

Part 4: Coordinate Descent

Mário A. T. Figueiredo¹ and Stephen J. Wright²

¹Instituto de Telecomunicações,
Instituto Superior Técnico, Lisboa, Portugal

²Computer Sciences Department,
University of Wisconsin,
Madison, WI, USA

ICCOPT, Lisbon, Portugal, July 2013

Coordinate Descent

Consider first $\min f(x)$ with $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

Iteration j of basic coordinate descent:

- Choose index $i_j \in \{1, 2, \dots, n\}$;
- Fix all components $i \neq i_j$, change x_{i_j} in a way that (hopefully) reduces f .

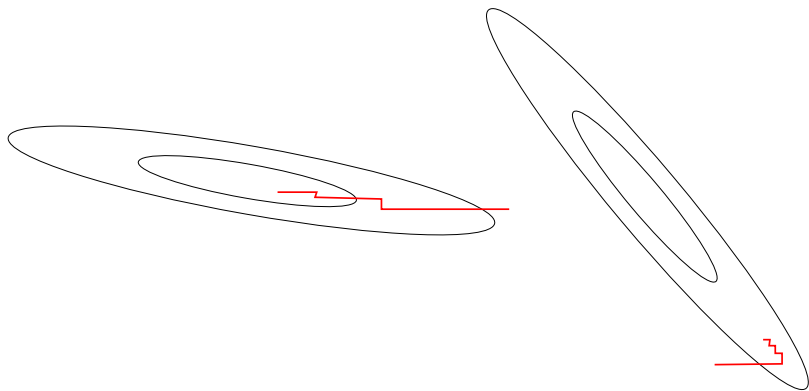
Variants for the reduced step:

- take a reduced gradient step: $-\nabla_{i_j} f(x)$;
- do a more rigorous search in the subspace defined by i_j ;
- actually minimize f in the i_j component.

Many extensions of this basic idea, discussed below.

An old approach, revived recently because of useful application to machine learning, and interesting possibilities as stochastic and parallel algorithms.

Coordinate Descent: Alternating Directions



Even if the subproblem is solved *exactly* in each coordinate, convergence can be slow. It does better when eigenvalues of the Hessian $\nabla^2 f(x)$ line up with the principal axes.

At iteration j , choose a subset $\mathcal{G}_j \subset \{1, 2, \dots, n\}$ and allow only the components in \mathcal{G}_j to change. Fix the components x_i for $i \notin \mathcal{G}_j$.

Again, the step could be a reduced gradient step along $-\nabla_{\mathcal{G}_j} f(x)$, or a more elaborate search.

There are many different heuristics for choosing \mathcal{G}_k (see below), often exploiting knowledge of the application.

Constraints and Regularizers Complicate Things

For $\min_{x \in \Omega} f(x)$, need to put enough components into \mathcal{G}_j to stay feasible, as well as make progress in reducing f .

Example: $\min f(x_1, x_2)$ with $x_1 + x_2 = 1$. Basic coordinate relaxation does not work from a feasible point!

(Given x_1 and x_2 satisfying the constraint, suppose we fix x_2 and minimize f over x_1 , maintaining feasibility. Because $x_1 = 1 - x_2$, we cannot move!)

For separable regularizer (e.g. Group LASSO) with

$$\psi(x) = \sum_{i \in G} \psi_i(x_{[i]}),$$

need to ensure that \mathcal{G}_k is a union of the some index subsets $[g]$. i.e. the relaxation components must be consonant with the partitioning.

We'll mostly drop the regularization terms in the description below, and assume that we work with smooth objective f . But we mention from time to time the extensions to constraints and regularizers.

Decomposition and Dual SVM

Decomposition has long been popular for solving the dual (QP) formulation of SVM, since the number of variables (N = number of training examples) may be very large.

$$\min_{\alpha \in \mathbb{R}^N} \frac{1}{2} \alpha^T K \alpha - \mathbf{1}^T \alpha \quad \text{s.t.} \quad 0 \leq \alpha \leq C \mathbf{1}, \quad y^T \alpha = 0.$$

SMO: Each \mathcal{G}_k has two components. (Thus can maintain feasibility with respect to the single linear constraint $y^T \alpha = 0$.)

LIBSVM: SMO approach (still $|\mathcal{G}_k| = 2$), with different heuristic.

LASVM: Again $|\mathcal{G}_k| = 2$, with focus on online setting.

SVM-light: Small $|\mathcal{G}_k|$ (default 10).

GPDT: Larger $|\mathcal{G}_k|$ (default 400) with gradient projection solver as the subproblem solver.

Choice of \mathcal{G}_k and Convergence Results

Some methods (e.g. Tseng and Yun, 2010) require \mathcal{G}_k to be chosen so that *the improvement in subproblem objective obtained over the subset \mathcal{G}_k is at least a fixed fraction of the improvement available over the whole space*. Undesirable, since to check it, usually need to evaluate the **full gradient** $\nabla f(x_k)$.

Alternative is a *generalized Gauss-Seidel* requirement, where each coordinate is “touched” at least once every T iterations:

$$\mathcal{G}_k \cup \mathcal{G}_{k+1} \cup \dots \cup \mathcal{G}_{k+T-1} = \{1, 2, \dots, n\}.$$

Can show global convergence (e.g. Tseng and Yun, 2009; Wright, 2010).

There are also results on

- global linear convergence rates
- optimal manifold identification
- fast local convergence for an algorithm that takes reduced steps on the estimated optimal manifold.

All are *deterministic* analyses.

Stochastic Coordinate Descent (SCD)

(Richtarik and Takac, 2012) (Nesterov, 2012)

Consider the basic (single-coordinate) setting, for $\min_x f(x)$ with smooth f .

Define L_i to be the component Lipschitz constant:

$$\|\nabla_i f(x + te_i) - \nabla_i f(x)\| \leq L_i t.$$

(Can think of L_i as a bound on the (i, i) component of the Hessian $\nabla^2 f(x)$ in the region of interest.)

Iteration j :

- Choose $i_j \in \{1, 2, \dots, n\}$ with equal probability;
- Set $x_{j+1} = x_j - e_{i_j} \nabla_{i_j} f(x_j) / L_{i_j}$.

This is **short-step steepest descent in the i_j component**.

SCD Convergence

For convex f , have **high probability convergence** of f to within a **specified threshold ϵ** of $f(x^*)$ in $O(1/\epsilon)$ iterations.

Given desired precision ϵ and error prob ρ , define

$$K := \frac{2n\mathcal{R}_L^2(x_0)}{\epsilon} \log \frac{f(x_0) - f^*}{\epsilon\rho},$$

(where $\mathcal{R}_L^2(x_0)$ is the radius of the level set in weighted norm $\|\cdot\|_L$). Have high-probability convergence in K iterations:

$$P(f(x_j) - f^* \leq \epsilon) \geq 1 - \rho, \quad \text{for } j \geq K.$$

If f is **strongly convex** with respect to $\|\cdot\|_L$, with modulus μ_L in this norm, there is expected convergence at a **linear rate**:

$$E[f(x_j) - f^*] \leq \left(1 - \frac{\mu_L}{4n}\right)^j (f(x_0) - f^*).$$

Extension to Blocks

It's straightforward. For partition of $\{1, 2, \dots, n\}$ into $[1], [2], \dots, [m]$, redefine L_i to be the block Lipschitz constant for block i :

$$\|\nabla_{[i]} f(x + U_i t) - \nabla_{[i]} f(x)\| \leq L_i \|t\|.$$

Iteration j :

- Choose block $i_j \in \{1, 2, \dots, m\}$ with equal probability;
- Set $x_{j+1} = x_j - E_{i_j} \nabla_{[i_j]} f(x_j) / L_{i_j}$.

(E_i is the matrix that adds 0 to the components $i \notin i_j$, to get a full vector in \mathbb{R}^n .)

Convergence results are similar to coordinate case, if we define L -weighted norm:

$$\|x\|_L := \left(\sum_{i=1}^m L_i \|x_{[i]}\|^2 \right)^{1/2}$$

and weighted measure of level set size:

$$\mathcal{R}_L(x) := \max_y \max_{x^* \in X^*} \{\|y - x^*\|_W : f(y) \leq f(x)\}.$$

Extension to Separable Regularizers

Consider regularized problem

$$\min_x f(x) + \psi(x).$$

For the given block partition $[1], [2], \dots, [m]$, suppose the regularizer is separable:

$$\psi(x) = \sum_{i=1}^m \psi_i(x_{[i]}).$$

For partition i , find the step by solving:

$$P(x, i) : \min_d d^T \nabla_{[i]} f(x) + \frac{L_i}{2} \|d\|^2 + \psi_i(x_{[i]} + d).$$

Iteration j :

- Choose partition $i_j \in \{1, 2, \dots, m\}$ with equal probability;
- Solve $P(x_j, i_j)$ to obtain step $d_{[i_j]}$;
- Set $x_{j+1} = x_j + E_{i_j} d_{[i_j]}$.

Stochastic Coordinate Descent: Key Constants

To prepare for a parallel asynchronous variant of SCD, consider again $\min f(x)$, where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is smooth and convex.

For simplicity, describe single-coordinate case (not blocks).

Need some more constants that characterize the problem:

- $L_{\max} = \max_{i=1,2,\dots,n} L_i$ (“max diagonal of Hessian”);
- L_{res} = restricted Lipschitz constant: $\|\nabla f(x) - \nabla f(x + te_i)\| \leq L_{\text{res}}t$ (“max row-norm of Hessian”)
- $R = \sup_k \text{dist}(x_k, \mathcal{S})$: maximum distance of iterates from solution set.

Diagonality of Hessian

The ratio $L_{\text{res}}/L_{\text{max}}$ is particularly important — it measures the degree of diagonal dominance in the Hessian $\nabla^2 f(x)$ (**Diagonality**).

We have

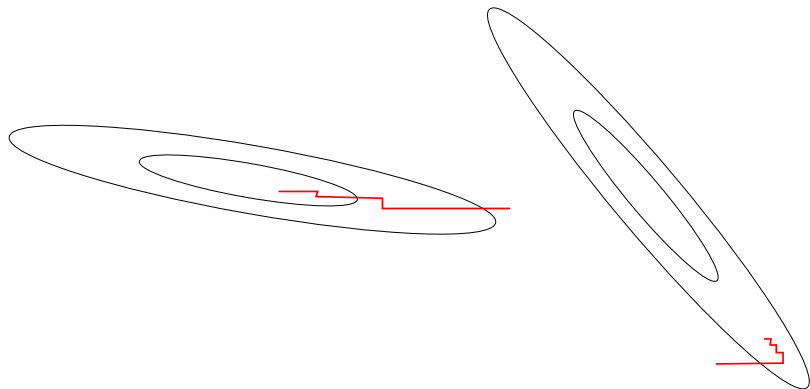
$$1 \leq \frac{L_{\text{res}}}{L_{\text{max}}} \leq \sqrt{n}.$$

- Closer to 1 if Hessian is nearly diagonally dominant (eigenvectors close to principal coordinate axes).
- Closer to \sqrt{n} otherwise.

If A is $m \times n$ Gaussian random matrix and $f(x) = (1/2)\|Ax - b\|_2^2$, the ratio is $1 + O(\sqrt{n/m})$ (good case!)

Smaller $L_{\text{res}}/L_{\text{max}} \Rightarrow$ easier to solve with coordinate descent!

That Picture Again!



$L_{\text{res}}/L_{\text{max}}$ is smaller in the left figure, larger in the right figure.

Asynchronous Unconstrained SCD

Similar computation model to HOGWILD!: asynchronous with maximum delay τ . Consider single-coordinate form.

At each iteration j :

- Choose i_j with equal probability from $\{1, 2, \dots, n\}$;
- Update the i_j component:

$$x_{j+1} = x_j - \frac{\gamma}{L_{\max}} \nabla_{i_j} f(x_{k(j)}),$$

where $k(j)$ is some iterate prior to j but no more than τ cycles old: $j - k(j) \leq \tau$. Here γ is a constant steplength.

Each core runs this process concurrently and asynchronously.

Choose some $\rho > 1$ and pick γ small enough to ensure that

$$\rho^{-1} \leq \frac{\mathbb{E}(\|\nabla f(x_{j+1})\|^2)}{\mathbb{E}(\|\nabla f(x_j)\|^2)} \leq \rho.$$

Not too much change in gradient over each iteration, so not too much price to pay for using old information, in the asynchronous setting.

Can choose γ small enough to satisfy this property but large enough to get a linear rate.

Analysis: Essentially Strongly Convex f

Essentially strongly convexity parameter μ :

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

for all $x, y \in \Omega$ with $P_S(x) = P_S(y)$. Weaker than usual strong convexity.¹

As a special case of the convergence analysis, consider the regime in which

$$\tau + 1 \leq \frac{\sqrt{n}L_{\max}}{2eL_{\text{res}}}.$$

Define

$$\rho = 1 + \frac{2eL_{\text{res}}}{\sqrt{n}L_{\max}}, \quad \psi = 1 + \frac{2\tau\rho^\tau L_{\text{res}}}{\sqrt{n}L_{\max}},$$

and choose steplength $\gamma = 1/\psi$.

Then obtain an expected linear convergence rate:

$$\mathbb{E}(f(x_j) - f^*) \leq \left(1 - \frac{\mu}{2nL_{\max}}\right)^j (f(x_0) - f^*).$$

¹Example: $f(Ax)$ is essentially strongly convex if $f(\cdot)$ is strongly convex.

Converges to precision ϵ with probability at least $1 - \eta$ for

$$K \geq \frac{2nL_{\max}}{\mu} \left| \log \frac{f(x_0) - f^*}{\eta\epsilon} \right|$$

The regime on τ is somewhat restrictive, but the degradation as τ exceeds this bound is not too severe. (Choose smaller γ .)

If $f(x) = \|Ax - b\|^2$ where $A \in \mathbb{R}^{m \times n}$ is a Gaussian random matrix, then L_{res}/L_{\max} is bounded by $1 + O(\sqrt{n/m})$.

Recall that short-step steepest descent on a strongly convex f gives linear convergence with rate approx:

$$1 - \frac{2}{(L/\mu) + 1} \approx 1 - \frac{2\mu}{L}.$$

By comparison, n steps of asynchronous short-step steepest descent gives decrease factor approx:

$$1 - \frac{\mu}{2L_{\max}}.$$

When $L_{\max} \sim L$, suggests that **about 4 times as many iterations would be needed by SCD**. But we can run it in parallel!

Bound on τ is a measure of potential parallelization. When ratio L_{res}/L_{\max} is favorable, get $\tau = O(\sqrt{n})$. Can expect linear rate even on $O(\sqrt{n})$ cores running asynchronously in parallel.

Analysis: Weakly Convex f ($\mu = 0$): Sublinear ($1/k$)

Defining ψ and γ as above, and assuming

$$\tau + 1 \leq \frac{\sqrt{n}L_{\max}}{2eL_{\text{res}}},$$

we have

$$\mathbb{E}(f(x_j) - f^*) \leq \frac{1}{(f(x_0) - f^*)^{-1} + \frac{j}{4nL_{\max}R^2}}.$$

Roughly “ $1/k$ ” behavior.

To achieve precision ϵ with probability at least $1 - \eta$, need

$$K \geq 4nL_{\max}R^2 \left(\frac{1}{\eta\epsilon} - \frac{1}{f(x_0) - f^*} \right).$$

Asynchronous Constrained SCD (CSCD)

$$\min f(x) \text{ subject to } x \in \Omega,$$

where Ω is separable $\Omega = \Omega_1 \times \Omega_2 \times \cdots \Omega_n$.

At each iteration j :

- Choose i_j with equal probability from $\{1, 2, \dots, n\}$;
- Update the i_j component:

$$x_{j+1} = P_{\Omega_{i_j}} \left(x_j - \frac{\gamma}{L_{\max}} \nabla_{i_j} f(x_{k(j)}) \right),$$

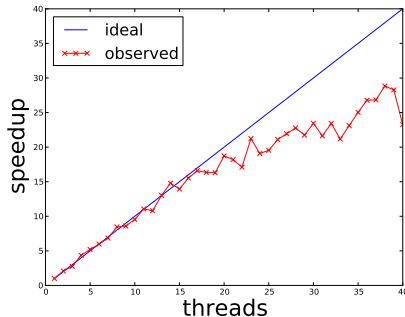
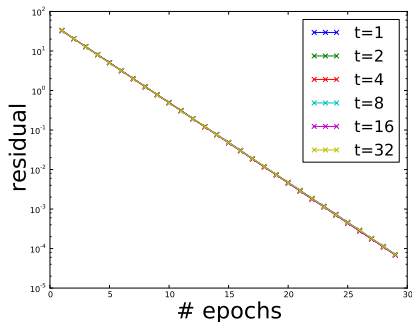
where $k(j)$ is some iterate prior to j but no more than τ cycles old:
 $j - k(j) \leq \tau$. Here γ is a constant steplength.

Each core runs this process concurrently and asynchronously.

Implemented on 4-socket, 40-core Intel Xeon

$$\min_x \|Ax - b\|^2 + 0.5\|x\|^2$$

where $A \in \mathbb{R}^{m \times n}$ is a Gaussian random matrix ($m = 6000$, $n = 20000$, columns are normalized to 1). $L_{\text{res}}/L_{\text{max}} \approx 2.2$. Choose $\gamma = 1$.

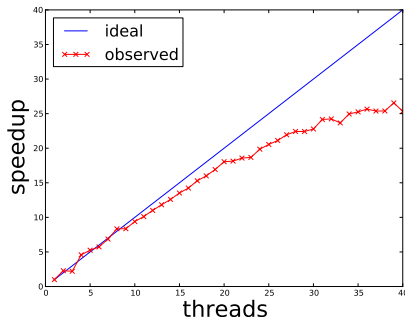
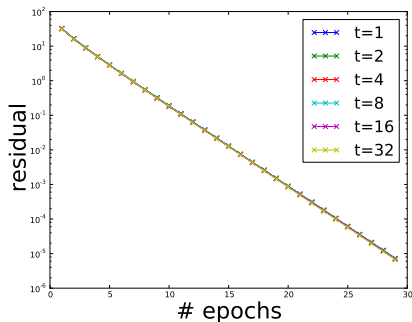


Implemented on 4-socket, 40-core Intel Xeon

$$\min_{x \geq 0} (x - z)^T (A^T A + 0.5I)(x - z) \quad ,$$

where $A \in \mathbb{R}^{m \times n}$ is a Gaussian random matrix ($m = 6000$, $n = 20000$, columns are normalized to 1) and z is a Gaussian random vector.

$L_{\text{res}}/L_{\text{max}} \approx 2.2$. Choose $\gamma = 1$.



Application: Sparse Inverse Covariance

Recall the sparse inverse covariance selection problem

$$\max_{X \succ 0} \log \det(X) - \langle X, S \rangle - \tau \|X\|_1,$$

where $\|X\|_1$ is the “element-wise” norm of the symmetric positive definite $n \times n$ matrix X .

Each step: choose a single entry of X , indexed by (i, j) (and its symmetric counterpart (j, i)) and find scalar θ that maximizes the objective for $X + \theta e_i e_j^T + \theta e_j e_i^T$.

Surprisingly, we can find θ exactly! Need some linear algebra tricks, facts about matrix logs and determinants, and simple calculus.

Suppose we know $W = X^{-1}$. Then can show

$$\det(X + \theta e_i e_j^T + \theta e_j e_i^T) = (\det X)(1 + 2W_{ij}\theta + \theta^2(W_{ij}^2 - W_{ii}W_{jj})).$$

The problem of maximizing along θ is equivalent to

$$\max_{\theta} \log(1 + 2W_{ij}\theta + \theta^2(W_{ij}^2 - W_{ii}W_{jj})) - 2S_{ij}\theta - 2\tau|X_{ij} + \theta|.$$

Using u to denote an element in the subdifferential of $\partial|\cdot|$ at $X_{ij} + \theta$, we set the derivative of the subproblem to zero:

$$2 \frac{W_{ij} + \theta(W_{ij}^2 - W_{ii}W_{jj})}{1 + 2W_{ij}\theta + \theta^2(W_{ij}^2 - W_{ii}W_{jj})} - 2S_{ij} - 2\tau u.$$

The θ and u that satisfy this expression can be obtained by following the same procedure as for the prox-operator for $\|x\|_1$.

- Try setting $u = +1$ and solve a quadratic for θ . If $X_{ij} + \theta \geq 0$, we are done.
- Try setting $u = -1$ and solve a quadratic for θ . If $X_{ij} + \theta \leq 0$, we are done.
- Otherwise set $\theta = -X_{ij}$; we must have $u \in [-1, 1]$ for this value of θ .

Finally, update $W = X^{-1}$ to reflect the rank-2 change to X using Sherman-Morrison-Woodbury.

Requires $O(p^2)$ steps per iteration.

Application: Extreme Linear Programming

State-of-the-art solvers for large linear programs (LPs) are based on simplex and interior-point methods. An alternative approach based on

- augmented Lagrangian / proximal-point
- iterative solvers for the bounded-QP subproblems (SOR, CG)

were studied in the late 1980s

- O. L. Mangasarian and R. DeLeone, “Serial and Parallel Solution of Large-Scale Linear Program by Augmented Lagrangian Successive Overrelaxations,” 1987.
- S. J. Wright, “Implementing Proximal-Point Methods for Linear Programming,” JOTA, 1990

These showed some promise on random, highly degenerate problems, but were **terrible** on the netlib test set and other problems arising in practice.

But this approach has potential appeal for:

- Cases in which only crude approximate LP solutions are needed.
- No matrix factorizations or multiplications are required. (Thus may be good for special problems, at extreme scale.)
- Multicore implementation is easy, when asynchronous solver is used on the QP subproblems.

Basics of the Approach

Primal-dual pair:

$$\min_x c^T x \text{ s.t. } Ax = b, \ x \geq 0$$

$$\max_u b^T u \text{ s.t. } A^T u \leq c.$$

“Proximal method of multipliers” subproblem is a bound-constrained convex QP:

$$x(\beta) := \arg \min_{x \geq 0} c^T x - \bar{u}^T (Ax - b) + \frac{\beta}{2} \|Ax - b\|^2 + \frac{1}{2\beta} \|x - \bar{x}\|_2^2,$$

where (\bar{x}, \bar{u}) is an estimate of the primal-dual solution and β is a penalty parameter.

Can solve a sequence of these, with updates to \bar{u} and \bar{x} , and possible increases in β , in the familiar style of augmented Lagrangian.

Solve the QP using SCD on 32 cores.

There are numerous NP-hard problems for which approximate solutions can be found using linear programming followed by rounding. Typical process:

- Construct a MIP formulation;
- Relax to an LP (replace binary variables by $[0, 1]$ intervals);
- Solve the LP approximately;
- Use LP solution to construct a feasible MIP solution (“rounding”).

Examples: Vertex cover, set cover, set packing, multiway cut, maximal independent set.

Given a graph with edge set E , vertex set V , seek a subset of vertices such that every edge touches the subset. Cost to select a vertex v is c_v .

Binary programming formulation:

$$\min \sum_{v \in V} c_v x_v \quad \text{s.t.} \quad x_u + x_v \geq 1 \quad \text{for } (u, v) \in E; \quad x_v \in \{0, 1\} \quad \text{for all } v \in V.$$

Relax the binary constraint to $x_v \in [0, 1]$ to get an LP. Very large, but matrix A is highly sparse and structured.

Sample Results

instance	vertex cover		multiway cuts	
	n	size(A)	n	size(A)
frb59-26-1	126K	616K	1.3M	3.6M
Amazon	203K	956K	6.8M	21.3 M
DBLP	146K	770K	10.7M	33.7M
Google+	82K	1.5M	7.6M	24.1M

Table: Problem Sizes

Computation Times (Seconds)

Run on 32 cores Intel machine for max of one hour. Compared with Cplex IP and LP solvers. Times shown for reaching solutions of similar quality.

instance	Cplex IP	Cplex LP	Us
frb59-26-1 VC	-	5.1	0.65
Amazon VC	44	22	4.7
DBLP VC	23	21	3.2
Google+ VC	-	62	6.2
frb59-26-1 MC	54	360	29
Amazon MC	-	-	131
DBLP MC	-	-	158
Google+ MC	-	-	570

(Cplex IP sometimes faster than LP because the IP preprocessing can drastically simplify the problem, for some data sets.)

Further Reading

- ① J. Liu and S. J. Wright, “An asynchronous parallel stochastic coordinate descent algorithm,” Manuscript, July, 2013.
- ② K. Scheinberg and S. Ma, “Optimization methods for sparse inverse covariance selection,” in *Optimization for Machine Learning*, MIT Press, 2011.
- ③ P. Tseng and S. Yun, “A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training.” *Computational Optimization and Applications*, 47, pp. 179–206, 2010.
- ④ P. Tseng and S. Yun, “A coordinate gradient descent method for nonsmooth separable minimization.” *Mathematical Programming, Series B*, 117. pp. 387–423, 2009.
- ⑤ P. Richtarik and M. Takac, “Iteration complexity of randomized block-coordinate gradient descent methods for minimizing a composite function,” *Mathematical Programming, Series A*, 2012.
- ⑥ S. J. Wright, “Accelerated block-coordinate relaxation for regularized optimization.” *SIAM J. Optimization* 22 (2012), pp. 159–186.
- ⑦ Y. Nesterov, “Efficiency of coordinate descent methods on huge-scale optimization problems.” *SIAM J. Optimization* 22 (2012), pp. 341–362.