Optimization for Data Science

F. Rinaldi[1]

1 Università degli Studi di Padova | Dipartimento Matematica

Padova
2020

Outline

Optimization for Data Science

1 Coordinate Minimization Methods

2 Block Coordinate Gradient Methods

Our Problem

GOAL

Solve the problem:

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

where f is convex and smooth (f is continuously differentiable and gradient is Lipschitz continuous).

- When n is large, computationally expensive to calculate full gradients.
- Gradient descent methods might not be efficient!
- What to do in this case?

Coordinate Minimization Schemes

- **NESOC:** x^* is an optimal solution if and only if $\nabla f(x^*) = 0$.
- Equivalently, $\nabla_i f(x^*) = 0, \ \forall\, i = 1, \ldots, n$.
- Search along each coordinate to get the optimal solution.
- When f not decreasing at every coordinate direction, then we get the optimum.
- These are *Coordinate Minimization* Algorithms (also known as *Coordinate Descent* Algorithms).

Basic Rules

- Split the optimization process into a sequence of simpler optimizations.
- Exploit the special structure of the problem.

General Scheme

Algorithm 1 `Coordinate minimization method`

1 Choose a point $x_1 \in \mathbb{R}^n$
2 For $k = 1, \ldots$
3 If x_k satisfies some specific condition, then STOP
4 Pick coordinate i from 1 to n and set

$$s_k^{(i)} = \operatorname*{Argmin}_{x^{(i)} \in \mathbb{R}} f(x^{(i)}, \mathbf{x}_{-i})$$

5 Use $s_k^{(i)}$, with $i = 1, \ldots, n$ to build x_{k+1}
6 End for

The symbol $\mathbf{x}_{-i}$ indicates the set of all variables but $x^{(i)}$.

Coordinate Minimization Schemes

- The scheme is very general.
- It embeds different strategies depending on the choices of $x^{(i)}$ and $\mathbf{x}_{-i}$.
- It makes practical sense if the minimizations to be done at Step 4 are fairly easy.
- We consider $x^{(i)}$ a scalar.
- In general it might be a block of variables.

Classic Example

Problem Features:

- Cost function involving terms $h(Ax)$.
- Computing $y = Ax$ much more expensive than computing $h(y)$.

Here minimization over a block component speeds up the calculations.

Examples of Schemes

Gauss-Seidel Scheme

- At Step 4, we set the coordinates $\mathbf{x}_{-i}$ to the most up-to-date value, that is

$$\mathbf{x}_{-i} = \big(s_k^{(1)}, \ldots, s_k^{(i-1)},\ x_k^{(i+1)}, \ldots, x_k^{(n)}\big).$$

- At Step 5, we simply set

$$x_{k+1}^{(i)} = s_k^{(i)},$$

to get the new iterate.

- PRO: Faster Convergence.
- CON: Minimizations to be done in **sequential order**.

Examples of Schemes II

Jacobi Scheme

- At Step 4, we set the coordinates $\mathbf{x}_{-i}$ to be the solution obtained from previous cycle, that is

$$\mathbf{x}_{-i} = (x_k^{(1)}, \ldots, x_k^{(i-1)},\ x_k^{(i+1)}, \ldots, x_k^{(n)}).$$

- At Step 5, gather information coming from the different minimization steps to generate a new iterate that guarantees function decrease.

- PRO: Coordinate minimization can be done in **parallel**.
- CON: Jacobi update does not take into account intermediary iterates until we complete all coordinates.

Comments

- Objective function values are non-decreasing, that is

$$f(x_{k+1}) \leq f(x_k).$$

- When f is strictly convex and smooth, the algorithm converges to a solution.
- If f is non-convex or non-smooth, the algorithm might not converge at all.

Block Minimization Schemes

- Up until this point, we have only considered coordinate updates in our schemes.
- Variables naturally partitioned into blocks when handling many data science applications (like, e.g., signal analysis or distributed computing problems).
- We may want to generalize the framework in order to use block coordinate updates.
- GOAL: develop and/or use those approaches that perform update of coordinate blocks.

What to do at Step 4?

- Give up running an exact minimization (this step becomes costly when dimension of the block increases).
- Use some gradient like steps instead.

Details and Notations

- Same unconstrained minimization problem as before.
- Variables partitioned in b blocks of dimension $n_i,\ i = 1, \dots, b$ such that $n = n_1 + \cdots + n_b$.
- Define matrices $U_i \in \mathbb{R}^{n \times n_i}$ and indicate the identity matrix $I_n = [U_1| \dots |U_b]$.
- We get the i-th block as follows

$$x^{(i)} = U_i^T x.$$

- We further indicate the i-th vector of partial derivatives as

$$\nabla_i f(x) = U_i^T \nabla f(x).$$

Assumption

Assumption [Lipschitz Continuity]

- f has Lipschitz continuous gradient, with constant L;
- $f(., \mathbf{x}_{-i})$ has Lipschitz continuous gradient with constant L_i, that is

$$\|\nabla f(x + U_i h_i) - \nabla f(x)\| \leq L_i \|h_i\|, \text{ for all } h_i \in \mathbb{R}^{n_i} \text{ and } x \in \mathbb{R}^n.$$

- We also denote with $L_{max} = \max_i L_i$ and $L_{min} = \min_i L_i$.
- It is possible to see that

$$L_i \leq L \leq \sum_i L_i \leq b \cdot L_{max}, \forall\, i \in \{1, \ldots, b\}.$$

Block Coordinate Gradient Descent (BCGD) Method

Algorithm 2 `BCGD method`

Choose a point $x_1 \in \mathbb{R}^n$
For $k = 1, \ldots$
    If x_k satisfies some specific condition, then STOP
    Set $y_0 = x_k$, pick blocks $S \subseteq \{1, \ldots, b\}$, and set $l = |S|$
    For $i = 1, \ldots, l$, select $j_i \in S$ and set

$$y_i = y_{i-1} - \alpha_i U_{j_i} \nabla_{j_i} f(y_{i-1})$$

    with $\alpha_i > 0$ calculated using a suitable line search
    Set $x_{k+1} = y_l$
End for

Comments to Main Steps

- **STEP 4:** Pick some blocks according to a given rule.
- **STEP 5:** Run an update in a (possibly) sequential fashion using gradient information.
- **STEP 6:** Build up the new point x_{k+1} before ending the iteration.

Classic Block Selection Strategies

Cyclic Order

Run all blocks in cyclic order, i.e. from 1 to b at each iteration (that is $S = \{1, \ldots, b\}$ in the scheme).

Almost cyclic order

Each block $i \in \{1, \ldots, b\}$ picked at least once every $B < \infty$ successive iterations

Gauss-Southwell (aka Greedy)

at each iteration, pick block i so that

$$i = \operatorname*{Argmax}_{j \in \{1,\ldots,b\}} \|\nabla_j f(x_k)\|.$$

Randomized Block Selection Strategies

Random Permutation

Run cyclic order on a set of permuted indices (that is $S = \{1, \ldots, b\}$ in the scheme but we pick the indices j_i at random).

Random Sampling

Randomly select some block i to update (we only pick one block in this case).

Example of How Strategies Work

Example

Assume that $n = 3$ and that we have blocks of dimension 1. We could have the following:

- Cyclic:

$$(1 \to 2 \to 3) \to (1 \to 2 \to 3) \to (1 \to 2 \to 3) \dots$$

- Random permutation :

$$(2 \to 1 \to 3) \to (3 \to 1 \to 2) \to (1 \to 2 \to 3) \to \dots$$

- Random sampling:

$$1 \to 1 \to 2 \to 3 \to 2 \to 1 \to \dots$$