Let's Make Block Coordinate Descent Go Fast!

Julie Nutini, Issam Laradji, Mark Schmidt and Warren Hare

University of British Columbia

EUROPT Workshop on Advances in Continuous Optimization
Montreal, Canada
July 12^{th}, 2017

Why Block Coordinate Descent?

- Block coordinate descent methods are key tools in large-scale optimization.
  - → Easy to implement.
  - → Low memory requirements.
  - → Cheap iteration costs.
  - → Adaptability to distributed settings.

Why Block Coordinate Descent?

- Block coordinate descent methods are key tools in large-scale optimization.
  - → Easy to implement.
  - → Low memory requirements.
  - → Cheap iteration costs.
  - → Adaptability to distributed settings.
- Used for almost two decades to solve LASSO and SVMs.

Why Block Coordinate Descent?

- Block coordinate descent methods are key tools in large-scale optimization.
  - → Easy to implement.
  - → Low memory requirements.
  - → Cheap iteration costs.
  - → Adaptability to distributed settings.
- Used for almost two decades to solve LASSO and SVMs.

→ **Any** improvements on convergence will affect many applications.

This Work

- We propose 4 ways to speed up Block Coordinate Descent (BCD) methods:

  1. New **greedy** block selection rules.
  2. New **second-order** update rule.
  3. New **exact** update rule for LASSO and SVMs.
  4. New **exact** update rule for graph-structured problems.

Block Coordinate Descent for Large-Scale Optimization

- We consider the basic convex optimization problem:

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

  where f is differentiable and n is large.

Block Coordinate Descent for Large-Scale Optimization

- We consider the basic convex optimization problem:

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

  where f is differentiable and n is large.

- At each iteration of the BCD algorithm, we
  - Select a block $b_k \subseteq \{1, 2, \ldots, n\}$.

Block Coordinate Descent for Large-Scale Optimization

- We consider the basic convex optimization problem:

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

  where f is differentiable and n is large.

- At each iteration of the BCD algorithm, we
  - Select a block $b_k \subseteq \{1, 2, \dots, n\}$.
  - Update iterate according to

$$x^{k+1} = x^k + U_{b_k} d^k,$$

Block Coordinate Descent for Large-Scale Optimization

- We consider the basic convex optimization problem:

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

  where f is differentiable and n is large.

- At each iteration of the BCD algorithm, we
  - Select a block $b_k \subseteq \{1, 2, \ldots, n\}$.
  - Update iterate according to

$$x^{k+1} = x^k + U_{b_k} d^k,$$

  where $d^k \in \mathbb{R}^M$ is a descent direction of the reduced dimensional subproblem,

$$\operatorname*{argmin}_{d \in \mathbb{R}^M} \; f(x^k + U_{b_k} d).$$

Block Coordinate Descent for Large-Scale Optimization

- We consider the basic convex optimization problem:

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

  where f is differentiable and n is large.

- At each iteration of the BCD algorithm, we
  - Select a block $b_k \subseteq \{1, 2, \ldots, n\}$.
  - Update iterate according to

$$x^{k+1} = x^k + U_{b_k} d^k,$$

    where $d^k \in \mathbb{R}^M$ is a descent direction of the reduced dimensional subproblem,

$$\operatorname*{argmin}_{d \in \mathbb{R}^M} \ f(x^k + U_{b_k} d).$$

→ E.g., gradient descent update $d^k = -\alpha_k \nabla_{b_k} f(x^k)$ for some $\alpha_k > 0$.

Block Selection Rules

- There are 4 common block selection strategies:
  - → **Cyclic**: Repeatedly cycle through blocks in order.

Block Selection Rules

- There are 4 common block selection strategies:
  - → **Cyclic**: Repeatedly cycle through blocks in order.
  - → **Random**: Uniformly sample the blocks (tries to avoid bad ordering).

Block Selection Rules

- There are 4 common block selection strategies:
  - → **Cyclic**: Repeatedly cycle through blocks in order.
  - → **Random**: Uniformly sample the blocks (tries to avoid bad ordering).
  - → **Greedy**: Choose “best” block according to criteria (too expensive in general).

Block Selection Rules

- There are 4 common block selection strategies:
  - → **Cyclic**: Repeatedly cycle through blocks in order.
  - → **Random**: Uniformly sample the blocks (tries to avoid bad ordering).
  - → **Greedy**: Choose “best” block according to criteria (too expensive in general).
  - → **Lipschitz**: Use Lipschitz constants of gradients to improve over random.

Block Selection Rules

- There are 4 common block selection strategies:
  - → **Cyclic**: Repeatedly cycle through blocks in order.
  - → **Random**: Uniformly sample the blocks (tries to avoid bad ordering).
  - → **Greedy**: Choose "best" block according to criteria (too expensive in general).
  - → **Lipschitz**: Use Lipschitz constants of gradients to improve over random.

- Assume that f is L_b-block-wise Lipschitz continuous,

$$\|\nabla_b f(x + U_b d) - \nabla_b f(x)\| \leq L_b \|d\|, \quad \text{for all } d.$$

  - → If f is twice-differentiable, this is equivalent to $\nabla^2_{bb} f(x) \preceq L_b \mathbb{I}$ for each block b.

Greedy Block Selection Rules

- There are 2 common greedy selection rules:
  - → **Gauss-Southwell (GS)**: Choose block with biggest gradient.

Greedy Block Selection Rules

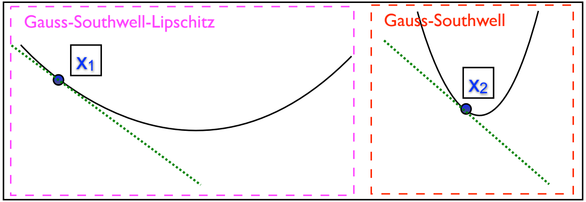
- There are 2 common greedy selection rules:
  - → **Gauss-Southwell (GS)**: Choose block with biggest gradient.
  - → **Maximum improvement (MI)**: Choose block that makes the most progress.

Greedy Block Selection Rules

- There are 2 common greedy selection rules:
  - → **Gauss-Southwell (GS)**: Choose block with biggest gradient.
  - → **Maximum improvement (MI)**: Choose block that makes the most progress.
- MI is usually too costly.

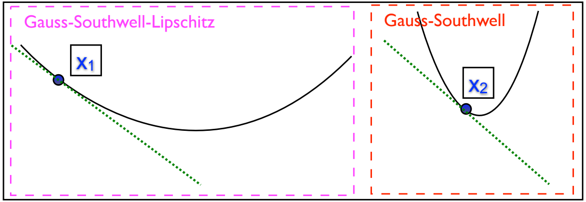
Greedy Block Selection Rules

- There are 2 common greedy selection rules:
  - → **Gauss-Southwell (GS)**: Choose block with biggest gradient.
  - → **Maximum improvement (MI)**: Choose block that makes the most progress.
- MI is usually too costly.
- Nutini et. al. (2015) showed GS can be efficiently calculated for some problem structures and also introduced the **Gauss-Southwell-Lipschitz** (GSL) rule.



Greedy Block Selection Rules

- There are 2 common greedy selection rules:
  - → **Gauss-Southwell (GS)**: Choose block with biggest gradient.
  - → **Maximum improvement (MI)**: Choose block that makes the most progress.
- MI is usually too costly.
- Nutini et. al. (2015) showed GS can be efficiently calculated for some problem structures and also introduced the **Gauss-Southwell-Lipschitz** (GSL) rule.



  - → Incorporates Lipschitz information in the rule.
  - → Equivalent to MI for quadratics.

Block Gauss-Southwell-Lipschitz

- As an obvious extension of the GSL rule to the block setting, we propose the **Block Gauss-Southwell-Lipschitz** (BGSL) rule:

$$b_k \in \underset{b \in \mathcal{B}}{\text{argmax}} \frac{\|\nabla_b f(x^k)\|_2}{\sqrt{L_b}}.$$

Block Gauss-Southwell-Lipschitz

- As an obvious extension of the GSL rule to the block setting, we propose the **Block Gauss-Southwell-Lipschitz** (BGSL) rule:

$$b_k \in \operatorname*{argmax}_{b \in \mathcal{B}} \frac{\|\nabla_b f(x^k)\|_2}{\sqrt{L_b}}.$$

- Derived by minimizing quadratic bound from block-wise Lipschitz continuity.
- Guarantees more progress than the block GS rule.

Block Gauss-Southwell-Lipschitz

- As an obvious extension of the GSL rule to the block setting, we propose the **Block Gauss-Southwell-Lipschitz** (BGSL) rule:

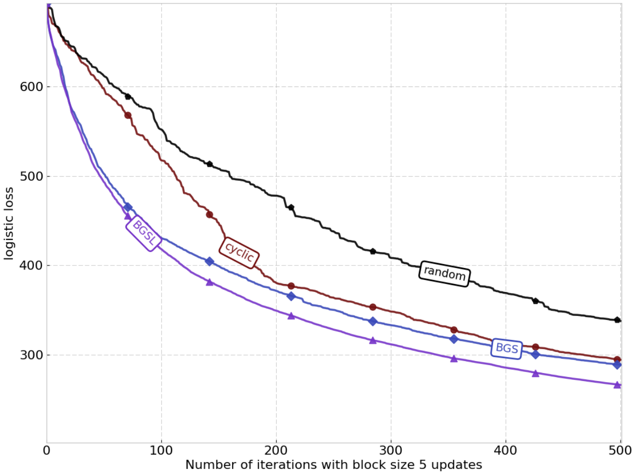
$$b_k \in \underset{b \in \mathcal{B}}{\text{argmax}} \frac{\|\nabla_b f(x^k)\|_2}{\sqrt{L_b}}.$$

- Derived by minimizing quadratic bound from block-wise Lipschitz continuity.
- Guarantees more progress than the block GS rule.

→ Unlike GSL, not equivalent to the MI rule for quadratic functions.

Experiment: L2-Regularized Logistic Regression

- Comparing block selection rules using fixed blocks.



Block Gauss-Southwell-Quadratic

- If we are updating more than one variable, we can obtain a better bound by measuring Lipschitz continuity using quadratic norms,

Block Gauss-Southwell-Quadratic

- If we are updating more than one variable, we can obtain a better bound by measuring Lipschitz continuity using quadratic norms,

$$\|\nabla_b f(x + U_b d) - \nabla_b f(x)\|_{H_b^{-1}} \leq \|d\|_{H_b} = \sqrt{d^T H_b d},$$

where $H_b = \beta H$ for any positive definite matrix H, sufficiently large β.

Block Gauss-Southwell-Quadratic

- If we are updating more than one variable, we can obtain a better bound by measuring Lipschitz continuity using quadratic norms,

$$\|\nabla_b f(x + U_b d) - \nabla_b f(x)\|_{H_b^{-1}} \leq \|d\|_{H_b} = \sqrt{d^T H_b d},$$

  where $H_b = \beta H$ for any positive definite matrix H, sufficiently large β.
- Simply changes measure of continuity $\rightarrow$ no new assumptions.

Block Gauss-Southwell-Quadratic

- If we are updating more than one variable, we can obtain a better bound by measuring Lipschitz continuity using quadratic norms,

$$\|\nabla_b f(x + U_b d) - \nabla_b f(x)\|_{H_b^{-1}} \leq \|d\|_{H_b} = \sqrt{d^T H_b d},$$

  where $H_b = \beta H$ for any positive definite matrix H, sufficiently large β.
- Simply changes measure of continuity $\rightarrow$ no new assumptions.

- **Block Gauss-Southwell-Quadratic** (BGSQ):

$$b_k \in \underset{b \in \mathcal{B}}{\text{argmax}} \left\{ \|\nabla_b f(x^k)\|_{H_b^{-1}} \right\}$$

$\rightarrow$ H_b is a global upper bound on Hessian $\nabla^2_{bb} f$.

Block Gauss-Southwell-Quadratic

- If we are updating more than one variable, we can obtain a better bound by measuring Lipschitz continuity using quadratic norms,

$$\|\nabla_b f(x + U_b d) - \nabla_b f(x)\|_{H_b^{-1}} \leq \|d\|_{H_b} = \sqrt{d^T H_b d},$$

  where $H_b = \beta H$ for any positive definite matrix H, sufficiently large β.
- Simply changes measure of continuity $\rightarrow$ no new assumptions.

- **Block Gauss-Southwell-Quadratic** (BGSQ):

$$b_k \in \underset{b \in \mathcal{B}}{\text{argmax}} \left\{ \|\nabla_b f(x^k)\|_{H_b^{-1}} \right\}$$

$\rightarrow$ H_b is a global upper bound on Hessian $\nabla^2_{bb} f$.

$\rightarrow$ Equivalent to the MI rule for quadratics.

Block Gauss-Southwell-Quadratic

- If we are updating more than one variable, we can obtain a better bound by measuring Lipschitz continuity using quadratic norms,

$$\|\nabla_b f(x + U_b d) - \nabla_b f(x)\|_{H_b^{-1}} \leq \|d\|_{H_b} = \sqrt{d^T H_b d},$$

  where $H_b = \beta H$ for any positive definite matrix H, sufficiently large β.
- Simply changes measure of continuity → no new assumptions.

- **Block Gauss-Southwell-Quadratic** (BGSQ):

$$b_k \in \underset{b \in \mathcal{B}}{\operatorname{argmax}} \left\{ \|\nabla_b f(x^k)\|_{H_b^{-1}} \right\}$$

→ H_b is a global upper bound on Hessian $\nabla^2_{bb} f$.

→ Equivalent to the MI rule for quadratics.

→ May be difficult to find Hessian bounds H_b, depends on how we define blocks.

Blocking Strategy

- There are 2 main strategies used to define a set of possible blocks:

Blocking Strategy

- There are 2 main strategies used to define a set of possible blocks:

  → **Fixed**: Partition variables into groups, select amongst these groups.

Blocking Strategy

- There are 2 main strategies used to define a set of possible blocks:

  → **Fixed**: Partition variables into groups, select amongst these groups.

  → **Variable**: Choose “best” M variables at each step, no pre-defined groups.

Blocking Strategy

- There are 2 main strategies used to define a set of possible blocks:

  → **Fixed**: Partition variables into groups, select amongst these groups.

  → **Variable**: Choose "best" M variables at each step, no pre-defined groups.

- Variable blocks guarantee more progress.
  - We show it is NP-hard to compute BGSL, BGSQ.

Blocking Strategy

- There are 2 main strategies used to define a set of possible blocks:

  → **Fixed**: Partition variables into groups, select amongst these groups.

  → **Variable**: Choose "best" M variables at each step, no pre-defined groups.

- Variable blocks guarantee more progress.
  - We show it is NP-hard to compute BGSL, BGSQ.
  - In practice, use approximate Block Gauss-Southwell-Diagonal rule,

$$b_k \in \underset{b \in \mathcal{B}}{\text{argmax}} \left\{ \sum_{i \in b} \left(\frac{\nabla_i f(x^k)}{D_i} \right)^2 \right\}.$$

Blocking Strategy

- There are 2 main strategies used to define a set of possible blocks:

  $\rightarrow$ **Fixed**: Partition variables into groups, select amongst these groups.

  $\rightarrow$ **Variable**: Choose “best” M variables at each step, no pre-defined groups.

- Variable blocks guarantee more progress.
  - We show it is NP-hard to compute BGSL, BGSQ.
  - In practice, use approximate Block Gauss-Southwell-Diagonal rule,

$$b_k \in \underset{b \in \mathcal{B}}{\text{argmax}} \left\{ \sum_{i \in b} \left(\frac{\nabla_i f(x^k)}{D_i} \right)^2 \right\}.$$

  $\rightarrow$ Each coordinate has its own step size $1/D_i$ (same across all blocks b).

Blocking Strategy

- There are 2 main strategies used to define a set of possible blocks:

  → **Fixed**: Partition variables into groups, select amongst these groups.

  → **Variable**: Choose "best" M variables at each step, no pre-defined groups.

- Variable blocks guarantee more progress.
  - We show it is NP-hard to compute BGSL, BGSQ.
  - In practice, use approximate Block Gauss-Southwell-Diagonal rule,

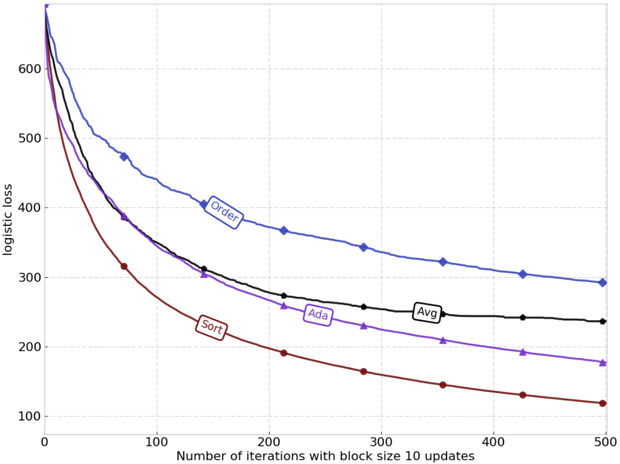
$$b_k \in \operatorname*{argmax}_{b \in \mathcal{B}} \left\{ \sum_{i \in b} \left(\frac{\nabla_i f(x^k)}{D_i} \right)^2 \right\}.$$

  → Each coordinate has its own step size $1/D_i$ (same across all blocks b).

- Fixed blocks we could use Lipschitz constants to help determine the partition.

Experiment: L2-Regularized Logistic Regression

- Comparing block partitioning strategies using BGSD rule.



Block Update Rules

- Assume that we have selected a block b_k.
- We now focus on how we define our update d^k.

Block Update Rules

- Assume that we have selected a block b_k.
- We now focus on how we define our update d^k.

→ **Gradient-style update**:

$$d^k = -\alpha_k \nabla_{b_k} f(x^k),$$

where $\alpha_k > 0$ is either constant or determined using a line search.

Block Update Rules

- Assume that we have selected a block b_k.
- We now focus on how we define our update d^k.

→ **Gradient-style update**:

$$d^k = -\alpha_k \nabla_{b_k} f(x^k),$$

where $\alpha_k > 0$ is either constant or determined using a line search.

→ **Hessian-bound udpate**:

$$d^k = -(H_{b_k})^{-1} \nabla_{b_k} f(x^k),$$

where H_{b_k} is a positive-definite global upper bound on the Hessian $\nabla^2_{bb} f$.

Block Update Rules

- Assume that we have selected a block b_k.
- We now focus on how we define our update d^k.

  → **Gradient-style update**:

  $$d^k = -\alpha_k \nabla_{b_k} f(x^k),$$

  where $\alpha_k > 0$ is either constant or determined using a line search.

  → **Hessian-bound udpate**:

  $$d^k = -(H_{b_k})^{-1} \nabla_{b_k} f(x^k),$$

  where H_{b_k} is a positive-definite global upper bound on the Hessian $\nabla^2_{bb} f$.

- Do better updates exist?

Block Update Rules

- Assume that we have selected a block b_k.
- We now focus on how we define our update d^k.

$\rightarrow$ **Gradient-style update**:

$$d^k = -\alpha_k \nabla_{b_k} f(x^k),$$

where $\alpha_k > 0$ is either constant or determined using a line search.

$\rightarrow$ **Hessian-bound udpate**:

$$d^k = -(H_{b_k})^{-1} \nabla_{b_k} f(x^k),$$

where H_{b_k} is a positive-definite global upper bound on the Hessian $\nabla^2_{bb} f$.

- Do better updates exist? Yes!

Why Not Newton?

- Why do we expect to develop better updates than the Hessian-bound update?

  $\rightarrow$ Uses an upper-bound on the Hessian everywhere.

Why Not Newton?

- Why do we expect to develop better updates than the Hessian-bound update?

  $\rightarrow$ Uses an upper-bound on the Hessian everywhere.

  $\rightarrow$ For non-quadratic functions, potentially make more progress by using instantaneous Hessian $\rightarrow$ cheap for a block.

Why Not Newton?

- Why do we expect to develop better updates than the Hessian-bound update?

  $\rightarrow$ Uses an upper-bound on the Hessian everywhere.

  $\rightarrow$ For non-quadratic functions, potentially make more progress by using instantaneous Hessian $\rightarrow$ cheap for a block.

- Classic Newton-style steps need line-search or trust-region method.

Why Not Newton?

- Why do we expect to develop better updates than the Hessian-bound update?

  $\rightarrow$ Uses an upper-bound on the Hessian everywhere.

  $\rightarrow$ For non-quadratic functions, potentially make more progress by using instantaneous Hessian $\rightarrow$ cheap for a block.

- Classic Newton-style steps need line-search or trust-region method.

$\rightarrow$ However, these:

  - Require more implementation effort.
  - Have tuning parameters.
  - Require extra evaluations of the function, which may be expensive.

Why Not Newton?

- Why do we expect to develop better updates than the Hessian-bound update?

  $\rightarrow$ Uses an upper-bound on the Hessian everywhere.

  $\rightarrow$ For non-quadratic functions, potentially make more progress by using instantaneous Hessian $\rightarrow$ cheap for a block.

- Classic Newton-style steps need line-search or trust-region method.

$\rightarrow$ However, these:

  - Require more implementation effort.
  - Have tuning parameters.
  - Require extra evaluations of the function, which may be expensive.

$\rightarrow$ We consider a Newton-style method based on a cubic regularization framework.

Cubic Regularization Updates

- While gradient-style methods are based on a quadratic upper-bound,

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), d^k \rangle + \frac{L}{2}\|d^k\|^2,$$

Cubic Regularization Updates

- While gradient-style methods are based on a quadratic upper-bound,

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), d^k \rangle + \frac{L}{2}\|d^k\|^2,$$

  Newton-like cubic regularization methods are based on a cubic upper-bound,

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), d^k \rangle + (d^k)^T \nabla^2 f(x^k) d^k + \frac{M}{6}\|d^k\|^3.$$

Cubic Regularization Updates

- While gradient-style methods are based on a quadratic upper-bound,

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), d^k \rangle + \frac{L}{2}\|d^k\|^2,$$

  Newton-like cubic regularization methods are based on a cubic upper-bound,

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), d^k \rangle + (d^k)^T \nabla^2 f(x^k) d^k + \frac{M}{6}\|d^k\|^3.$$

- By defining next iterate as minimizer of the bound (or minimum among both), we can incorporate the instantaneous Hessian.

Cubic Regularization Updates

- While gradient-style methods are based on a quadratic upper-bound,

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), d^k \rangle + \frac{L}{2}\|d^k\|^2,$$

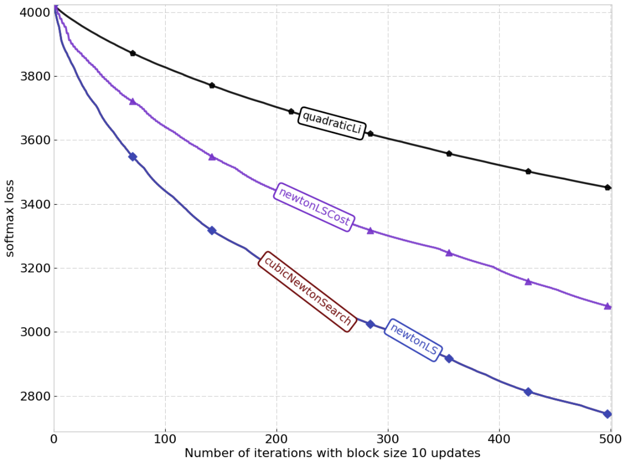
  Newton-like cubic regularization methods are based on a cubic upper-bound,

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), d^k \rangle + (d^k)^T \nabla^2 f(x^k) d^k + \frac{M}{6}\|d^k\|^3.$$

- By defining next iterate as minimizer of the bound (or minimum among both), we can incorporate the instantaneous Hessian.
  - → Guaranteed to decrease the objective without needing extra objective function evaluations required for a line search.

Experiment: Multi-class Logistic Regression

- Comparing update rules using variable blocks with greedy block selection.



Superlinear Convergence?

- Looks like Newton's method, which has superlinear convergence!

Superlinear Convergence?

- Looks like Newton's method, which has superlinear convergence!
- Can we achieve superlinear convergence in general?

Superlinear Convergence?

- Looks like Newton's method, which has superlinear convergence!
- Can we achieve superlinear convergence in general?
- No, not even with exact updates.
  - E.g., 2-variable non-separable quadratic

Superlinear Convergence?

- Looks like Newton's method, which has superlinear convergence!
- Can we achieve superlinear convergence in general?
- No, not even with exact updates.
  - E.g., 2-variable non-separable quadratic

$\rightarrow$ Possible to get superlinear convergence for problems with certain structures.

Superlinear Convergence for L1-Regularized Problems

- Consider minimizing a differentiable function f with L1-regularization,

$$\min_x F(x) := f(x) + \lambda\|x\|_1,$$

  - E.g., LASSO: $F(x) = \frac{1}{2]}\|Ax - b\|^2 + \lambda\|x\|_1$ for $\lambda > 0$.

Superlinear Convergence for L1-Regularized Problems

- Consider minimizing a differentiable function f with L1-regularization,

$$\min_{x} F(x) := f(x) + \lambda\|x\|_1,$$

  - E.g., LASSO: $F(x) = \frac{1}{2]}\|Ax - b\|^2 + \lambda\|x\|_1$ for $\lambda > 0$.

→ Using the BCD method with:

  - Variable blocks to choose the “best” M variables to update at each step.
  - Greedy BGS-q selection rule (or variations like those presented earlier).

Superlinear Convergence for L1-Regularized Problems

- Consider minimizing a differentiable function f with L1-regularization,

$$\min_{x} F(x) := f(x) + \lambda\|x\|_1,$$

  - E.g., LASSO: $F(x) = \frac{1}{2]}\|Ax - b\|^2 + \lambda\|x\|_1$ for $\lambda > 0$.

→ Using the BCD method with:

  - Variable blocks to choose the "best" M variables to update at each step.
  - Greedy BGS-q selection rule (or variations like those presented earlier).

→ We prove identification of the sparsity pattern of x^* for **finite** k.

Superlinear Convergence for L1-Regularized Problems

- Consider minimizing a differentiable function f with L1-regularization,

$$\min_x F(x) := f(x) + \lambda\|x\|_1,$$

  - E.g., LASSO: $F(x) = \frac{1}{2]}\|Ax - b\|^2 + \lambda\|x\|_1$ for $\lambda > 0$.

→ Using the BCD method with:
  - Variable blocks to choose the "best" M variables to update at each step.
  - Greedy BGS-q selection rule (or variations like those presented earlier).

→ We prove identification of the sparsity pattern of x^* for **finite** k.
  - Similar results are known for cyclic and random selection.

Superlinear Convergence for L1-Regularized Problems

- Consider minimizing a differentiable function f with L1-regularization,

$$\min_x F(x) := f(x) + \lambda\|x\|_1,$$

  - E.g., LASSO: $F(x) = \frac{1}{2]}\|Ax - b\|^2 + \lambda\|x\|_1$ for $\lambda > 0$.

→ Using the BCD method with:

  - Variable blocks to choose the "best" M variables to update at each step.
  - Greedy BGS-q selection rule (or variations like those presented earlier).

→ We prove identification of the sparsity pattern of x^* for **finite** k.

  - Similar results are known for cyclic and random selection.
  - **BUT** with greedy, if $M >$ nnz(x^*), then after identifying the sparsity pattern, the cubic update is equivalent to cubic-regularized Newton on the non-zeroes...

Superlinear Convergence for L1-Regularized Problems

- Consider minimizing a differentiable function f with L1-regularization,

$$\min_x F(x) := f(x) + \lambda\|x\|_1,$$

  - E.g., LASSO: $F(x) = \frac{1}{2]}\|Ax - b\|^2 + \lambda\|x\|_1$ for $\lambda > 0$.

$\rightarrow$ Using the BCD method with:

  - Variable blocks to choose the "best" M variables to update at each step.
  - Greedy BGS-q selection rule (or variations like those presented earlier).

$\rightarrow$ We prove identification of the sparsity pattern of x^* for **finite** k.

  - Similar results are known for cyclic and random selection.
  - **BUT** with greedy, if $M >$ nnz(x^*), then after identifying the sparsity pattern, the cubic update is equivalent to cubic-regularized Newton on the non-zeroes...

$\rightarrow$ SUPERLINEAR CONVERGENCE!

Exact Updates for LASSO and SVMs

- For LASSO and SVMs, possible to do **exact block updates** using homotopy methods.
  - Roughly cost $O(M^3) \rightarrow$ efficient for updating thousands of variables at once.

Exact Updates for LASSO and SVMs

- For LASSO and SVMs, possible to do **exact block updates** using homotopy methods.
  - Roughly cost $O(M^3) \rightarrow$ efficient for updating thousands of variables at once.
  - Once we identify nnz(x^*), using $M >$ nnz(x^*) and exact updates yields exact optimal solution...

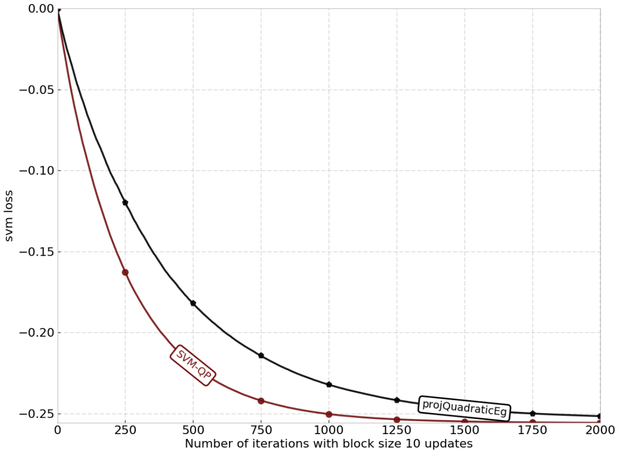
Exact Updates for LASSO and SVMs

- For LASSO and SVMs, possible to do **exact block updates** using homotopy methods.
  - Roughly cost $O(M^3) \rightarrow$ efficient for updating thousands of variables at once.
  - Once we identify nnz(x^*), using $M >$ nnz(x^*) and exact updates yields exact optimal solution...

$\rightarrow$ FINITE TERMINATION!

Experiment: Dual SVM

- Comparing update methods using variable blocks with greedy selection.



Message-Passing for Sparse Quadratics

- Given the amazing properties we get for LASSO/SVMs with exact updates, are there other problems that allow efficient exact updates?

Message-Passing for Sparse Quadratics

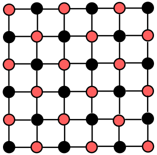
- Given the amazing properties we get for LASSO/SVMs with exact updates, are there other problems that allow efficient exact updates?
- Obvious choice is quadratics: can do exact updates in $O(M^3)$.
  - Allows big-but-not-too-big blocks.

Message-Passing for Sparse Quadratics

- Given the amazing properties we get for LASSO/SVMs with exact updates, are there other problems that allow efficient exact updates?
- Obvious choice is quadratics: can do exact updates in $O(M^3)$.
  - Allows big-but-not-too-big blocks.
  - $\rightarrow$ For **sparse** quadratics we can often do exact updates for much larger M.
  - E.g., Quadratic with lattice-structured non-zero pattern.

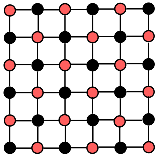
Message-Passing for Sparse Quadratics

- Given the amazing properties we get for LASSO/SVMs with exact updates, are there other problems that allow efficient exact updates?
- Obvious choice is quadratics: can do exact updates in $O(M^3)$.
  - Allows big-but-not-too-big blocks.
  - $\rightarrow$ For **sparse** quadratics we can often do exact updates for much larger M.
  - E.g., Quadratic with lattice-structured non-zero pattern.
    - Classic red-black ordering.
    - Allows blocks of size $n/2$ for $O(n)$, but loses dependencies.



Message-Passing for Sparse Quadratics

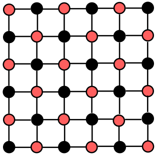
- Given the amazing properties we get for LASSO/SVMs with exact updates, are there other problems that allow efficient exact updates?
- Obvious choice is quadratics: can do exact updates in $O(M^3)$.
  - Allows big-but-not-too-big blocks.
  - → For **sparse** quadratics we can often do exact updates for much larger M.
  - E.g., Quadratic with lattice-structured non-zero pattern.



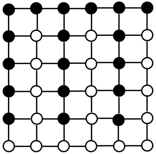
    - Classic red-black ordering.
    - Allows blocks of size $n/2$ for $O(n)$, but loses dependencies.
  - → Exploit connection to Gaussian Markov random fields, update tree-structured blocks in $O(M)$ using Gaussian belief propagation.

Message-Passing for Sparse Quadratics

- Given the amazing properties we get for LASSO/SVMs with exact updates, are there other problems that allow efficient exact updates?
- Obvious choice is quadratics: can do exact updates in $O(M^3)$.
  - Allows big-but-not-too-big blocks.
  - $\rightarrow$ For **sparse** quadratics we can often do exact updates for much larger M.
  - E.g., Quadratic with lattice-structured non-zero pattern.
    - Classic red-black ordering.
    - Allows blocks of size $n/2$ for $O(n)$, but loses dependencies.

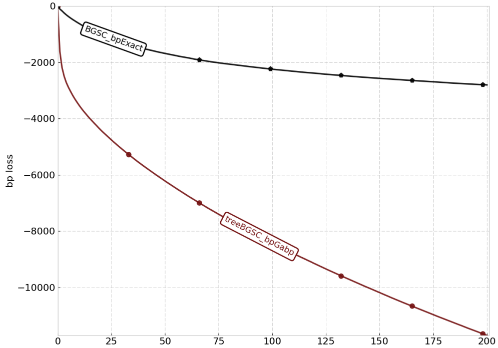


  - $\rightarrow$ Exploit connection to Gaussian Markov random fields, update tree-structured blocks in $O(M)$ using Gaussian belief propagation.
    - For lattice-structured graphs, can use blocks of size $n/2$ in $O(n)$.
    - Maintains modelling dependencies.



Experiment: Sparse Quadratic Problem

- Comparing exact updates using variable blocks with greedy selection.



- Exact solver uses $M = 8$, Gaussian belief propagation method uses $M = 8^3$.
- NP-hard to choose best “tree-structure” block.
  - $\rightarrow$ Use approximation method that performs substantially better than BGSD.

Discussion

- Propose several greedy block selection rules for fixed and variable blocks.

Discussion

- Propose several greedy block selection rules for fixed and variable blocks.
- Propose cubic regularization update.
  - Uses instantaneous Hessian.
  - Achieves superlinear convergence for problems with certain structure.

Discussion

- Propose several greedy block selection rules for fixed and variable blocks.
- Propose cubic regularization update.
  - Uses instantaneous Hessian.
  - Achieves superlinear convergence for problems with certain structure.
- Propose using exact homotopy update rule for LASSO and SVMs.
  - Greedy block updates have “active set” identification property for LASSO, SMVs.
  - Superlinear convergence with variable blocks and higher order updates.
  - Finite convergence with variable blocks and exact updates.

Discussion

- Propose several greedy block selection rules for fixed and variable blocks.
- Propose cubic regularization update.
  - Uses instantaneous Hessian.
  - Achieves superlinear convergence for problems with certain structure.
- Propose using exact homotopy update rule for LASSO and SVMs.
  - Greedy block updates have “active set” identification property for LASSO, SMVs.
  - Superlinear convergence with variable blocks and higher order updates.
  - Finite convergence with variable blocks and exact updates.
- Propose optimal block update strategy for sparse quadratic problems.
  - Use “tree-structured” blocks.
  - Exploits Gaussian belief propagation algorithm developed for GMRFs.
  - Requires linear time in block size.