# Optimization Techniques for Big Data Analysis

## Chapter 6. Coordinate Descent Methods

**Master of Science in Signal Theory and Communications**
Dpto. de Señales, Sistemas y Radiocomunicaciones
E.T.S. Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

2023

# Coordinate descent

$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{x} = [x_0, x_1, \cdots, x_j, \cdots, x_d]$$

# Coordinate descent

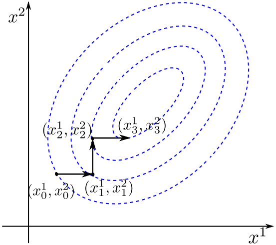
$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{x} = [x_0, x_1, \cdots, x_j, \cdots, x_d]$$

The coordinate descent (CD) method proposes to minimize $f(\cdot)$ across **one dimension at a time**, turning the problem into consecutive one-dimensional optimization problems [1].

# Coordinate descent

$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{x} = [x_0, x_1, \cdots, x_j, \cdots, x_d]$$

The coordinate descent (CD) method proposes to minimize $f(\cdot)$ across **one dimension at a time**, turning the problem into consecutive one-dimensional optimization problems [1].

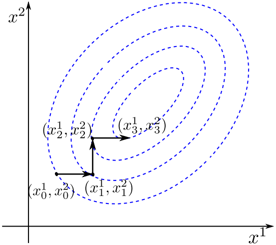



Why would you use CD instead of GD?

# Coordinate descent

$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{x} = [x_0, x_1, \cdots, x_j, \cdots, x_d]$$

The coordinate descent (CD) method proposes to minimize $f(\cdot)$ across **one dimension at a time**, turning the problem into consecutive one-dimensional optimization problems [1].



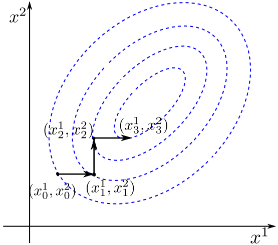

Why would you use CD instead of GD?

- The gradient is impossible to calculate

# Coordinate descent

$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{x} = [x_0, x_1, \cdots, x_j, \cdots, x_d]$$

The coordinate descent (CD) method proposes to minimize $f(\cdot)$ across **one dimension at a time**, turning the problem into consecutive one-dimensional optimization problems [1].



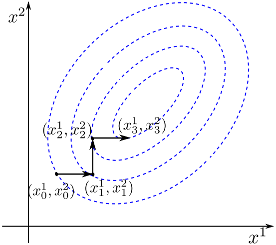

Why would you use CD instead of GD?

- The gradient is impossible to calculate
- The feasible region is constrained

# Coordinate descent

$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{x} = [x_0, x_1, \cdots, x_j, \cdots, x_d]$$

The coordinate descent (CD) method proposes to minimize $f(\cdot)$ across **one dimension at a time**, turning the problem into consecutive one-dimensional optimization problems [1].



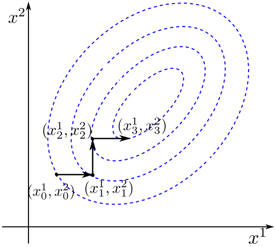

Why would you use CD instead of GD?

- The gradient is impossible to calculate
- The feasible region is constrained
- A massive amount of variables to optimize

# Coordinate descent

$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{x} = [x_0, x_1, \cdots, x_j, \cdots, x_d]$$

The coordinate descent (CD) method proposes to minimize $f(\cdot)$ across **one dimension at a time**, turning the problem into consecutive one-dimensional optimization problems [1].




Why would you use CD instead of GD?

- The gradient is impossible to calculate
- The feasible region is constrained
- A massive amount of variables to optimize

We can also group the variables into block of dimension $m_j$, and optimise one block at a time; that's call **Block Coordinate Descent**.

# Block Coordinate Descent

The BCD algorithm consists of solving our block-structured problem in an iterative manner. On iteration $k$ we compute

$$\begin{aligned} x_{k+1,j} &= \arg\min_{x_j \in X_j} f\left(x_j, x_{k,-j}\right) \\ x_{k+1,l} &= x_{k,l}, \qquad \forall l \neq j \end{aligned}$$

where $x_{k,-j} \triangleq (x_{k,1}, \cdots, x_{k,j-1}, x_{k,j+1}, \cdots, x_{k,d})$. In the next iteration, a different coordinate, for instance, $j+1$, is updated.

# Block Coordinate Descent

The BCD algorithm consists of solving our block-structured problem in an iterative manner. On iteration $k$ we compute

$$\begin{aligned} x_{k+1,j} &= \arg\min_{x_j \in X_j} f\left(x_j, x_{k,-j}\right) \\ x_{k+1,l} &= x_{k,l}, \qquad \forall l \neq j \end{aligned}$$

where $x_{k,-j} \triangleq (x_{k,1}, \cdots, x_{k,j-1}, x_{k,j+1}, \cdots, x_{k,d})$. In the next iteration, a different coordinate, for instance, $j+1$, is updated.

The method is very intuitive and simple to implement and very popular in many applications. However, it does not have guaranteed convergence for an arbitrary function $f$.

# Example_6_1

Ridge regression:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2}\left\|\mathbf{y} - \mathbf{X}\mathbf{w}\right\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

minimizing over $w_j$ with all $w_l$, $l \neq j$ fixed,

# Example_6_1

Ridge regression:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

minimizing over $w_j$ with all $w_l$, $l \neq j$ fixed,

$$\nabla_j f(\mathbf{w}) = -\mathbf{X}_{:,j}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda w_j = 0$$

# Example_6_1

Ridge regression:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

minimizing over $w_j$ with all $w_l$, $l \neq j$ fixed,

$$\nabla_j f(\mathbf{w}) = -\mathbf{X}_{:,j}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda w_j = 0$$

$$-\mathbf{X}_{:,j}^T(\mathbf{y} - \mathbf{X}_{:,-j}\mathbf{w}_{-j} - \mathbf{X}_{:,j}w_j) + \lambda w_j = 0$$

# Example_6_1

Ridge regression:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

minimizing over $w_j$ with all $w_l$, $l \neq j$ fixed,

$$\nabla_j f(\mathbf{w}) = -\mathbf{X}_{:,j}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda w_j = 0$$

$$-\mathbf{X}_{:,j}^T(\mathbf{y}-\mathbf{X}_{:,-j}\mathbf{w}_{-j}-\mathbf{X}_{:,j}w_j)+\lambda w_j = 0$$

$$w_j = \frac{\mathbf{X}_{:,j}^T(\mathbf{y}-\mathbf{X}_{:,-j}\mathbf{w}_{-j})}{\mathbf{X}_{:,j}^T\mathbf{X}_{:,j}+\lambda}$$

# Example_6_1

Ridge regression:

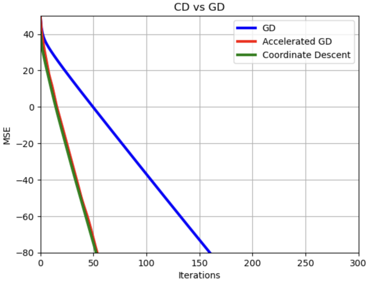
$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

minimizing over $w_j$ with all $w_l$, $l \neq j$ fixed,

$$\nabla_j f(\mathbf{w}) = -\mathbf{X}_{:,j}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda w_j = 0$$

$$-\mathbf{X}_{:,j}^T(\mathbf{y}-\mathbf{X}_{:,-j}\mathbf{w}_{-j}-\mathbf{X}_{:,j}w_j)+\lambda w_j = 0$$

$$w_j = \frac{\mathbf{X}_{:,j}^T(\mathbf{y}-\mathbf{X}_{:,-j}\mathbf{w}_{-j})}{\mathbf{X}_{:,j}^T\mathbf{X}_{:,j}+\lambda}$$



# Gradient Coordinate Descent.

$$x_{k+1,j} = x_{k,j} - \eta \nabla_{x_j} f(\mathbf{x}_k)$$
$$x_{k+1,j} = x_{k,j} \quad \forall j \neq i$$

where $\mathbf{x}_k = x_{k,1}, \cdots, x_{k,j-1}, x_{k,j}, \cdots, x_{k,d}$ is the pivoting point around whom we have evaluated the gradient over block variable $x_j$ at instant $k$.

# Gradient Coordinate Descent.

$$x_{k+1,j} = x_{k,j} - \eta \nabla_{x_j} f(\mathbf{x}_k)$$
$$x_{k+1,j} = x_{k,j} \quad \forall j \neq i$$

where $\mathbf{x}_k = x_{k,1}, \cdots, x_{k,j-1}, x_{k,j}, \cdots, x_{k,d}$ is the pivoting point around whom we have evaluated the gradient over block variable $x_j$ at instant $k$.

1. If $f$ is non-smooth, we could incorporate projected or proximal updates.
2. The SGD is also applicable, where an instantaneous estimate substitutes the gradient.
3. It could also be improved using Nesterov or Quasi-Newton principles.

# Parallel settings

BCD can be applied in different settings:

# Parallel settings

BCD can be applied in different settings:

1. Cyclic rule: the block coordinates are chosen cyclically, in a sequential manner. This scheme is frequently referred to as Gauss-Seidel scheme.

# Parallel settings

BCD can be applied in different settings:

1. Cyclic rule: the block coordinates are chosen cyclically, in a sequential manner. This scheme is frequently referred to as Gauss-Seidel scheme.
2. Parallel rule: all blocks are updated based on the same approximation point $\mathbf{x}_k$. This scheme is frequently referred to as the Jacobi scheme.

# Parallel settings

BCD can be applied in different settings:

1. Cyclic rule: the block coordinates are chosen cyclically, in a sequential manner. This scheme is frequently referred to as Gauss-Seidel scheme.
2. Parallel rule: all blocks are updated based on the same approximation point $\mathbf{x}_k$. This scheme is frequently referred to as the Jacobi scheme.
3. Mixed scheme: for big data, it is useful that some blocks are updated in parallel (in different processors) while the variables of each block are updated sequentially (within the same processor). This scheme is usually referred to as Gauss-Jacobi scheme.

# Parallel settings

BCD can be applied in different settings:

1. Cyclic rule: the block coordinates are chosen cyclically, in a sequential manner. This scheme is frequently referred to as Gauss-Seidel scheme.
2. Parallel rule: all blocks are updated based on the same approximation point $\mathbf{x}_k$. This scheme is frequently referred to as the Jacobi scheme.
3. Mixed scheme: for big data, it is useful that some blocks are updated in parallel (in different processors) while the variables of each block are updated sequentially (within the same processor). This scheme is usually referred to as Gauss-Jacobi scheme.
4. Randomized rule: In the randomized scheme, every block has a non-zero probability of being updated, and these probabilities are varied according to some information over the estimated errors.

## case_study_6_1

Let us assume the general problem

$$\arg\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

Instead of updating one single variable at a time, we can make blocks of variables of size $m_j \geq 1$ and solve the problem in a parallel fashion.

# case_study_6_1

Let us assume the general problem

$$\arg\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

Instead of updating one single variable at a time, we can make blocks of variables of size $m_j \geq 1$ and solve the problem in a parallel fashion.

The gradient takes the same form as before, but the $j$ represents a block of variables:

$$\nabla_j f(\mathbf{w}) = \frac{2}{n}\mathbf{X}_{:,j}^T(\mathbf{X}_{:,j}\mathbf{w}_j + \mathbf{X}_{:,-j}\mathbf{w}_{-j} - \mathbf{y}) + \lambda\mathbf{w}_j = 0$$

# case_study_6_1

Let us assume the general problem

$$\arg\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

Instead of updating one single variable at a time, we can make blocks of variables of size $m_j \geq 1$ and solve the problem in a parallel fashion.

The gradient takes the same form as before, but the $j$ represents a block of variables:

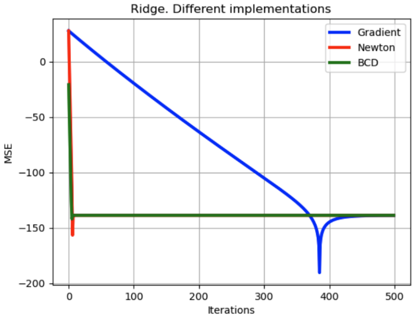
$$\nabla_j f(\mathbf{w}) = \frac{2}{n}\mathbf{X}_{:,j}^T(\mathbf{X}_{:,j}\mathbf{w}_j + \mathbf{X}_{:,-j}\mathbf{w}_{-j} - \mathbf{y}) + \lambda\mathbf{w}_j = 0$$

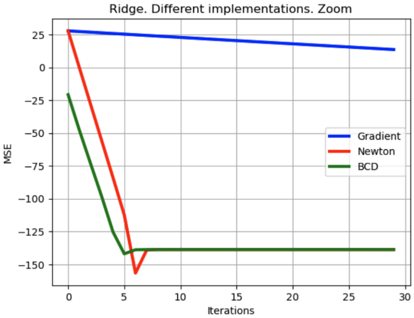
So, the closed-form solution for the iteration $k+1$ results into

$$\mathbf{w}_{k+1,j} = \left(\mathbf{X}_{:,j}^T\mathbf{X}_{:,j} + \frac{n}{2}\lambda\mathbf{I}_{m_j}\right)^{-1}\mathbf{X}_{:,j}^T\left(\mathbf{y} - \mathbf{X}_{:,-j}\mathbf{w}_{k,-j}\right)$$

# Case_study_6_1

Follow the code provided in the notebook `Case_study_6_1` to obtain results as those presented in the next Figure. Pay attention to how high-speed these algorithms are.



# Case_study_6_2

Let's now consider the LASSO case

$$\arg\min_{\mathbf{w}} \frac{1}{n} \left\| \mathbf{X}\mathbf{w} - \mathbf{y} \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1$$

# Case_study_6_2

Let's now consider the LASSO case

$$\arg\min_{\mathbf{w}} \frac{1}{n} \left\| \mathbf{X}\mathbf{w} - \mathbf{y} \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1$$

By differentiating it with respect to $j$-th weight, we get

$$\nabla_j f(\mathbf{w}) = \frac{2}{n} \mathbf{X}_{:,j}^T \left( \mathbf{X}_{:,j} w_j + \mathbf{X}_{:,-j} \mathbf{w}_{-j} - \mathbf{y} \right) + \lambda \partial |w_j| \in 0$$

# Case_study_6_2

Let's now consider the LASSO case

$$\arg\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_1$$

By differentiating it with respect to $j$-th weight, we get

$$\nabla_j f(\mathbf{w}) = \frac{2}{n} \mathbf{X}_{:,j}^T \left(\mathbf{X}_{:,j} w_j + \mathbf{X}_{:,-j} \mathbf{w}_{-j} - \mathbf{y}\right) + \lambda \partial |w_j| \in 0$$

Solving for $w_j$

$$w_j = \frac{\mathbf{X}_{:,j}^T \left(\mathbf{y} - \mathbf{X}_{:,-j} \mathbf{w}_{-j}\right)}{\mathbf{X}_{:,j}^T \mathbf{X}_{:,j}} - \frac{n\lambda \mathrm{sgn}(w_j)}{2\mathbf{X}_{:,j}^T \mathbf{X}_{:,j}}$$

# Case_study_6_2

Let's now consider the LASSO case

$$\arg\min_{\mathbf{w}} \frac{1}{n} \left\| \mathbf{X}\mathbf{w} - \mathbf{y} \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1$$

By differentiating it with respect to $j$-th weight, we get

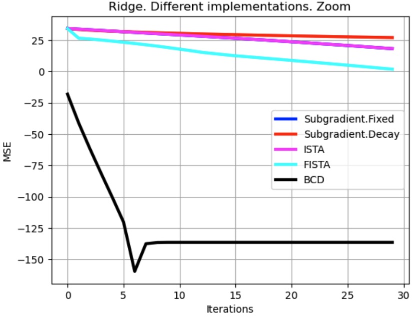
$$\nabla_j f(\mathbf{w}) = \frac{2}{n} \mathbf{X}_{:,j}^T \left( \mathbf{X}_{:,j} w_j + \mathbf{X}_{:,-j} \mathbf{w}_{-j} - \mathbf{y} \right) + \lambda \partial |w_j| \in 0$$

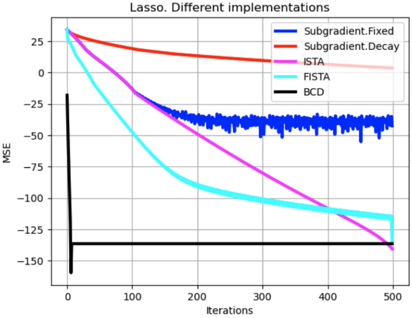
Solving for $w_j$

$$\begin{aligned} w_j &= \frac{\mathbf{X}_{:,j}^T \left( \mathbf{y} - \mathbf{X}_{:,-j} \mathbf{w}_{-j} \right)}{\mathbf{X}_{:,j}^T \mathbf{X}_{:,j}} - \frac{n\lambda \mathrm{sgn}(w_j)}{2\mathbf{X}_{:,j}^T \mathbf{X}_{:,j}} \\ &= \mathrm{Soft}\left( \frac{\mathbf{X}_{:,j}^T \left( \mathbf{y} - \mathbf{X}_{:,-j} \mathbf{w}_{-j} \right)}{\mathbf{X}_{:,j}^T \mathbf{X}_{:,j}}, \frac{n\lambda}{2\mathbf{X}_{:,j}^T \mathbf{X}_{:,j}} \right) \end{aligned}$$

# Case_study_6_2

Follow the code provided in the notebook `Case_study_6_2` to obtain the following results



# Questions?

# References

[1] Jorge Nocedal and J. Wright Stephen. *Numerical optimization.* Spinger, 2006.

# *Thank You*

Julián D. Arias-Londoño
julian.arias@upm.es