Is Greedy Coordinate Descent a Terrible Algorithm?

Julie Nutini, Mark Schmidt, Issam Laradji, Behrooz Sepehry, Michael Friedlander, Hoyt Koepke

University of British Columbia
We consider the basic convex optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f$ is differentiable and $n$ is large.
We consider the basic **convex optimization** problem:

\[
\min_{x \in \mathbb{R}^n} f(x),
\]

where \( f \) is differentiable and \( n \) is large.

A popular approach is **coordinate descent**:

1. Select a coordinate to update.
2. Take a small gradient step along coordinate.
Why use coordinate descent?

- Theoretically, it is a **provably bad** algorithm:
  - The convergence rate is **slower than gradient descent**.
  - The iteration cost can be **similar to gradient descent**.
Why use coordinate descent?

- Theoretically, it is a *provably bad* algorithm:
  - The convergence rate is *slower* than gradient descent.
  - The iteration cost can be *similar to* gradient descent.

- But it is *widely-used* in practice:
  - Nothing works better for certain problems.
  - Certain fields think it is the ‘ultimate’ algorithm.
Why use coordinate descent?

- Theoretically, it is a provably bad algorithm:
  - The convergence rate is slower than gradient descent.
  - The iteration cost can be similar to gradient descent.

- But it is widely-used in practice:
  - Nothing works better for certain problems.
  - Certain fields think it is the ‘ultimate’ algorithm.
Why use coordinate descent?

- Theoretically, it is a **provably bad** algorithm:
  - The convergence rate is **slower than gradient descent**.
  - The iteration cost can be **similar to gradient descent**.

- But it is **widely-used** in practice:
  - Nothing works better for certain problems.
  - Certain fields think it is the ‘ultimate’ algorithm.

- Renewed theoretical interest began with Nesterov [2010]:
  - Global convergence rate for **randomized** coordinate selection.
  - **Faster than gradient descent** if iterations are \( n \) times cheaper.
Problems Suitable for Coordinate Descent

Coordinate update is $n$ times faster than gradient update for:

$h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i)$, or

$h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)$

$f$ and $f_{ij}$ smooth

$A$ is a matrix and $f$ is cheap

$\{V, E\}$ is a graph

$g_i$ general convex functions

Examples $h_1$: least squares, logistic regression, lasso, SVMs (e.g., machine learning).

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|_2^2 + \lambda \sum_{i=1}^{n} |x_i|.$$  

Examples $h_2$: quadratics, graph-based label propagation, graphical models.

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T Ax + b^T x = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} b_i x_i.$$
Problems Suitable for Coordinate Descent

Coordinate update is $n$ times faster than gradient update for:

$$h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \text{ or } h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)$$

- $f$ and $f_{ij}$ smooth
- $A$ is a matrix and $f$ is cheap
- $\{V, E\}$ is a graph
- $g_i$ general convex functions
Problems Suitable for Coordinate Descent

Coordinate update is \textit{n times faster} than gradient update for:

\[ h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \quad \text{or} \quad h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j) \]

- \( f \) and \( f_{ij} \) smooth
- \( A \) is a matrix and \( f \) is cheap
- \( \{V, E\} \) is a graph
- \( g_i \) general convex functions

\textbf{Examples} \( h_1 \): least squares, logistic regression, lasso, SVMs (e.g., machine learning).

\[
\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|^2 + \lambda \sum_{i=1}^{n} |x_i|.
\]
Problems Suitable for Coordinate Descent

Coordinate update is *n times faster* than gradient update for:

\[ h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \quad \text{or} \quad h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j) \]

- \( f \) and \( f_{ij} \) smooth
- \( A \) is a matrix and \( f \) is cheap
- \( \{V, E\} \) is a graph
- \( g_i \) general convex functions

**Examples** \( h_1 \): least squares, logistic regression, lasso, SVMs (e.g., machine learning).

\[ \min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|^2 + \lambda \sum_{i=1}^{n} |x_i|. \]

**Examples** \( h_2 \): quadratics, graph-based label propagation, graphical models.

\[ \min_{x \in \mathbb{R}^n} \frac{1}{2} x^T Ax + b^T x = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} b_i x_i. \]
This talk:

Instead of random, consider classic Gauss Southwell (GS) rule:

$$\arg\max_i |\nabla f(x)|,$$

also known as greedy selection.

GS is at least as expensive as random. But Nesterov showed the rate is the same. But this theory disagrees with practice...
This talk:

- Instead of random, consider classic Gauss Southwell (GS) rule:

\[
\arg\max_i |\nabla_i f(x)|,
\]

also known as greedy selection.
This talk:

- Instead of random, consider classic Gauss Southwell (GS) rule:

\[ \underset{i}{\text{argmax}} |\nabla_i f(x)|, \]

also known as greedy selection.
This talk:

- Instead of random, consider classic Gauss Southwell (GS) rule:

\[ \arg\max_i |\nabla_i f(x)|, \]

also known as greedy selection.
This talk:

- Instead of random, consider classic Gauss Southwell (GS) rule:

\[
\arg\max_i |\nabla_i f(x)|,
\]

also known as greedy selection.
This talk:

- Instead of random, consider classic Gauss Southwell (GS) rule:

\[
\arg\max_i |\nabla_i f(x)|,
\]

also known as greedy selection.

- GS is at least as expensive as random.
- But Nesterov showed the rate is the same.
- But this theory disagrees with practice...
GS works much better when random and GS have similar costs.
- GS works much better when random and GS have similar costs.
- This work: refined analysis of GS.
In general, GS rule may be as expensive as gradient even for $h_1$ and $h_2$.

$$h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \text{ or } h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)$$

But in special cases GS is $\approx n$ times faster.
Problems where can apply Gauss-Southwell

- In general, GS rule may be as expensive as gradient even for $h_1$ and $h_2$.
  \[
  h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \quad \text{or} \quad h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)
  \]

- But in special cases GS is $\approx n$ times faster.

- GS is efficient for $h_2$ if maximum degree similar to average degree.

  (can track gradient using max-heap, Meshi et al. [2012])
Problems where can apply Gauss-Southwell

- In general, GS rule may be as expensive as gradient even for \( h_1 \) and \( h_2 \).

\[
h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \quad \text{or} \quad h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)
\]

- But in special cases GS is \( \approx n \) times faster.

- GS is efficient for \( h_2 \) if maximum degree similar to average degree.
  (can track gradient using max-heap, Mesi et al. [2012])

- Grid-based models, max degree = 4 and average degree \( \approx 4 \).
- Dense quadratic: max degree = \((n - 1)\), average degree = \((n - 1)\).
- Facebook graph: max degree < 7000, average is \( \approx 200 \).
Problems where can apply Gauss-Southwell

- In general, GS rule **may be as expensive as gradient** even for \(h_1\) and \(h_2\).

\[
h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \quad \text{or} \quad h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)
\]

- But in **special cases** GS is \(\approx n\) times faster.

- GS is efficient for \(h_2\) if **maximum degree similar to average degree**.
  
  (can track gradient using max-heap, Meshi et al. [2012])

  - Grid-based models, max degree = 4 and average degree \(\approx 4\).
  - Dense quadratic: max degree = \((n - 1)\), average degree = \((n - 1)\).
  - Facebook graph: max degree < 7000, average is \(\approx 200\).

- For problem \(h_1\):
  - Often solvable in \(O(c r \log n)\) with \(c\) and \(r\) non-zeros per column/row.
In general, GS rule may be as expensive as gradient even for $h_1$ and $h_2$.

$$h_1(x) = f(Ax) + \sum_{i=1}^{n} g_i(x_i), \quad \text{or} \quad h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)$$

But in special cases GS is $\approx n$ times faster.

GS is efficient for $h_2$ if maximum degree similar to average degree.

(can track gradient using max-heap, Meshi et al. [2012])

- Grid-based models, max degree = 4 and average degree $\approx 4$.
- Dense quadratic: max degree $= (n - 1)$, average degree $= (n - 1)$.
- Facebook graph: max degree $< 7000$, average is $\approx 200$.

For problem $h_1$:

- Often solvable in $O(cr \log n)$ with $c$ and $r$ non-zeros per column/row.
- GS can be approximated as nearest neighbour problem.

[Dhillon et al., 2011, Shrivastava & Li, 2014].
We focus on the convex optimization problem

\[
\min_{x \in \mathbb{R}^n} f(x),
\]

where \( \nabla f \) is coordinate-wise \( L \)-Lipschitz continuous,

\[
|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L|\alpha|.
\]
We focus on the convex optimization problem

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $\nabla f$ is coordinate-wise $L$-Lipschitz continuous,

$$|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L|\alpha|.$$

We focus on the case where $f$ is $\mu$-strongly convex, meaning that

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

for all $y$ and $x$ and some $\mu > 0$. 
We focus on the convex optimization problem

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $\nabla f$ is coordinate-wise $L$-Lipschitz continuous,

$$|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L|\alpha|.$$

We focus on the case where $f$ is $\mu$-strongly convex, meaning that

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

for all $y$ and $x$ and some $\mu > 0$.

If twice-differentiable, equivalent to

$$\nabla^2_{ii} f(x) \leq L, \quad \nabla^2 f(x) \succeq \mu I.$$
Convergence of Randomized Coordinate Descent

- Coordinate descent with constant step size $\frac{1}{L}$ uses

$$x^{k+1} = x^k - \frac{1}{L} \nabla_{i_k} f(x^k) e_{i_k},$$

for some variable $i_k$. 

- The convergence rate of gradient descent with step-size $\frac{1}{L}$ is

$$f(x^{k+1}) - f(x^*) \leq (1 - \mu L) \left[ f(x^k) - f(x^*) \right],$$

so we require $O\left(\frac{L}{\mu} \log \left(\frac{1}{\epsilon}\right)\right)$ iterations to reach accuracy $\epsilon$.

- With $i_k$ chosen uniformly, coordinate descent has

$$E[f(x^{k+1}) - f(x^*)] \leq (1 - \mu L_n) \left[ f(x^k) - f(x^*) \right],$$

so we require $O\left(\frac{L_n}{\mu} \log \left(\frac{1}{\epsilon}\right)\right)$ iterations to reach accuracy $\epsilon$.

Since $L_n \geq L f \geq L$, coordinate descent is slower per iteration, but converge faster if they are $n$ times cheaper.
Convergence of Randomized Coordinate Descent

- Coordinate descent with constant step size $\frac{1}{L}$ uses

\[ x^{k+1} = x^k - \frac{1}{L} \nabla_{i_k} f(x^k)e_{i_k}, \]

for some variable $i_k$.

- Convergence rate of gradient descent with step-size $1/L_f$ is

\[ f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu}{L_f}\right) [f(x^k) - f(x^*)], \]

so we require $O(\frac{L_f}{\mu} \log(1/\epsilon))$ iterations to reach accuracy $\epsilon$. 
Convergence of Randomized Coordinate Descent

- Coordinate descent with constant step size $\frac{1}{L}$ uses
  \[ x^{k+1} = x^k - \frac{1}{L} \nabla_{i_k} f(x^k)e_{i_k}, \]
  for some variable $i_k$.
- Convergence rate of gradient descent with step-size $1/L_f$ is
  \[ f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu}{L_f}\right) [f(x^k) - f(x^*)], \]
  so we require $O\left(\frac{L_f}{\mu} \log(1/\epsilon)\right)$ iterations to reach accuracy $\epsilon$.
- With $i_k$ chosen uniformly, coordinate descent has
  \[ \mathbb{E}[f(x^{k+1})] - f(x^*) \leq \left(1 - \frac{\mu}{Ln}\right) [f(x^k) - f(x^*)], \]
  so we require $O\left(\frac{Ln}{\mu} \log(1/\epsilon)\right)$ iterations to reach accuracy $\epsilon$. 
Convergence of Randomized Coordinate Descent

- Coordinate descent with constant step size $\frac{1}{L}$ uses

$$x^{k+1} = x^k - \frac{1}{L} \nabla_{i_k} f(x^k)e_{i_k},$$

for some variable $i_k$.

- Convergence rate of gradient descent with step-size $1/L_f$ is

$$f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu}{L_f}\right) [f(x^k) - f(x^*)],$$

so we require $O\left(\frac{L_f}{\mu} \log(1/\epsilon)\right)$ iterations to reach accuracy $\epsilon$.

- With $i_k$ chosen uniformly, coordinate descent has

$$\mathbb{E}[f(x^{k+1})] - f(x^*) \leq \left(1 - \frac{\mu}{Ln}\right) [f(x^k) - f(x^*)],$$

so we require $O\left(\frac{Ln}{\mu} \log(1/\epsilon)\right)$ iterations to reach accuracy $\epsilon$.

- Since $Ln \geq L_f \geq L$, coordinate descent is slower per iteration, but converge faster if they are $n$ times cheaper.
GS rule chooses coordinate with largest directional derivative,

\[ i_k = \arg\max_i |\nabla_i f(x^k)|. \]
Classic Analysis of Gauss-Southwell Rule

- GS rule chooses coordinate with largest directional derivative,

\[ i_k = \arg\max_i |\nabla_i f(x^k)|. \]

- From Lipschitz-continuity assumption this rule satisfies

\[
f(x^{k+1}) \leq f(x^k) - \frac{1}{2L} \|\nabla f(x^k)\|^2_\infty.
\]

From strong-convexity we have

\[
f(x^*) \geq f(x^k) - \frac{1}{2\mu} \|\nabla f(x^k)\|^2_\infty.
\]

Using \( \|\nabla f(x^k)\|_2 \leq \sqrt{n} \|\nabla f(x^k)\|_\infty \), we get

\[
f(x^{k+1}) - f(x^*) \leq (1 - \mu L n) [f(x^k) - f(x^*)],
\]

same rate as randomized [Boyd & Vandenberghe, 2004, §9.4.3].
Classic Analysis of Gauss-Southwell Rule

- GS rule chooses coordinate with largest directional derivative,

\[ i_k = \operatorname{argmax}_i |\nabla_i f(x^k)|. \]

- From Lipschitz-continuity assumption this rule satisfies

\[ f(x^{k+1}) \leq f(x^k) - \frac{1}{2L} \|\nabla f(x^k)\|_\infty^2. \]

- From strong-convexity we have

\[ f(x^*) \geq f(x^k) - \frac{1}{2\mu} \|\nabla f(x^k)\|^2. \]
Classic Analysis of Gauss-Southwell Rule

- GS rule chooses coordinate with largest directional derivative,
  \[ i_k = \arg\max_i |\nabla_i f(x^k)|. \]
- From Lipschitz-continuity assumption this rule satisfies
  \[ f(x^{k+1}) \leq f(x^k) - \frac{1}{2L} \|\nabla f(x^k)\|^2. \]
- From strong-convexity we have
  \[ f(x^*) \geq f(x^k) - \frac{1}{2\mu} \|\nabla f(x^k)\|^2. \]
- Using \( \|\nabla f(x^k)\|^2 \leq n \|\nabla f(x^k)\|^2 \|\nabla f(x^k)\|_{\infty} \), we get
  \[ f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu}{Ln}\right) [f(x^k) - f(x^*)], \]
  same rate as randomized [Boyd & Vandenberghe, 2004, §9.4.3].
Classic Analysis of Gauss-Southwell Rule

- GS rule chooses coordinate with largest directional derivative,
  \[ i_k = \arg\max_i |\nabla_i f(x^k)|. \]

- From Lipschitz-continuity assumption this rule satisfies
  \[ f(x^{k+1}) \leq f(x^k) - \frac{1}{2L} \|\nabla f(x^k)\|_\infty^2. \]

- From strong-convexity we have
  \[ f(x^*) \geq f(x^k) - \frac{1}{2\mu} \|\nabla f(x^k)\|_\infty^2. \]

- Using \( \|\nabla f(x^k)\|^2 \leq n \|\nabla f(x^k)\|_\infty^2 \), we get
  \[ f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu}{Ln}\right) [f(x^k) - f(x*)]. \]

same rate as randomized [Boyd & Vandenberghe, 2004, §9.4.3].
To avoid norm inequality, measure strong-convexity in $1$-norm,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_1}{2} \|y - x\|_1^2.$$
To avoid norm inequality, measure strong-convexity in $1$-norm,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_1}{2} \|y - x\|_1^2.$$  

Using convex conjugate of $\| \cdot \|_1^2$ we now have that

$$f(x^*) \geq f(x^k) - \frac{1}{2\mu_1} \|\nabla f(x^k)\|_\infty^2.$$
To avoid norm inequality, measure strong-convexity in 1-norm,

\[ f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_1}{2} \|y - x\|_1^2. \]

Using convex conjugate of \( \| \cdot \|_1^2 \) we now have that

\[ f(x^*) \geq f(x^k) - \frac{1}{2\mu_1} \| \nabla f(x^k) \|_\infty^2. \]

Combining with \( \| \cdot \|_\infty^2 \) GS progress bound gives a rate of

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

This would be the same as random if \( \mu_1 = \mu/n \).
To avoid norm inequality, measure strong-convexity in 1-norm,

\[ f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_1}{2} \|y - x\|_1^2. \]

Using convex conjugate of \( \| \cdot \|_1^2 \) we now have that

\[ f(x^*) \geq f(x^k) - \frac{1}{2\mu_1} \|\nabla f(x^k)\|_{\infty}^2. \]

Combining with \( \| \cdot \|_{\infty}^2 \) GS progress bound gives a rate of

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

This would be the same as random if \( \mu_1 = \mu/n \).

The relationship between \( \mu \) and \( \mu_1 \) is given by

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

Worst case same as random, but may be faster by factor up to \( n \).
In $f$ is a quadratic with diagonal Hessian, we can show

$$\mu = \min_i \lambda_i, \quad \text{and} \quad \mu_1 = \frac{1}{\sum_{i=1}^{n} \frac{1}{\lambda_i}}.$$
In $f$ is a quadratic with diagonal Hessian, we can show

$$\mu = \min_i \lambda_i, \quad \text{and} \quad \mu_1 = \frac{1}{\sum_{i=1}^{n} \frac{1}{\lambda_i}}.$$

$\mu_1$ is harmonic mean of $\lambda_i$ divided by $n$:

- Time needed for workers “working together” to finish task is $\mu_1$.
  
  [Ferger, 1931].

- Dominated by minimum $\lambda_i$. 

Comparison for Separable Quadratic

- In $f$ is a quadratic with diagonal Hessian, we can show

$$
\mu = \min_i \lambda_i, \quad \text{and} \quad \mu_1 = \frac{1}{\sum_{i=1}^n \frac{1}{\lambda_i}}.
$$

- $\mu_1$ is harmonic mean of $\lambda_i$ divided by $n$:
  - Time needed for workers “working together” to finish task is $\mu_1$.
    - Dominated by minimum $\lambda_i$.
  - If all $\lambda_i$ equal:
    - There is no advantage to GS ($\mu_1 = \mu/n$).

[Ferger, 1931].

In $f$ is a quadratic with diagonal Hessian, we can show

$$\mu = \min_i \lambda_i, \quad \text{and} \quad \mu_1 = \frac{1}{\sum_{i=1}^{n} \frac{1}{\lambda_i}}.$$  

$\mu_1$ is harmonic mean of $\lambda_i$ divided by $n$:

- Time needed for workers “working together” to finish task is $\mu_1$. [Ferger, 1931].
- Dominated by minimum $\lambda_i$.

If all $\lambda_i$ equal:
- There is no advantage to GS ($\mu_1 = \mu/n$).

With one very large $\lambda_i$:
- Here you would think that GS would be faster.
- But GS and random are still similar ($\mu_1 \approx \mu/n$).
Comparison for Separable Quadratic

- In $f$ is a quadratic with diagonal Hessian, we can show

$$\mu = \min_{i} \lambda_i, \quad \text{and} \quad \mu_1 = \frac{1}{\sum_{i=1}^{n} \frac{1}{\lambda_i}}.$$ 

- $\mu_1$ is harmonic mean of $\lambda_i$ divided by $n$:
  - Time needed for workers “working together” to finish task is $\mu_1$.
  - [Ferger, 1931].
  - Dominated by minimum $\lambda_i$.

- If all $\lambda_i$ equal:
  - There is no advantage to GS ($\mu_1 = \mu/n$).

- With one very large $\lambda_i$:
  - Here you would think that GS would be faster.
  - But GS and random are still similar ($\mu_1 \approx \mu/n$).

- With one very small $\lambda_i$:
  - Here GS bound can be better by a factor of $n$ ($\mu_1 \approx \mu$).
  - In this case, GS can actually be faster than gradient descent.
We’ve analyze using constant $1/L$ step-size.

Common alternative is exact coordinate optimization.
We’ve analyze using constant $1/L$ step-size.

Common alternative is exact coordinate optimization.

We can get same rates because

$$f(x_k^{k+1}) = \min_{\alpha} \{ f(x_k^k - \alpha \nabla_i f(x_k^k)e_i^k) \}$$

$$\leq f \left( x_k^k - \frac{1}{L} \nabla_i f(x_k^k)e_i^k \right),$$

and rate is not know to be better for exact minimization.
We’ve analyze using constant $1/L$ step-size.

Common alternative is exact coordinate optimization.

We can get same rates because

$$f(x^{k+1}) = \min_{\alpha} \{ f(x^k - \alpha \nabla_i f(x^k)e_{i_k}) \}$$

$$\leq f \left( x^k - \frac{1}{L} \nabla_i f(x^k)e_{i_k} \right),$$

and rate is not know to be better for exact minimization.

But empirically, exact minimization is much faster.

Can we show that exact optimization gives a better bound?
Let’s assume we have an $L_i$ for each coordinate,

$$|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L_i |\alpha|.$$
Let’s assume we have an $L_i$ for each coordinate,

$$|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L_i |\alpha|.$$ 

If we use a step-size of $1/L_{i_k}$ our rate becomes

$$f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu_1}{L_{i_k}}\right) [f(x^k) - f(x^*)],$$

but this isn’t faster because we may have $L_{i_k} = L$ for all $k$. 

---

**Rates with Different Lipschitz Constants**
Let’s assume we have an $L_i$ for each coordinate,

$$|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L_i |\alpha|.$$  

If we use a step-size of $1/L_{i,k}$ our rate becomes

$$f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu_1}{L_{i,k}}\right)[f(x^k) - f(x^*)],$$

but this isn’t faster because we may have $L_{i,k} = L$ for all $k$.

But for certain sparse problems, worst case can’t happen...
Key insight is that we never repeat same coordinate:

- After exact update, $\nabla f(x_{k+1}) = 0$.
Key insight is that we never repeat same coordinate:

- After exact update, $\nabla_{i_k} f(x^{k+1}) = 0$.
- Since $i_{k+1} = \arg\max_i |\nabla_i f(x^{k+1})|$, we never have $i_{k+1} \neq i_k$. 

\[
\begin{bmatrix}
0 & 0.67 & -1.21 & 0.72 & 1.63 & 0.49 \\
0 & 0.65 & -1.31 & 0.81 & 0 & 0.53
\end{bmatrix}
\]
Key insight is that we never repeat same coordinate:

- After exact update, $\nabla_{i_k} f(x^{k+1}) = 0$.
- Since $i_{k+1} = \arg\max_i |\nabla_i f(x^{k+1})|$, we never have $i_{k+1} \neq i_k$.

$$\nabla f(x^k) = \begin{bmatrix} 0.67 \\ -1.21 \\ 0.72 \\ 1.63 \\ 0.49 \end{bmatrix},$$
Key insight is that we never repeat same coordinate:

- After exact update, $\nabla_{i_k} f(x^{k+1}) = 0$.
- Since $i_{k+1} = \arg\max_i |\nabla_i f(x^{k+1})|$, we never have $i_{k+1} \neq i_k$.

$$\nabla f(x^k) = \begin{bmatrix} 0.67 \\ -1.21 \\ 0.72 \\ 1.63 \\ 0.49 \end{bmatrix}, \quad \nabla f(x^{k+1}) = \begin{bmatrix} 0.65 \\ -1.31 \\ 0.81 \\ 0 \\ 0.53 \end{bmatrix}.$$
Key insight is that we never repeat same coordinate:

- After exact update, \( \nabla_{i_k} f(x^{k+1}) = 0 \).
- Since \( i_{k+1} = \arg\max_i |\nabla_i f(x^{k+1})| \), we never have \( i_{k+1} \neq i_k \).

\[
\nabla f(x^k) = \begin{bmatrix} 0.67 \\ -1.21 \\ 0.72 \\ 1.63 \\ 0.49 \end{bmatrix}, \quad \nabla f(x^{k+1}) = \begin{bmatrix} 0.65 \\ -1.31 \\ 0.81 \\ 0 \\ 0.53 \end{bmatrix}.
\]

We can use this to bound the sequence of \( L_{i_k} \) values.
Consider a graph describing dependencies between variables:

After update, only derivatives of neighbours can change. After an exact update we have
\[ \nabla f(x_k + m) = 0, \]
for all \( m \) until we update a neighbour of \( i_k \) in the graph.

We derive a bound on the \( L_{i_k} \) depending on worst non-empty star-structured subgraph.

Rate is much faster if large \( L_{i_k} \) are at least 2 nodes apart.
Consider a graph describing dependencies between variables:

After update, only derivatives of neighbours can change. After an exact update we have
\[ \nabla f_i(x_k + m) = 0 \]
for all \( m \) until we update a neighbour of \( i \) in the graph. We derive a bound on the \( L_i \) depending on worst non-empty star-structured subgraph. Rate is much faster if large \( L_i \) are at least 2 nodes apart.
Consider a graph describing dependencies between variables:

After update, only derivatives of neighbours can change.
Consider a graph describing dependencies between variables:

After update, only derivatives of neighbours can change.
Consider a graph describing dependencies between variables:

After update, only derivatives of neighbours can change.
Consider a graph describing dependencies between variables:

After update, only derivatives of neighbours can change.

After an exact update we have $\nabla_{i_k} f(x^{k+m}) = 0$, for all $m$ until we update a neighbour of $i_k$ in the graph.
Consider a graph describing dependencies between variables:

After update, only derivatives of neighbours can change.

After an exact update we have $\nabla_{i_k} f(x^{k+m}) = 0$, for all $m$ until we update a neighbour of $i_k$ in the graph.

We derive a bound on the $L_{i_k}$ depending on worst non-empty star-structured subgraph.

Rate is much faster if large $L_i$ are at least 2 nodes apart.
What about Nesterov’s non-uniform sampling strategy?
What about Nesterov’s non-uniform sampling strategy?

- If we sample proportional to $L_i$ we get

$$
\mathbb{E}[f(x^{k+1})] - f(x^*) \leq \left(1 - \frac{\mu}{n\bar{L}}\right) [f(x^k) - f(x^*)],
$$

where $\bar{L} = \frac{1}{n} \sum_{i=1}^{n} L_i$ so its faster than uniform.
What about Nesterov’s non-uniform sampling strategy?

- If we sample proportional to $L_i$ we get

$$\mathbb{E}[f(x^{k+1})] - f(x^*) \leq \left(1 - \frac{\mu}{n\bar{L}}\right) [f(x^k) - f(x^*)],$$

where $\bar{L} = \frac{1}{n} \sum_{i=1}^{n} L_i$ so its faster than uniform.

- Could be faster or slower than our GS bound.

- So what should we do?
We obtain a faster rate than both by using $L_i$ in the GS rule,

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

which we call the Gauss-Southwell-Lipschitz (GSL) rule.
We obtain a faster rate than both by using $L_i$ in the GS rule,

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

which we call the Gauss-Southwell-Lipschitz (GSL) rule.

Intuition: if gradients are similar, more progress if $L_i$ is small.
We obtain a faster rate than both by using $L_i$ in the GS rule,

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

which we call the Gauss-Southwell-Lipschitz (GSL) rule.

Intuition: if gradients are similar, more progress if $L_i$ is small.
We obtain a faster rate than both by using $L_i$ in the GS rule,

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

which we call the Gauss-Southwell-Lipschitz (GSL) rule.

Intuition: if gradients are similar, more progress if $L_i$ is small.
The GSL rule obtains a rate of

\[ f(x^{k+1}) - f(x^k) \leq (1 - \mu_L)[f(x^k) - f(x^*)]. \]

where \( \mu_L \) comes from weighted L1-norm
The GSL rule obtains a rate of

\[ f(x^{k+1}) - f(x^k) \leq (1 - \mu_L)[f(x^k) - f(x^*)]. \]

where \( \mu_L \) comes from weighted L1-norm and satisfies

\[
\max \left\{ \frac{\mu}{n\overline{L}}, \frac{\mu_1}{L_i} \right\} \leq \mu_L \leq \frac{\mu_1}{\min_i \{L_i\}},
\]

so GSL is at least as fast as GS and Lipschitz sampling.
The GSL rule obtains a rate of

\[ f(x^{k+1}) - f(x^k) \leq (1 - \mu_L)[f(x^k) - f(x^*)], \]

where \( \mu_L \) comes from weighted L1-norm and satisfies

\[
\max\left\{ \frac{\mu}{nL}, \frac{\mu_1}{L_i}, \frac{\mu_1}{GS} \right\} \leq \mu_L \leq \frac{\mu_1}{\min_i\{L_i\}},
\]

so GSL is at least as fast as GS and Lipschitz sampling.

GSL using \( \frac{1}{L_{i_k}} \) is unimprovable for quadratics,

\[ f(x^{k+1}) = \arg\min_{i,\alpha} \left\{ f(x^k + \alpha e_i) \right\}. \]
The GSL rule obtains a rate of

\[ f(x^{k+1}) - f(x^k) \leq (1 - \mu_L) [f(x^k) - f(x^*)]. \]

where \( \mu_L \) comes from weighted L1-norm and satisfies

\[
\max \left\{ \frac{\mu}{nL}, \frac{\mu_1}{L_i}, \frac{\mu_1}{GS} \right\} \leq \mu_L \leq \frac{\mu_1}{\min_i \{L_i\}},
\]

so GSL is at least as fast as GS and Lipschitz sampling.

GSL using \( \frac{1}{L_{i_k}} \) is unimprovable for quadratics,

\[ f(x^{k+1}) = \arg\min_{i, \alpha} \{ f(x^k + \alpha e_i) \}. \]

Gives tighter bound on maximum improvement rule.
• GSL rule gives modest but consistent improvements.
What if the exact GS rule is too expensive?
What if the exact GS rule is too expensive?

We can also analyze approximate GS rule.

For linear models, can approximate GS as nearest neighbour.
What if the exact GS rule is too expensive?

We can also analyze approximate GS rule.

- For linear models, can approximate GS as nearest neighbour.
- With multiplicative error,

\[
|\nabla_{i_k} f(x^k)| \geq \|\nabla f(x^k)\|_\infty (1 - \epsilon_k),
\]

we have a fast rate and do not need \(\epsilon_k \to 0\),

\[
f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu_1 (1 - \epsilon_k)^2}{L}\right)[f(x^k) - f(x^*)].
\]
• What if the exact GS rule is too expensive?
• We can also analyze approximate GS rule.
  • For linear models, can approximate GS as nearest neighbour.
• With multiplicative error,

\[ |\nabla_{i_k} f(x^k)| \geq \| \nabla f(x^k) \|_\infty (1 - \epsilon_k), \]

we have a fast rate and do not need \( \epsilon_k \to 0 \),

\[ f(x^{k+1}) - f(x^*) \leq \left( 1 - \frac{\mu_1 (1 - \epsilon_k)^2}{L} \right) [f(x^k) - f(x^*)]. \]

• With additive error, we have a fast rate if \( \epsilon_k \to 0 \) fast enough.
Approximate GS can still be faster than random sampling.
Consider a special case of $h_1$,

$$\min_x h_1(x) = \sum_{i=1}^{n} f(a_i^T x),$$

where GS rule has the form

$$i_k = \arg\max_i |a_i^T r(x^k)|.$$
Consider a special case of $h_1$, 

$$\min_x h_1(x) = \sum_{i=1}^{n} f(a_i^T x),$$

where GS rule has the form 

$$i_k = \arg\max_i |a_i^T r(x^k)|.$$

Dhillon et al. [2011] approximate GS as nearest neighbour,
Consider a special case of $h_1$,

$$
\min_x h_1(x) = \sum_{i=1}^{n} f(a_i^T x),
$$

where GS rule has the form

$$
i_k = \arg\max_i |a_i^T r(x^k)|.
$$

Dhillon et al. [2011] approximate GS as nearest neighbour,

$$
\arg\min_i \|r(x^k) - a_i\|^2 = \arg\min_i \left\{ |\nabla_i f(x^k)| - \frac{1}{2} \|a_i\|^2 \right\}.
$$
Gauss-Southwell-Lipschitz as Nearest Neighbour

- Consider a special case of $h_1$,

$$
\min_x h_1(x) = \sum_{i=1}^n f(a_i^T x),
$$

where GS rule has the form

$$
i_k = \arg\max_i |a_i^T r(x_k)|.
$$

- Dhillon et al. [2011] approximate GS as nearest neighbour,

$$
\arg\min_i \| r(x_k) - a_i \|^2 = \arg\min_i \left\{ |\nabla_i f(x_k)| - \frac{1}{2} \|a_i\|^2 \right\}.
$$

- Usually $L_i = \gamma \|a_i\|^2$, and exact GSL is nearest neighbours,

$$
\arg\min_i \left\| r(x_k) - \frac{a_i}{\|a_i\|} \right\|^2 = \arg\min_i \left\{ \frac{|\nabla_i f(x_k)|}{\sqrt{L_i}} \right\}.
$$
Important application of coordinate descent is for problems

\[
\min_{x \in \mathbb{R}^n} F(x) \equiv f(x) + \sum_i g_i(x_i),
\]

where \( f \) is smooth but \( g_i \) might be non-smooth.
Important application of coordinate descent is for problems

$$\min_{x \in \mathbb{R}^n} F(x) \equiv f(x) + \sum_i g_i(x_i),$$

where $f$ is smooth but $g_i$ might be non-smooth.

For example, $\ell_1$-regularization and/or bound constraints.
Proximal Coordinate Descent

- Important application of coordinate descent is for problems
  \[
  \min_{x \in \mathbb{R}^n} F(x) \equiv f(x) + \sum_i g_i(x_i),
  \]
  where \( f \) is smooth but \( g_i \) might be non-smooth.

- For example, \( \ell_1 \)-regularization and/or bound constraints.

- Richtárik and Takác [2014] show that
  \[
  \mathbb{E}[F(x^{k+1}) - F(x^k)] \leq \left( 1 - \frac{\mu}{Ln} \right) [F(x^k) - F(x^*)],
  \]
  the same rate as if non-smooth \( g_i \) was not there.
Important application of coordinate descent is for problems

\[
\min_{x \in \mathbb{R}^n} F(x) = f(x) + \sum_i g_i(x_i),
\]

where \( f \) is smooth but \( g_i \) might be non-smooth.

For example, \( \ell_1 \)-regularization and/or bound constraints.

Richtárik and Takác [2014] show that

\[
\mathbb{E}[F(x^{k+1}) - F(x^k)] \leq \left( 1 - \frac{\mu}{L_n} \right) [F(x^k) - F(x^*)],
\]

the same rate as if non-smooth \( g_i \) was not there.

Applies to exact minimization or proximal-gradient:

\[
x^{k+1} = \text{prox}_{\frac{1}{L} g_i} \left[ x^k - \frac{1}{L} \nabla_{i_k} f(x^k) e_{i_k} \right],
\]
Proximal Coordinate Descent

- Important application of coordinate descent is for problems

\[
\min_{x \in \mathbb{R}^n} F(x) \equiv f(x) + \sum_i g_i(x_i),
\]

where \(f\) is smooth but \(g_i\) might be non-smooth.

- For example, \(\ell_1\)-regularization and/or bound constraints.

- Richtárik and Takác [2014] show that

\[
\mathbb{E}[F(x^{k+1}) - F(x^k)] \leq \left(1 - \frac{\mu}{Ln}\right) [F(x^k) - F(x^*)],
\]

the same rate as if non-smooth \(g_i\) was not there.

- Applies to exact minimization or proximal-gradient:

\[
x^{k+1} = \text{prox}_{\frac{1}{L} g_{i_k}} \left[ x^k - \frac{1}{L} \nabla_{i_k} f(x^k) e_{i_k} \right],
\]

where

\[
\text{prox}_{\alpha g}[y] = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2} \|x - y\|^2 + \alpha g(x).
\]
There are several generalizations of GS to this setting:

- **GS-\( s \):** Minimize directional derivative,
  \[
  i_k = \arg\max_i \left\{ \min_{s \in \partial g_i} |\nabla_i f(x^k) + s| \right\}.
  \]
There are several generalizations of GS to this setting:

- **GS-\(s\):** Minimize directional derivative,
  
  \[ i_k = \argmax_i \left\{ \min_{s \in \partial g_i} |\nabla_i f(x^k) + s| \right\}. \]

  Used for \(\ell_1\)-regularization, but \(\|x^{k+1} - x^k\|\) could be tiny.

- **GS-\(r\):** Maximize how far we move,
  
  \[ i_k = \arg\max \left\{ \left| x_i^{k+1} - \text{prox}_1 Lg_i [x_i^k] \right| \right\}. \]

  Use for bound constraints, but ignores \(g_i(x^{k+1}) - g_i(x^k)\).

- **GS-\(q\):** Maximize progress under quadratic approximation of \(f\).
  
  \[ i_k = \arg\min \left\{ \min d f(x^k) + \nabla_i f(x^k) d + L d^2 + g_i(x^{k+1}) - g_i(x^k) \right\}. \]
There are several generalizations of GS to this setting:

- **GS-\(s\)**: Minimize directional derivative, 
  \[
  i_k = \argmax \left\{ \min_{s \in \partial g_i} |\nabla_i f(x^k) + s| \right\}.
  \]
  - Used for \(\ell_1\)-regularization, but \(\|x^{k+1}_i - x^k_i\| \) could be tiny.

- **GS-\(r\)**: Maximize how far we move, 
  \[
  i_k = \argmax \left\{ \left| x^k_i - \text{prox}_{\frac{1}{L} g_i} \left[ x^k_i - \frac{1}{L} \nabla_i f(x^k) \right] \right| \right\}.
  \]
  - Use for bound constraints, but ignores \(g_i(x^{k+1}_i) - g_i(x^k_i)\).
There are several generalizations of GS to this setting:

- **GS-ₘ:** Minimize directional derivative,

  \[ i_k = \argmax_i \left\{ \min_{s \in \partial g_i} |\nabla_i f(x^k) + s| \right\}. \]

  - Used for $\ell_1$-regularization, but $\|x^{k+1} - x^k\|$ could be tiny.

- **GS-ᵣ:** Maximize how far we move,

  \[ i_k = \argmax_i \left\{ \left\| x_i^k - \text{prox} \frac{1}{L} g_i \left[ x_i^k - \frac{1}{L} \nabla_i f(x^k) \right] \right\} \right\}. \]

  - Use for bound constraints, but ignores $g_i(x_i^{k+1}) - g_i(x_i^k)$.

- **GS-ₚ:** Maximize progress under quadratic approximation of $f$.

  \[ i_k = \argmin_i \left\{ \min_d f(x^k) + \nabla_i f(x^k)d + \frac{Ld^2}{2} + g_i(x_i^k + d) - g_i(x_i^k) \right\}. \]
There are several generalizations of GS to this setting:

- **GS-$s$:** Minimize directional derivative,
  
  \[ i_k = \arg\max_i \left\{ \min_{s \in \partial g_i} \left| \nabla_i f(x^k) + s \right| \right\}. \]
  
  Used for $\ell_1$-regularization, but $\|x^{k+1} - x^k\|$ could be tiny.

- **GS-$r$:** Maximize how far we move,
  
  \[ i_k = \arg\max_i \left\{ \left| x^k_i - \text{prox}_{\frac{1}{L} g_{i_k}} \left[ x^k_i - \frac{1}{L} \nabla_{i_k} f(x^k) \right] \right| \right\}. \]
  
  Use for bound constraints, but ignores $g_i(x^{k+1}_i) - g_i(x^k_i)$.

- **GS-$q$:** Maximize progress under quadratic approximation of $f$.
  
  \[ i_k = \arg\min_i \left\{ \min_d f(x^k) + \nabla_i f(x^k) d + \frac{Ld^2}{2} + g_i(x^k_i + d) - g_i(x^k_i) \right\}. \]
  
  Least intuitive, but has the best theoretical properties.
For the GS-$q$ rule, we show that

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

where $\epsilon_k \to 0$ as the algorithm converges.
For the GS-$q$ rule, we show that

$$f(x^{k+1}) - f(x^k) \leq \min \left\{ \left(1 - \frac{\mu}{L \bar{n}}\right) \left[f(x^k) - f(x^*)\right], \left(1 - \frac{\mu_1}{L}\right) \left[f(x^0) - f(x^*)\right] + \epsilon_k \right\},$$

where $\epsilon_k \to 0$ as the algorithm converges.

We conjecture that the above always holds with $\epsilon_k = 0$.

We also show GS-$s$ and GS-$r$ have worse rates than random.
For the GS-$q$ rule, we show that

$$f(x^{k+1}) - f(x^k) \leq \min \left\{ \left(1 - \frac{\mu}{Ln}\right) [f(x^k) - f(x^*)], \right. $$

$$\left. \left(1 - \frac{\mu_1}{L}\right) [f(x^0) - f(x^*)] + \epsilon_k \right\},$$

where $\epsilon^k \rightarrow 0$ as the algorithm converges.

We conjecture that the above always holds with $\epsilon_k = 0$.

We also show GS-$s$ and GS-$r$ have worse rates than random.

But again theory disagrees with practice...
All three rules seem to work pretty well,
All three rules seem to work pretty well, but

- We can make GS-\(s\) work badly with different initialization.
- GS-\(r\) works badly if you use the \(L_i\) to make a GSL-\(r\) rule.
- If you use the \(L_i\) to make a GSL-\(q\) rule, generalizes GSL.
• GS is not always practical.
  • But it is efficient for certain problems.
  • And it does converge faster than random.
Discussion

- GS is not always practical.
  - But it is efficient for certain problems.
  - And it does converge faster than random.
- We also gave better bounds for exact coordinate minimization.
  - Previously they were the same as using constant step size.
- GS is not always practical.
  - But it is efficient for certain problems.
  - And it does converge faster than random.
- We also gave better bounds for exact coordinate minimization.
  - Previously they were the same as using constant step size.
- We proposed GSL rule and analyzed non-smooth case.
- GS is not always practical.
  - But it is efficient for certain problems.
  - And it does converge faster than random.
- We also gave better bounds for exact coordinate minimization.
  - Previously they were the same as using constant step size.
- We proposed GSL rule and analyzed non-smooth case.
- Analysis extends to block updates.
- Could be used for accelerated/parallel methods [Fercocq & Richtárik, 2013], primal-dual methods [Shalev-Schwartz & Zhang, 2013], and without strong-convexity [Luo & Tseng, 1993].