Coordinate Descent Survey

He Wang

School of Information Science and Technology
ShanghaiTech University

July 31, 2021

Outline

Coordinate Descent

Introduction to CD

- Solve an optimization problem by solving a sequence of simpler (lower dimensional, even scalar) optimization problems.
- Core idea: The cost is minimized along one (block) coordinate(s) at each iteration.
- Motivations/Advantages:
  - Reduce the cost of each iteration, both in generating the search direction and in performing the update of variables.
  - Reduce the sensitivity of asynchronization.

The Problem Settings of CD

- General structured formulation:

$$\min_x h(x) := f(x) + \lambda\Omega(x) \tag{1}$$

  - f: L_f-smooth; L_i-component-wise Lipschitz continous gradient, i.e.

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq L_i|h| \quad x \in \mathbb{R}^n,\ h \in \mathbb{R},\ i = 1, ...n$$

  - Ω: regularization function, may be nonsmooth but often assume to be separable or block separable: $\Omega(x) = \sum_{i=1}^n \Omega_i(x_i)$
  - Note: $L_{\max} \leq L \leq nL_{\max}$, where $L_{\max} = \max_{1,...,n} L_i$

The Procedures of Coordinate Descent

- Initialization: Choose any $x^0 = (x_1^0, ..., x_N^0) \in \mathbf{dom} f$
- Iteration $r+1$, $r > 0$. Given $x^r = (x_1^r, ..., x_N^r) \in \mathbf{dom} f$
- Choose an index $s \in \{1, ..., N\}$ and compute a new iterate

$$x^{r+1} = (x_1^{r+1}, ..., x_N^{r+1}) \in \mathbf{dom} f$$

satisfying

$$x_s^{r+1} \in \arg\min_{x_s} f(x_1^r, ..., x_{s-1}^r, x_s, ..., x_N^r),$$

$$x_j^{r+1} = x_j^r, \forall j \neq s$$

Coordinate Update Rules

- Cyclic CD: Gauss-Seidel fashion.
- Randomized CD: The coordinates are selected at random.
- Greedy CD: Choose the coordinate(s) with the best values, for example, a coordinate of the gradient vector with the maximal absolute value.
- Mixed CD: Mix the strategy above three.

Cyclic CD

- Be closely to the Gauss-Seidel for equation solving, such as

$$Q\alpha = \mathbf{b}$$

Gauss-Seidel iterations take the following form.

$$\alpha_i^{r+1} = \frac{b_i - \sum_{j=1}^{i-1} Q_{ij}\alpha_j^{r+1} - \sum_{j=i+1}^{l} Q_{ij}\alpha_j^r}{Q_{ii}}$$

Further, α_i^{r+1} is the solution of

$$\min_{\alpha \in \mathbf{R}^t} \frac{1}{2}\alpha^T Q\alpha - \mathbf{b}^T\alpha$$

over α_i, while fixing $\alpha_1^{r+1}, ..., \alpha_{i-1}^{r+1}, \alpha_{i+1}^r, ..., \alpha_l^r$.

The Outline of Cyclic CD

Algorithm 1 Coordinate Descent Algorithm of Cyclic Version

1: Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
2: **for** $k = 0$ to N **do**
3: $\quad x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$
4: $\quad i_{k+1} = [i_k \bmod n] + 1$
5: **end for**

- Starting from the simple problem: $\Omega(x) = 0$.
- More general, it can be required to satisfy an "essentially cyclic" condition, in which for some $T > n$, each component is modified at least once in every stretch of T iterations, i.e.

$$\cup_{j=0}^{T} \{i_{k-j}\} = \{1, 2, ..., n\}, \text{for all } k \geq T$$

Randomized CD

- Key Idea of RCD: The coordinates are selected at random.
- Key Idea of SGD: Minimize a smooth function f by taking a (negative) step along a unbaised estimate g^k of the gradient $\nabla f(x^k)$.
- Randomized CD can be viewed as a special case of SGD methods, in which $g^k = n[\nabla f(x^k)]_{i_k} e_{i_k}$
- CD algorithms have the advantage over general SGD: descent in f can be guaranteed at every iteration.

The Outline of Randomized CD

Algorithm 2 Coordinate Descent Algorithm of Randomized Version

1: Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
2: **for** $k = 0$ to N **do**
3: Choose index $i_k \in \{1, 2, ..n\}$ randomly;
4: $x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$
5: **end for**

- Starting from the simple problem: $\Omega(x) = 0$.

The Outline for Randomized CD

Algorithm 3 Coordinate Descent Algorithm of Randomized Version

1: Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
2: **while** termination test satisfied **do**
3: Choose index $i_k \in \{1, 2, ..n\}$ randomly;
4: $z_{i_k}^k \leftarrow \arg\min_{\mathcal{X}} (\mathcal{X} - x_{i_k}^k)^T [\nabla f(x^k)]_{i_k} + \frac{1}{2\alpha_k} \| \mathcal{X} - x_{i_k}^k \|_2^2 + \lambda \Omega_i(\mathcal{X})$ for some $\alpha_k > 0$
5: $x^{k+1} \leftarrow x^k + (z_{i_k}^k - x_{i_k}^k) e_{i_k}$
6: $k \leftarrow k + 1$
7: **end while**

- Make a linear approximation of f along the i_k coordinate direction at current iterate ,add a quadratic damping term and tread the Ω_i explicitly.

The Outline of Greedy CD

Algorithm 4 Coordinate Descent Algorithm of Greedy Version

1: Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
2: **for** $k = 0$ to N **do**
3: Choose index $i_k = \arg\max_{1 \leq i \leq} |\nabla_i f(x_k)|$;
4: $x^{k+1} \leftarrow x^k - \frac{1}{M}[\nabla f(x^k)]_{i_k} e_{i_k}$
5: **end for**

- Starting from the simple problem: $\Omega(x) = 0$.
- Extensive works used GCD in sparse optimization problems.

The Convergence of Greedy CD

- Easy to prove the convergence of the above algorithm. $f(x_k) - f^* \leq \frac{2nMR^2}{k+4}$, for $k \geq 0$ and where $R \geq \|x_0 - x^*\|$
- Require computation of the $whole$ gradient vector.
- If f is L-smooth, then $M > O(L(f))$.
- The convergence rate of simple Gradient Method is better.
- The work [1] shows that GCD has faster convergence than RCD in theory, if f is M coordinate-wise smooth and μ-strongly convex in the 1-norm.

Mixed CD

- Mix the above three methods: Cylic, Randomized, Greedy
- Example: Block-Greedy CD [2] partitions the coordinates into B blocks and randomly selects P blocks, within each of which a coordinate is selected for update in a greedy manner.

Parallel Coordinate Descent

The General Framework for PCD Methods [3]

Algorithm 5 GenCD

1: **while** not converged **do**
2: Select a set of coordinates J
3: Propose increment $\delta_j, j \in J$ {parallel}
4: Accpet some subset $J' \subset J$ of the proposals
5: Update weight w_j for all $j \in J'$ {parallel}
6: **end while**

PCDM [4]: Parallel Version of Coordinate Descent Method

- Problem Formulation:

$$f^* = \min_{\mathbf{u}_1 \in \mathbf{U}_1, \ldots, \mathbf{u}_M \in \mathbf{U}_M} f(\mathbf{u}^1, \ldots, \mathbf{u}^M) \tag{2}$$

- Assume that the gradient of f is coordinate-wise Lipschitz continous with constants $L_i > 0$.

PCDM [4]: Parallel Version of Coordinate Descent Method

Algorithm 6 PCDM

1: **while** not converged **do**
2: Choose $u_0^i \in U^i \quad \forall i = 1, ...M$
3: Compute in parallel:

$$\hat{v}^i(u) = \arg\min_{v^i \in U^i} \langle \nabla_i f(u), v^i - u^i \rangle + \frac{L_i}{2} \|v^i - u^i\|^2$$

4: Update in parallel: $u_{k+1}^i = \frac{1}{M} \hat{v^i}(u_k) + \frac{M-1}{M} u_k^i, \; i = 1, ..., M$
5: **end while**

- Parallelization of Algorithm 3
- However, not "Coordinate Descent" enough.

GRock[5]: A Greedy Block Coordinate Descent Method

- Problem Formulation:

$$\min_{x \in \mathcal{R}^n} F(\mathbf{x}) = \lambda R(\mathbf{x}) + L(\mathbf{Ax}, \mathbf{b}) \tag{3}$$

  - $R(\mathbf{x})$: a separable regularizer, $R(\mathbf{x}) = \sum_{i=1}^{n} r(\mathbf{x}_i)$.
  - $L(\mathbf{Ax}, \mathbf{b})$: smooth and convex, assuming that

$$L(\mathbf{A}(\mathbf{x}+\mathbf{d}), \mathbf{b}) \leq L(\mathbf{Ax}, \mathbf{b}) + \mathbf{g}^T\mathbf{d} + \frac{\beta}{2}\mathbf{d}^T\mathbf{A}^T\mathbf{Ad}$$

  where $\mathbf{g} = \mathbf{A}^T\nabla\mathbf{L}(\mathbf{Ax}, \mathbf{b})$.

- Update Rules: Greedy Strategy + Parallel Scenario

GRock[5]: A Greedy Block Coordinate Descent Method

Procedures:

1. Divide the coordinates into N blocks.
2. At each iteration, out of the N blocks, the best coordinate of each of the best P blocks are updated.
   - What does **Best** coordinate means?
     - $d_i = \arg\min_d \lambda \cdot r(x_i + d) + g_i d + \frac{\beta}{2} d^2$
     - $m_i = \max\{|d| : d \text{ is an element of } \mathbf{d_i}\}$
     - Best coordinate s_i: such $m_i = d_{s_i}$
3. Repeat Step 2 until converge.

GRock[5]: A Greedy Block Coordinate Descent Method

- Convergence Analysis:
- Without additional conditions, GRock is guaranteed to converge only for $P = 1$.
- Define a block spectral radius: $\rho_P = \max_{\mathbf{M} \in \mathcal{M}} \rho(\mathbf{M})$
- Assume $R(x) = \|\mathbf{x}\|_1$, $\rho_P \leq 2$, function value converges at $O(1/k)$

Distributed Coordinate Descent

Hydra[6]: Hybrid Coordinate Descent

- Problem Formulation:

$$\min_{x \in \mathbb{R}^d} L(x) := f(x) + R(x) \tag{4}$$

  - $f(x)$: smooth and convex, assuming that

$$f(x+h) \leq f(x) + f'(x)^T h + \frac{1}{2} h^T \mathbf{M} h$$

    and where $\mathbf{M}$ is positive definite matrix
  - $R(x)$: possibly nonsmooth, convex and separable ($R(\mathbf{x}) = \sum_{i=1}^{n} r(\mathbf{x}_i)$)
- Update Rules: Random Strategy + Distributed Scenario (with Master Node)

Hydra[6]: Hybrid Coordinate Descent

Procedures:

1. Partition d coordinates into c sets $\mathcal{P}_1, ..., \mathcal{P}_c$ of equal cardinality, $s := d/c$, and assign set $\mathcal{P}_l$ to node l.
2. Set $x_0 \in \mathbb{R}^d$, $k \leftarrow 0$
3. For each node $l \in \{1, ..., c\}$ in parallel:
   - Pick a random subset $S_l \subset \mathcal{P}_l$ with $|S_l| = \tau$.
   - For each coordinate $i \in S_l$ in parallel:
     - $h_k^i \leftarrow \arg\min_t f_i'(x_k)t + \frac{M_{ii}\beta}{2}t^2 + R_i(x_k^i + t)$
     - Apply the update: $x_{k+1}^i \leftarrow x_{k+1}^i + h_k^i$
   - $k \leftarrow k + 1$

Hydra[6]: Hybrid Coordinate Descent

Convergence Analysis:

- Depend on its partition.
- Neither f nor R is strongly convex: $O(\frac{s\beta}{\tau\epsilon})$
- L is strongly convex: linear convergence

Limitation:

- Need master nodes to compute the gradients.
- No communication with neighborhoods.

Hydra^2 [7]: Fast version of Hydra

Algorithm 1 Hydra^2

INPUT: $\{\mathcal{P}_l\}_{l=1}^c$, $1 \leqslant \tau \leqslant s$, $\{\mathbf{D}_{ii}\}_{i=1}^d$, $z_0 \in \mathbb{R}^d$
set $\theta_0 = \tau/s$ and $u_0 = 0$
for $k \geqslant 0$ **do**
  $z_{k+1} \leftarrow z_k$, $u_{k+1} \leftarrow u_k$
  for each computer $l \in \{1, \ldots, c\}$ **in parallel do**
    pick a random set of coordinates $\hat{S}_l \subseteq \mathcal{P}_l$, $|\hat{S}_l| = \tau$
    for each $i \in \hat{S}_l$ **in parallel do**
      $t_k^i = \operatorname{argmin}_t f_i'(\theta_k^2 u_k + z_k)t + \frac{s\theta_k \mathbf{D}_{ii}}{2\tau} t^2 + R_i(z_k^i + t)$
      $z_{k+1}^i \leftarrow z_k^i + t_k^i$, $u_{k+1}^i \leftarrow u_k^i - (\frac{1}{\theta_k^2} - \frac{s}{\tau\theta_k})t_k^i$
    end parallel for
  end parallel for
  $\theta_{k+1} = \frac{1}{2}(\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2)$
end for
OUTPUT: $\theta_k^2 u_{k+1} + z_{k+1}$

- Idea: Hydra + FISTA
- Convergence Rate: $O(1/k^2)$

DCD-Lasso [8]: Coordinate Descent Distributed Lasso

Problem Formulation:

- Lasso estimator in a distributed fashion:

$$\hat{\beta}_{Lasso} = \arg\min_{\beta_0,\beta} \frac{1}{2} \sum_{j=1}^{J} \|\mathbf{y}_j - \mathbf{1}_N \beta_0 - \mathbf{X}_j \beta\|_2^2 + \lambda \|\beta\|_1 \tag{5}$$

- A consensus-based reformulation of the Lasso:

$$\{\hat{\beta}_j\}_{j=1}^{J} =: \arg\min_{\{\beta_j\}} \frac{1}{2} \sum_{j=1}^{J} [\|\mathbf{y}_j - \mathbf{X}_j \beta_j\|_2^2 + \frac{2\lambda}{J} \|\beta_j\|_1] \tag{6}$$

  s.t. $\beta_j = \beta_{j'}$, $j \in \mathcal{J}$, $j' \in \mathcal{N}_j$

- Update Rules: Cyclic Strategy + Distributed Scenario

DCD-Lasso [8]: Coordinate Descent Distributed Lasso

The Outline of its deduction of update equations:

- Add auxiliary local variables $\gamma := \{\{\hat{\gamma}_j^{j'}{}_{j' \in \mathcal{N}_j}, \{\bar{\gamma}_j^{j'}\}_{j' \in \mathcal{N}_j}\}$ to (6)
- Then, form the quadratically augmented Lagrangian function.
- Obtain that

$$\mathbf{p}_j(k) = \mathbf{p}_j(k-1) + c \sum_{j' \in \mathcal{N}_j} [\beta_j(k) - \beta_{j'}(k)] \tag{7}$$

$$\begin{aligned} \hat{\beta}_j(k+1) =: \arg\min_{\beta_j} \{ & \frac{1}{2} \|\mathbf{y}_j - \mathbf{X}_j \beta_j\|_2^2 + \frac{\lambda}{J} \|\beta_j\|_1 \\ & + \mathbf{p}_j^T(k) \beta_j + c \sum_{j' \in \mathcal{N}_j} \|\beta_j - \frac{\beta_j(k) + \beta_{j'}(k)}{2}\|_2^2 \} \end{aligned} \tag{8}$$

DCD-Lasso [8]: Coordinate Descent Distributed Lasso

Procudures:

- All agents $j \in \mathcal{J}$ initialize to zero $\{\beta_j(0), \mathbf{p}_j\}$, and locally run
- For k=0,... do
  - Transmit $\beta_j(k)$ to neighbors in $\mathcal{N}_j$.
  - Update $\mathbf{p}_j(k)$ via (7).
  - for i =1,...p do
    - Update

$$[\beta_j(k+1)]_i = (2c|\mathcal{N}_j| + \|x_{ji}\|^2)^{-1}$$
$$\times \mathcal{S}(\mathbf{x}_{ji}^T \mathbf{y}_j^{-i} + [c \sum_{j' \in \mathcal{N}_j} \beta_j(k) + \beta_{j'}(k) - \mathbf{p_j}(k)]_i, \frac{\lambda}{J})$$

Convergent but no convergence rate provided.

Dual Coordinate Ascent

CoCoA [9]: Communication-Efficient Distributed Dual Coordinate Ascent

Problem Formulation:

- Primal Problem: a convex loss function of linear predictors with a convex regularization term:

$$\min_{w \in \mathbb{R}^d} [P(\mathbf{w}) := \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} l_i(\mathbf{w})^T \mathbf{x}_i]$$

- Dual Problem:

$$\max_{\alpha \in \mathbb{R}^n} [D(\alpha) := -\frac{\lambda}{2}\|A\alpha\|^2 - \frac{1}{n}\sum_{i=1}^{n} l_i^*(-\alpha_i)]$$

  - The data matrix $A \in \mathbb{R}^{d \times n}$: collects the (normalized) data examples $A_i := \frac{1}{\lambda n} x_i$ and $\mathbf{w}(\alpha) = A\alpha$

CoCoA: Communication-Efficient Distributed Dual Coordinate Ascent

Algorithm 1: CoCoA: Communication-Efficient Distributed Dual Coordinate Ascent

Input: $T \geq 1$, scaling parameter $1 \leq \beta_K \leq K$ (default: $\beta_K := 1$).
Data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ distributed over K machines
Initialize: $\boldsymbol{\alpha}_{[k]}^{(0)} \leftarrow \mathbf{0}$ for all machines k, and $\boldsymbol{w}^{(0)} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots, T$
  for ***all machines*** $k = 1, 2, \ldots, K$ ***in parallel***
    $(\Delta\boldsymbol{\alpha}_{[k]}, \Delta\boldsymbol{w}_k) \leftarrow \textsc{LocalDualMethod}(\boldsymbol{\alpha}_{[k]}^{(t-1)}, \boldsymbol{w}^{(t-1)})$
    $\boldsymbol{\alpha}_{[k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[k]}^{(t-1)} + \frac{\beta_K}{K} \Delta\boldsymbol{\alpha}_{[k]}$
  end
  reduce $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \frac{\beta_K}{K} \sum_{k=1}^K \Delta\boldsymbol{w}_k$
end

Figure: CoCoA

CoCoA: Communication-Efficient Distributed Dual Coordinate Ascent

- Advantages:
  1. Significant reduction in commuication cost comes with only a very moderate increase in the amount of total computation.
  2. More general, since the inner optimizer can use any dual optimization method.
  3. No strong assumptions on the data like the data is orthogonal between the different works.
- Disadvantages:
  1. Need a master node.
  2. Just for convex linear predictors with a convex regularization term (2-norm).

Combine with Other Problems

Parameter Servers [10]

- A General regularized optimization problem:

$$\min_{w} F(w) := f(w) + h(w)$$

  - Loss function f: continuously differentiable but not necessarily convex
  - Regularizer h: convex, left side continous, block separable, but possibly non-smooth.
- Update Rules: Cyclic Strategy + Distributed Scenario (with master node).

Parameter Servers

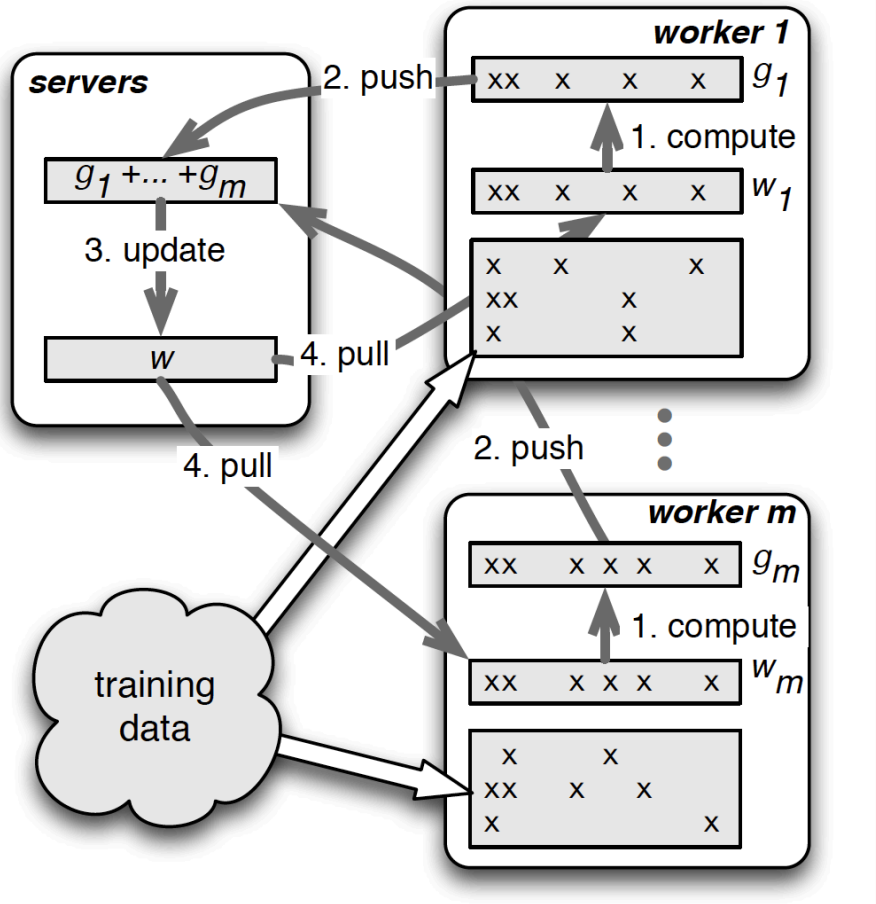

Figure: The Parameter Server Architecture

- Workers: store only a portion of the training data, compute the local gradients or other statistics and only communicate with severs.
- Server: global parameters and communicate with other severs and belonged workers.

Delayed Block Proximal Gradient Method

- Proximal Gradient Method:
  - $w^{t+1} = \text{Prox}_{\eta_t}^{U}[w^t - \eta_t \nabla f(w^t)]$, for $t = 1, 2, ...$
  - $\text{Prox}_{\eta_t}^{U}(x) := \arg\min_{y \in \mathbf{R}^p} h(y) + \frac{1}{2\eta}\|x - y\|_U^2$
  - A forward step performing steepest gradient descent on f.
  - A backward step carrying out projection using h.
- Bounded Delay: To solve the inconsistency.

Delayed Block Proximal Gradient Method

Algorithm 2 Delayed Block Proximal Gradient Method Solving (1)

Scheduler:

1: Partition parameters into k blocks $b_1, \ldots, b_k$
2: **for** $t = 1$ **to** T**:** Pick a block b_{i_t} and issue the task to workers

Worker r at iteration t

1: Wait until all iterations before $t - \tau$ are finished
2: Compute first-order gradient $g_r^{(t)}$ and coordinate-specific learning rates $u_r^{(t)}$ on block b_{i_t}
3: Push $g_r^{(t)}$ and $u_r^{(t)}$ to servers with user-defined filters, e.g., the random skip or the KKT filter
4: Pull $w_r^{(t+1)}$ from servers with user-defined filters, e.g., the significantly modified filter

Servers at iteration t

1: Aggregate $g^{(t)}$ and $u^{(t)}$
2: Solve the generalized proximal operator (2) $w^{(t+1)} \leftarrow \text{Prox}_{\gamma_t}^{U}(w^{(t)})$ with $U = \text{diag}(u^{(t)})$.

Figure: Delayed Block Proximal Gradient Method

References I

Nutini, J., Schmidt, M., Laradji, I., Friedlander, M. & Koepke, H. *Coordinate descent converges faster with the gauss-southwell rule than random selection.* in *International Conference on Machine Learning* (2015), 1632–1641.

Li, Y. & Osher, S. Coordinate descent optimization for l1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Problems and Imaging* **3,** 487–503 (2009).

Scherrer, C., Tewari, A. & Haglin, D. Scaling Up Coordinate Descent Algorithms for Large1 Regularization Problems. (2012).

Necoara, I. & Clipici, D. Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed MPC. *Journal of Process Control* **23,** 243–253 (2013).

References II


Peng, Z., Yan, M. & Yin, W. *Parallel and distributed sparse optimization.* in *2013 Asilomar conference on signals, systems and computers* (2013), 659–646.


Richtárik, P. & Takáč, M. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research* **17,** 2657–2681 (2016).


Fercoq, O., Qu, Z., Richtárik, P. & Takáč, M. *Fast distributed coordinate descent for non-strongly convex losses.* in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)* (2014), 1–6.


Mateos, G., Bazerque, J. A. & Giannakis, G. B. Distributed sparse linear regression. *IEEE Transactions on Signal Processing* **58,** 5262–5276 (2010).

References III

Jaggi, M. *et al.* *Communication-efficient distributed dual coordinate ascent.* in *Advances in neural information processing systems* (2014), 3068–3076.

Li, M., Andersen, D. G., Smola, A. J. & Yu, K. *Communication efficient distributed machine learning with the parameter server.* in *Advances in Neural Information Processing Systems* (2014), 19–27.

The End