Stochastic Variance-Reduced Optimization for Machine Learning
Parts 2: Weakening the Assumptions

Presenters: Francis Bach and Mark Schmidt

2017 SIAM Conference on Optimization

May 23, 2017
Linear of Convergence of Gradient-Based Methods

- We’ve seen a variety of results of the form:

\[
\text{Smoothness } + \text{ Strong-Convexity } \implies \text{Linear Convergence}
\]

- Error on iteration \( t \) is \( O(\rho^t) \), or we need \( O(\log(1/\epsilon)) \) iterations.
Linear of Convergence of Gradient-Based Methods

- We’ve seen a variety of results of the form:

\[ \text{Smoothness} + \text{Strong-Convexity} \Rightarrow \text{Linear Convergence} \]

- Error on iteration \( t \) is \( O(\rho^t) \), or we need \( O(\log(1/\epsilon)) \) iterations.

- But even simple models are often not strongly-convex.
  - Least squares, logistic regression, SVMs with bias, etc.
Linear of Convergence of Gradient-Based Methods

- We’ve seen a variety of results of the form:

  \[ \text{Smoothness} + \text{Strong-Convexity} \implies \text{Linear Convergence} \]

  \[ \text{Error on iteration } t \text{ is } O(\rho^t), \text{ or we need } O(\log(1/\epsilon)) \text{ iterations.} \]

- But even simple models are often not strongly-convex.
  - Least squares, logistic regression, SVMs with bias, etc.

- How much can we relax strong-convexity?

  \[ \text{Smoothness} + \text{???} \implies \text{Linear Convergence} \]
Polyak-Łojasiewicz (PL) Inequality

For example, in 1963 Polyak showed linear convergence of GD only assuming

$$\frac{1}{2} \| \nabla f(x) \|^2 \geq \mu (f(x) - f^*),$$

that gradient grows as quadratic function of sub-optimality.
Polyak-Łojasiewicz (PL) Inequality

- For example, in 1963 Polyak showed linear convergence of GD only assuming

$$\frac{1}{2} \| \nabla f(x) \|^2 \geq \mu (f(x) - f^*),$$

that gradient grows as quadratic function of sub-optimality.
- Holds for SC problems, but also problems of the form

$$f(x) = g(Ax), \quad \text{for strongly-convex } g.$$
- Includes least squares, logistic regression (on compact set), etc.
Polyak-Łojasiewicz (PL) Inequality

- For example, in 1963 Polyak showed linear convergence of GD only assuming

\[
\frac{1}{2} \| \nabla f(x) \|^2 \geq \mu (f(x) - f^*),
\]

that gradient grows as quadratic function of sub-optimality.
- Holds for SC problems, but also problems of the form

\[ f(x) = g(Ax), \quad \text{for strongly-convex } g. \]

- Includes least squares, logistic regression (on compact set), etc.
- A special case of the Łojasiewicz' inequality [1963].
  - We’ll call this the Polyak-Łojasiewicz (PL) inequality.
Polyak-Łojasiewicz (PL) Inequality

- For example, in 1963 Polyak showed linear convergence of GD only assuming
\[
\frac{1}{2} \|\nabla f(x)\|^2 \geq \mu (f(x) - f^*),
\]
that gradient grows as quadratic function of sub-optimality.
- Holds for SC problems, but also problems of the form
\[f(x) = g(Ax),\text{ for strongly-convex } g.\]
- Includes least squares, logistic regression (on compact set), etc.
- A special case of the Łojasiewicz' inequality [1963].
  - We’ll call this the Polyak-Łojasiewicz (PL) inequality.
- Using the PL inequality we can show

\[
\text{Smoothness } + \text{ PL Inequality } \Rightarrow \text{ Linear Convergence}
\]
PL Inequality and Invexity

- PL inequality doesn’t require uniqueness or convexity.
PL Inequality and Invexity

- PL inequality doesn’t require uniqueness or convexity.
- However, it implies invexity.
  - For smooth $f$, invexity $\iff$ all stationary points are global optimum.
PL Inequality and Invexity

- PL inequality doesn’t require uniqueness or convexity.
- However, it implies invexity.
  - For smooth $f$, invexity $\iff$ all stationary points are global optimum.
- Example of invex but non-convex function satisfying PL:

$$f(x) = x^2 + 3\sin^2(x).$$

- Gradient descent converges linearly on this non-convex problem.
Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
  - QG: quadratic growth [Anitescu, 2000]
Non-Convex  Non-IID  Non-Stochastic  Non-Serial

Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
  - QG: quadratic growth [Anitescu, 2000]
  - ESC: essential strong convexity [Liu et al., 2013].
Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
  - QG: quadratic growth [Anitescu, 2000]
  - ESC: essential strong convexity [Liu et al., 2013].
  - RSI: restricted secant inequality [Zhang & Yin, 2013].
    - RSI plus convexity is “restricted strong-convexity”.
Non-Convex Non-IID Non-Stochastic Non-Serial

Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
  - QG: quadratic growth [Anitescu, 2000]
  - ESC: essential strong convexity [Liu et al., 2013].
  - RSI: restricted secant inequality [Zhang & Yin, 2013].
    - RSI plus convexity is “restricted strong-convexity”.
  - Semi-strong convexity [Gong & Ye, 2014].
    - Equivalent to QG plus convexity.
Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
  - QG: quadratic growth [Anitescu, 2000]
  - ESC: essential strong convexity [Liu et al., 2013].
  - RSI: restricted secant inequality [Zhang & Yin, 2013].
    - RSI plus convexity is “restricted strong-convexity”.
  - Semi-strong convexity [Gong & Ye, 2014].
    - Equivalent to QG plus convexity.
  - Optimal strong convexity [Liu & Wright, 2015].
    - Equivalent to QG plus convexity.
How does PL inequality [1963] relate to more recent conditions?

- **EB**: error bounds [Luo and Tseng, 1993].
- **QG**: quadratic growth [Anitescu, 2000]
- **ESC**: essential strong convexity [Liu et al., 2013].
- **RSI**: restricted secant inequality [Zhang & Yin, 2013].
  - RSI plus convexity is “restricted strong-convexity”.
- **Semi-strong convexity** [Gong & Ye, 2014].
  - Equivalent to QG plus convexity.
- **Optimal strong convexity** [Liu & Wright, 2015].
  - Equivalent to QG plus convexity.
- **WSC**: weak strong convexity [Necoara et al., 2015].
Weaker Conditions than Strong Convexity (SC)

- How does PL inequality [1963] relate to more recent conditions?
  - QG: quadratic growth [Anitescu, 2000]
  - ESC: essential strong convexity [Liu et al., 2013].
  - RSI: restricted secant inequality [Zhang & Yin, 2013].
    - RSI plus convexity is “restricted strong-convexity”.
  - Semi-strong convexity [Gong & Ye, 2014].
    - Equivalent to QG plus convexity.
  - Optimal strong convexity [Liu & Wright, 2015].
    - Equivalent to QG plus convexity.
  - WSC: weak strong convexity [Necoara et al., 2015].

- Proofs are often more complicated under these conditions.
- Are they more general?
For a function $f$ with a Lipschitz-continuous gradient, we have:

$$(SC) \rightarrow (ESC) \rightarrow (WSC) \rightarrow (RSI) \rightarrow (EB) \equiv (PL) \rightarrow (QG).$$

$QG$ is the weakest condition but allows non-global local minima. $PL \equiv EB$ are most general conditions giving global min.
For a function $f$ with a Lipschitz-continuous gradient, we have:

$$(SC) \rightarrow (ESC) \rightarrow (WSC) \rightarrow (RSI) \rightarrow (EB) \equiv (PL) \rightarrow (QG).$$

If we further assume that $f$ is convex, then

$$(RSI) \equiv (EB) \equiv (PL) \equiv (QG).$$
For a function $f$ with a Lipschitz-continuous gradient, we have:

$$(SC) \rightarrow (ESC) \rightarrow (WSC) \rightarrow (RSI) \rightarrow (EB) \equiv (PL) \rightarrow (QG).$$

If we further assume that $f$ is convex, then

$$(RSI) \equiv (EB) \equiv (PL) \equiv (QG).$$

- QG is the weakest condition but allows non-global local minima.
For a function $f$ with a Lipschitz-continuous gradient, we have:

$$ (SC) \to (ESC) \to (WSC) \to (RSI) \to (EB) \equiv (PL) \to (QG). $$

If we further assume that $f$ is convex, then

$$ (RSI) \equiv (EB) \equiv (PL) \equiv (QG). $$

- QG is the weakest condition but allows non-global local minima.
- PL $\equiv$ EB are most general conditions giving global min.
Convergence of Huge-Scale Methods

- For large datasets, we typically don’t use GD.
  - But the PL inequality can be used to analyze other algorithms.
Convergence of Huge-Scale Methods

- For large datasets, we typically don’t use GD.
  - But the PL inequality can be used to analyze other algorithms.

- It has now been used to analyze:
  - **Classic stochastic gradient** methods [Karimi et al., 2016]:
    - \(O(1/k)\) without strong-convexity using basic method.
  - Coordinate descent methods [Karimi et al., 2016].
  - Frank-Wolfe [Garber & Hazan, 2015].
For large datasets, we typically don’t use GD. But the PL inequality can be used to analyze other algorithms.

It has now been used to analyze:
- **Classic stochastic gradient** methods [Karimi et al., 2016]:
  - $O(1/k)$ without strong-convexity using basic method.
- Coordinate descent methods [Karimi et al., 2016].
- Frank-Wolfe [Garber & Hazan, 2015].
- **Variance-reduced stochastic gradient** (like SAGA and SVRG) [Reddi et al., 2016].
  - Linear convergence without strong-convexity.
Relevant Problems for Proximal-PL

- **Proximal-PL** is a generalization for non-smooth composite problems.
  - Reddi et al. [2016] analyze proximal-SVRG and proximal-SAGA.

- Proximal-PL is satisfied when:
  - $f$ is strongly-convex.
  - $f$ satisfies PL and $g$ is constant.
  - $f = h(Ax)$ for strongly-convex $h$ and $g$ is indicator of polyhedral set.
Relevant Problems for Proximal-PL

- **Proximal-PL** is a generalization for non-smooth composite problems.
  - Reddi et al. [2016] analyze proximal-SVRG and proximal-SAGA.

- Proximal-PL is satisfied when:
  - $f$ is strongly-convex.
  - $f$ satisfies PL and $g$ is constant.
  - $f = h(Ax)$ for strongly-convex $h$ and $g$ is indicator of polyhedral set.
  - $F$ is convex and satisfies QG (SVM and LASSO)
  - Any problem satisfying KL inequality or error bounds (equivalent to these).
    - Group L1-regularization, nuclear-norm regularization.

- Another important problem class: principal component analysis (PCA)
  - Non-convex and doesn’t satisfy PL, but we can find global optimum.
  - But it satisfies PL on Riemannian manifold [Zhang et al., 2016].
  - New faster method based on SVRG [Shamir, 2015, Garber & Hazan, 2016].
Relevant Problems for Proximal-PL

- **Proximal-PL** is a generalization for non-smooth composite problems.
  - Reddi et al. [2016] analyze **proximal-SVRG** and **proximal-SAGA**.

- Proximal-PL is satisfied when:
  - $f$ is strongly-convex.
  - $f$ satisfies PL and $g$ is constant.
  - $f = h(Ax)$ for strongly-convex $h$ and $g$ is indicator of polyhedral set.
  - $F$ is convex and satisfies QG (SVM and LASSO)
  - Any problem satisfying KL inequality or error bounds (equivalent to these).
    - Group L1-regularization, nuclear-norm regularization.

- Another important problem class: **principal component analysis** (PCA)
  - Non-convex and doesn’t satisfy PL, but we can find global optimum.
Relevant Problems for Proximal-PL

- **Proximal-PL** is a generalization for non-smooth composite problems.
  - Reddi et al. [2016] analyze *proximal-SVRG* and *proximal-SAGA*.

Proximal-PL is satisfied when:
- $f$ is strongly-convex.
- $f$ satisfies PL and $g$ is constant.
- $f = h(Ax)$ for strongly-convex $h$ and $g$ is indicator of polyhedral set.
- $F$ is convex and satisfies QG (SVM and LASSO)
- Any problem satisfying KL inequality or error bounds (equivalent to these).
  - Group L1-regularization, nuclear-norm regularization.

Another important problem class: **principal component analysis** (PCA)
- Non-convex and doesn’t satisfy PL, but we can find global optimum.
- But it satisfies PL on Riemannian manifold [Zhang et al., 2016].
- New faster method based on SVRG [Shamir, 2015, Garber & Hazan, 2016].
• But can we say anything about **general non-convex** functions?

• What if all we know is $\nabla f$ is Lipschitz and $f$ is bounded below?
Non-Convex Non-IID Non-Stochastic Non-Serial

Non-Convex Rates for Gradient Descent

- For strongly-convex functions, GD satisfies
  \[ \| x_t - x^* \|^2 = O(\rho^t). \]

- For convex functions, for GD still satisfies
  \[ f(x^t) - f(x^*) = O(1/t). \]
Non-Convex Rates for Gradient Descent

- For **strongly-convex** functions, GD satisfies
  \[ \|x_t - x^*\|^2 = O(\rho^t). \]

- For **convex** functions, for GD still satisfies
  \[ f(x^t) - f(x^*) = O(1/t). \]

- For **non-convex and bounded below** functions, GD still satisfies
  \[ \min_{k \leq t} \|\nabla f(x^k)\|^2 = O(1/t), \]

  a convergence rate in terms of getting to a critical point [Nesterov, 2003].
Non-Convex Rates for Stochastic Gradient

- For stochastic gradient methods, Ghadimi & Lan [2013] show a similar result,

\[ \mathbb{E}[\|\nabla f(x^k)\|^2] = O(1/\sqrt{t}), \]

for a randomly-chosen \( k \leq t \).
Non-Convex Rates for Stochastic Gradient

- For stochastic gradient methods, Ghadimi & Lan [2013] show a similar result,

\[ \mathbb{E}[\|\nabla f(x^k)\|^2] = O(1/\sqrt{t}), \]

for a randomly-chosen \( k \leq t \).

- For variance-reduced methods, Reddi et al. [2016] show we get faster rate,

\[ \mathbb{E}[\|\nabla f(x^k)\|^2] = O(1/t), \]

for a randomly-chosen \( k \leq t \).
Non-Convex Non-IID Non-Stochastic Non-Serial

Non-Convex Rates for Stochastic Gradient

CIFAR10 dataset; 2-layer NN
Non-Convex Rates for Stochastic Gradient

- Number of gradient evaluations to guarantee $\epsilon$-close to critical:
  - Gradient descent $O(n/\epsilon)$
  - Stochastic gradient $O(1/\epsilon^2)$

We have analogous results for variance-reduced proximal + stochastic methods.

We cannot show analogous results for classic proximal stochastic methods.

Open problem that needs to be resolved: are analogous results possible?
Non-Convex Rates for Stochastic Gradient

- Number of gradient evaluations to guarantee $\epsilon$-close to critical:
  - Gradient descent: $O(n/\epsilon)$
  - Stochastic gradient: $O(1/\epsilon^2)$
  - Variance-reduced: $O(n + n^{2/3}/\epsilon)$

We have analogous results for variance-reduced proximal+stochastic methods.

[Reddi et al., 2016]

We cannot show analogous results for classic proximal stochastic methods. All existing proximal+stochastic results require noise to go to zero. Open problem that needs to be resolved: are analogous results possible?
Non-Convex Non-IID Non-Stochastic Non-Serial

Non-Convex Rates for Stochastic Gradient

- Number of gradient evaluations to guarantee $\epsilon$-close to critical:
  - Gradient descent $O(n/\epsilon)$
  - Stochastic gradient $O(1/\epsilon^2)$
  - Variance-reduced $O(n + n^{2/3}/\epsilon)$

- We have analogous results for variance-reduced proximal+stochastic methods.

[Reddi et al., 2016]
Non-Convex Non-IID Non-Stochastic Non-Serial

Non-Convex Rates for Stochastic Gradient

- Number of gradient evaluations to guarantee $\epsilon$-close to critical:
  - Gradient descent: $O(n/\epsilon)$
  - Stochastic gradient: $O(1/\epsilon^2)$
  - Variance-reduced: $O(n + n^{2/3}/\epsilon)$

- We have analogous results for variance-reduced proximal+stochastic methods. [Reddi et al., 2016]

- We cannot show analogous results for classic proximal stochastic methods.
  - All existing proximal+stochastic results require noise to go to zero.
  - Open problem that needs to be resolved: are analogous results possible?
Outline

1. Non-Convex
2. Non-IID
3. Non-Stochastic
4. Non-Serial
Non-IID Setting

- We discussed stochastic minimization problems

$$\arg\min_x \mathbb{E}[f_i(x)],$$

where we have the ability to generate IID samples $f_i(x)$.

- Using IID samples is justified by the law of large numbers.
Non-IID Setting

- We discussed stochastic minimization problems

\[
\argmin_x \mathbb{E}[f_i(x)],
\]

where we have the ability to generate IID samples \( f_i(x) \).

- Using IID samples is justified by the law of large numbers.
  - But it’s almost never true.

- What if we can’t get IID samples?
Non-IID Setting

- We discussed **stochastic minimization** problems

$$\arg\min_x \mathbb{E}[f_i(x)],$$

where we have the ability to generate IID samples $f_i(x)$.

- Using IID samples is justified by the **law of large numbers**.
  - But it’s **almost never true**.

- What if we **can’t get IID samples**?

  - Classic non-IID sampling scheme [Bertsekas & Tsitsiklis, 1996]:
    - **Samples follow a Markov chain** with stationary distribution of $\mathbb{E}[f_i(x)]$.
    - Obtain standard guarantees if Markov chain mixes fast enough [Duchi et al., 2012].
General Sampling

- What about general non-IID sampling schemes?
General Sampling

- What about general non-IID sampling schemes?
- What if our samples $f_i$ come from an adversary?
- Can we say anything in this case?
General Sampling

- What about general non-IID sampling schemes?
- What if our samples $f_i$ come from an adversary?
- Can we say anything in this case?
- Optimization error can be arbitrarily bad, but we can bound regret...
Online Convex Optimization

- Consider the online convex optimization (OCO) framework [Zinkevich, 2003]:
  - At time $t$, make a prediction $x^t$. 
Consider the online convex optimization (OCO) framework [Zinkevich, 2003]:

- At time $t$, make a prediction $x^t$.
- Receive next arbitrary convex loss $f_t$. 

The regret at time $t$ is given by

$$\sum_{k=1}^{t} [f_k(x_k) - f_k(x^*)]$$

The $x^*$ is not the solution to the problem, it's just the best we could have done. The $x^*$ depends on $t$, the "solution" is changing over time.
Online Convex Optimization

- Consider the online convex optimization (OCO) framework [Zinkevich, 2003]:
  - At time $t$, make a prediction $x^t$.
  - Receive next arbitrary convex loss $f_t$.
  - Pay a penalty of $f_t(x^t)$. 

The regret at time $t$ is given by $\sum_{k=1}^{t} [f_k(x_k) - f_k(x^*)]$, the total error compared to the best $x^*$ we could have chosen for first $t$ functions. The $x^*$ is not the solution to the problem, it’s just the best we could have done. The $x^*$ depends on $t$, the “solution” is changing over time.
Online Convex Optimization

Consider the online convex optimization (OCO) framework [Zinkevich, 2003]:

- At time $t$, make a prediction $x^t$.
- Receive next arbitrary convex loss $f_t$.
- Pay a penalty of $f_t(x^t)$.

The regret at time $t$ is given by

$$\sum_{k=1}^{t} [f_k(x^k) - f_k(x^*)],$$

the total error compared to the best $x^*$ we could have chosen for first $t$ functions.
Online Convex Optimization

- Consider the online convex optimization (OCO) framework [Zinkevich, 2003]:
  - At time $t$, make a prediction $x^t$.
  - Receive next arbitrary convex loss $f_t$.
  - Pay a penalty of $f_t(x^t)$.

- The regret at time $t$ is given by

\[
\sum_{k=1}^{t} [f_k(x^k) - f_k(x^*)],
\]

the total error compared to the best $x^*$ we could have chosen for first $t$ functions.

- The $x^*$ is not the solution to the problem, it’s just the best we could have done.
- The $x^*$ depends on $t$, the “solution” is changing over time.
Online Convex Optimization

- Assuming everything is bounded, doing **nothing** has a regret of $O(t)$. 
Online Convex Optimization

- Assuming everything is bounded, doing nothing has a regret of $O(t)$.

- Consider applying stochastic gradient, treating the $f_t$ as the samples.
  - For convex functions, has a regret of $O(\sqrt{t})$ [Zinkevich, 2003].
  - For strongly-convex, has a regret of $O(\log(t))$ [Hazan et al., 2006].
Online Convex Optimization

- Assuming everything is bounded, doing nothing has a regret of $O(t)$.

- Consider applying stochastic gradient, treating the $f_t$ as the samples.
  - For convex functions, has a regret of $O(\sqrt{t})$ [Zinkevich, 2003].
  - For strongly-convex, has a regret of $O(\log(t))$ [Hazan et al., 2006].
  - These are optimal.

- Key idea: $x^*$ isn’t moving faster than stochastic gradient is converging.
AdaGrad is a very-popular online method [Duchi et al., 2011]:
- Improves on constants in regret bounds using diagonal-scaling

\[ x^{t+1} = x^t - \alpha_t D_t^{-1} \nabla f_t(x^t) , \]

with diagonal entries \((D_t)_{ii} = \delta + \sqrt{\sum_{k=1}^{t} \nabla f_k(x^k)}\).
Online Convex Optimization

- **AdaGrad** is a very-popular online method [Duchi et al., 2011]:
  - Improves on constants in regret bounds using diagonal-scaling
    \[ x^{t+1} = x^t - \alpha_t D_t^{-1} \nabla f_t(x^t), \]
    with diagonal entries \((D_t)_{ii} = \delta + \sqrt{\sum_{k=1}^{t} \nabla_i f_k(x^k)}.\)

- **Adam** is a generalization that is incredibly-popular for deep learning. [Kingma & Ba, 2015]
  - Though trend is returning to variations on accelerated stochastic gradient.
  - Online learning remains active area and many variations exist:
    - **Bandit** methods only receive evaluation \(f_t(x^t)\) rather than function \(f_t\).
    - Main application: internet advertising and recommender systems.
Non-Convex Non-IID Non-Stochastic Non-Serial

Outline

1. Non-Convex
2. Non-IID
3. Non-Stochastic
4. Non-Serial
Graph-Structured Optimization

Another structure arising in machine learning is graph-structured problems,

$$\arg\min_x \sum_{(i,j) \in E} f_{ij}(x_i, x_j) + \sum_{i=1}^{n} f_i(x_i).$$

where $E$ is the set of edges in graph.
Graph-Structured Optimization

- Another structure arising in machine learning is **graph-structured problems**, 

  \[
  \arg\min_x \sum_{(i,j) \in E} f_{ij}(x_i, x_j) + \sum_{i=1}^{n} f_i(x_i).
  \]

  where \( E \) is the set of edges in graph.

- Includes quadratic functions, 

  \[
  x^T A x + b^T x = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} b_i x_i,
  \]

  and other models like **label propagation** for semi-supervised learning.

  - The **graph is sparsity pattern of** \( A \).
Coordinate Descent for Graph-Structured Optimization

- **Coordinate descent** seems well-suited to this problem structure:

\[
\arg\min_x \sum_{(i,j) \in E} f_{ij}(x_i, x_j) + \sum_{i=1}^{n} f_i(x_i).
\]

- To update \(x_i\), we only need to consider \(f_i\) and the \(f_{ij}\) for each neighbour.
Coordinate Descent for Graph-Structured Optimization

- **Coordinate descent** seems well-suited to this problem structure:

  \[
  \arg\min_x \sum_{(i,j) \in E} f_{ij}(x_i, x_j) + \sum_{i=1}^{n} f_i(x_i).
  \]

  To update \( x_i \), we only need to consider \( f_i \) and the \( f_{ij} \) for each neighbour.

- With **random selection** of coordinates, expected iteration cost is \( O(|E|/n) \).
  - This is \( n \)-times faster than GD iteration which cost \( O(|E|) \).
Coordinate Descent for Graph-Structured Optimization

- But for many problems randomized coordinate descent doesn’t work well...

- Often outperformed by the greedy Gauss-Southwell rule.
Coordinate Descent for Graph-Structured Optimization

- But for many problems randomized coordinate descent doesn’t work well...

- Often outperformed by the greedy Gauss-Southwell rule.
- But is plotting “epochs” cheating because Gauss-Southwell is more expensive?
Greedy Coordinate Descent

- **Gauss-Southwell** greedy rule for picking a coordinate to update:

\[
\argmax_i |\nabla_i f(x)|.
\]
Greedy Coordinate Descent

- **Gauss-Southwell** greedy rule for picking a coordinate to update:

\[
\arg\max_i |\nabla_i f(x)|.
\]
Greedy Coordinate Descent

- **Gauss-Southwell** greedy rule for picking a coordinate to update:

\[
\arg\max_i |\nabla_i f(x)|.
\]
Greedy Coordinate Descent

- **Gauss-Southwell** greedy rule for picking a coordinate to update:

\[
\text{argmax}_i |\nabla_i f(x)|.
\]
Greedy Coordinate Descent

- **Gauss-Southwell** greedy rule for picking a coordinate to update:
  \[
  \arg\max_i |\nabla_i f(x)|.
  \]

- **Looks expensive** because computing the gradient costs \(O(|E|)\).
Cost of Greedy Coordinate Descnet

- Gauss-Southwell cost depends on graph structure.
  - Same is true of Lipschitz sampling.
Cost of Greedy Coordinate Descent

- Gauss-Southwell cost depends on graph structure.
  - Same is true of Lipschitz sampling.

- Consider problems where maximum degree and average degree are similar:
  - Lattice graphs (max is 4, average is ≈ 4).
Cost of Greedy Coordinate Descent

- Gauss-Southwell cost depends on graph structure.
  - Same is true of Lipschitz sampling.

- Consider problems where maximum degree and average degree are similar:
  - Lattice graphs (max is 4, average is $\approx 4$).
  - Complete graphs (max and average degrees are $n - 1$).
Cost of Greedy Coordinate Descnet

- Gauss-Southwell cost depends on graph structure.
  - Same is true of Lipschitz sampling.

- Consider problems where maximum degree and average degree are similar:
  - Lattice graphs (max is 4, average is \( \approx 4 \)).
  - Complete graphs (max and average degrees are \( n - 1 \)).
  - Facebook graph (max is 7000, average is \( \approx 200 \)).
Cost of Greedy Coordinate Descnet

- Gauss-Southwell cost depends on graph structure.
  - Same is true of Lipschitz sampling.

- Consider problems where maximum degree and average degree are similar:
  - Lattice graphs (max is 4, average is ≈ 4).
  - Complete graphs (max and average degrees are \( n - 1 \)).
  - Facebook graph (max is 7000, average is ≈ 200).

- Here we can efficiently track the gradient and it’s max [Meshi et al., 2012].
Cost of Greedy Coordinate Descnet

- Gauss-Southwell cost depends on graph structure.
  - Same is true of Lipschitz sampling.

- Consider problems where maximum degree and average degree are similar:
  - Lattice graphs (max is 4, average is $\approx 4$).
  - Complete graphs (max and average degrees are $n - 1$).
  - Facebook graph (max is 7000, average is $\approx 200$).

- Here we can efficiently track the gradient and it’s max [Meshi et al., 2012].
  - Updating $x_i$, it only changes $|\nabla_j f(x^k)|$ for $i$ and its neighbours.
  - We can use a max-heap to track the maximum.
Convergence Rate of Greedy Coordinate Descent

But don’t random and greedy have the same rate?

Nutini et al. [2015] show that rate for Gauss-Southwell is

$$f(x_k) - f^* \leq (1 - \mu_1 L) k [f(x_0) - f^*],$$

where $\mu_1$ is strong-convexity constant in the 1-norm. Constant $\mu_1$ satisfies

$$\mu_{\text{random}} \leq \mu_1 \leq \mu_{\text{gradient}},$$

so we should expect more progress under Gauss-Southwell.
Convergence Rate of Greedy Coordinate Descent

- But don’t random and greedy have the same rate?

- Nutini et al. [2015] show that rate for Gauss-Southwell is

\[
f(x^k) - f^* \leq \left(1 - \frac{\mu_1}{L}\right)^k [f(x^0) - f^*],
\]

where \( \mu_1 \) is strong-convexity constant in the 1-norm.
Convergence Rate of Greedy Coordinate Descent

- But don’t random and greedy have the same rate?
- Nutini et al. [2015] show that rate for Gauss-Southwell is

\[ f(x^k) - f^* \leq \left(1 - \frac{\mu_1}{L}\right)^k [f(x^0) - f^*], \]

where \( \mu_1 \) is strong-convexity constant in the 1-norm.
- Constant \( \mu_1 \) satisfies

\[ \frac{\mu}{n} \leq \mu_1 \leq \mu, \]

so we should expect more progress under Gauss-Southwell.
Non-Convex Non-IID Non-Stochastic Non-Serial

Gauss-Southwell-Lipschitz Rule

- Nutini et al. [2015] also give a rule with faster rate by incorporating the $L_i$,

$$i_k = \arg\max_i \frac{|\nabla_i f(x^k)|}{\sqrt{L_i}}$$

which is called the **Gauss-Southwell-Lipschitz rule**.

- At least as fast as GS and Lipschitz sampling rules.
Gauss-Southwell-Lipschitz Rule

- Nutini et al. [2015] also give a rule with faster rate by incorporating the $L_i$,
  
  \[ i_k = \arg \max_i \frac{|\nabla_i f(x^k)|}{\sqrt{L_i}} , \]

  which is called the Gauss-Southwell-Lipschitz rule.

- At least as fast as GS and Lipschitz sampling rules.

- Intuition: if gradients are similar, more progress if $L_i$ is small.
Gauss-Southwell-Lipschitz Rule

- Nutini et al. [2015] also give a rule with faster rate by incorporating the $L_i$,

$$i_k = \arg\max_i \frac{\|\nabla f(x^k)\|}{\sqrt{L_i}},$$

which is called the Gauss-Southwell-Lipschitz rule.
- At least as fast as GS and Lipschitz sampling rules.
- Intuition: if gradients are similar, more progress if $L_i$ is small.
Nutini et al. [2015] also give a rule with faster rate by incorporating the $L_i$,

$$i_k = \arg\max_i \frac{|\nabla_i f(x^k)|}{\sqrt{L_i}}$$

which is called the Gauss-Southwell-Lipschitz rule.

- At least as fast as GS and Lipschitz sampling rules.
- Intuition: if gradients are similar, more progress if $L_i$ is small.

Greedy rules have lead to new methods for computing leading eigenvectors.

- Coordinate-wise power methods [Wei et al., 2016, Wang et al., 2017].
Outline

1. Non-Convex
2. Non-IID
3. Non-Stochastic
4. Non-Serial
Motivation for Parallel and Distributed

- Two recent trends:
  - We aren’t making large gains in serial computation speed.
  - Datasets no longer fit on a single machine.
Motivation for Parallel and Distributed

- Two recent trends:
  - We aren’t making large gains in serial computation speed.
  - Datasets no longer fit on a single machine.

- Result: we must use parallel and distributed computation.
Motivation for Parallel and Distributed

- **Two recent trends:**
  - We aren’t making large gains in serial computation speed.
  - Datasets no longer fit on a single machine.

- **Result:** we must use *parallel and distributed* computation.

- **Two major new issues:**
  - **Synchronization:** we can’t wait for the slowest machine.
  - **Communication:** we can’t transfer all information.
Embarrassing Parallelism in Machine Learning

- A lot of machine learning problems are **embarrassingly parallel**:
  - Split task across $M$ machines, solve independently, combine.
A lot of machine learning problems are **embarrassingly parallel**:
- Split task across $M$ machines, solve independently, combine.
- E.g., computing the gradient in deterministic gradient method,

$$
\frac{1}{N} \sum_{i=1}^{N} \nabla f_{i}(x) = \frac{1}{N} \left( \sum_{i=1}^{N/M} \nabla f_{i}(x) + \sum_{i=(N/M)+1}^{2N/M} \nabla f_{i}(x) + \ldots \right).
$$
A lot of machine learning problems are **embarrassingly parallel**:
- Split task across $M$ machines, solve independently, combine.

E.g., computing the gradient in deterministic gradient method,

$$
\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x) = \frac{1}{N} \left( \sum_{i=1}^{N/M} \nabla f_i(x) + \sum_{i=(N/M)+1}^{2N/M} \nabla f_i(x) + \ldots \right).
$$

These allow optimal **linear** speedups.
- You should always consider this first!
Asynchronous Computation

For stochastic gradient and SVRG, we can compute \textit{batch of gradients in parallel}:

\[
x^{k+1} = x^k - \alpha_k \frac{1}{|B|} \sum_{i \in B} \nabla f_i(x^k),
\]

for example computing \textit{one gradient} \( \nabla f_i(x^k) \) per processor.
For stochastic gradient and SVRG, we can compute batch of gradients in parallel:

\[ x^{k+1} = x^k - \alpha_k \frac{1}{|B|} \sum_{i \in B} \nabla f_i(x^k), \]

for example computing one gradient \( \nabla f_i(x^k) \) per processor.

Do we have to wait for the last computer to finish?
Asynchronous Computation

- For stochastic gradient and SVRG, we can compute batch of gradients in parallel:

  \[ x^{k+1} = x^k - \alpha_k \frac{1}{|B|} \sum_{i \in B} \nabla f_i(x^k), \]

  for example computing one gradient \( \nabla f_i(x^k) \) per processor.

- Do we have to wait for the last computer to finish?
  - No!
  - Updating asynchronously saves a lot of time.
Asynchronous Computation

- For stochastic gradient and SVRG, we can compute batch of gradients in parallel:

\[ x^{k+1} = x^k - \alpha_k \frac{1}{|B|} \sum_{i \in B} \nabla f_i(x^k), \]

for example computing one gradient \( \nabla f_i(x^k) \) per processor.

- Do we have to wait for the last computer to finish?
  - No!
  - Updating asynchronously saves a lot of time.
- E.g., stochastic gradient method on shared memory:

\[ x^{k+1} = x^k - \alpha_k \nabla f_i(x^{k-m}). \]
Asynchronous Computation

- For stochastic gradient and SVRG, we can compute \textit{batch of gradients in parallel}:

\[
x^{k+1} = x^k - \alpha_k \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla f_i(x^k),
\]

for example computing \textit{one gradient} \( \nabla f_i(x^k) \) per processor.

- Do we have to wait for the last computer to finish?  
  No! 
- Updating asynchronously saves a lot of time. 
- E.g., stochastic gradient method on shared memory:

\[
x^{k+1} = x^k - \alpha_k \nabla f_i(x^k - m).
\]

- You need to decrease step-size in proportion to asynchrony. 
- Convergence rate decays elegantly with delay \( m \) [Niu et al., 2011].
  - Now exists asynchronous variance-reduced methods.

[Reddi et al., 2015, Leblond et al., 2016, Mania et al., 2016]
Reduced Communication: Parallel Coordinate Descnet

- It may be expensive to communicate parameters $x$. 
It may be expensive to communicate parameters $x$.

One solution: use parallel coordinate descent:

$$
x_{j_1} = x_{j_1} - \alpha_{j_1} \nabla_{j_1} f(x) \\
x_{j_2} = x_{j_2} - \alpha_{j_2} \nabla_{j_2} f(x) \\
x_{j_3} = x_{j_3} - \alpha_{j_3} \nabla_{j_3} f(x)
$$

Only needs to communicate single coordinates.
Reduced Communication: Parallel Coordinate Descnet

- It may be **expensive to communicate** parameters $x$.
- One solution: use **parallel coordinate descent**:

  \[
  x_{j_1} = x_{j_1} - \alpha_{j_1} \nabla_{j_1} f(x)
  \]
  \[
  x_{j_2} = x_{j_2} - \alpha_{j_2} \nabla_{j_2} f(x)
  \]
  \[
  x_{j_3} = x_{j_3} - \alpha_{j_3} \nabla_{j_3} f(x)
  \]

- Only needs to communicate single coordinates.
- Again need to decrease step-size for convergence.
- Speedup is based on dependencies between variables [Richtarik & Takac, 2013].
Reduced Communication: Decentralized Gradient

- We may need to **distribute the data across machines**.
Reduced Communication: Decentralized Gradient

- We may need to distribute the data across machines.
- One solution: decentralized gradient method [Nedic & Ozdaglar, 2009]:
  - Each processor has its own data samples $f_1, f_2, \ldots f_m$.
  - Each processor has its own parameter vector $x_c$. 

$$x_c = \frac{1}{|\text{nei}(c)|} \sum_{c' \in \text{nei}(c)} x_{c'} - \alpha_c M \sum_{i=1}^m \nabla f_i(x_c).$$

Gradient descent is a special case where all neighbours communicate.

Modified update has fast rate in terms of graph Laplacian [Shi et al., 2014].

Can also consider communication failures [Agarwal & Duchi, 2011].

An active area with several other recent distributed methods.

[Jaggi et al., 2014, Shamir et al., 2013, Lee et al., 2015]
Reduced Communication: Decentralized Gradient

- We may need to **distribute the data across machines**.
- One solution: **decentralized gradient method** [Nedic & Ozdaglar, 2009]:
  - Each processor has its own data samples $f_1, f_2, \ldots f_m$.
  - Each processor has its own parameter vector $x_c$.
  - Each processor only communicates with a limited number of neighbours $\text{nei}(c)$.

Gradient descent is a special case where all neighbours communicate.
Modified update has fast rate in terms of graph Laplacian [Shi et al., 2014].
Can also consider communication failures [Agarwal & Duchi, 2011].
An active area with several other recent distributed methods.

[Jaggi et al., 2014, Shamir et al., 2013, Lee et al. 2015]
Reduced Communication: Decentralized Gradient

- We may need to distribute the data across machines.
- One solution: decentralized gradient method [Nedic & Ozdaglar, 2009]:
  - Each processor has its own data samples $f_1, f_2, \ldots, f_m$.
  - Each processor has its own parameter vector $x_c$.
  - Each processor only communicates with a limited number of neighbours $\text{nei}(c)$.

\[
x_c = \frac{1}{|\text{nei}(c)|} \sum_{c' \in \text{nei}(c)} x_{c'} - \frac{\alpha_c}{M} \sum_{i=1}^{M} \nabla f_i(x_c).
\]
We may need to distribute the data across machines.

One solution: decentralized gradient method [Nedic & Ozdaglar, 2009]:

- Each processor has its own data samples $f_1, f_2, \ldots f_m$.
- Each processor has its own parameter vector $x_c$.
- Each processor only communicates with a limited number of neighbours $\text{nei}(c)$.

\[
x_c = \frac{1}{|\text{nei}(c)|} \sum_{c' \in \text{nei}(c)} x_{c'} - \frac{\alpha_c}{M} \sum_{i=1}^{M} \nabla f_i(x_c).
\]

Gradient descent is special case where all neighbours communicate.

Modified update has fast rate in terms of graph Laplacian [Shi et al., 2014].
Reduced Communication: Decentralized Gradient

- We may need to **distribute the data across machines**.
- One solution: **decentralized gradient method** [Nedic & Ozdaglar, 2009]:
  - Each processor has its own data samples $f_1, f_2, \ldots f_m$.
  - Each processor has its own parameter vector $x_c$.
  - Each processor only communicates with a limited number of neighbours $\text{nei}(c)$.

\[
x_c = \frac{1}{|\text{nei}(c)|} \sum_{c' \in \text{nei}(c)} x_{c'} - \alpha_c \frac{M}{M} \sum_{i=1}^{M} \nabla f_i(x_c).
\]

- Gradient descent is special case where all neighbours communicate.
- Modified update has **fast rate in terms of graph Laplacian** [Shi et al., 2014].
- Can also consider communication failures [Agarwal & Duchi, 2011].
- An active area with several other recent distributed methods.

[Jaggi et al., 2014, Shamir et al., 2013, Lee et al. 2015]
Summary

- **PL inequality**: linear convergence for somewhat-non-convex functions.
Summary

- **PL inequality**: linear convergence for somewhat-non-convex functions.
- Convergence **rate of gradient norm**, and variance-reduction appears.
Summary

- **PL inequality**: linear convergence for somewhat-non-convex functions.
- Convergence rate of gradient norm, and variance-reduction appears.
- Stochastic algorithms have good regret for arbitrary sequences.
Summary

- **PL inequality**: linear convergence for somewhat-non-convex functions.
- Convergence rate of gradient norm, and variance-reduction appears.
- Stochastic algorithms have good regret for arbitrary sequences.
- Greedy coordinate descent seems like the right tool for some problems.
Summary

- **PL inequality**: linear convergence for somewhat-non-convex functions.
- Convergence rate of gradient norm, and variance-reduction appears.
- Stochastic algorithms have good regret for arbitrary sequences.
- Greedy coordinate descent seems like the right tool for some problems.
- Parallel/distributed methods are the future, but pose new challenges.