Is Greedy Coordinate Descent a Terrible Algorithm?

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

University of British Columbia

Optimization and Big Data, 2015
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.

- Renewed theoretical interest began with Nesterov [2010]:
  - Global convergence rate for randomized coordinate selection.
  - Faster than gradient descent if iterations are $n$ times cheaper.
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:

\[
\begin{align*}
\text{h}_1(x) &= f(Ax) + \sum_{i=1}^n g_i(x_i), \\
\text{h}_2(x) &= \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i,x_j).
\end{align*}
\]

\( f \) and \( f_{ij} \) smooth

\( A \) is a matrix and \( f \) is cheap

\{ \( V, E \) \} is a graph

\( g_i \) general convex functions

Examples \( \text{h}_1 \): 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 \( \text{h}_2 \): 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 \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
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).

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

Coordinate update is \( n \) times faster than gradient update for:

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

- \( f \) 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 steepest descent rule:

$$\arg\max_i |\nabla_i f(x)|$$

also known as greedy rule or Gauss-Southwell (GS).

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 **steepest descent** rule:

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

  also known as **greedy rule** or **Gauss-Southwell (GS)**.
This talk:

- Instead of random, consider classic **steepest descent** rule:

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

also known as **greedy rule** or **Gauss-Southwell (GS)**.
This talk:

- Instead of random, consider classic **steepest descent** rule:

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

also known as **greedy rule** or **Gauss-Southwell (GS)**.

Nesterov showed the rate is the same. But this theory disagrees with practice...
This talk:

- Instead of random, consider classic **steepest descent** rule:

\[
\text{argmax} \left| \nabla_i f(x) \right|, \\
\text{Gauss-Southwell (GS)}
\]

also known as **greedy rule** or Gauss-Southwell (GS).
This talk:

- Instead of random, consider classic **steepest descent** rule:

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

also known as **greedy rule** or **Gauss-Southwell** (GS).

- GS is at least as expensive as random.
- But Nesterov showed the rate is the same.
- But this theory **disagrees** with practice...
If random and GS have similar costs, GS works much better.
If random and GS have similar costs, GS works much better.

This work: refined analysis of GS.
In general, GS rule may be as expensive as gradient even for $h_1$ and $h_2$.

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

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

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, 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 $\approx 200$. 
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.
Problems where can apply Gauss-Southwell

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

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

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

- 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,

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

- 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

$$\nabla_{ii}^2 f(x) \leq L, \quad \nabla^2 f(x) \succeq \mu I.$$
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 $i_k$. 

Since $\text{Ln} \geq L \geq L_f \geq \text{Ln}$, coordinate descent is slower per iteration, but $n$ coordinate iterations are faster than one gradient iteration.
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 $i_k$.

With $i_k$ chosen uniformly from $\{1, 2, \ldots, n\}$ [Nesterov, 2010],

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

With $i_k$ chosen uniformly from $\{1, 2, \ldots, n\}$ [Nesterov, 2010],
\[ \mathbb{E}[f(x^{k+1})] - f(x^*) \leq \left(1 - \frac{\mu}{Ln}\right) [f(x^k) - f(x^*)]. \]

Compare this to the rate of gradient descent,
\[ f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu}{Lf}\right) [f(x^k) - f(x^*)]. \]

Since $Ln \geq L_f \geq L$, coordinate descent is slower per iteration, but $n$ coordinate iterations are faster than one gradient iteration.
Classic Analysis of Gauss-Southwell Rule

- 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

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

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

- From strong-convexity we have

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

- Using \( \|\nabla f(x^k)\|^2 \leq n \|\nabla f(x^k)\|_\infty^2 \), 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].
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

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

- Using \( \| \nabla f(x^k) \|^2 \leq n \| \nabla f(x^k) \|^2_\infty \), 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].
To avoid norm inequality, measure strong-convexity in $1$-norm.
To avoid norm inequality, measure strong-convexity in 1-norm.

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.

We now have that

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

This gives a rate of

$$f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu_1}{L}\right) [f(x^k) - f(x^*)].$$
To avoid norm inequality, measure strong-convexity in 1-norm.

We now have that

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

This gives a rate of

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

The relationship between $\mu$ and $\mu_1$ is given by

$$\frac{\mu}{n} \leq \mu_1 \leq \mu.$$
To avoid norm inequality, measure strong-convexity in $1$-norm.

We now have that

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

This gives a rate of

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

The relationship between $\mu$ and $\mu_1$ is given by

$$\frac{\mu}{n} \leq \mu_1 \leq \mu.$$

GS bound is the same as random when $\mu_1 = \mu/n$.

Otherwise, GS can be faster by as large as $n$. 
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

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

If all $\lambda_i$ equal:

- There is no advantage to GS ($\mu_1 = \mu/n$).
In $f$ is a quadratic with diagonal Hessian, we can show

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

- 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

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

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

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

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.

$\mu_1$ is harmonic mean of $\lambda_i$ divided by $n$:
- $\mu_1$ is dominated by minimum of $\lambda_i$.
- If each worker takes $\lambda_i$ time to finish a task on their own, $\mu_1$ is time needed when ‘working together’ [Ferger, 1931].
Consider the linear-prediction framework in statistics,

$$\arg\min_{x, \beta} \sum_{i=1}^{n} f(a_i^T x + \beta) + \frac{\lambda}{2} \|x\|^2 + \frac{\sigma}{2} \beta^2,$$

where we’ve included a bias $\beta$. 
Consider the linear-prediction framework in statistics,

$$\arg\min_{x,\beta} \sum_{i=1}^{n} f(a_i^T x + \beta) + \frac{\lambda}{2} \|x\|^2 + \frac{\sigma}{2} \beta^2,$$

where we’ve included a bias $\beta$.

Typically $\sigma \ll \lambda$ to avoid biasing against a global shift.

This is an instance of $h_1$ where GS has the most benefit.
In many applications, can only approximate GS rule.
In many applications, can only approximate GS rule.

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 $\epsilon_k \to 0$,

$$f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu_1 (1 - \epsilon_k)^2}{L}\right) \left[f(x^k) - f(x^*)\right].$$
In many applications, can only approximate GS rule.

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 \( \epsilon_k \to 0 \),

\[ f(x_k^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu L_1 (1 - \epsilon_k)^2}{L}\right) [f(x_k) - f(x^*)]. \]

With additive error,

\[ |\nabla_i k f(x_k)| \geq \|\nabla f(x_k)\|_\infty - \epsilon_k, \]

we have a fast rate if \( \epsilon_k \to 0 \) fast enough.
What about **exact coordinate minimization**?
What about \textbf{exact coordinate minimization}?

We can get the same rates for exact optimization 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 known to be better for exact minimization.
What about exact coordinate minimization?

We can get the same rates for exact optimization because

\[ f(x^{k+1}) = \min_{\alpha} \{ f(x^k - \alpha \nabla_{i_k} f(x^k) e_{i_k}) \} \]

\[ \leq f \left( x^k - \frac{1}{L} \nabla_{i_i} f(x^k) e_{i_k} \right), \]

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

But theory again disagrees with practice:
Empirically, exact minimization is much faster.
Empirically, exact minimization is much faster.

Can we show that exact optimization gives a better bound?
Consider the case where we have an $L_i$ for each coordinate,

$$|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L_i |\alpha|.$$
Consider the case where we have an $L_i$ for each coordinate,

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

In this setting, exact coordinate optimization gives a rate of

$$f(x^k) - f(x^*) \leq \left[ \prod_{j=1}^{k} \left( 1 - \frac{\mu_1}{L_{i,j}} \right) \right] [f(x^0) - f(x^*)],$$

which is faster if $L_{i,k} < L$ (the maximum $L_i$) for any $i_k$. 

Worst case is $L_{i,k} = L$ for all $i$, even with distinct $L_i$. But this can never happen...
Consider the case where we have an $L_i$ for each coordinate,

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

In this setting, exact coordinate optimization gives a rate of

$$f(x^k) - f(x^*) \leq \left[ \prod_{j=1}^{k} \left( 1 - \frac{\mu_1}{L_{ij}} \right) \right] [f(x^0) - f(x^*)],$$

which is faster if $L_{i_k} < L$ (the maximum $L_i$) for any $i_k$.

Worst case is $L_{i_k} = L$ for all $i$, even with distinct $L_i$.

But this can never happen...
Key insight is that we never repeat same coordinate:

- After exact update, $\nabla_{i_k} f(x^{k+1}) = 0$. 

\[
\begin{bmatrix}
0.67 \\
-1.21 \\
0.72 \\
1.63 \\
0.49
\end{bmatrix}
\]

Worst case is alternating between largest two $L_i$. 
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 $i_{k+1} \neq i_k$. 

\[
\begin{pmatrix}
0 & -1 & 0 & 0 & 0 \\
0 & 0.67 & -1 & 0 & 0 \\
0 & 0 & 0.72 & 1.63 & 0 \\
0 & 0 & 0 & 0.49 & 0 \\
0 & 0 & 0 & 0 & 0.53
\end{pmatrix}
\]

Worst case is alternating between largest two $L_i$. 
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 \( i_{k+1} \neq i_k \).

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

- Worst case is alternating between largest two \( L_i \).
For **sparse** problems, **exact** can be much better:
For sparse problems, exact can be much better:

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.
For sparse problems, exact can be much better:

- 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.
For **sparse** problems, **exact** can be much better:

- 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.
For **sparse** problems, exact can be much better:

- 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.
For **sparse** problems, exact can be much better:

- 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.
For **sparse** problems, exact can be much better:

- 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.
For sparse problems, exact can be much better:

- 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.
For **sparse** problems, exact can be much better:

- 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.

Worst case cannot alternate between non-neighbouring $L_i$:

- For chain-structured graphs we can show

$$f(x^k) - f(x^*) = O(\rho_k^*)[f(x^0) - f(x^*)],$$

where $\rho_*$ is largest average among adjacent 2 or 3 nodes.

- Much faster if large $L_i$ at least than 2 nodes apart.
Can we use non-uniform randomized sampling?
Can we use non-uniform randomized sampling?

- Assume that we know the $L_i$ or approximate them.
- Nesterov [2010] shows that sampling proportional to $L_i$ yields

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

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

- Faster than uniform sampling when the $L_i$ are distinct.
Can we use non-uniform randomized sampling?

- Assume that we know the $L_i$ or approximate them.
- Nesterov [2010] shows that sampling proportional to $L_i$ yields

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

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

- Faster than uniform sampling when the $L_i$ are distinct.
- If we know gradients and $L_i$, should we use this or GS?
We obtain a faster rate than both by using $L_i$ in the GS rule,

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

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

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

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.
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 \( \mu_L \) satisfies the inequality

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

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 $\mu_L$ satisfies the inequality

$$\max \left\{ \frac{\mu}{nL}, \frac{\mu_1}{L} \right\} \leq \mu_L \leq \frac{\mu_1}{\min_i \{L_i\}},$$

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

GSL using $\frac{1}{L_{ik}}$ is unimprovable for quadratics,

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

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

where $\mu_L$ satisfies the inequality

$$\max \left\{ \frac{\mu}{nL}, \frac{\mu_1}{L} \right\} \leq \mu_L \leq \frac{\mu_1}{\min_i \{L_i\}},$$

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.
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

$$
\hat{i}_k = \arg\max_i |a_i^T r(x^k)|.
$$
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,

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

Approximation is exact if $\|a_i\| = 1$ for all $i$. 
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,

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

Approximation is exact if $\|a_i\| = 1$ for all $i$.

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

$$\arg\min_i \left\| r(x^k) - \frac{a_i}{\sqrt{\|a_i\|}} \right\|^2 = \arg\min_i \left\{ \frac{|\nabla_i f(x^k)|}{\sqrt{L_i}} \right\}.$$
Approximate GS is still faster than random sampling.
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 and $g_i$ might be non-smooth.

E.g., $\ell_1$-regularization or bound constraints.
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 and $g_i$ might be non-smooth.

E.g., $\ell_1$-regularization or bound constraints.
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 and \( g_i \) might be non-smooth.

E.g., \( \ell_1 \)-regularization or bound constraints.

Here we can apply proximal-gradient style of update,
\[
x^{k+1} = \text{prox}_{\frac{1}{L} g_i k} \left[ x^k - \frac{1}{L} \nabla_i f(x^k) e_i \right],
\]
where
\[
\text{prox}_{\alpha g}[y] = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2} \|x - y\|^2 + \alpha g(x).
\]
Proximal Coordinate Descent

- 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 and \( g_i \) might be non-smooth.

- E.g., \( \ell_1 \)-regularization or bound constraints.

- Here we can apply proximal-gradient style of update,

\[
x^{k+1} = \text{prox}_{\frac{1}{L}g_i} \left[ x^k - \frac{1}{L} \nabla f_i(x^k) e_i \right],
\]

where

\[
\text{prox}_{\alpha g}[y] = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2} \|x - y\|^2 + \alpha g(x).
\]

- 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.
There are several generalizations of GS to this setting:

- **GS-\(s\):** Minimize directional derivative,

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

- **GS-\(r\):** Maximize how far we move,

\[
    i_k = \mathop{\text{argmax}}_i \{ |x^i_k - \text{prox}_L g(x^i_k)| \}.
\]

Effective for bound constraints, but ignores \(g_i(x^i_k + 1) - g_i(x^i_k)\).

- **GS-\(q\):** Maximize progress under quadratic approximation of \(f\).

\[
    i_k = \mathop{\text{argmin}}_i \left\{ \min_{d \in \partial g_i} |\nabla f(x^k) + Ld| + g_i(x^i_k + d) - g_i(x^i_k) \right\}.
\]

Least intuitive, but has the best theoretical properties. Generalizes GSL if you use \(L_i\) instead of \(L\).
There are several generalizations of GS to this setting:

- **GS-\(s\):** Minimize directional derivative,

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

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

- **GS-\(r\):** Maximize how far we move,

  \[ i_k = \arg\max \{ |x^i_k - \text{prox}_{\frac{1}{L}} g_i[x^i_k - \frac{1}{L} \nabla_i f(x^k)]| \}. \]

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

- **GS-\(q\):** Maximize progress under quadratic approximation of \(f\).

  \[ i_k = \arg\min \left\{ \min d f(x^k) + \nabla_i f(x^k) d + \frac{L}{2} d^2 + g_i(x^k + d) - g_i(x^k) \right\}. \]

  Least intuitive, but has the best theoretical properties. Generalizes GSL if you use \(L_i\) instead of \(L\).
There are several generalizations of GS to this setting:

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

- **GS-r**: Maximize how far we move,
  \[
  i_k = \arg\max_i \left\{ \left| x_i^k - \text{prox}_{1\over L} g_{i_k} \left[ x_i^k - {1\over L} \nabla_i f(x^k) \right] \right| \right\}.
  \]
  - Effective for bound constraints, but ignores $g_i(x_i^{k+1}) - g_i(x_i^k)$.
Proximal Gauss-Southwell

There are several generalizations of GS to this setting:

- **GS-\(s\):** Minimize directional derivative,

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

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

- **GS-\(r\):** Maximize how far we move,

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

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

- **GS-\(q\):** Maximize progress under quadratic approximation of \(f\).

\[
    i_k = \arg\min_i \left\{ \min_d f(x^k) + \nabla_i f(x^k)d + \frac{Ld^2}{2} + g_i(x_i^k + d) - g_i(x_i^k) \right\}.
\]

  - Least intuitive, but has the best theoretical properties.
  - Generalizes GSL if you use \(L_i\) instead of \(L\).
For the GS-\( q \) rule, we show that

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

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

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

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

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

The above rate does not hold for GS-$s$ or GS-$r$.

(even if you change min to max)
For the GS-\(q\) rule, we show that

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

where \(\epsilon_k \to 0\) as the algorithm converges.

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

The above rate does not hold for GS-\(s\) or GS-\(r\).

(even if you change \(\min\) to \(\max\))

But again theory **disagrees** with practice:

- All three rules seem to work pretty well.
- Though GS-\(s\) works badly with poor initialization, and GS-\(r\) works badly if you use the \(L_i\).
Comparison of Proximal Gauss-Southwell Rules

\( \ell_1 \)-regularized underdetermined sparse least squares

Epochs

Objective

Random
Cyclic
Lipschitz
GS−q
GS−r
GS−s
GSL−q
GSL−r
GS not always practical.

But even approximate GS rules may outperform random.
Discussion

- GS not always practical.
- But even approximate GS rules may outperform random.
- We’ve given a justification for line-search in certain scenarios.
- We proposed GSL rule, and approximate/proximal variants.
GS not always practical.
But even approximate GS rules may outperform random.
We’ve given a justification for line-search in certain scenarios.
We proposed GSL rule, and approximate/proximal variants.
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].