (1 - \frac{1}{N}) \sigma^2_t + \frac{2L}{N} D_F(x_t, x^*)
\]

\[ \rho = \frac{1}{N}, \quad C = \frac{2L}{N} \]
## Loopless SVRG: Summary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \frac{L}{\mu} \times \log\left(\frac{1}{\epsilon}\right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \frac{L}{\mu} \times \frac{1}{\epsilon}$</td>
<td>$d$</td>
</tr>
<tr>
<td>SAGA</td>
<td>$\max\left{N, \frac{L}{\mu}\right} \times \log\left(\frac{1}{\epsilon}\right)$</td>
<td>$dN$</td>
</tr>
<tr>
<td>L-SVRG</td>
<td>$\max\left{N, \frac{L}{\mu}\right} \times \log\left(\frac{1}{\epsilon}\right)$</td>
<td>$d$</td>
</tr>
</tbody>
</table>
Loopless SVRG: Summary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \frac{L}{\mu} \times \log\left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \frac{L}{\mu} \times \frac{1}{\epsilon}$</td>
<td>$d$</td>
</tr>
<tr>
<td>SAGA</td>
<td>max({N, \frac{L}{\mu}}) \times \log\left( \frac{1}{\epsilon} \right)$</td>
<td>$dN$</td>
</tr>
<tr>
<td>L-SVRG</td>
<td>max({N, \frac{L}{\mu}}) \times \log\left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
</tbody>
</table>

Loopless SVRG has almost same number of gradient calculations as SAGA but requires same storage as SGD
Outline

1. Context
2. Background
3. Vanilla Stochastic Gradient Descent: Large $N$
4. Variance-Reduced SGD: Moderate $N$
   - SAGA and SVRG
   - State-of-the-art and Open Problems
5. High-dimensional problems: large $d$
6. Conclusion
Accelerated Variants

- Accelerated GD proposed by Nesterov in 1983: uses a momentum term

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times L \mu \times \log \left( \frac{1}{\epsilon} \right) d$</td>
<td></td>
</tr>
<tr>
<td>Accelerated GD</td>
<td>$N \times \sqrt{L} \mu \times \log \left( \frac{1}{\epsilon} \right) d$</td>
<td></td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times L \mu \times \frac{1}{\epsilon} d$</td>
<td></td>
</tr>
<tr>
<td>L-SVRG</td>
<td>$\max { N, L \mu } \times \log \left( \frac{1}{\epsilon} \right) d$</td>
<td></td>
</tr>
<tr>
<td>Accelerated SVRG</td>
<td>$(N + \sqrt{NL}) \mu \times \log \left( \frac{1}{\epsilon} \right) d$</td>
<td></td>
</tr>
</tbody>
</table>
Accelerated Variants

- Accelerated GD proposed by Nesterov in 1983: uses a momentum term
- But acceleration has not been achieved for classical SGD

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \frac{1}{\epsilon} \times L \mu \times \log\left(\frac{1}{\epsilon}\right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>Accelerated GD</td>
<td>$N \times \frac{1}{\sqrt{L}} \times \frac{1}{\epsilon} \times L \mu \times \log\left(\frac{1}{\epsilon}\right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \frac{1}{\epsilon} \times L \mu \times \frac{1}{\epsilon}$</td>
<td>$d$</td>
</tr>
<tr>
<td>L-SVRG</td>
<td>$\max{N, L \mu} \times \log\left(\frac{1}{\epsilon}\right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>Accelerated SVRG</td>
<td>$(N + \sqrt{NL}) \times \frac{1}{\epsilon} \times L \mu \times \log\left(\frac{1}{\epsilon}\right)$</td>
<td>$d$</td>
</tr>
</tbody>
</table>
Accelerated Variants

- Accelerated GD proposed by Nesterov in 1983: uses a momentum term
- But acceleration has not been achieved for classical SGD
- Indeed, momentum SGD is prone to error accumulation [Konevčný et al., 2015]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times L \mu \times \log(1/\epsilon)$</td>
<td>$d$</td>
</tr>
<tr>
<td>Accelerated GD</td>
<td>$N \times \sqrt{L} \mu \times \log(1/\epsilon)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times L \mu \times 1/\epsilon$</td>
<td>$d$</td>
</tr>
<tr>
<td>L-SVRG</td>
<td>$\max{N, L \mu} \times \log(1/\epsilon)$</td>
<td>$d$</td>
</tr>
<tr>
<td>Accelerated SVRG</td>
<td>$(N + \sqrt{NL} \mu) \times \log(1/\epsilon)$</td>
<td>$d$</td>
</tr>
</tbody>
</table>
Accelerated Variants

• Accelerated GD proposed by Nesterov in 1983: uses a momentum term
• But acceleration has not been achieved for classical SGD
• Indeed, momentum SGD is prone to error accumulation [Konevcnỳ et al., 2015]
• But can it work for variance-reduced algorithms?
Accelerated GD proposed by Nesterov in 1983: uses a momentum term
But acceleration has not been achieved for classical SGD
Indeed, momentum SGD is prone to error accumulation [Konevčnỳ et al., 2015]
But can it work for variance-reduced algorithms?
Resolved partially in [Lin et al., 2015] and completely in [Allen-Zhu, 2017]
Accelerated Variants

- Accelerated GD proposed by Nesterov in 1983: uses a momentum term
- But acceleration has not been achieved for classical SGD
- Indeed, momentum SGD is prone to error accumulation [Konevčnỳ et al., 2015]
- But can it work for variance-reduced algorithms?
- Resolved partially in [Lin et al., 2015] and completely in [Allen-Zhu, 2017]
- Several variants since then, active area of research
Accelerated Variants

- Accelerated GD proposed by Nesterov in 1983: uses a momentum term
- But acceleration has not been achieved for classical SGD
- Indeed, momentum SGD is prone to error accumulation [Konevchny et al., 2015]
- But can it work for variance-reduced algorithms?
- Resolved partially in [Lin et al., 2015] and completely in [Allen-Zhu, 2017]
- Several variants since then, active area of research

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \frac{L}{\mu} \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>Accelerated GD</td>
<td>$N \times \sqrt{\frac{L}{\mu}} \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \frac{L}{\mu} \times \frac{1}{\epsilon}$</td>
<td>$d$</td>
</tr>
<tr>
<td>L-SVRG</td>
<td>$\max \left{ N, \frac{L}{\mu} \right} \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>Accelerated SVRG</td>
<td>$\left( N + \sqrt{\frac{NL}{\mu}} \right) \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
</tbody>
</table>
## Accelerated Variants: Smooth + Convex

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times L \times \frac{1}{\epsilon}$</td>
</tr>
<tr>
<td>Accelerated GD</td>
<td>$N \times \sqrt{L} \times \frac{1}{\sqrt{\epsilon}}$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times L \times \frac{1}{\epsilon^2}$</td>
</tr>
<tr>
<td>SAGA</td>
<td>$(N + L) \times \frac{1}{\epsilon}$</td>
</tr>
<tr>
<td>SVRG+</td>
<td>$N \log \left( \frac{1}{\epsilon} \right) + \frac{L}{\epsilon}$</td>
</tr>
<tr>
<td>Accelerated SVRG</td>
<td>$N \log \left( \frac{1}{\epsilon} \right) + \sqrt{\frac{NL}{\epsilon}}$</td>
</tr>
</tbody>
</table>
• Moderately large $N \leq \epsilon^{-2}$
Non-Convex Finite Sum: SPIDER

- Moderately large $N \leq \epsilon^{-2}$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \epsilon^{-1}$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \epsilon^{-2}$</td>
</tr>
<tr>
<td>SVRG/SAGA</td>
<td>$N^{2/3} \times \epsilon^{-1}$</td>
</tr>
<tr>
<td>SPIDER/SPIDERBoost</td>
<td>$N^{1/2} \times \epsilon^{-1}$</td>
</tr>
</tbody>
</table>
Non-Convex Finite Sum: SPIDER

- Moderately large $N \leq \epsilon^{-2}$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$N \times \epsilon^{-1}$</td>
</tr>
<tr>
<td>SGD</td>
<td>$1 \times \epsilon^{-2}$</td>
</tr>
<tr>
<td>SVRG/SAGA</td>
<td>$N^{2/3} \times \epsilon^{-1}$</td>
</tr>
<tr>
<td>SPIDER/SPIDERBoost</td>
<td>$N^{1/2} \times \epsilon^{-1}$</td>
</tr>
</tbody>
</table>

- SPIDER [Fang et al., 2018] and SPIDERBoost [Wang et al., 2018] rate optimal in terms of $N$ and $\epsilon$

- **Open problem:** Adaptive step-size variant of SPIDER?
Non-Convex Online: STORM

- SAGA/SVRG not meant for large $N$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>$\epsilon^{-2}$</td>
</tr>
<tr>
<td>SVRG+</td>
<td>$\epsilon^{-5/3}$</td>
</tr>
<tr>
<td>SPIDER/SPIDERBoost</td>
<td>$\epsilon^{-3/2}$</td>
</tr>
<tr>
<td>STORM</td>
<td>$\epsilon^{-3/2}$</td>
</tr>
</tbody>
</table>
Non-Convex Online: STORM

- SAGA/SVRG not meant for large $N$
- SARAH [Nguyen et al., 2017], SPIDER proposed calculating “checkpoint” gradients every $\epsilon^{-1}$ samples: mega batches hard to tune
Non-Convex Online: STORM

- SAGA/SVRG not meant for large $N$
- SARAH [Nguyen et al., 2017], SPIDER proposed calculating “checkpoint” gradients every $\epsilon^{-1}$ samples: mega batches hard to tune
- STORM uses momentum + adaptive step-size to achieve optimal rate using single loop
Non-Convex Online: STORM

- SAGA/SVRG not meant for large $N$
- SARAH [Nguyen et al., 2017], SPIDER proposed calculating “checkpoint” gradients every $\epsilon^{-1}$ samples: mega batches hard to tune
- STORM uses momentum + adaptive step-size to achieve optimal rate using single loop

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>$\epsilon^{-2}$</td>
</tr>
<tr>
<td>SVRG+</td>
<td>$\epsilon^{-5/3}$</td>
</tr>
<tr>
<td>SPIDER/SPIDERBoost</td>
<td>$\epsilon^{-3/2}$</td>
</tr>
<tr>
<td>STORM</td>
<td>$\epsilon^{-3/2}$</td>
</tr>
</tbody>
</table>
Non-Convex Online: STORM

- SAGA/SVRG not meant for large $N$
- SARAH [Nguyen et al., 2017], SPIDER proposed calculating “checkpoint” gradients every $\epsilon^{-1}$ samples: mega batches hard to tune
- STORM uses momentum + adaptive step-size to achieve optimal rate using single loop

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>$\epsilon^{-2}$</td>
</tr>
<tr>
<td>SVRG+</td>
<td>$\epsilon^{-5/3}$</td>
</tr>
<tr>
<td>SPIDER/SPIDERBoost</td>
<td>$\epsilon^{-3/2}$</td>
</tr>
<tr>
<td>STORM</td>
<td>$\epsilon^{-3/2}$</td>
</tr>
</tbody>
</table>

- Open problem: can STORM to handle $\mathcal{X}$, regularizers, etc?
Consider the problem

\[ \min_{x \in \mathcal{X}} \sum_{k \in \mathcal{V}} F_k(x) \]

- Data points \( \{\xi_{i}^{k}\}_{i=1}^{N} \) available only at \( k \)-th node
- Central server aids in parallelizing: \( K \) nodes can offer \( K \)-fold speedup in wall-clock time
- **State-of-the-art:** Parallel Restarted SPIDER matches centralized \( \mathcal{O}(\epsilon^{-3/2}) \) for online non-convex
- **Open problems:** Distributed version of STORM? Accelerated variants?
Open Problem: Decentralized Setting

- Again consider the problem

\[
\min_{x \in \mathcal{X}} \sum_{k \in \mathcal{V}} F_k(x)
\]

- No central server, only communication between peers is allowed
- All existing approaches are either suboptimal or cannot handle
- For non-convex, optimal \(O(\epsilon^{-3/2})\) achieved in [Sun et al., 2019]
- Open problem: can accelerated rates be obtained for convex decentralized case?
Open Problem: Decentralized Setting

• Again consider the problem

\[ \min_{x \in X} \sum_{k \in V} F_k(x) \]

• No central server, only communication between peers is allowed
Open Problem: Decentralized Setting

• Again consider the problem

$$\min_{x \in \mathcal{X}} \sum_{k \in V} F_k(x)$$

• No central server, only communication between peers is allowed
• All existing approaches are either suboptimal or cannot handle $\mathcal{X}$
Open Problem: Decentralized Setting

• Again consider the problem

\[
\min_{x \in \mathcal{X}} \sum_{k \in \mathcal{V}} F_k(x)
\]

• No central server, only communication between peers is allowed

• All existing approaches are either suboptimal or cannot handle \( \mathcal{X} \)

• For non-convex, optimal \( \mathcal{O}(\epsilon^{-3/2}) \) achieved in [Sun et al., 2019]
Open Problem: Decentralized Setting

- Again consider the problem

\[
\min_{x \in \mathcal{X}} \sum_{k \in \mathcal{V}} F_k(x)
\]

- No central server, only communication between peers is allowed
- All existing approaches are either suboptimal or cannot handle \( \mathcal{X} \)
- For non-convex, optimal \( \mathcal{O}(\epsilon^{-3/2}) \) achieved in [Sun et al., 2019]
- **Open problem**: can accelerated rates be obtained for convex decentralized case?
High-dimensional problems: large $d$
• When $d$ is large, accessing $\nabla F(x)$ becomes difficult

E.g.: in matrix completion, $\nabla F(X) \in \mathbb{R}^{m \times n}$ may be unwieldy ($d = mn$)

But a few coordinates of $\nabla F(X)$ may be available

Motivates coordinate descent and sketched gradient methods
• When $d$ is large, accessing $\nabla F(\mathbf{x})$ becomes difficult
• E.g.: in matrix completion, $\nabla F(\mathbf{X}) \in \mathbb{R}^{m \times n}$ may be unwieldy ($d = mn$)
• When $d$ is large, accessing $\nabla F(x)$ becomes difficult
• E.g.: in matrix completion, $\nabla F(X) \in \mathbb{R}^{m \times n}$ may be unwieldy ($d = mn$)
• But a few coordinates of $\nabla F(X)$ may be available
• When $d$ is large, accessing $\nabla F(x)$ becomes difficult
• E.g.: in matrix completion, $\nabla F(X) \in \mathbb{R}^{m \times n}$ may be unwieldy ($d = mn$)
• But a few coordinates of $\nabla F(X)$ may be available
• Motivates coordinate descent and sketched gradient methods
Sketched Gradient Descent

- Consider recently proposed SEGA [Hanzely et al., 2018]
Sketched Gradient Descent

- Consider recently proposed SEGA [Hanzely et al., 2018]
- Assumes availability of $P \nabla F(x)$ where $P \in \mathbb{R}^{p \times d}$ where $p \ll d$
Sketched Gradient Descent

- Consider recently proposed SEGA [Hanzely et al., 2018]
- Assumes availability of $P \nabla F(x)$ where $P \in \mathbb{R}^{p \times d}$ where $p \ll d$
- We look at the special case of $p = 1$ and

$$P = e_{i_t}^{\top} = \begin{bmatrix} 0 & 0 & \ldots & 1 & \ldots & 0 & 0 \end{bmatrix}$$

where $i_t$ is randomly selected from $\{1, \ldots, N\}$
Consider recently proposed SEGA [Hanzely et al., 2018]
Assumes availability of $P \nabla F(x)$ where $P \in \mathbb{R}^{p \times d}$ where $p \ll d$
We look at the special case of $p = 1$ and

$$P = e_{i_t}^\top = \begin{bmatrix} 0 & 0 & \ldots & 1 & \ldots & 0 & 0 \end{bmatrix}$$

where $i_t$ is randomly selected from $\{1, \ldots, N\}$

Sketched gradient is not an unbiased estimator!
• Unbiased gradient estimate must be maintained
SEGA: single coordinate update

- Unbiased gradient estimate must be maintained
- Starting with \( h_1 = 0 \), we have

\[
\begin{align*}
    h_{t+1}^j &= \begin{cases} 
    [\nabla F(x_t)]_j & j = i_t \\
    h_t^j & j \neq i_t 
    \end{cases} \\
    [g_t]_j &= \begin{cases} 
    d[\nabla F(x_t)]_j + (1 - d)h_t^j & j = i_t \\
    h_t^j & j \neq i_t 
    \end{cases}
\end{align*}
\]
SEGA: single coordinate update

• Unbiased gradient estimate must be maintained
• Starting with $h_1 = 0$, we have

$$h_{t+1}^j = \begin{cases} 
[\nabla F(x_t)]_j & j = i_t \\
h_t^j & j \neq i_t 
\end{cases}$$

$$[g_t]_j = \begin{cases} 
d[\nabla F(x_t)]_j + (1 - d)h_t^j & j = i_t \\
h_t^j & j \neq i_t 
\end{cases}$$

• Maintain two $d \times 1$ vectors, but update only 1 coordinate at a time
SEGA: single coordinate update

• Unbiased gradient estimate must be maintained
• Starting with $h_1 = 0$, we have

$$h_{t+1}^j = \begin{cases} 
\nabla F(x_t)_j & j = i_t \\
h_{t}^j & j \neq i_t 
\end{cases}$$

$$[g_t]_j = \begin{cases} 
\nabla F(x_t)_j + (1 - d)h_t^j & j = i_t \\
h_t^j & j \neq i_t 
\end{cases}$$

• Maintain two $d \times 1$ vectors, but update only 1 coordinate at a time
• Can we get GD-like performance with such sporadic updates?
Let us write in compact form:

\[ h_{t+1} = h_t + e_{it} \odot (\nabla F(x_t) - h_t) \]

\[ g_t = h_t + d e_{it} \odot (\nabla F(x_t) - h_t) \]

where \( \odot \) denotes element-wise product
• Let us write in compact form:

\[
    h_{t+1} = h_t + e_{it} \odot (\nabla F(x_t) - h_t)
\]

\[
    g_t = h_t + d e_{it} \odot (\nabla F(x_t) - h_t)
\]

where \( \odot \) denotes element-wise product

• Note that \( \mathbb{E}[e_{it}] = \frac{1}{d} \)
Let us write in compact form:

\[ h_{t+1} = h_t + e_{it} \odot (\nabla F(x_t) - h_t) \]
\[ g_t = h_t + d e_{it} \odot (\nabla F(x_t) - h_t) \]

where \( \odot \) denotes element-wise product.

Note that \( \mathbb{E}[e_{it}] = \frac{1}{d} \)

Unbiased gradient:

\[ \mathbb{E}_{i_t} [g_t] = h_t + d \mathbb{E}_{i_t} [e_{it}] \odot (\nabla F(x_t) - h_t) = \nabla F(x_t) \]
Proceeding as earlier (since $\nabla F(x^*) = 0$)

$$g_t = d(e_{it} \odot \nabla F(x_t)) - de_{it} \odot h_t + \mathbb{E}_{it} [de_{it} \odot h_t]$$

$$= X + Y - \mathbb{E}_{it} [Y]$$
SEGA: Approximation Properties

Proceeding as earlier (since $\nabla F(x^*) = 0$)

$$
g_t = d(e_i \otimes \nabla F(x_t)) - de_i \otimes h_t + \mathbb{E}_{i_t}[de_i \otimes h_t]
= X + Y - \mathbb{E}_{i_t}[Y]
$$

$$
\mathbb{E}_{i_t}[\|g_t\|^2] \leq 2d^2\mathbb{E}_{i_t}[\|e_i \otimes \nabla F(x_t)\|^2] + 2d^2\mathbb{E}_{i_t}[\|e_i \otimes h_t\|^2]
$$

$$
\mathbb{E}[\|X + Y - \mathbb{E}[Y]\|^2] \leq 2\mathbb{E}[\|X\|^2] + 2\mathbb{E}[\|Y\|^2]
$$
Proceeding as earlier (since $\nabla F(x^*) = 0$)

$$\begin{align*}
g_t &= d(e_{it} \circ \nabla F(x_t)) - de_{it} \circ h_t + \mathbb{E}_{it} [de_{it} \circ h_t] \\
&= X + Y - \mathbb{E}_{it} [Y] \\
\mathbb{E}_{it} [\|g_t\|^2] &\leq 2d^2 \mathbb{E}_{it} [\|e_{it} \circ \nabla F(x_t)\|^2] + 2d^2 \mathbb{E}_{it} [\|e_{it} \circ h_t\|^2] \\
&= 2d \|\nabla F(x_t)\|^2 + 2d \|h_t\|^2
\end{align*}$$
SEGA: Approximation Properties

Proceeding as earlier (since $\nabla F(x^*) = 0$)

$$g_t = d(e_{it} \odot \nabla F(x_t)) - de_{it} \odot h_t + \mathbb{E}_{it}[de_{it} \odot h_t]$$

$$= X + Y - \mathbb{E}_{it}[Y]$$

$$\mathbb{E}_{it}[\|g_t\|^2] \leq 2d^2\mathbb{E}_{it}[\|e_{it} \odot \nabla F(x_t)\|^2] + 2d^2\mathbb{E}_{it}[\|e_{it} \odot h_t\|^2]$$

$$= 2d \|\nabla F(x_t)\|^2 + 2d \|h_t\|^2$$

$$\leq 4dLD_F(x_t, x^*) + 2d\sigma_t^2$$

$L$-smoothness

$$\frac{1}{2L} \|\nabla F(x_t) - \nabla F(x^*)\|^2 \leq F(x) - F(x^*) = D_F(x_t, x^*)$$
SEGA: Approximation Properties

Proceeding as earlier (since $\nabla F(x^*) = 0$)

$$g_t = d(e_{it} \odot \nabla F(x_t)) - de_{it} \odot h_t + \mathbb{E}_{it}[de_{it} \odot h_t]$$

$$= X + Y - \mathbb{E}_{it}[Y]$$

$$\mathbb{E}_{it}[\|g_t\|^2] \leq 2d^2\mathbb{E}_{it}[\|e_{it} \odot \nabla F(x_t)\|^2] + 2d^2\mathbb{E}_{it}[\|e_{it} \odot h_t\|^2]$$

$$= 2d\|\nabla F(x_t)\|^2 + 2d\|h_t\|^2$$

$$\leq 4dLD_F(x_t, x^*) + 2d\sigma_t^2$$

$$A = 2dL, \ B = 2d$$
Recall that $h_{t+1} = h_t + e_i \circ (\nabla F(x_t) - h_t)$, so
SEGA Approximation: $\sigma_t^2$

Recall that $h_{t+1} = h_t + e_{it} \odot (\nabla F(x_t) - h_t)$, so

$$E_{it} [\sigma_{t+1}^2] = E_{it} [\|h_{t+1}\|^2] = E_{it} [\|h_t + e_{it} \odot (\nabla F(x_t) - h_t)\|^2]$$
Recall that $h_{t+1} = h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)$, so

$$
E_{i_t} [\sigma^2_{t+1}] = E_{i_t} [\|h_{t+1}\|^2] = E_{i_t} [\|h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)\|^2] \\
= E_{i_t} \left[ \| (I - e_{i_t} e_{i_t}^\top) h_t + e_{i_t} e_{i_t}^\top \nabla F(x_t) \|^2 \right]
$$
Recall that $h_{t+1} = h_t + e_{it} \odot (\nabla F(x_t) - h_t)$, so

$$
\mathbb{E}_{it} \left[ \sigma_{t+1}^2 \right] = \mathbb{E}_{it} \left[ \|h_{t+1}\|^2 \right] = \mathbb{E}_{it} \left[ \|h_t + e_{it} \odot (\nabla F(x_t) - h_t)\|^2 \right] \\
= \mathbb{E}_{it} \left[ \|(I - e_{it} e_{it}^T)h_t + e_{it} e_{it}^T \nabla F(x_t)\|^2 \right] \\
= \mathbb{E}_{it} \left[ \|(I - e_{it} e_{it}^T)h_t\|^2 \right] + \mathbb{E}_{it} \left[ \|e_{it} \odot (\nabla F(x_t))\|^2 \right]
$$

$$
\mathbb{E}_{it} \left[ (I - e_{it} e_{it}^T) e_{it} e_{it}^T \right] = \\
\mathbb{E}_{it} \left[ e_{it} e_{it}^T \right] - \mathbb{E}_{it} \left[ e_{it} e_{it}^T e_{it} e_{it}^T \right] = 0
$$
Recall that $h_{t+1} = h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)$, so

$$
\mathbb{E}_{i_t} \left[ \sigma^2_{t+1} \right] = \mathbb{E}_{i_t} \left[ \|h_{t+1}\|^2 \right] = \mathbb{E}_{i_t} \left[ \|h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)\|^2 \right]
$$

$$
= \mathbb{E}_{i_t} \left[ \left\| (I - e_{i_t} e_{i_t}^\top) h_t + e_{i_t} e_{i_t}^\top \nabla F(x_t) \right\|^2 \right]
$$

$$
= \mathbb{E}_{i_t} \left[ \left\| (I - e_{i_t} e_{i_t}^\top) h_t \right\|^2 \right] + \mathbb{E}_{i_t} \left[ \left\| e_{i_t} \odot (\nabla F(x_t)) \right\|^2 \right]
$$

$$
= \left( 1 - \frac{1}{d} \right) \mathbb{E}_{i_t} \left[ \|h_t\|^2 \right] + \frac{1}{d} \|\nabla F(x_t)\|^2
$$
Recall that $h_{t+1} = h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)$, so

\[
\mathbb{E}_{i_t} [\sigma_{t+1}^2] = \mathbb{E}_{i_t} [\|h_{t+1}\|^2] = \mathbb{E}_{i_t} [\|h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)\|^2]
\]

\[
= \mathbb{E}_{i_t} [\|(I - e_{i_t}e_{i_t}^\top)h_t + e_{i_t}e_{i_t}^\top \nabla F(x_t)\|^2]
\]

\[
= \mathbb{E}_{i_t} [\|(I - e_{i_t}e_{i_t}^\top)h_t\|^2] + \mathbb{E}_{i_t} [\|e_{i_t} \odot (\nabla F(x_t))\|^2]
\]

\[
= \left(1 - \frac{1}{d}\right) \mathbb{E}_{i_t} [\|h_t\|^2] + \frac{1}{d} \|\nabla F(x_t)\|^2
\]

\[
\leq \left(1 - \frac{1}{d}\right) \sigma_t^2 + \frac{2L}{d} D_F(x_t, x^*)
\]

$L$-smoothness

\[
\frac{1}{2L} \|\nabla F(x_t)\|^2 \leq D_F(x_t, x^*)
\]
Recall that $h_{t+1} = h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)$, so

$$
\mathbb{E}_{i_t} \left[ \sigma_{t+1}^2 \right] = \mathbb{E}_{i_t} \left[ \|h_{t+1}\|^2 \right] = \mathbb{E}_{i_t} \left[ \|h_t + e_{i_t} \odot (\nabla F(x_t) - h_t)\|^2 \right]
$$

$$
= \mathbb{E}_{i_t} \left[ \| (I - e_{i_t} e_{i_t}^T) h_t + e_{i_t} e_{i_t}^T \nabla F(x_t) \|^2 \right]
$$

$$
= \mathbb{E}_{i_t} \left[ \| (I - e_{i_t} e_{i_t}^T) h_t \|^2 \right] + \mathbb{E}_{i_t} \left[ \| e_{i_t} \odot (\nabla F(x_t)) \|^2 \right]
$$

$$
= \left( 1 - \frac{1}{d} \right) \mathbb{E}_{i_t} \left[ \| h_t \|^2 \right] + \frac{1}{d} \| \nabla F(x_t) \|^2
$$

$$
\leq \left( 1 - \frac{1}{d} \right) \sigma_t^2 + \frac{2L}{d} D_F(x_t, x^*)
$$

$$
\rho = \frac{1}{d}, \quad C = \frac{2L}{d}
$$
• GD uses $d$ gradient entries per iteration
SEGA Summary

- GD uses $d$ gradient entries per iteration
- SEGA uses 1 gradient entry per iteration
• GD uses $d$ gradient entries per iteration
• SEGA uses 1 gradient entry per iteration
• Equivalently, GD incurs $d \times \text{per iteration cost}$
SEGA Summary

- GD uses $d$ gradient entries per iteration
- SEGA uses 1 gradient entry per iteration
- Equivalently, GD incurs $d \times \text{per iteration cost}$
- Define oracle complexity $= d \times \text{number of gradients required to achieve } \epsilon\text{-accuracy}$

Algorithm

<table>
<thead>
<tr>
<th>Oracle Complexity</th>
<th>Per-iteration cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$d \times L \mu \times \log(\frac{1}{\epsilon})$</td>
</tr>
<tr>
<td>SEGA</td>
<td>$d \times L \mu \times \log(\frac{1}{\epsilon})$</td>
</tr>
</tbody>
</table>

SEGA is competitive with GD even while looking at one entry at a time!
SEGA Summary

- GD uses $d$ gradient entries per iteration
- SEGA uses 1 gradient entry per iteration
- Equivalently, GD incurs $d \times$ per iteration cost
- Define oracle complexity $= d \times$ number of gradients required to achieve $\epsilon$-accuracy

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Complexity</th>
<th>Per-iteration cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>$d \times \frac{L}{\mu} \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>$d$</td>
</tr>
<tr>
<td>SEGA</td>
<td>$d \times \frac{L}{\mu} \times \log \left( \frac{1}{\epsilon} \right)$</td>
<td>1</td>
</tr>
</tbody>
</table>

SEGA is competitive with GD even while looking at one entry at a time!
Outline

1. Context
2. Background
3. Vanilla Stochastic Gradient Descent: Large $N$
4. Variance-Reduced SGD: Moderate $N$
5. High-dimensional problems: large $d$
   - Gradient sketching
   - Hogwild!
6. Conclusion
Large $N$ and $d$

- Large $N$ $\Rightarrow$ cannot compute even one entry exactly
Large $N$ and $d$

- Large $N \Rightarrow$ cannot compute even one entry exactly
- Large $d \Rightarrow$ cannot compute full stochastic gradient
Large $N$ and $d$

- Large $N \Rightarrow$ cannot compute even one entry exactly
- Large $d \Rightarrow$ cannot compute full stochastic gradient
- Large-scale matrix completion
Large $N$ and $d$

- Large $N \Rightarrow$ cannot compute even one entry exactly
- Large $d \Rightarrow$ cannot compute full stochastic gradient
- Large-scale matrix completion
  - Observations $Z \in \mathbb{R}^{N_r \times N_c}$
  - $\min_{L, R} \|Z - LR^\top\|_F^2 + \frac{\mu}{2} \|L\|_F^2 + \frac{\mu}{2} \|R\|_F^2$

where $L \in \mathbb{R}^{N_r \times r}$, and $R \in \mathbb{R}^{N_c \times r}$
Large $N$ and $d$

- Large $N \Rightarrow$ cannot compute even one entry exactly
- Large $d \Rightarrow$ cannot compute full stochastic gradient
- Large-scale matrix completion
  - Observations $Z \in \mathbb{R}^{N_r \times N_c}$
    - \[
    \min_{L, R} \left\| Z - LR^\top \right\|_F^2 + \frac{\mu}{2} \left\| L \right\|_F^2 + \frac{\mu}{2} \left\| R \right\|_F^2
    \]
    where $L \in \mathbb{R}^{N_r \times r}$, and $R \in \mathbb{R}^{N_c \times r}$
  - Low-rank assumption $\Rightarrow r \ll N_c, N_r$
Large $N$ and $d$

• Large $N \Rightarrow$ cannot compute even one entry exactly
• Large $d \Rightarrow$ cannot compute full stochastic gradient
• Large-scale matrix completion
  • Observations $\mathbf{Z} \in \mathbb{R}^{N_r \times N_c}$
  
  $$\min_{\mathbf{L}, \mathbf{R}} \|\mathbf{Z} - \mathbf{LR}^\top\|_F^2 + \frac{\mu}{2} \|\mathbf{L}\|_F^2 + \frac{\mu}{2} \|\mathbf{R}\|_F^2$$

  where $\mathbf{L} \in \mathbb{R}^{N_r \times r}$, and $\mathbf{R} \in \mathbb{R}^{N_c \times r}$
• Low-rank assumption $\Rightarrow r \ll N_c, N_r$
• Number of observations $N = N_r N_c$ is extremely large
Large $N$ and $d$

- Large $N \Rightarrow$ cannot compute even one entry exactly
- Large $d \Rightarrow$ cannot compute full stochastic gradient
- Large-scale matrix completion
  - Observations $\mathbf{Z} \in \mathbb{R}^{N_r \times N_c}$
  
  $$\min_{\mathbf{L}, \mathbf{R}} \left\| \mathbf{Z} - \mathbf{L} \mathbf{R}^T \right\|_F^2 + \frac{\mu}{2} \left\| \mathbf{L} \right\|_F^2 + \frac{\mu}{2} \left\| \mathbf{R} \right\|_F^2$$
  
  where $\mathbf{L} \in \mathbb{R}^{N_r \times r}$, and $\mathbf{R} \in \mathbb{R}^{N_c \times r}$
  
  - Low-rank assumption $\Rightarrow r \ll N_c, N_r$
  
  - Number of observations $N = N_r N_c$ is extremely large
  
  - Number of variables $d = (N_c + N_r)r$ is also very large
Large $N$ and $d$

- Large $N \Rightarrow$ cannot compute even one entry exactly
- Large $d \Rightarrow$ cannot compute full stochastic gradient
- Large-scale matrix completion
  - Observations $Z \in \mathbb{R}^{N_r \times N_c}$
    
    $$\min_{\mathbf{L}, \mathbf{R}} \|Z - LR^\top\|_F^2 + \frac{\mu}{2} \|L\|_F^2 + \frac{\mu}{2} \|R\|_F^2$$

    where $\mathbf{L} \in \mathbb{R}^{N_r \times r}$, and $\mathbf{R} \in \mathbb{R}^{N_c \times r}$
  - Low-rank assumption $\Rightarrow r \ll N_c, N_r$
  - Number of observations $N = N_r N_c$ is extremely large
  - Number of variables $d = (N_c + N_r)r$ is also very large
- Cannot load the variables or observations into the RAM
• SGD is inherently serial
• SGD is inherently serial

• Consider system with \( m \) cores or \( m \) distributed servers

\[
\text{SGD achieves } \epsilon \text{ accuracy in } \mathcal{O}(\sigma^2 \epsilon) \text{ oracle calls}
\]

To use multi-core systems, one must parallelize, e.g., using minibatch

\[
m-\text{SGD} \quad x_{t+1} = x_t - \eta \sum_{j \in I_t} \nabla f(x_t, \xi_j)
\]

where \( m = |I_t| \) stochastic gradients are computed in parallel

What is the wall-clock time?
• SGD is inherently serial
• Consider system with $m$ cores or $m$ distributed servers
• SGD achieves $\epsilon$ accuracy in $O\left(\frac{\sigma^2}{\epsilon}\right)$ oracle calls

\[x_{t+1} = x_t - \eta \sum_{j \in I_t} \nabla f(x_t, \xi_j)\]
• SGD is inherently serial
• Consider system with $m$ cores or $m$ distributed servers
• SGD achieves $\epsilon$ accuracy in $\mathcal{O}\left(\frac{\sigma^2}{\epsilon}\right)$ oracle calls
• To use multi-core systems, one must parallelize, e.g., using minibatch

$$m\text{-SGD} \quad x_{t+1} = x_t - \frac{\eta}{m} \sum_{j \in \mathcal{I}_t} \nabla f(x_t, \xi_j)$$

where $m = |\mathcal{I}_t|$ stochastic gradients are computed in parallel
• SGD is inherently serial
• Consider system with $m$ cores or $m$ distributed servers
• SGD achieves $\epsilon$ accuracy in $O\left(\frac{\sigma^2}{\epsilon}\right)$ oracle calls
• To use multi-core systems, one must parallelize, e.g., using minibatch m-SGD

$$x_{t+1} = x_t - \frac{\eta}{m} \sum_{j \in I_t} \nabla f(x_t, \xi_j)$$

where $m = |I_t|$ stochastic gradients are computed in parallel
• What is the wall-clock time?
• Let $t_g =$ time to calculate $\nabla f(x, \xi_j)$ and $t_r =$ time to read/write $x_t$
• Let $t_g =$ time to calculate $\nabla f(x, \xi_j)$ and $t_r =$ time to read/write $x_t$
• If $t_r \ll t_g$, then
Curse of Parallelization: Wall Clock Time

- Let $t_g =$ time to calculate $\nabla f(x, \xi_j)$ and $t_r =$ time to read/write $x_t$
- If $t_r \ll t_g$, then
  
  SGD: Total wall-clock time $= t_g \times \sigma^2/\epsilon$
  
  m-SGD: Total wall-clock time $= t_g \times \sigma^2/m\epsilon \approx O(\sigma^2/\epsilon)$

• Gains due to parallelization offset by the limited memory throughput
• Synchronization requirement cause idling of cores
• Memory is locked while being written
Curse of Parallelization: Wall Clock Time

- Let $t_g = \text{time to calculate } \nabla f(x, \xi_j) \text{ and } t_r = \text{time to read/write } x_t$
- If $t_r \ll t_g$, then
  - SGD: Total wall-clock time $= t_g \times \sigma^2 / \epsilon$
  - m-SGD: Total wall-clock time $= t_g \times \sigma^2 / m\epsilon$

- Gains due to parallelization offset by the limited memory throughput
- Synchronization requirement cause idling of cores
- Memory is locked while being written
Curse of Parallelization: Wall Clock Time

- Let $t_g =$ time to calculate $\nabla f(x, \xi_j)$ and $t_r =$ time to read/write $x_t$

- If $t_r \ll t_g$, then
  
  SGD: Total wall-clock time $= t_g \times \sigma^2/\epsilon$
  
  m-SGD: Total wall-clock time $= t_g \times \sigma^2/m\epsilon$

- If $t_r \approx t_g$, writes are not concurrent
• Let $t_g =$ time to calculate $\nabla f(x, \xi_j)$ and $t_r =$ time to read/write $x_t$

• If $t_r \ll t_g$, then

SGD: Total wall-clock time $= t_g \times \sigma^2 / \epsilon$

m-SGD: Total wall-clock time $= t_g \times \sigma^2 / m\epsilon$

• If $t_r \approx t_g$, writes are not concurrent

SGD: Total wall-clock time $= (t_g + 2t_r) \times \sigma^2 / \epsilon \approx O(\sigma^2 / \epsilon)$
Curse of Parallelization: Wall Clock Time

- Let $t_g = \text{time to calculate } \nabla f(x, \xi_j)$ and $t_r = \text{time to read/write } x_t$
- If $t_r \ll t_g$, then
  - SGD: Total wall-clock time $= t_g \times \sigma^2/\epsilon$
  - m-SGD: Total wall-clock time $= t_g \times \sigma^2/m\epsilon$
- If $t_r \approx t_g$, writes are not concurrent
  - SGD: Total wall-clock time $= (t_g + 2t_r) \times \sigma^2/\epsilon \approx O(\sigma^2/\epsilon)$
  - m-SGD: Total wall-clock time $= (t_g + (m+1)t_r) \times \sigma^2/m\epsilon \approx O(\sigma^2/\epsilon)$
Curse of Parallelization: Wall Clock Time

- Let \( t_g = \) time to calculate \( \nabla f(x, \xi_j) \) and \( t_r = \) time to read/write \( x_t \)
- If \( t_r \ll t_g \), then
  \[ \text{SGD: Total wall-clock time} = t_g \times \sigma^2/\epsilon \]
  \[ \text{m-SGD: Total wall-clock time} = t_g \times \sigma^2/m\epsilon \]
- If \( t_r \approx t_g \), writes are not concurrent
  \[ \text{SGD: Total wall-clock time} = (t_g + 2t_r) \times \sigma^2/\epsilon \approx O(\sigma^2/\epsilon) \]
  \[ \text{m-SGD: Total wall-clock time} = (t_g + (m + 1)t_r) \times \sigma^2/m\epsilon \approx O(\sigma^2/\epsilon) \]
- Gains due to parallelization offset by the limited memory throughput
Curse of Parallelization: Wall Clock Time

- Let $t_g = \text{time to calculate } \nabla f(x, \xi_j)$ and $t_r = \text{time to read/write } x_t$
- If $t_r \ll t_g$, then
  
  SGD: Total wall-clock time $= t_g \times \sigma^2/\epsilon$
  
  m-SGD: Total wall-clock time $= t_g \times \sigma^2/m\epsilon$
- If $t_r \approx t_g$, writes are not concurrent
  
  SGD: Total wall-clock time $= (t_g + 2t_r) \times \sigma^2/\epsilon \approx O(\sigma^2/\epsilon)$
  
  m-SGD: Total wall-clock time $= (t_g + (m + 1)t_r) \times \sigma^2/m\epsilon \approx O(\sigma^2/\epsilon)$
- Gains due to parallelization offset by the limited memory throughput
- **Synchronization** requirement cause idling of cores
Curse of Parallelization: Wall Clock Time

- Let $t_g =$ time to calculate $\nabla f(x, \xi_j)$ and $t_r =$ time to read/write $x_t$
- If $t_r \ll t_g$, then
  - SGD: Total wall-clock time $= t_g \times \frac{\sigma^2}{\epsilon}$
  - m-SGD: Total wall-clock time $= t_g \times \frac{\sigma^2}{m\epsilon}$
- If $t_r \approx t_g$, writes are not concurrent
  - SGD: Total wall-clock time $= (t_g + 2t_r) \times \frac{\sigma^2}{\epsilon} \approx O(\sigma^2/\epsilon)$
  - m-SGD: Total wall-clock time $= (t_g + (m + 1)t_r) \times \frac{\sigma^2}{m\epsilon} \approx O(\sigma^2/\epsilon)$
- Gains due to parallelization offset by the limited memory throughput
- **Synchronization** requirement cause idling of cores
- Memory is **locked** while being written
• Consider the problem [Recht et al., 2011]

\[
\mathbf{x}^* = \arg \min_{\mathbf{x}} F(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}, \xi_i)
\]

where \( \xi_i \subseteq \{1, \ldots, n\} \) is an hyperedge
Sparse Problem Structure

- Consider the problem [Recht et al., 2011]

\[ x^* = \arg \min_x F(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i) \]

where \( \xi_i \subseteq \{1, \ldots, n\} \) is an hyperedge

- E.g., \( \xi_i = \{1, 3, 4\} \) and \( f(x, \xi_i) \) depends on \( x_1, x_3, x_4 \)
Sparse Problem Structure

- Consider the problem [Recht et al., 2011]

\[ x^* = \arg\min_x F(x) := \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i) \]

where \( \xi_i \subseteq \{1, \ldots, n\} \) is an hyperedge

- E.g., \( \xi_i = \{1, 3, 4\} \) and \( f(x, \xi_i) \) depends on \( x_1, x_3, x_4 \)

- Sparsity: \( |\xi_i| \ll d \)

**Figure 3:** (a) Bipartite graph (b) conflict graph representation
Sparse Problem Structure

• Consider the problem [Recht et al., 2011]

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} F(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}, \xi_i)$$

where $\xi_i \subseteq \{1, \ldots, n\}$ is a hyperedge

• E.g., $\xi_i = \{1, 3, 4\}$ and $f(\mathbf{x}, \xi_i)$ depends on $x_1, x_3, x_4$

• Sparsity: $|\xi_i| \ll d$

• Function $f : \mathbb{R}^n \times \mathcal{E} \rightarrow \mathbb{R}$ depends only on the subset of variables in $\xi_i$
Sparse Problem Structure

- Consider the problem [Recht et al., 2011]

$$x^* = \arg \min_x F(x) := \frac{1}{N} \sum_{i=1}^N f(x, \xi_i)$$

where $\xi_i \subseteq \{1, \ldots, n\}$ is an hyperedge

- E.g., $\xi_i = \{1, 3, 4\}$ and $f(x, \xi_i)$ depends on $x_1, x_3, x_4$

- Sparsity: $|\xi_i| \ll d$

- Function $f : \mathbb{R}^n \times \mathcal{E} \to \mathbb{R}$ depends only on the subset of variables in $\xi_i$

- So only a few entries of $\nabla f(x, \xi_i)$ are non-zero
• Consider the problem [Recht et al., 2011]

\[
\mathbf{x}^* = \arg \min_{\mathbf{x}} F(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}, \xi_i)
\]

where \(\xi_i \subseteq \{1, \ldots, n\}\) is an hyperedge

• E.g., \(\xi_i = \{1, 3, 4\}\) and \(f(\mathbf{x}, \xi_i)\) depends on \(x_1, x_3, x_4\)

• Sparsity: \(|\xi_i| \ll d\)

• Function \(f : \mathbb{R}^n \times \mathcal{E} \rightarrow \mathbb{R}\) depends only on the subset of variables in \(\xi_i\)

• So only a few entries of \(\nabla f(\mathbf{x}, \xi_i)\) are non-zero

• Indeed, \([\nabla f(\mathbf{x}, \xi_i)]_j = 0\) for all \(j \notin \xi_i\)
Go hog wild: read and write \(x\) without locking
Go hog wild: read and write $x$ without locking

Each core does the following:

without synchronizing with other cores
• Go hog wild: read and write $x$ without locking
• Each core does the following:
  • reads $x$ from the memory;

without synchronizing with other cores
Go hog wild: read and write $x$ without locking

Each core does the following:
- reads $x$ from the memory;
- evaluates $\nabla f(x, \xi)$;
- updates $x$;
- writes $x$ to memory one entry at a time without synchronizing with other cores

This will lead to inconsistent reads and overwrites: recipe for disaster?

Key idea: collisions rare if $\xi_i \cap \xi_j = \emptyset$ with high probability
Hogwild!

- Go hog wild: read and write $x$ without locking
- Each core does the following:
  - reads $x$ from the memory;
  - evaluates $\nabla f(x, \xi)$;
  - updates $x$; and

without synchronizing with other cores
• Go hog wild: read and write $x$ without locking

• Each core does the following:
  • reads $x$ from the memory;
  • evaluates $\nabla f(x, \xi)$;
  • updates $x$; and
  • writes $x$ to memory one entry at a time

without synchronizing with other cores
• Go hog wild: read and write $x$ without locking
• Each core does the following:
  • reads $x$ from the memory;
  • evaluates $\nabla f(x, \xi)$;
  • updates $x$; and
  • writes $x$ to memory one entry at a time without synchronizing with other cores
• This will lead to inconsistent reads and overwrites: recipe for disaster?
Hogwild!

• Go hog wild: read and write $x$ without locking

• Each core does the following:
  • reads $x$ from the memory;
  • evaluates $\nabla f(x, \xi)$;
  • updates $x$; and
  • writes $x$ to memory one entry at a time

  without synchronizing with other cores

• This will lead to inconsistent reads and overwrites: recipe for disaster?

• Key idea: collisions rare if $\xi_i \cap \xi_j = \emptyset$ with high probability
Hogwild Algorithm

• Define $[x]_\xi \in \mathbb{R}^{d \times 1}$ to contain only those entries that are in $\xi$, i.e.,

$$([x]_\xi)_j = \begin{cases} 0 & j \notin \xi \\ x_j & j \in \xi \end{cases}$$
Hogwild Algorithm

- Define \([x]_\xi \in \mathbb{R}^{d \times 1}\) to contain only those entries that are in \(\xi\), i.e.,

\[
([x]_\xi)_j = \begin{cases} 
0 & \text{if } j \notin \xi \\
x_j & \text{if } j \in \xi
\end{cases}
\]

- The full algorithm takes the form:
Hogwild Algorithm

- Define \([x]_\xi \in \mathbb{R}^{d \times 1}\) to contain only those entries that are in \(\xi\), i.e.,

\[
([x]_\xi)_j = \begin{cases} 
0 & j \notin \xi \\
x_j & j \in \xi 
\end{cases}
\]

- The full algorithm takes the form:

**Algorithm 3** Hogwild! (at each core, in parallel)

1: repeat
2: Sample an hyperedge \(\xi\)
3: Let \(\hat{x}_\xi\) = an inconsistent read of \([x]_\xi\)
4: Evaluate \([u]_\xi = -\eta \nabla f([\hat{x}]_\xi, \xi)\)
5: for \(v \in \xi\) do:
6: \(x_v \leftarrow x_v + u_v\)
7: end for
8: until number of edges \(\leq T\)
Perturbed SGD

• Cannot write Hogwild in classical SGD form

Lemma (Perturbed SGD: Strongly Convex + Smooth [Mania et al., 2017])

For $L$-smooth, $\mu$-convex functions $f$, perturbed SGD satisfies

$$
\delta_{t+1} \leq (1 - \eta \mu) \delta_t + \eta^2 \mathbb{E}[\|\nabla f(\hat{x}_t, \xi_t)\|^2] + 2\eta \mu \mathbb{E}[\|\hat{x}_t - x_t\|^2] + 2\eta \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(x_t, \xi_t) \rangle]
$$
Perturbed SGD

• Cannot write Hogwild in classical SGD form
• Instead consider perturbed SGD with some random variable $\xi_t$

$$x_{t+1} = x_t - \eta \nabla f(\hat{x}_t, \xi_t)$$

where $\hat{x}_t = x_t + n_t$ with noise $n_t$ independent of $\xi_t$

Lemma (Perturbed SGD: Strongly Convex + Smooth [Mania et al., 2017])

For $L$-smooth, $\mu$-convex functions $f$, perturbed SGD satisfies

$$\delta_{t+1} \leq (1 - \eta \mu)\delta_t + \eta^2 \mathbb{E}[\|\nabla f(\hat{x}_t, \xi_t)\|^2] + 2\eta \mu \mathbb{E}[\|\hat{x}_t - x_t\|^2] + 2\eta \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(x_t, \xi_t) \rangle]$$
Perturbed SGD

• Cannot write Hogwild in classical SGD form
• Instead consider perturbed SGD with some random variable $\xi_t$

$$x_{t+1} = x_t - \eta \nabla f(\hat{x}_t, \xi_t)$$

where $\hat{x}_t = x_t + n_t$ with noise $n_t$ independent of $\xi_t$

• Defining $\delta_t :=\mathbb{E}[\|x_t - x^*\|]$, then

Lemma (Perturbed SGD: Strongly Convex + Smooth [Mania et al., 2017])

For $L$-smooth, $\mu$-convex functions $f$, perturbed SGD satisfies

$$\delta_{t+1} \leq (1 - \eta \mu)\delta_t + \eta^2 \mathbb{E}[\|\nabla f(\hat{x}_t, \xi_t)\|^2] + 2\eta \mu \mathbb{E}[\|\hat{x}_t - x_t\|^2] + 2\eta \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(x_t, \xi_t) \rangle]$$
Lemma (Perturbed SGD: Strongly Convex + Smooth)

For $L$-smooth, $\mu$-convex functions $f$, perturbed SGD satisfies

$$\delta_{t+1} \leq (1 - \eta \mu)\delta_{t} + \eta^2 \mathbb{E}[\|\nabla f(\hat{x}_t, \xi_t)\|^2] + 2\eta \mu \mathbb{E}[\|\hat{x}_t - x_t\|^2] + 2\eta \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(x_t, \xi_t) \rangle]$$

Proof: Expand the optimality gap

$$\|x_{t+1} - x^*\|^2 = \|x_t - x^* - \eta \nabla f(\hat{x}_t, \xi_t)\|$$

$$= \|x_t - x^*\|^2 - 2\eta \langle \hat{x}_t - x^*, \nabla f(\hat{x}_t, \xi_t) \rangle + \eta^2 \|\nabla f(\hat{x}_t, \xi_t)\|^2 + 2\eta \langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle$$
Lemma (Perturbed SGD: Strongly Convex + Smooth)

For $L$-smooth, $\mu$-convex functions $f$, perturbed SGD satisfies

$$\delta_{t+1} \leq (1 - \eta \mu)\delta_t + \eta^2 \mathbb{E}[\|\nabla f(\hat{x}_t, \xi_t)\|^2] + 2\eta\mu \mathbb{E}[\|\hat{x}_t - x_t\|^2] + 2\eta \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(x_t, \xi_t) \rangle]$$

Proof: Expand the optimality gap and add-subtract $\langle \hat{x}_t, \nabla f(\hat{x}_t, \xi_t) \rangle$

$$\|x_{t+1} - x^*\|^2 = \|x_t - x^* - \eta \nabla f(\hat{x}_t, \xi_t)\|$$

$$= \|x_t - x^*\|^2 - 2\eta \langle \hat{x}_t - x^*, \nabla f(\hat{x}_t, \xi_t) \rangle + \eta^2 \|\nabla f(\hat{x}_t, \xi_t)\|^2 + 2\eta \langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle$$
Lemma (Perturbed SGD: Strongly Convex + Smooth)

For $L$-smooth, $\mu$-convex functions $f$, perturbed SGD satisfies

$$\delta_{t+1} \leq (1 - \eta \mu) \delta_t + \eta^2 \mathbb{E} \left[ \| \nabla f(\hat{x}_t, \xi_t) \|^2 \right] + 2\eta \mu \mathbb{E} \left[ \| \hat{x}_t - x_t \|^2 \right] + 2\eta \mathbb{E} \left[ \langle \hat{x}_t - x_t, \nabla f(x_t, \xi_t) \rangle \right]$$

Proof: Expand the optimality gap and add-subtract $\langle \hat{x}_t, \nabla f(\hat{x}_t, \xi_t) \rangle$

$$\| x_{t+1} - x^* \|^2 = \| x_t - x^* - \eta \nabla f(\hat{x}_t, \xi_t) \|$$

$$= \| x_t - x^* \|^2 - 2\eta \langle \hat{x}_t - x^*, \nabla f(\hat{x}_t, \xi_t) \rangle + \eta^2 \| \nabla f(\hat{x}_t, \xi_t) \|^2 + 2\eta \langle \hat{x}_t - x_t, \nabla (\hat{x}_t, \xi_t) \rangle$$

$$\mathbb{E}_t [\| x_{t+1} - x^* \|^2] = \| x_t - x^* \|^2 - 2\eta \langle \hat{x}_t - x^*, \nabla F(\hat{x}_t) \rangle + \eta^2 \| \nabla f(\hat{x}_t, \xi_t) \|^2$$

$$+ 2\eta \mathbb{E} \langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle$$
Lemma (Perturbed SGD: Strongly Convex + Smooth)

For $L$-smooth, $\mu$-convex functions $f$, perturbed SGD satisfies

$$
\delta_{t+1} \leq (1 - \eta \mu) \delta_t + \eta^2 \mathbb{E}[\|\nabla f(\hat{x}_t, \xi_t)\|^2] + 2\eta \mu \mathbb{E}[\|\hat{x}_t - x_t\|^2] + 2\eta \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(x_t, \xi_t) \rangle]
$$

Proof: Expand the optimality gap and add-subtract $\langle \hat{x}_t, \nabla f(\hat{x}_t, \xi_t) \rangle$

$$
\begin{align*}
\|x_{t+1} - x^*\|^2 &= \|x_t - x^* - \eta \nabla f(\hat{x}_t, \xi_t)\| \\
&= \|x_t - x^*\|^2 - 2\eta \langle \hat{x}_t - x^*, \nabla f(\hat{x}_t, \xi_t) \rangle + \eta^2 \|\nabla f(\hat{x}_t, \xi_t)\|^2 + 2\eta \langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle \\
\mathbb{E}_t[\|x_{t+1} - x^*\|^2] &= \|x_t - x^*\|^2 - 2\eta \langle \hat{x}_t - x^*, \nabla F(\hat{x}_t) \rangle + \eta^2 \|\nabla f(\hat{x}_t, \xi_t)\|^2 \\
&+ 2\eta \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle]
\end{align*}
$$

Lemma follows from using $\mu$-strong convexity and triangle inequality:

$$
\langle \hat{x}_t - x^*, \nabla F(\hat{x}_t) \rangle \geq \mu \|\hat{x}_t - x^*\|^2 \geq \frac{\mu}{2} \|x_t - x^*\|^2 - \mu \|\hat{x}_t - x_t\|^2
$$
Hogwild as Perturbed SGD

• Let $\xi_t$ be the $t$-th sampled hyperedge
Hogwild as Perturbed SGD

- Let $\xi_t$ be the $t$-th sampled hyperedge
- Let $\bar{x}_t$ be the contents before $t$-th read
Hogwild as Perturbed SGD

• Let $\xi_t$ be the $t$-th sampled hyperedge
• Let $\bar{x}_t$ be the contents before $t$-th read
• Also, recall that $[x]_{\xi_t}$ is an inconsistent read, and define full vector $\hat{x}_t$:

$$[\hat{x}_t]_v = \begin{cases} 
[\hat{x}_t]_v & v \in \xi_t \text{ – these are changed} \\
[\bar{x}_t]_v & v \notin \xi_t \text{ – these remain same as before the read}
\end{cases}$$
Hogwild as Perturbed SGD

- Let $\xi_t$ be the $t$-th sampled hyperedge
- Let $\bar{x}_t$ be the contents before $t$-th read
- Also, recall that $[x]_{\xi_t}$ is an inconsistent read, and define full vector $\hat{x}_t$:

$$
[\hat{x}_t]_v = \begin{cases} 
[\hat{x}_t]_v & v \in \xi_t \text{ – these are changed} \\
[\bar{x}_t]_v & v \notin \xi_t \text{ – these remain same as before the read}
\end{cases}
$$

- $\hat{x}_t$ independent of $\xi_t$ (can be relaxed)
Hogwild as Perturbed SGD

- Let $\xi_t$ be the $t$-th sampled hyperedge
- Let $\bar{x}_t$ be the contents before $t$-th read
- Also, recall that $[x]_{\xi_t}$ is an inconsistent read, and define full vector $\hat{x}_t$:

$$[\hat{x}_t]_v = \begin{cases} 
[\hat{x}_t]_v & v \in \xi_t \text{ - these are changed} \\
[\bar{x}_t]_v & v \notin \xi_t \text{ - these remain same as before the read} 
\end{cases}$$

- $\hat{x}_t$ independent of $\xi_t$ (can be relaxed)
- Bounded gradients: $\|f(\hat{x},\xi)\| \leq M$ (can be relaxed)
**Hogwild as Perturbed SGD**

- Let $\xi_t$ be the $t$-th sampled hyperedge
- Let $\bar{x}_t$ be the contents before $t$-th read
- Also, recall that $[x]_{\xi_t}$ is an inconsistent read, and define full vector $\hat{x}_t$:
  
  $$[\hat{x}_t]_v = \begin{cases} 
  [\hat{x}_t]_v & v \in \xi_t \text{ – these are changed} \\
  [\bar{x}_t]_v & v \notin \xi_t \text{ – these remain same as before the read} 
  \end{cases}$$

- $\hat{x}_t$ independent of $\xi_t$ (can be relaxed)
- Bounded gradients: $\|f(\hat{x}, \xi)\| \leq M$ (can be relaxed)
- Key idea: after $T$ updates are written to the memory:
  
  $$x_T = x_1 - \eta \nabla f(\hat{x}_1, \xi_1) - \eta \nabla f(\hat{x}_2, \xi_2) - \ldots - \eta \nabla f(\hat{x}_{T-1}, \xi_{T-1})$$
  
  or
  
  $$x_{t+1} = x_t - \eta \nabla f(\hat{x}_t, \xi_t)$$
Hogwild Abstractions: $\mathcal{T}$ and $\Delta$

- $\Delta = \text{average degree of conflict graph}$
Hogwild Abstractions: $\tau$ and $\Delta$

- $\Delta = \text{average degree of conflict graph}$
- Max. number of hyperedges that overlap with a given hyperedge $= \tau$
Hogwild Abstractions: $\tau$ and $\Delta$

- $\Delta = \text{average degree of conflict graph}$
- Max. number of hyperedges that overlap with a given hyperedge $= \tau$
- $\tau = 0$ implies no overlap (classical SGD)
Hogwild Abstractions: $\Delta$ and $\tau$

- $\Delta =$ average degree of conflict graph
- Max. number of hyperedges that overlap with a given hyperedge $= \tau$
- $\tau = 0$ implies no overlap (classical SGD)
- $\tau$ can be proxy for number of cores: $\tau$ read-writes in parallel
Hogwild Abstractions: $\tau$ and $\Delta$

- $\Delta =$ average degree of conflict graph
- Max. number of hyperedges that overlap with a given hyperedge $=$ $\tau$
- $\tau = 0$ implies no overlap (classical SGD)
- $\tau$ can be proxy for number of cores: $\tau$ read-writes in parallel
- Consider, for instance, times $i$ and $j$:
Hogwild Abstractions: $\tau$ and $\Delta$

- $\Delta =$ average degree of conflict graph
- Max. number of hyperedges that overlap with a given hyperedge $= \tau$
- $\tau = 0$ implies no overlap (classical SGD)
- $\tau$ can be proxy for number of cores: $\tau$ read-writes in parallel
- Consider, for instance, times $i$ and $j$:
  - if $i < j$ and $\xi_i \cap \xi_j = \emptyset$, $\nabla f(\hat{x}_i, \xi_i)$ written before $\hat{x}_j$ read: contribution of $\nabla f(\hat{x}_i, \xi_i)$ included into $\hat{x}_j$ and $x_j$
Hogwild Abstractions: $\tau$ and $\Delta$

- $\Delta =$ average degree of conflict graph
- Max. number of hyperedges that overlap with a given hyperedge $= \tau$
- $\tau = 0$ implies no overlap (classical SGD)
- $\tau$ can be proxy for number of cores: $\tau$ read-writes in parallel
- Consider, for instance, times $i$ and $j$:
  - if $i < j$ and $\xi_i \cap \xi_j = \emptyset$, $\nabla f(\hat{x}_i, \xi_i)$ written before $\hat{x}_j$ read: contribution of $\nabla f(\hat{x}_i, \xi_i)$ included into $\hat{x}_j$ and $x_j$
  - If $i > j$ and $\xi_i \cap \xi_j = \emptyset$, then neither $\hat{x}_j$ nor $x_j$ contain any contribution of $\nabla f(\hat{x}_i, \xi_i)$
Hogwild Abstractions: $\tau$ and $\Delta$

- $\Delta =$ average degree of conflict graph
- Max. number of hyperedges that overlap with a given hyperedge $= \tau$
- $\tau = 0$ implies no overlap (classical SGD)
- $\tau$ can be proxy for number of cores: $\tau$ read-writes in parallel

Consider, for instance, times $i$ and $j$:

- If $i < j$ and $\xi_i \cap \xi_j = \emptyset$, $\nabla f(\hat{x}_i, \xi_i)$ written before $\hat{x}_j$ read: contribution of $\nabla f(\hat{x}_i, \xi_i)$ included into $\hat{x}_j$ and $x_j$
- If $i > j$ and $\xi_i \cap \xi_j = \emptyset$, then neither $\hat{x}_j$ nor $x_j$ contain any contribution of $\nabla f(\hat{x}_i, \xi_i)$

- Edges $\xi_i \cap \xi_j = \emptyset$ if $|i - j| > \tau$
Let $S^t_l$ be diagonal matrix with entries in $\{-1, 0, 1\}$

Define conflicting edges: $\mathcal{I}_t := \{t - \tau, t - \tau + 1, \ldots t - 1, t + 1, \ldots, t + \tau\}$

Then, all possible update orders can be written as

$$\hat{x}_t - x_t = \eta \sum_{i \in \mathcal{I}_t} S^t_l \nabla f(\hat{x}_l, \xi_l)$$

Models all patterns of possibly partial updates while $\xi_t$ is being processed
Hogwild Analysis

Lemma

The following bounds hold:

\[ \mathbb{E}[\|\hat{x}_t - x_t\|^2] \leq \eta^2 M \left( 2\tau + 8\tau^2 \frac{\Delta}{d} \right) \]

\[ \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, e_t) \rangle] \leq 4\eta M^2 \tau \frac{\Delta}{d} \]

We use \( \|\nabla f(\hat{x}_t, \xi_t)\| \leq M \)

\[ \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle] = \eta \sum_{t \in I_t} \mathbb{E}[\langle S_t \nabla f(\hat{x}_t, \xi_t), \nabla f(\hat{x}_t, \xi_t) \rangle] \]
Lemma

The following bounds hold:

\[ \mathbb{E}[\|\hat{x}_t - x_t\|^2] \leq \eta^2 M \left( 2\tau + 8\tau^2 \frac{\Delta}{d} \right) \]

\[ \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, e_t) \rangle] \leq 4\eta M^2 \tau \frac{\Delta}{d} \]

We use \( \|\nabla f(\hat{x}_t, \xi_t)\| \leq M \)

\[ \mathbb{E}[\langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle] = \eta \sum_{i \in \mathcal{I}_t} \mathbb{E}[\langle S_i^t \nabla f(\hat{x}_t, \xi_t), \nabla f(\hat{x}_t, \xi_t) \rangle] \]

\[ \leq \eta M^2 \sum_{i} \Pr [\xi_i \cap \xi_t \neq \emptyset] \]
Hogwild Analysis

Lemma

The following bounds hold:

\[ E[\|\hat{x}_t - x_t\|^2] \leq \eta^2 M \left( 2\tau + 8\tau^2 \frac{\Delta}{d} \right) \]

\[ E[\langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, e_t) \rangle] \leq 4\eta M^2 \tau \frac{\Delta}{d} \]

We use \( \|\nabla f(\hat{x}_t, \xi_t)\| \leq M \) and \( \Pr(\xi_t \cap \xi_t \neq \emptyset) = \frac{2\Delta}{d} \)

\[ E[\langle \hat{x}_t - x_t, \nabla f(\hat{x}_t, \xi_t) \rangle] = \eta \sum_{i \in I_t} E[\langle \mathbf{S}_i^t \nabla f(\hat{x}_i, \xi_i), \nabla f(\hat{x}_t, \xi_t) \rangle] \]

\[ \leq \eta M^2 \sum_{i} \Pr[\xi_t \cap \xi_t \neq \emptyset] \]

\[ \leq 2\eta M^2 \tau \frac{2\Delta}{d} \]
Hogwild Analysis

Since $\|Su\|_2 \leq \|u\|$, it holds that

$$E[\|\hat{x}_t - x_t\|^2] = \eta^2 E[\|\sum_{t \in I_t} S_t \nabla f(\hat{x}_t, \xi_t)\|^2]$$
Since $\|S\u\|_2 \leq \|\u\|$, it holds that

$$
\mathbb{E}[\|\hat{x}_t - x_t\|^2] = \eta^2 \mathbb{E}[\| \sum_{i \in I_t} S_i^t \nabla f(\hat{x}_i, \xi_i) \|^2]
$$

$$
= \eta^2 \sum_{i \in I_t} \mathbb{E} \| S_i^t \nabla f(\hat{x}_i, \xi_i) \|^2 + \eta^2 \sum_{i \neq \kappa} \mathbb{E}[\langle S_i^t \nabla f(\hat{x}_i, \xi_i), S_{\kappa}^t \nabla f(\hat{x}_{\kappa}, \xi_{\kappa}) \rangle]
$$
Since $\|Su\|_2 \leq \|u\|$, it holds that

$$
\mathbb{E}[\|\hat{x}_t - x_t\|^2] = \eta^2 \mathbb{E}[\| \sum_{i \in I_t} S_t \nabla f(\hat{x}_i, \xi_i) \|^2] \\
= \eta^2 \sum_{i \in I_t} \mathbb{E} \| S_t \nabla f(\hat{x}_i, \xi_i) \|^2 + \eta^2 \sum_{i \neq \kappa} \mathbb{E}[\langle S_t \nabla f(\hat{x}_i, \xi_i), S_{\kappa} \nabla f(\hat{x}_\kappa, \xi_\kappa) \rangle] \\
\leq \eta^2 \sum_{i} \mathbb{E} \| \nabla f(\hat{x}_i, \xi_i) \|^2 + \eta^2 \sum_{i \neq \kappa} \mathbb{E}[\| \nabla f(\hat{x}_i, \xi_i) \| \| \nabla f(\hat{x}_\kappa, \xi_\kappa) \| \mathbf{1}_{\xi_i \cap \xi_\kappa \neq \emptyset}] 
$$
Hogwild Analysis

Since \( \| S u \|_2 \leq \| u \| \), it holds that

\[
E[\| \hat{x}_t - x_t \|^2] = \eta^2 E[\| \sum_{i \in I_t} S_t^i \nabla f(\hat{x}_i, \xi_i) \|^2]
\]

\[
= \eta^2 \sum_{i \in I_t} E \| S_t^i \nabla f(\hat{x}_i, \xi_i) \|^2 + \eta^2 \sum_{i \neq \kappa} E[\langle S_t^i \nabla f(\hat{x}_i, \xi_i), S_t^\kappa \nabla f(\hat{x}_\kappa, \xi_\kappa) \rangle]
\]

\[
\leq \eta^2 \sum_{i} E \| \nabla f(\hat{x}_i, \xi_i) \|^2 + \eta^2 \sum_{i \neq \kappa} E[\| \nabla f(\hat{x}_i, \xi_i) \| \| \nabla f(\hat{x}_\kappa, \xi_\kappa) \| 1_{\xi_i \cap \xi_\kappa \neq \emptyset}]
\]

\[
\leq \eta^2 M^2 (2\tau + 4\tau^2 \Pr[\xi_i \cap \xi_\kappa \neq \emptyset]) = 2\eta^2 M^2 \tau (1 + 2\tau (2\Delta / d))
\]
Hogwild Analysis

Since $\| Su \|_2 \leq \| u \|$, it holds that

$$\mathbb{E}[\| \hat{x}_t - x_t \|^2] = \eta^2 \mathbb{E}[\| \sum_{i \in I_t} S^t_i \nabla f(\hat{x}_i, \xi_i) \|^2]$$

$$= \eta^2 \sum_{i \in I_t} \mathbb{E} \| S^t_i \nabla f(\hat{x}_i, \xi_i) \|^2 + \eta^2 \sum_{i \neq \kappa} \mathbb{E}[\langle S^t_i \nabla f(\hat{x}_i, \xi_i), S^t_{\kappa} \nabla f(\hat{x}_{\kappa}, \xi_{\kappa}) \rangle]$$

$$\leq \eta^2 \sum_{i} \mathbb{E} \| \nabla f(\hat{x}_i, \xi_i) \|^2 + \eta^2 \sum_{i \neq \kappa} \mathbb{E}[\| \nabla f(\hat{x}_i, \xi_i) \| \| \nabla f(\hat{x}_{\kappa}, \xi_{\kappa}) \| 1_{\xi_i \cap \xi_{\kappa} \neq \emptyset}]$$

$$\leq \eta^2 M^2 (2\tau + 4\tau^2 \Pr[\xi_i \cap \xi_{\kappa} \neq \emptyset]) = 2\eta^2 M^2 \tau (1 + 2\tau(2\Delta/d))$$

Substituting all bounds,

$$\delta_{t+1} \leq (1 - \eta \mu) \delta_t + \eta^2 M^2 C_1$$

where $C_1 = 1 + 8\tau \Delta/d + 4\eta \mu \tau + 16\eta \mu \tau^2 \Delta/d$. 

Yields $O(L \mu \epsilon)$ oracle complexity (same as SGD) provided $\tau$ is not too large.
Hogwild Analysis

Since $\|S_u\|_2 \leq \|u\|$, it holds that

$$E[\|\hat{x}_t - x_t\|^2] = \eta^2 E[\| \sum_{i \in \mathcal{I}_t} S_i^t \nabla f(\hat{x}_i, \xi_i)\|^2]$$

$$= \eta^2 \sum_{i \in \mathcal{I}_t} E[\|S_i^t \nabla f(\hat{x}_i, \xi_i)\|^2] + \eta^2 \sum_{i \neq \kappa} E[\langle S_i^t \nabla f(\hat{x}_i, \xi_i), S_k^t \nabla f(\hat{x}_\kappa, \xi_\kappa) \rangle]$$

$$\leq \eta^2 \sum_{i} E[\|\nabla f(\hat{x}_i, \xi_i)\|^2] + \eta^2 \sum_{i \neq \kappa} E[\|\nabla f(\hat{x}_i, \xi_i)\| \|\nabla f(\hat{x}_\kappa, \xi_\kappa)\| 1_{\xi_i \cap \xi_\kappa \neq \emptyset}]$$

$$\leq \eta^2 M^2(2\tau + 4\tau^2 \Pr[\xi_i \cap \xi_\kappa \neq \emptyset]) = 2\eta^2 M^2\tau(1 + 2\tau(2\Delta/d))$$

Substituting all bounds,

$$\delta_{t+1} \leq (1 - \eta \mu)\delta_t + \eta^2 M^2 C_1$$

where $C_1 = 1 + 8\tau\Delta/d + 4\eta \mu \tau + 16\eta \mu \tau^2 \Delta/d$.

Yields $O\left(\frac{L}{\mu \epsilon}\right)$ oracle complexity (same as SGD) provided $\tau$ is not too large.
Asynchronous SVRG [Mania et al., 2017] is the variance-reduced version of Hogwild!
State-of-the-art for high-dimensional

- Asynchronous SVRG [Mania et al., 2017] is the variance-reduced version of Hogwild!
- Extensions to non-convex settings with more realistic assumptions [Cannelli et al., 2019]
State-of-the-art for high-dimensional

- Asynchronous SVRG [Mania et al., 2017] is the variance-reduced version of Hogwild!
- Extensions to non-convex settings with more realistic assumptions [Cannelli et al., 2019]
- Very large delays [Zhou et al., 2018]
State-of-the-art for high-dimensional

- Asynchronous SVRG [Mania et al., 2017] is the variance-reduced version of Hogwild!
- Extensions to non-convex settings with more realistic assumptions [Cannelli et al., 2019]
- Very large delays [Zhou et al., 2018]
- Proximal variants [Zhu et al., 2018]
State-of-the-art for high-dimensional

- Asynchronous SVRG [Mania et al., 2017] is the variance-reduced version of Hogwild!
- Extensions to non-convex settings with more realistic assumptions [Cannelli et al., 2019]
- Very large delays [Zhou et al., 2018]
- Proximal variants [Zhu et al., 2018]
- Decentralized variants? Skewed sparsity profile?
Conclusion
• Oracle complexity results for different SGD variants
• Intuition regarding variance reduction and coordinate descent
• When to apply which version?
• Unified and simplified proofs (extend to non-strongly convex settings also)
• State-of-the-art and open problems

*Katyusha: The first direct acceleration of stochastic gradient methods.*


*First-order methods in optimization, volume 25.*

SIAM.


*Optimization methods for large-scale machine learning.*


**Sebastien Bubeck’s blog: I’m a bandit.**


**Convex optimization: Algorithms and complexity.**


**Asynchronous parallel algorithms for nonconvex optimization.**
*Mathematical Programming*, pages 1–34.

Notes on large scale optimization for data science.


Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator.


A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent.
**SEGA: Variance reduction via gradient sketching.**

**Accelerating stochastic gradient descent using predictive variance reduction.**

**Mini-batch semi-stochastic gradient descent in the proximal setting.**
Kovalev, D., Horváth, S., and Richtárik, P. (2019). *Don’t jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop.* 


*In Advances in neural information processing systems, pages 3384–3392.*


**Notes on first-order methods for minimizing smooth functions.**  

**Improving the sample and communication complexity for decentralized non-convex optimization: A joint gradient estimation and tracking approach.**  

**Optimization methods for large-scale systems.**  

