Optimization Techniques for Big Data Analysis

Chapter 7. Augmented Lagrangian 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

Motivation

This lecture is devoted to the method known as **Alternating Direction Method of Multipliers (ADMM)**, which applies to linearly constrained problems (most Machine Learning optimization problems can be expressed in this way).

This method represents a potent approach to solving machine learning problems in both centralized and distributed frameworks.

Before we get into the steps of the ADMM algorithm, we will review the basic theory of constrained linear optimization.

Precursor 1. Lagrangian relaxation (1).

Let us define the following problem, which we will denote as primal:

$$(P)\left[\begin{array}{c}\min f(x)\\ \text{s.t.:} g_i(x) \leq 0\ , i=1,\cdots,m\\ h_i(x)=0\ , i=1,\cdots,p\end{array}\right]$$

whose solution is going to be denoted as p*: the value of x^* that minimizes $f(x)$ while fulfilling the constraints.

Precursor 1. Lagrangian relaxation (1).

Let us define the following problem, which we will denote as primal:

$$(P)\left[\begin{array}{c}\min f(x)\\ \text{s.t.:} g_i(x) \leq 0\ , i=1,\cdots,m\\ h_i(x)=0\ , i=1,\cdots,p\end{array}\right]$$

whose solution is going to be denoted as p*: the value of x^* that minimizes $f(x)$ while fulfilling the constraints.

A powerful optimization approach consists of defining a simpler problem, called **relaxation**, solving it, and mapping this solution into the original problem solution.

- This is known as the dual problem (and its solution is denoted as d*).
- Desirably, d* = p* in order to have an useful relaxation.

Precursor 1. Lagrangian relaxation (2).

Lagrange proposed using as a relaxed problem the sum of the original function and the weighted contribution of the constraints: the **Lagrangian**:

$$L\left(x, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) = f\left(x\right) + \sum_{i=1}^{m} \beta_i g_i\left(x\right) + \sum_{i=1}^{p} \alpha_i h_i\left(x\right)$$

where $\boldsymbol{\beta} \succeq 0, \alpha_i \in \mathbb{R}$ are Lagrangian multipliers.

Precursor 1. Lagrangian relaxation (2).

Lagrange proposed using as a relaxed problem the sum of the original function and the weighted contribution of the constraints: the **Lagrangian**:

$$L\left(x, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) = f\left(x\right) + \sum_{i=1}^{m} \beta_i g_i\left(x\right) + \sum_{i=1}^{p} \alpha_i h_i\left(x\right)$$

where $\boldsymbol{\beta} \succeq 0, \alpha_i \in \mathbb{R}$ are Lagrangian multipliers.

We obtain the Lagrangian dual function:

$$L\left(\boldsymbol{\beta}, \boldsymbol{\alpha}\right) = \min_{x} \left(f\left(x\right) + \sum_{i=1}^{m} \beta_i g_i\left(x\right) + \sum_{i=1}^{p} \alpha_i h_i\left(x\right) \right)$$

whose principal property is that

$$L\left(\boldsymbol{\beta}, \boldsymbol{\alpha}\right) \leq \mathrm{p}^* \quad \forall \boldsymbol{\beta} \succeq 0, \boldsymbol{\alpha}$$

Precursor 1. Lagrangian relaxation (3).

Hence, the Lagrange dual problem is

$$(D) \left[\begin{array}{c} \max\limits_{\boldsymbol{\beta},\boldsymbol{\alpha}} L(\boldsymbol{\beta},\boldsymbol{\alpha}) \\ \text{s.t.:} \boldsymbol{\beta} \succeq 0 \end{array} \right]$$

- The solution of the dual problem is $\mathrm{d}^* = L(\boldsymbol{\beta}^*, \boldsymbol{\alpha}^*) \leq p^*$.
  - This property is known as weak duality.
  - If all functions are convex (always in our Machine Learning problems), we have the strong duality property, that is $\mathrm{d}^* = \mathrm{p}^*$, and hence, we can use this solution to obtain the solution of (P).

Example 7.1

Let us study the following primal problem

$$(P)\left[\begin{array}{c}\min\limits_{x_1,x_2}\left(x_1^2+x_2^2-2x_1\right)\\ \text{s.t.:}x_1^2+x_2^2-2x_2\leq 0\end{array}\right]$$

Example 7.1

Let us study the following primal problem

$$(P)\left[\begin{array}{c}\min\limits_{x_1,x_2}\left(x_1^2+x_2^2-2x_1\right)\\ \text{s.t.:}x_1^2+x_2^2-2x_2\leq 0\end{array}\right]$$

a) Construct the Lagrangian relaxation

$$L\left(x_1,x_2,\beta\right)=\left(x_1^2+x_2^2-2x_1\right)+\beta\left(x_1^2+x_2^2-2x_2\right)$$

Example 7.1

Let us study the following primal problem

$$(P)\left[\begin{array}{c}\min\limits_{x_1,x_2}\left(x_1^2+x_2^2-2x_1\right)\\ \text{s.t.:}x_1^2+x_2^2-2x_2\leq 0\end{array}\right]$$

a) Construct the Lagrangian relaxation

$$L\left(x_1,x_2,\beta\right)=\left(x_1^2+x_2^2-2x_1\right)+\beta\left(x_1^2+x_2^2-2x_2\right)$$

b) Obtain the Lagrangian dual function

$$L\left(\beta\right)=\min_{x_1,x_2}\left(x_1^2+x_2^2-2x_1\right)+\beta\left(x_1^2+x_2^2-2x_2\right)$$

Example 7.1

Let us study the following primal problem

$$(P)\left[\begin{array}{c}\min\limits_{x_1,x_2}\left(x_1^2+x_2^2-2x_1\right)\\ \text{s.t.:}x_1^2+x_2^2-2x_2\leq 0\end{array}\right]$$

a) Construct the Lagrangian relaxation

$$L\left(x_1,x_2,\beta\right)=\left(x_1^2+x_2^2-2x_1\right)+\beta\left(x_1^2+x_2^2-2x_2\right)$$

b) Obtain the Lagrangian dual function

$$L\left(\beta\right)=\min_{x_1,x_2}\left(x_1^2+x_2^2-2x_1\right)+\beta\left(x_1^2+x_2^2-2x_2\right)$$

Taking derivatives, we have:

$$\frac{\partial L\left(x_1,x_2,\beta\right)}{\partial x_1}=2x_1-2+2\beta x_1=0\rightarrow x_1^*=\frac{1}{1+\beta}$$

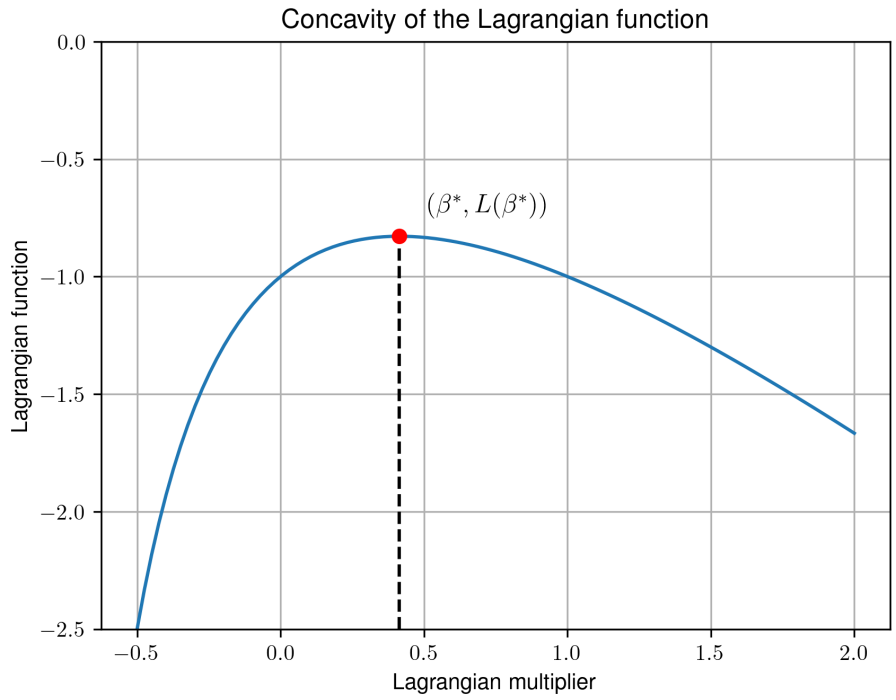
$$\frac{\partial L\left(x_1,x_2,\beta\right)}{\partial x_2}=2x_2+\beta\left(2x_2-2\right)=0\rightarrow x_2^*=\frac{\beta}{1+\beta}$$

Example 7.1

Therefore we have

$$L(\beta) = L(x_1^*, x_2^*, \beta) = -\frac{1+\beta^2}{1+\beta}$$

The maximum is $L(\beta^*) = -0.8284$.

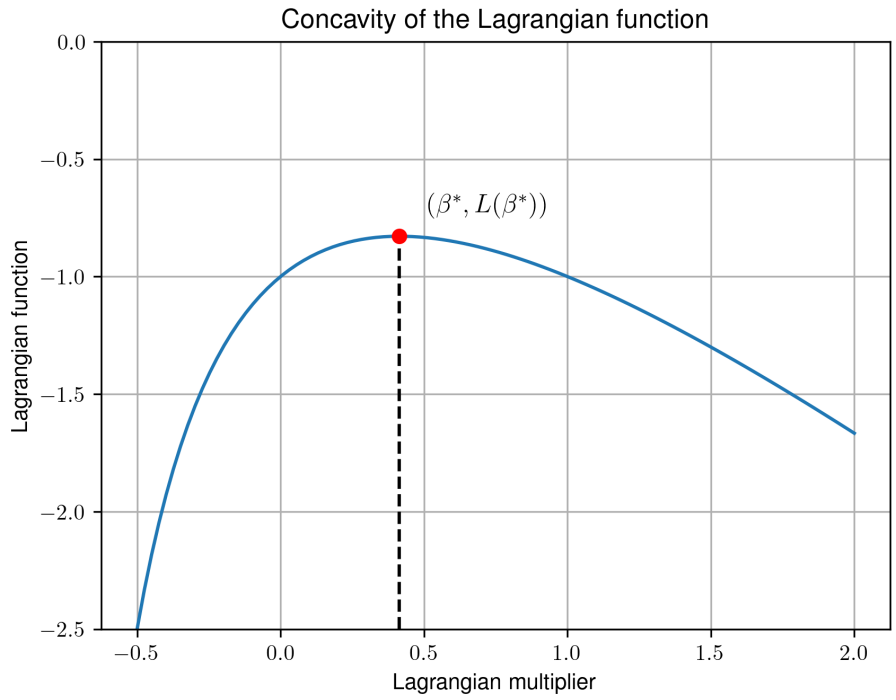


Example 7.1

Therefore we have

$$L(\beta) = L(x_1^*, x_2^*, \beta) = -\frac{1+\beta^2}{1+\beta}$$

The maximum is $L(\beta^*) = -0.8284$.



Let's prove it!

Example 7.1

c) Solve the Lagrange dual problem $\max_{\beta \geq 0} L(\beta)$.

Taking derivatives

$$\frac{dL(\beta)}{d\beta} = \frac{d}{d\beta}\left(-\frac{1+\beta^2}{1+\beta}\right) = 0$$

Example 7.1

c) Solve the Lagrange dual problem $\max_{\beta \geq 0} L(\beta)$.

Taking derivatives

$$\frac{dL(\beta)}{d\beta} = \frac{d}{d\beta}\left(-\frac{1+\beta^2}{1+\beta}\right) = 0$$

we obtain:

$$\beta = -1 \pm \sqrt{2}$$

Example 7.1

c) Solve the Lagrange dual problem $\max_{\beta \geq 0} L(\beta)$.

Taking derivatives

$$\frac{dL(\beta)}{d\beta} = \frac{d}{d\beta}\left(-\frac{1+\beta^2}{1+\beta}\right) = 0$$

we obtain:

$$\beta = -1 \pm \sqrt{2} \rightarrow \beta^* = \sqrt{2} - 1 > 0$$

Example 7.1

c) Solve the Lagrange dual problem $\max_{\beta \geq 0} L(\beta)$.

Taking derivatives

$$\frac{dL(\beta)}{d\beta} = \frac{d}{d\beta}\left(-\frac{1+\beta^2}{1+\beta}\right) = 0$$

we obtain:

$$\beta = -1 \pm \sqrt{2} \rightarrow \beta^* = \sqrt{2} - 1 > 0 \rightarrow L(\beta^*) = \mathrm{d}^* = -0.8284$$

Example 7.1

c) Solve the Lagrange dual problem $\max_{\beta \geq 0} L(\beta)$.

Taking derivatives

$$\frac{dL(\beta)}{d\beta} = \frac{d}{d\beta}\left(-\frac{1+\beta^2}{1+\beta}\right) = 0$$

we obtain:

$$\beta = -1 \pm \sqrt{2} \rightarrow \beta^* = \sqrt{2} - 1 > 0 \rightarrow L(\beta^*) = \mathrm{d}^* = -0.8284$$

d) Obtain the optimum solution of the primal variables

$$x_1^* = \frac{1}{1+\beta^*} = \frac{1}{\sqrt{2}}, \quad x_2^* = \frac{\beta^*}{1+\beta^*} = \frac{\sqrt{2}-1}{\sqrt{2}}$$

Example 7.1

c) Solve the Lagrange dual problem $\max_{\beta \geq 0} L(\beta)$.

Taking derivatives

$$\frac{dL(\beta)}{d\beta} = \frac{d}{d\beta}\left(-\frac{1+\beta^2}{1+\beta}\right) = 0$$

we obtain:

$$\beta = -1 \pm \sqrt{2} \rightarrow \beta^* = \sqrt{2} - 1 > 0 \rightarrow L(\beta^*) = \mathrm{d}^* = -0.8284$$

d) Obtain the optimum solution of the primal variables

$$x_1^* = \frac{1}{1+\beta^*} = \frac{1}{\sqrt{2}}, \quad x_2^* = \frac{\beta^*}{1+\beta^*} = \frac{\sqrt{2}-1}{\sqrt{2}}$$

e) Calculate the optimum solution of the primal problem

$$\mathrm{p}^* = (x_1^*)^2 + (x_2^*)^2 - 2x_1^* = -0.8284$$

Precursor 2. Dual ascent (1).

This approach is advantageous when the solution of the dual problem is not as easy to find as in the previous example, as it relies on using gradient methods.

Precursor 2. Dual ascent (1).

This approach is advantageous when the solution of the dual problem is not as easy to find as in the previous example, as it relies on using gradient methods.

Starting again from the Lagrangian

$$L\left(x, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) = f\left(x\right) + \sum_{i=1}^{m} \beta_i g_i\left(x\right) + \sum_{i=1}^{p} \alpha_i h_i\left(x\right)$$

Precursor 2. Dual ascent (1).

This approach is advantageous when the solution of the dual problem is not as easy to find as in the previous example, as it relies on using gradient methods.

Starting again from the Lagrangian

$$L\left(x, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) = f\left(x\right) + \sum_{i=1}^{m} \beta_i g_i\left(x\right) + \sum_{i=1}^{p} \alpha_i h_i\left(x\right)$$

If we assume that strict duality holds, we know that the optimal values of the primal and dual problems are the same:

$$\boldsymbol{\beta}^*, \boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\beta}, \boldsymbol{\alpha}} L\left(x^*, \boldsymbol{\beta}, \boldsymbol{\alpha}\right)$$
$$x^* = \arg\min_{x} L\left(x, \boldsymbol{\beta}^*, \boldsymbol{\alpha}^*\right)$$

Precursor 2. Dual ascent (2).

The dual ascent method consists of iterating the updates:

$$x_{k+1} = \arg\min_{x} L\left(x, \beta_k, \alpha_k\right)$$

$$\beta_{k+1}, \alpha_{k+1} = \arg\max_{\beta,\alpha} L\left(x_{k+1}, \beta, \alpha\right)$$

Precursor 2. Dual ascent (2).

The dual ascent method consists of iterating the updates:

$$x_{k+1} = \arg\min_{x} L\left(x, \beta_k, \alpha_k\right)$$
$$\beta_{k+1}, \alpha_{k+1} = \arg\max_{\beta,\alpha} L\left(x_{k+1}, \beta, \alpha\right)$$

whose iterative implementation is:

$$x_{k+1} = x_k - \mu_k \frac{\partial L(x,\beta_k,\alpha_k)}{\partial x}\bigg|_{x=x_k}$$
$$\beta_{k+1} = \beta_k + \gamma_k \max\left(0, \frac{\partial L(x_{k+1},\beta,\alpha_k)}{\partial \beta}\bigg|_{\beta=\beta_k}\right)$$
$$\alpha_{k+1} = \alpha_k + \gamma_k \frac{\partial L(x_{k+1},\beta_{k+1},\alpha)}{\partial \alpha}\bigg|_{\alpha=\alpha_k}$$

Precursor 2. Dual ascent (2).

The dual ascent method consists of iterating the updates:

$$x_{k+1} = \arg\min_{x} L\left(x, \beta_k, \alpha_k\right)$$
$$\beta_{k+1}, \alpha_{k+1} = \arg\max_{\beta,\alpha} L\left(x_{k+1}, \beta, \alpha\right)$$

whose iterative implementation is:

$$x_{k+1} = x_k - \mu_k \frac{\partial L(x,\beta_k,\alpha_k)}{\partial x}\bigg|_{x=x_k}$$
$$\beta_{k+1} = \beta_k + \gamma_k \max\left(0, \frac{\partial L(x_{k+1},\beta,\alpha_k)}{\partial \beta}\bigg|_{\beta=\beta_k}\right)$$
$$\alpha_{k+1} = \alpha_k + \gamma_k \frac{\partial L(x_{k+1},\beta_{k+1},\alpha)}{\partial \alpha}\bigg|_{\alpha=\alpha_k}$$

With appropriate choice of γ_k:

- The dual function increases in each step, i.e., $L\left(x_{k+1}, \beta_{k+1}, \alpha_{k+1}\right) \geq L\left(x_{k+1}, \beta_k, \alpha_k\right)$.
- This method can also be used when $L\left(x, \beta, y\right)$ is not differentiable using the subgradient of L.

Example 7.2

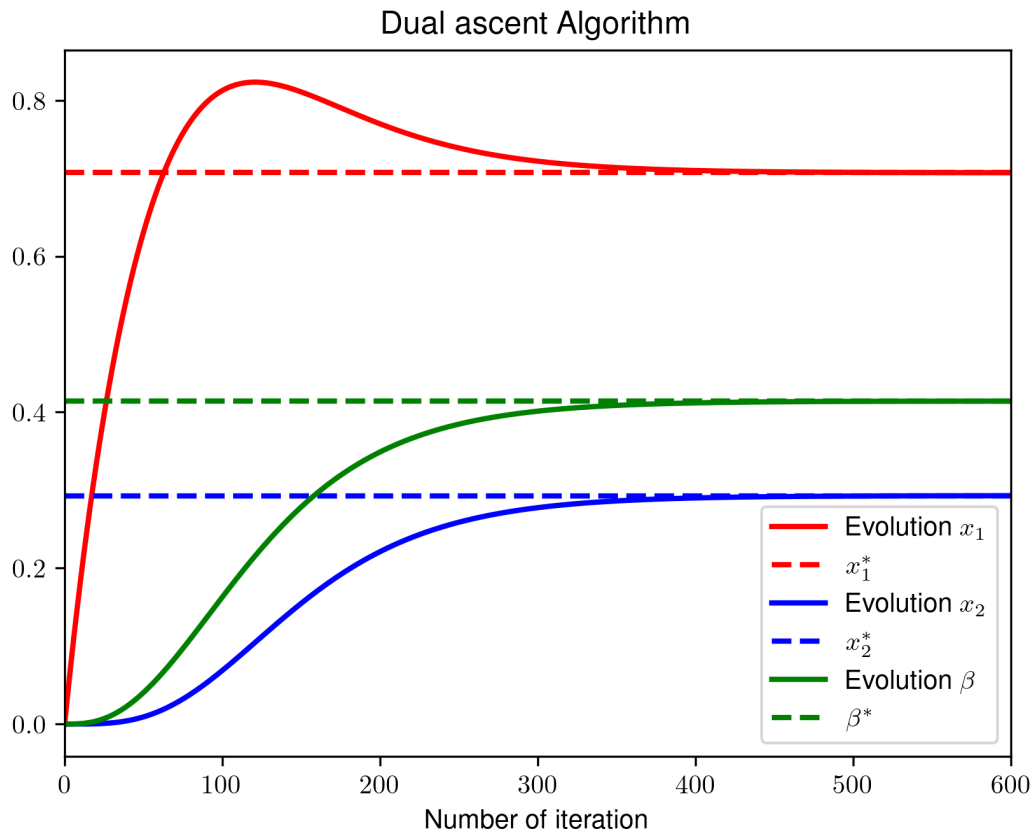
Implement the dual ascent problem of `Example 7.1` described by the following iterations (per coordinate):

$$x_{k+1,1} = x_{k,1} - \mu \left(2x_{k,1} - 2 + 2\beta_k x_{k,1}\right)$$
$$x_{k+1,2} = x_{k,2} - \mu \left(2x_{k,2} + \beta_k \left(2x_{k,2} - 2\right)\right)$$
$$\beta_{k+1} = \beta_k + \gamma \max\left(0, x_{k+1,1}^2 + x_{k+1,2}^2 - 2x_{k+1,2}\right)$$

Example 7.2

Implement the dual ascent problem of `Example 7.1` described by the following iterations (per coordinate):

$$x_{k+1,1} = x_{k,1} - \mu\left(2x_{k,1} - 2 + 2\beta_k x_{k,1}\right)$$
$$x_{k+1,2} = x_{k,2} - \mu\left(2x_{k,2} + \beta_k\left(2x_{k,2} - 2\right)\right)$$
$$\beta_{k+1} = \beta_k + \gamma \max\left(0, x_{k+1,1}^2 + x_{k+1,2}^2 - 2x_{k+1,2}\right)$$



Equality constrained convex problem

Now, we will focus on an equality-constrained convex optimization problem

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t. } \mathbf{Ax} = \mathbf{b} \end{aligned}$$

Equality constrained convex problem

Now, we will focus on an equality-constrained convex optimization problem

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

The Lagrangian for that problem is

$$L(\mathbf{x}, \boldsymbol{\beta}) = f(\mathbf{x}) + \boldsymbol{\beta}^T (\mathbf{A}\mathbf{x} - \mathbf{b})$$

Equality constrained convex problem

Now, we will focus on an equality-constrained convex optimization problem

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

The Lagrangian for that problem is

$$L(\mathbf{x}, \boldsymbol{\beta}) = f(\mathbf{x}) + \boldsymbol{\beta}^T (\mathbf{A}\mathbf{x} - \mathbf{b})$$

so the dual ascent method consists of iterating the updates:

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\beta}_k) \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \gamma_k (\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}) \end{aligned}$$

With appropriate choice of γ_k the dual function increases in each step, i.e., $L(\boldsymbol{\beta}_{k+1}) > L(\boldsymbol{\beta}_k)$.

Dual decomposition (1).

Dual ascent can lead to a decentralized algorithm if f is separable $f(\mathbf{x}) = \sum_{j=1}^{d} f_j(x_j)$ where x_j represents the corresponding coordinate.

Dual decomposition (1).

Dual ascent can lead to a decentralized algorithm if f is separable $f(\mathbf{x}) = \sum_{j=1}^{d} f_j(x_j)$ where x_j represents the corresponding coordinate.

- **Note** that matrix $\mathbf{A}$ can be also correspondingly partitioned into columns $\mathbf{A} = \left[\begin{array}{ccc} \mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_d \end{array} \right]$ so $\mathbf{Ax} = \sum_{j=1}^{d} \mathbf{A}_j x_j$.

Dual decomposition (1).

Dual ascent can lead to a decentralized algorithm if f is separable $f(\mathbf{x}) = \sum_{j=1}^{d} f_j(x_j)$ where x_j represents the corresponding coordinate.

- **Note** that matrix $\mathbf{A}$ can be also correspondingly partitioned into columns $\mathbf{A} = \left[\mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_d \right]$ so $\mathbf{A}\mathbf{x} = \sum_{j=1}^{d} \mathbf{A}_j x_j$.

Taking into account that

$$\boldsymbol{\beta}^T (\mathbf{A}\mathbf{x} - \mathbf{b}) = \sum_{j=1}^{d} \left(\boldsymbol{\beta}^T \mathbf{A}_j x_j - \frac{1}{d} \boldsymbol{\beta}^T \mathbf{b} \right)$$

Dual decomposition (1).

Dual ascent can lead to a decentralized algorithm if f is separable $f(\mathbf{x}) = \sum_{j=1}^{d} f_j(x_j)$ where x_j represents the corresponding coordinate.

- **Note** that matrix $\mathbf{A}$ can be also correspondingly partitioned into columns $\mathbf{A} = \left[\mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_d \right]$ so $\mathbf{A}\mathbf{x} = \sum_{j=1}^{d} \mathbf{A}_j x_j$.

Taking into account that

$$\boldsymbol{\beta}^T (\mathbf{A}\mathbf{x} - \mathbf{b}) = \sum_{j=1}^{d} \left(\boldsymbol{\beta}^T \mathbf{A}_j x_j - \frac{1}{d} \boldsymbol{\beta}^T \mathbf{b} \right)$$

The Lagrangian can be written as:

$$L(\mathbf{x}, \boldsymbol{\beta}) = \sum_{j=1}^{d} L(x_j, \boldsymbol{\beta}) = \sum_{j=1}^{d} \left(f_j(x_j) + \boldsymbol{\beta}^T \mathbf{A}_j x_j - \frac{1}{d} \boldsymbol{\beta}^T \mathbf{b} \right)$$

Dual decomposition (1).

Dual ascent can lead to a decentralized algorithm if f is separable $f(\mathbf{x}) = \sum_{j=1}^{d} f_j(x_j)$ where x_j represents the corresponding coordinate.

- **Note** that matrix $\mathbf{A}$ can be also correspondingly partitioned into columns $\mathbf{A} = \left[\mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_d \right]$ so $\mathbf{A}\mathbf{x} = \sum_{j=1}^{d} \mathbf{A}_j x_j$.

Taking into account that

$$\boldsymbol{\beta}^T (\mathbf{A}\mathbf{x} - \mathbf{b}) = \sum_{j=1}^{d} \left(\boldsymbol{\beta}^T \mathbf{A}_j x_j - \frac{1}{d} \boldsymbol{\beta}^T \mathbf{b} \right)$$

The Lagrangian can be written as:

$$L(\mathbf{x}, \boldsymbol{\beta}) = \sum_{j=1}^{d} L(x_j, \boldsymbol{\beta}) = \sum_{j=1}^{d} \underbrace{\left(f_j(x_j) + \boldsymbol{\beta}^T \mathbf{A}_j x_j - \frac{1}{d} \boldsymbol{\beta}^T \mathbf{b} \right)}_{L_j(x_j, y)}$$

Dual decomposition (2).

Thus, the minimization step can be split in d separate problems that can be solved in parallel:

$$x_{k+1,j} = \arg\min_{x_j} L_j\left(x_j, \boldsymbol{\beta}_k\right) \qquad \forall j$$
$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \gamma_k\left(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}\right)$$

Dual decomposition (2).

Thus, the minimization step can be split in d separate problems that can be solved in parallel:

$$x_{k+1,j} = \arg\min_{x_j} L_j\left(x_j, \boldsymbol{\beta}_k\right) \qquad \forall j$$
$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \gamma_k\left(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}\right)$$

Each iteration of the dual decomposition method requires a broadcast $(x_{k+1,j})$ to the rest of the nodes and a gather operation to obtain $\boldsymbol{\beta}_{k+1}$ locally, and hence, it is similar to the **Map-Reduce** scheme introduced previously.

Precursor 3. Augmented Lagrangian and the method of multipliers

To guarantee convergence without the assumption of strict convexity of f, we can define the augmented Lagrangian:

$$L_\rho(\mathbf{x}, \boldsymbol{\beta}) = f(\mathbf{x}) + \boldsymbol{\beta}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

This problem is equivalent to the original one and can be solved by the dual ascent method, which is known in this case as the method of multipliers.

Precursor 3. Augmented Lagrangian and the method of multipliers

To guarantee convergence without the assumption of strict convexity of f, we can define the augmented Lagrangian:

$$L_\rho(\mathbf{x}, \boldsymbol{\beta}) = f(\mathbf{x}) + \boldsymbol{\beta}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

This problem is equivalent to the original one and can be solved by the dual ascent method, which is known in this case as the method of multipliers.

By definition, $\mathbf{x}_{k+1}$ minimises $L_\rho(\mathbf{x}, \boldsymbol{\beta}_k)$, so:

$$\begin{aligned} 0 &= \nabla_{\mathbf{x}} L_\rho(\mathbf{x}_{k+1}, \boldsymbol{\beta}_k) \\ &= \nabla_{\mathbf{x}} f(\mathbf{x}_{k+1}) + \mathbf{A}^T(\boldsymbol{\beta}_k + \rho(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b})) \\ &= \nabla_{\mathbf{x}} f(\mathbf{x}_{k+1}) + \mathbf{A}^T\boldsymbol{\beta}_{k+1} \end{aligned}$$

Augmented Lagrangian and the method of multipliers

Thus, the minimisation step looks similar:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} L_\rho\left(\mathbf{x}, \boldsymbol{\beta}_k\right)$$
$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \rho\left(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}\right)$$

Augmented Lagrangian and the method of multipliers

Thus, the minimisation step looks similar:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} L_\rho\left(\mathbf{x}, \boldsymbol{\beta}_k\right)$$
$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \rho\left(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}\right)$$

This method has superior convergence due to its strong convexity properties, but it is not separable as the former one.

Augmented Lagrangian and the method of multipliers

Thus, the minimisation step looks similar:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} L_\rho\left(\mathbf{x}, \boldsymbol{\beta}_k\right)$$
$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \rho\left(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}\right)$$

This method has superior convergence due to its strong convexity properties, but it is not separable as the former one.

ADMM intends to integrate these two advantages [1].

Alternating Direction Method of Multipliers (1)

ADMM solves problems in the form:

$$\min_{\mathbf{x},\mathbf{z}} g(\mathbf{x}) + h(\mathbf{z})$$
$$s.t.\ \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$$

where g and h are convex and the former variable $\mathbf{x}$ has been split into two parts $\mathbf{x}, \mathbf{z}$.

Alternating Direction Method of Multipliers (1)

ADMM solves problems in the form:

$$\min_{\mathbf{x},\mathbf{z}} g(\mathbf{x}) + h(\mathbf{z})$$
$$s.t.\ \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$$

where g and h are convex and the former variable $\mathbf{x}$ has been split into two parts $\mathbf{x}, \mathbf{z}$.

The augmented Lagrangian is:

$$L_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\beta}) = g(\mathbf{x}) + h(\mathbf{z}) + \boldsymbol{\beta}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2$$

Alternating Direction Method of Multipliers (1)

ADMM solves problems in the form:

$$\min_{\mathbf{x},\mathbf{z}} g(\mathbf{x}) + h(\mathbf{z})$$
$$s.t.\ \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$$

where g and h are convex and the former variable $\mathbf{x}$ has been split into two parts $\mathbf{x}, \mathbf{z}$.

The augmented Lagrangian is:

$$L_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\beta}) = g(\mathbf{x}) + h(\mathbf{z}) + \boldsymbol{\beta}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2$$

so, ADMM consists of the iterations:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}_k, \boldsymbol{\beta}_k)$$
$$\mathbf{z}_{k+1} = \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}_{k+1}, \mathbf{z}, \boldsymbol{\beta}_k)$$
$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \rho(\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c})$$

Alternating Direction Method of Multipliers (2)

1. Although $(\mathbf{x}, \mathbf{z})$ can be updated simultaneously, ADMM proposes a sequential fashion in alternating directions. This precisely allows for decomposition when g or h are separable.

Alternating Direction Method of Multipliers (2)

1. Although $(\mathbf{x}, \mathbf{z})$ can be updated simultaneously, ADMM proposes a sequential fashion in alternating directions. This precisely allows for decomposition when g or h are separable.
2. ADMM can be written in a slightly different form. Defining the residual $\mathbf{r} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}$ and combining the linear and quadratic terms [2]:

$$\begin{aligned} \boldsymbol{\beta}^T\mathbf{r} + (\rho/2)\|\mathbf{r}\|_2^2 &= (\rho/2)\|\mathbf{r} + (1/\rho)\boldsymbol{\beta}\|_2^2 - (1/2\rho)\|\boldsymbol{\beta}\|_2^2 \\ &= (\rho/2)\|\mathbf{r} + \mathbf{u}\|_2^2 - (\rho/2)\|\mathbf{u}\|_2^2 \end{aligned}$$

where $\mathbf{u} = (1/\rho)\boldsymbol{\beta}$ is the *scaled dual variable*.

Alternating Direction Method of Multipliers (3)

Using $\mathbf{u}$, ADMM looks like:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \left(g(\mathbf{x}) + \frac{\rho}{2} \left\| \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k - \mathbf{c} + \mathbf{u}_k \right\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}} \left(h(\mathbf{z}) + \frac{\rho}{2} \left\| \mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}_k \right\|_2^2 \right) \\
\mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c}
\end{aligned}$$

Alternating Direction Method of Multipliers (3)

Using $\mathbf{u}$, ADMM looks like:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \left(g(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k - \mathbf{c} + \mathbf{u}_k\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}} \left(h(\mathbf{z}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}_k\|_2^2 \right) \\
\mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c}
\end{aligned}$$

Under very mild conditions, as $k \to \infty$ ADMM satisfies:

Residual convergence : $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{z}_k - \mathbf{c} \to 0$

Objective convergence : $g(\mathbf{x}_k) + h(\mathbf{z}_k)$ approaches the optimal

Dual variable convergence : $\boldsymbol{\beta}_k \to \boldsymbol{\beta}^*$ (dual optimal point)

ADMM in Machine Learning problems

ML problems can be straightforwardly rewritten in ADMM form as:

$$\min_{\mathbf{x},\mathbf{z}} g(\mathbf{x}) + h(\mathbf{z})$$
$$s.t.\ \mathbf{x} - \mathbf{z} = 0$$

ADMM in Machine Learning problems

ML problems can be straightforwardly rewritten in ADMM form as:

$$\min_{\mathbf{x},\mathbf{z}} g(\mathbf{x}) + h(\mathbf{z})$$
$$s.t.\ \mathbf{x} - \mathbf{z} = 0$$

Whose solution is:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \left(g(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}_k + \mathbf{u}_k\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}} \left(h(\mathbf{z}) + \frac{\rho}{2} \|\mathbf{x}_{k+1} - \mathbf{z} + \mathbf{u}_k\|_2^2 \right) \\
\mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1}
\end{aligned}$$

ADMM in Machine Learning problems

ML problems can be straightforwardly rewritten in ADMM form as:

$$\min_{\mathbf{x},\mathbf{z}} g(\mathbf{x}) + h(\mathbf{z})$$
$$s.t.\ \mathbf{x} - \mathbf{z} = 0$$

Whose solution is:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \left(g(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}_k + \mathbf{u}_k\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}} \left(h(\mathbf{z}) + \frac{\rho}{2} \|\mathbf{x}_{k+1} - \mathbf{z} + \mathbf{u}_k\|_2^2 \right) \\
\mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1}
\end{aligned}$$

Observe that the two first equations are proximal problems of the form:

$$\text{Prox}_{\eta f}(z) = \arg\min_{x} \left(f(x) + \frac{1}{2\eta} \|x - z\|_2^2 \right)$$

Example 7.3

Let's consider the ridge regression problem from Example 6.1:

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

Example 7.3

Let's consider the ridge regression problem from `Example 6.1`:

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

To solve it using ADMM, we define $g(\mathbf{w})=\frac{1}{n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2$ and $h(\mathbf{z})=\frac{\lambda}{2}\|\mathbf{z}\|_2^2$

Example 7.3

Let's consider the ridge regression problem from `Example 6.1`:

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

To solve it using ADMM, we define $g(\mathbf{w})=\frac{1}{n}\left\|\mathbf{X}\mathbf{w}-\mathbf{y}\right\|_2^2$ and $h(\mathbf{z})=\frac{\lambda}{2}\left\|\mathbf{z}\right\|_2^2$

Using the augmented Lagrangian formulation:

$$L_\rho\left(\mathbf{w},\mathbf{z},\boldsymbol{\beta}\right)=g\left(\mathbf{w}\right)+h\left(\mathbf{z}\right)+\boldsymbol{\beta}^T\left(\mathbf{w}-\mathbf{z}\right)+\frac{\rho}{2}\left\|\mathbf{w}-\mathbf{z}\right\|_2^2$$

Example 7.3

Let's consider the ridge regression problem from `Example 6.1`:

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

To solve it using ADMM, we define $g(\mathbf{w})=\frac{1}{n}\left\|\mathbf{X}\mathbf{w}-\mathbf{y}\right\|_2^2$ and $h(\mathbf{z})=\frac{\lambda}{2}\left\|\mathbf{z}\right\|_2^2$

Using the augmented Lagrangian formulation:

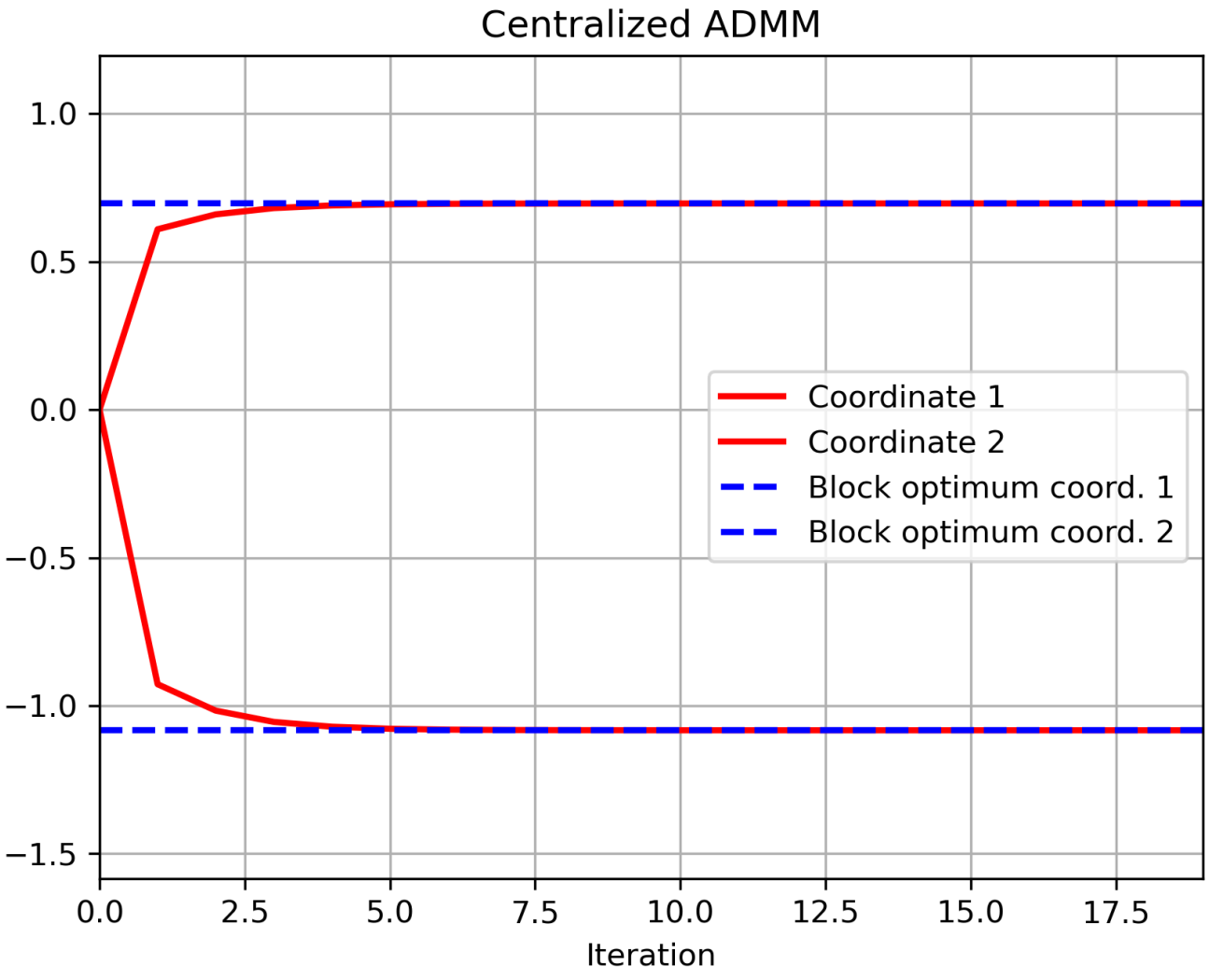
$$L_\rho\left(\mathbf{w},\mathbf{z},\boldsymbol{\beta}\right)=g\left(\mathbf{w}\right)+h\left(\mathbf{z}\right)+\boldsymbol{\beta}^T\left(\mathbf{w}-\mathbf{z}\right)+\frac{\rho}{2}\left\|\mathbf{w}-\mathbf{z}\right\|_2^2$$

It is easy to find the updating formulas:

$$\begin{aligned}\mathbf{w}_{k+1} &= \left(2\mathbf{X}^T\mathbf{X}-n\rho\mathbf{I}\right)^{-1}\left(2\mathbf{X}^T\mathbf{b}+n\rho\mathbf{z}_k-n\boldsymbol{\beta}_k\right)\\ \mathbf{z}_{k+1} &= \left(\boldsymbol{\beta}_k+\rho\mathbf{w}_k\right)/(\lambda+\rho)\\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k+\rho(\mathbf{w}_{k+1}-\mathbf{z}_{k+1})\end{aligned}$$

Example 7.3

Look at the notebook `Example_7_3`, which applies the solution for a problem with two variables and four samples.



General L_1 Regularized Loss Minimization (1)

Consider the generic problem:

$$\min_{\mathbf{x}} \left(g\left(\mathbf{x}\right) + \lambda \left\|\mathbf{x}\right\|_1 \right) \rightarrow \begin{array}{c} \min\limits_{\mathbf{x},\mathbf{z}} \left(g\left(\mathbf{x}\right) + h\left(\mathbf{z}\right) \right) \\ s.t. : \mathbf{x} - \mathbf{z} = 0 \end{array}$$

where g is any convex loss function and $h\left(\mathbf{z}\right) = \lambda \left\|\mathbf{z}\right\|_1$.

General L_1 Regularized Loss Minimization (1)

Consider the generic problem:

$$\min_{\mathbf{x}} \left(g\left(\mathbf{x}\right) + \lambda \left\|\mathbf{x}\right\|_1 \right) \rightarrow \begin{array}{c} \min\limits_{\mathbf{x},\mathbf{z}} \left(g\left(\mathbf{x}\right) + h\left(\mathbf{z}\right) \right) \\ s.t. : \mathbf{x} - \mathbf{z} = 0 \end{array}$$

where g is any convex loss function and $h\left(\mathbf{z}\right) = \lambda \left\|\mathbf{z}\right\|_1$.

Using the scaled dual variable ADMM form, the problem can be written as:

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \left(g\left(\mathbf{x}\right) + \frac{\rho}{2} \left\|\mathbf{x} - \mathbf{z}_k + \mathbf{u}_k\right\|_2^2 \right) \\ \mathbf{z}_{k+1} &= \text{Soft}\left(\mathbf{x}_{k+1} + \mathbf{u}_k, \frac{\lambda}{\rho} \right) \\ \mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1} \end{aligned}$$

General L_1 Regularized Loss Minimization (2)

- Soft $(\cdot, \cdot)$ refers to the thresholding operator defined in Chapter 3.
- Depending on g term $\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} \left(g(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}_k + \mathbf{u}_k\|_2^2 \right)$ can be solved using different approaches.
  - If g is smooth, it can be solved using Newton, quasi-Newton, or the conjugate gradient algorithms.
  - If quadratic, it leads to a set of linear equations.
  - If non-smooth, proximal, or subdifferential methods have to be applied.

case_study_7_1

Use the ADMM formulation to solve the LASSO cost function. In this case, $g(\mathbf{w}) = \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ and $g(\mathbf{z}) = \lambda \|\mathbf{z}\|_1$

case_study_7_1

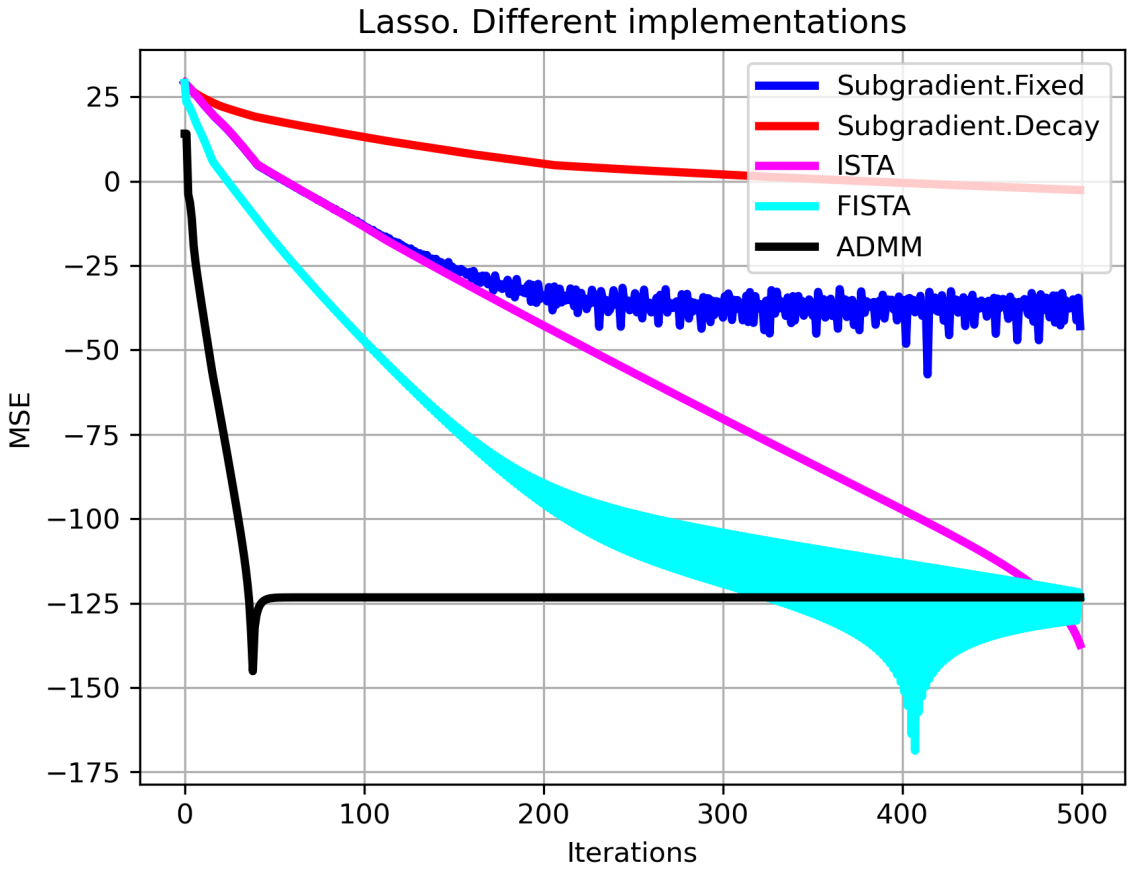
Use the ADMM formulation to solve the LASSO cost function. In this case, $g(\mathbf{w}) = \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ and $g(\mathbf{z}) = \lambda \|\mathbf{z}\|_1$

Thus, the algorithm will look like this:

$$\begin{aligned}
\mathbf{w}_{k+1} &= \left(\frac{2}{n}\mathbf{X}^T\mathbf{X} + \rho\mathbf{I}\right)^{-1}\left(\frac{2}{n}\mathbf{X}^T\mathbf{y} + \rho\left(\mathbf{z}_k - \mathbf{u}_k\right)\right) \\
\mathbf{z}_{k+1} &= \text{Soft}\left(\mathbf{w}_{k+1} + \mathbf{u}_k, \frac{\lambda}{\rho}\right) \\
\mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{w}_{k+1} - \mathbf{z}_{k+1}
\end{aligned}$$

case_study_7_1

You should obtain a result like the following one:



The ADMM Logistic-L_2 algorithm

Let's remember the logistic loss function and add a L_2 regularisation term:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp\left(-y_i \mathbf{w}^T \mathbf{x}_i\right)) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

The ADMM Logistic-L_2 algorithm

Let's remember the logistic loss function and add a L_2 regularisation term:

$$\mathcal{L} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\log(1+\exp\left(-y_i\mathbf{w}^T\mathbf{x}_i\right))}_{g(\mathbf{w})} + \underbrace{\frac{\lambda}{2}\left\|\mathbf{z}\right\|_2^2}_{h(\mathbf{z})}$$

$$\nabla g(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} -y_i\mathbf{x}_i\left(\frac{1}{1+\exp\left(y_i\mathbf{w}^T\mathbf{x}_i\right)}\right)$$

The ADMM Logistic-L_2 algorithm

Let's remember the logistic loss function and add a L_2 regularisation term:

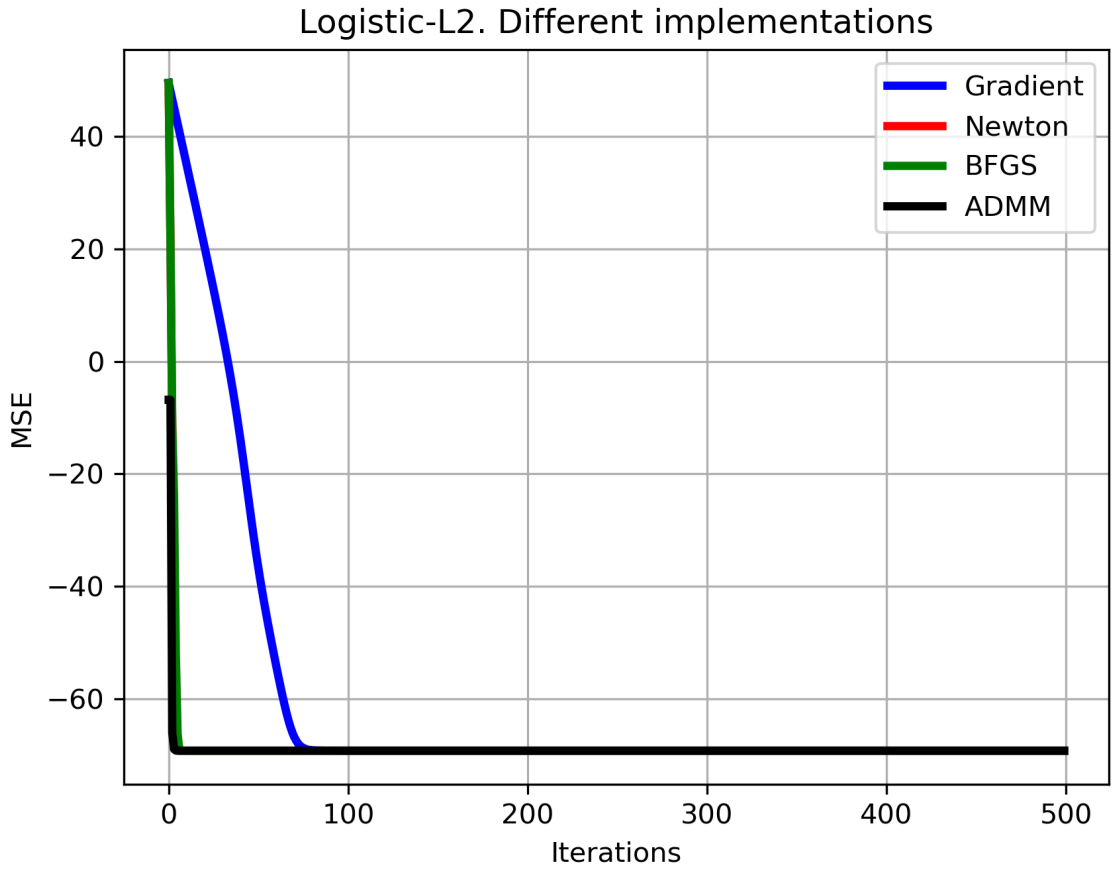
$$\mathcal{L} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\log(1+\exp\left(-y_i\mathbf{w}^T\mathbf{x}_i\right))}_{g(\mathbf{w})} + \underbrace{\frac{\lambda}{2}\left\|\mathbf{z}\right\|_2^2}_{h(\mathbf{z})}$$

$$\nabla g(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} -y_i\mathbf{x}_i\left(\frac{1}{1+\exp\left(y_i\mathbf{w}^T\mathbf{x}_i\right)}\right)$$

Therefore, the minimisation of $\mathbf{w}_{k+1}$ step can addressed using a gradient-based algorithm and $\mathbf{z}_{k+1}$ can be solved using the `prox_quadratic` function.

case_study_7_2

You should obtain a result like the following one:



Consensus ADMM

The main idea is to formulate **ML problems** using the ADMM framework within a new perspective known as ***Consensus***.

Consensus ADMM

The main idea is to formulate **ML problems** using the ADMM framework within a new perspective known as ***Consensus***.

Let's consider a simple problem:

$$\min_{\mathbf{x}} f(x) = \min_{\mathbf{x}} \sum_{i=1}^{n_b} g_i(\mathbf{x})$$

Consensus ADMM

The main idea is to formulate **ML problems** using the ADMM framework within a new perspective known as ***Consensus***.

Let's consider a simple problem:

$$\min_{\mathbf{x}} f(x) = \min_{\mathbf{x}} \sum_{i=1}^{n_b} g_i(\mathbf{x})$$

where

- $\mathbf{x} \in \mathbb{R}^{d+1}$ and $g_i : \mathbb{R}^{d+1} \to \mathbb{R} \cup \{+\infty\}$ are convex.
- Initially, let's consider $h(\mathbf{x}) = 0$.
- n_b refers to the number of blocks or nodes where our data is collected.

Initial simple approach (2)

Suppose that we intend to solve a LS problem with n samples:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{a}_i^T \mathbf{x} - b_i \right)^2$$

Initial simple approach (2)

Suppose that we intend to solve a LS problem with n samples:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{a}_i^T\mathbf{x} - b_i\right)^2$$

but we want to split our calculations into two nodes:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \left(\frac{1}{n}\sum_{i=1}^{n/2}\left(\mathbf{a}_i^T\mathbf{x} - b_i\right)^2 + \frac{1}{n}\sum_{n/2+1}^{n}\left(\mathbf{a}_i^T\mathbf{x} - b_i\right)^2\right)$$

Initial simple approach (2)

Suppose that we intend to solve a LS problem with n samples:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{a}_i^T \mathbf{x} - b_i \right)^2$$

but we want to split our calculations into two nodes:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \left(\frac{1}{n} \sum_{i=1}^{n/2} \left(\mathbf{a}_i^T \mathbf{x} - b_i \right)^2 + \frac{1}{n} \sum_{n/2+1}^{n} \left(\mathbf{a}_i^T \mathbf{x} - b_i \right)^2 \right)$$

We can define for each node the following functions to optimise:

$$g_1(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n/2} \left(\mathbf{a}_i^T \mathbf{x} - b_i \right)^2, \quad g_2(\mathbf{x}) = \frac{1}{n} \sum_{n/2+1}^{n} \left(\mathbf{a}_i^T \mathbf{x} - b_i \right)^2$$

Initial simple approach (3)

The **goal** now is to solve this problem so that a different processing unit can handle each term.

Initial simple approach (3)

The **goal** now is to solve this problem so that a different processing unit can handle each term.

This problem can be rewritten assuming that each node has its local version of the vector $\mathbf{x}_i$ and a shared global variable $\mathbf{z}$

$$\begin{aligned} &\min_{\mathbf{x}_i} \sum_{i=1}^{n_b} g_i\left(\mathbf{x}_i\right) \\ &\text{s.t. } \mathbf{x}_i - \mathbf{z} = 0 \quad i = 1, \cdots, n_b \end{aligned}$$

with $n_b = 2$

Initial simple approach (3)

The **goal** now is to solve this problem so that a different processing unit can handle each term.

This problem can be rewritten assuming that each node has its local version of the vector $\mathbf{x}_i$ and a shared global variable $\mathbf{z}$

$$\begin{aligned} &\min_{\mathbf{x}_i} \sum_{i=1}^{n_b} g_i\left(\mathbf{x}_i\right) \\ &\text{s.t. } \mathbf{x}_i - \mathbf{z} = 0 \quad i = 1, \cdots, n_b \end{aligned}$$

with $n_b = 2$

This formulation is called the **Global Consensus Problem**, since the constraint is that all the local variables should agree.

Initial simple approach (4)

ADMM can be derived directly:

$$\mathrm{L}_\rho\left(\mathbf{x}_1, \cdots, \mathbf{x}_{n_b}, \mathbf{z}, \boldsymbol{\beta}_1, \cdots \boldsymbol{\beta}_{n_b}\right) = \sum_{i=1}^{n_b} \left(g_i\left(\mathbf{x}_i\right) + \boldsymbol{\beta}_i^T\left(\mathbf{x}_i - \mathbf{z}\right) + \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{z}\right\|_2^2 \right)$$

Initial simple approach (4)

ADMM can be derived directly:

$$\mathrm{L}_\rho\left(\mathbf{x}_1,\cdots,\mathbf{x}_{n_b},\mathbf{z},\boldsymbol{\beta}_1,\cdots\boldsymbol{\beta}_{n_b}\right)=\sum_{i=1}^{n_b}\left(g_i\left(\mathbf{x}_i\right)+\boldsymbol{\beta}_i^T\left(\mathbf{x}_i-\mathbf{z}\right)+\frac{\rho}{2}\left\|\mathbf{x}_i-\mathbf{z}\right\|_2^2\right)$$

Making a similar development as in the centralized ADMM, we can obtain the following algorithm:

$$\begin{aligned}
\mathbf{x}_{i,k+1} &= \arg\min_{\mathbf{x}_i}\left(g_i\left(\mathbf{x}_i\right)+\boldsymbol{\beta}_{i,k}^T\left(\mathbf{x}_i-\mathbf{z}_k\right)+\frac{\rho}{2}\left\|\mathbf{x}_i-\mathbf{z}_k\right\|_2^2\right)\\
\mathbf{z}_{k+1} &= \frac{1}{n_b}\sum_{i=1}^{n_b}\left(\mathbf{x}_{i,k+1}+\frac{1}{\rho}\boldsymbol{\beta}_{i,k}\right)\\
\boldsymbol{\beta}_{i,k+1} &= \boldsymbol{\beta}_{i,k}+\rho\left(\mathbf{x}_{i,k+1}-\mathbf{z}_{k+1}\right)
\end{aligned}$$

Initial simple approach (4)

ADMM can be derived directly:

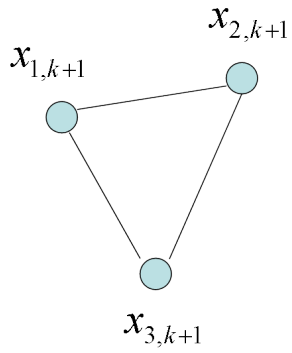
$$\mathrm{L}_\rho\left(\mathbf{x}_1,\cdots,\mathbf{x}_{n_b},\mathbf{z},\boldsymbol{\beta}_1,\cdots\boldsymbol{\beta}_{n_b}\right)=\sum_{i=1}^{n_b}\left(g_i\left(\mathbf{x}_i\right)+\boldsymbol{\beta}_i^T\left(\mathbf{x}_i-\mathbf{z}\right)+\frac{\rho}{2}\left\|\mathbf{x}_i-\mathbf{z}\right\|_2^2\right)$$

Making a similar development as in the centralized ADMM, we can obtain the following algorithm:

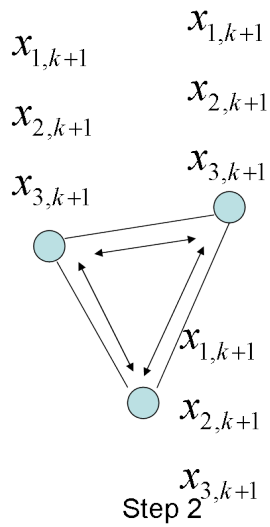
$$\begin{aligned}\mathbf{x}_{i,k+1} &= \arg\min_{\mathbf{x}_i}\left(g_i\left(\mathbf{x}_i\right)+\boldsymbol{\beta}_{i,k}^T\left(\mathbf{x}_i-\mathbf{z}_k\right)+\frac{\rho}{2}\left\|\mathbf{x}_i-\mathbf{z}_k\right\|_2^2\right)\\ \mathbf{z}_{k+1} &= \frac{1}{n_b}\sum_{i=1}^{n_b}\left(\mathbf{x}_{i,k+1}+\frac{1}{\rho}\boldsymbol{\beta}_{i,k}\right)\\ \boldsymbol{\beta}_{i,k+1} &= \boldsymbol{\beta}_{i,k}+\rho\left(\mathbf{x}_{i,k+1}-\mathbf{z}_{k+1}\right)\end{aligned}$$

- Steps carried out independently for every node.
- Step performed in a fusion or data collector.

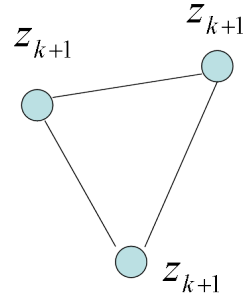
Initial simple approach (5)



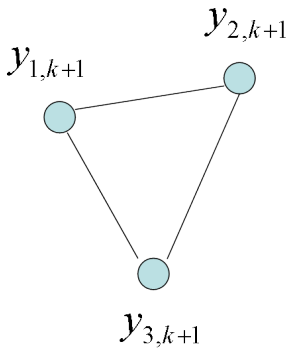
Step 1
Local calculation



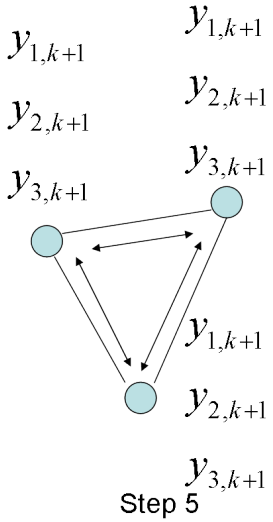
Step 2
Sharing



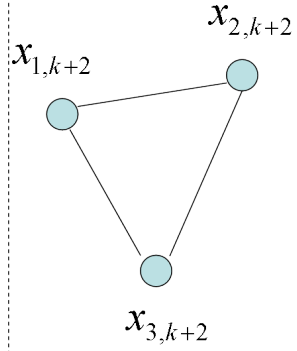
Step 3
Joint calculation



Step 4
Local calculation



Step 5
Sharing



Step 6 = Step 1
Local calculation

Fully connected networks (1)

Let's now include the regularisation effect:

$$\min_{\mathbf{x}_i} \sum_{i=1}^{n_b} g_i\left(\mathbf{x}_i\right) + h(\mathbf{z})$$
$$\text{s.t. } \mathbf{x}_i - \mathbf{z} = 0 \quad i = 1, \cdots, n_b$$

Fully connected networks (1)

Let's now include the regularisation effect:

$$\begin{aligned} &\min_{\mathbf{x}_i} \sum_{i=1}^{n_b} g_i\left(\mathbf{x}_i\right) + h(\mathbf{z}) \\ &\text{s.t. } \mathbf{x}_i - \mathbf{z} = 0 \quad i = 1, \cdots, n_b \end{aligned}$$

The resulting ADMM algorithm is:

$$\begin{aligned} \mathbf{x}_{i,k+1} &= \arg\min_{\mathbf{x}_i} \left(g_i\left(\mathbf{x}_i\right) + \boldsymbol{\beta}_{i,k}^T \left(\mathbf{x}_i - \mathbf{z}_k\right) + \frac{\rho}{2} \left\|\mathbf{x}_i - \mathbf{z}_k\right\|_2^2 \right) \\ \mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}} \left(h(\mathbf{z}) + \sum_{i=1}^{n_b} \left(-\boldsymbol{\beta}_{i,k}^T \mathbf{z} + \frac{\rho}{2} \left\|\mathbf{x}_{i,k+1} - \mathbf{z}\right\|_2^2 \right) \right) \\ \boldsymbol{\beta}_{i,k+1} &= \boldsymbol{\beta}_{i,k} + \rho \left(\mathbf{x}_{i,k+1} - \mathbf{z}_{k+1}\right) \end{aligned}$$

Fully connected networks (2)

Defining again the residual $\mathbf{r} = (\mathbf{x}_i - \mathbf{z}_k)$ as before, and using the scaled dual variable $\mathbf{u}_{i,k} = \frac{1}{\rho}\boldsymbol{\beta}_{i,k}$, ADMM looks like:

$$\begin{aligned}
\mathbf{x}_{i,k+1} &= \arg\min_{\mathbf{x}_i} \left(g_i(\mathbf{x}_i) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}_k + \mathbf{u}_{i,k}\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}} \left(h(\mathbf{z}) + \frac{n_b \rho}{2} \|\mathbf{z} - \bar{\mathbf{x}}_{k+1} - \bar{\mathbf{u}}_k\|_2^2 \right) \\
\mathbf{u}_{i,k+1} &= \mathbf{u}_{i,k} + \mathbf{x}_{i,k+1} - \mathbf{z}_{k+1}
\end{aligned}$$

where $\bar{\mathbf{x}}_k = \frac{1}{n_b}\sum_{i=1}^{n_b} \mathbf{x}_{i,k}$ and $\bar{\mathbf{u}}_k = \frac{1}{n_b}\sum_{i=1}^{n_b} \mathbf{u}_{i,k}$.

Example 7.4

Calculate the solution of `Example 6.1` again, but this time as an ADMM distributed model splitting the samples into four nodes.

$$\min_{\mathbf{w}} f(\mathbf{w}) \rightarrow \begin{cases} \min_{\mathbf{w}} \frac{1}{4} \sum_{i=1}^{4} \left(\mathbf{x}_i^T \mathbf{w}_i - \mathbf{y}_i \right)^2 + \frac{\lambda}{2} \|\mathbf{z}\|_2^2 \\ s.t.\ \mathbf{x}_i - \mathbf{z} = 0 \quad i = 1, \cdots, 4 \end{cases}$$

Example 7.4

Calculate the solution of `Example 6.1` again, but this time as an ADMM distributed model splitting the samples into four nodes.

$$\min_{\mathbf{w}} f(\mathbf{w}) \rightarrow \begin{cases} \min_{\mathbf{w}} \frac{1}{4} \sum_{i=1}^{4} \left(\mathbf{x}_i^T \mathbf{w}_i - \mathbf{y}_i \right)^2 + \frac{\lambda}{2} \|\mathbf{z}\|_2^2 \\ s.t.\ \mathbf{x}_i - \mathbf{z} = 0 \quad i = 1, \cdots, 4 \end{cases}$$

The Lagrangian in this case is

$$L_\rho(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\beta}_3, \boldsymbol{\beta}_4, \mathbf{z}) = \frac{1}{4} \sum_{i=1}^{4} \left(\mathbf{x}_i^T \mathbf{w}_i - b_i \right)^2 + \frac{\lambda}{2} \|\mathbf{z}\|_2^2 +$$

$$+ \sum_{i=1}^{4} \boldsymbol{\beta}_i^T (\mathbf{w}_i - z) + \frac{\rho}{2} \sum_{i=1}^{4} \|\mathbf{w}_i - \mathbf{z}\|_2^2$$

Example 7.4

Taking derivatives with respect to $\mathbf{w}$, $\mathbf{z}$ and $\boldsymbol{\beta}$...

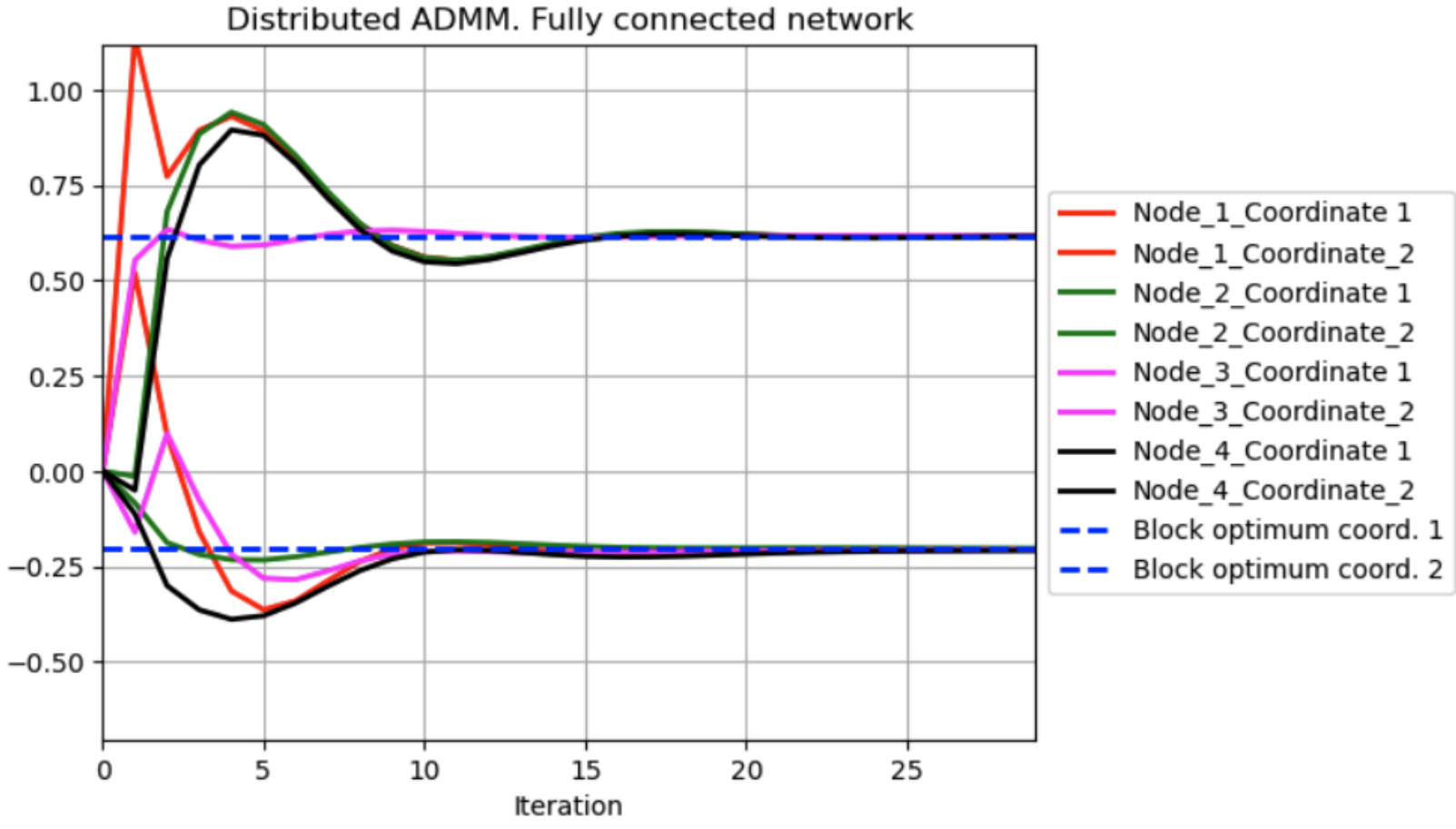
$$\mathbf{w}_i^* = \left(\mathbf{x}_i\mathbf{x}_i^T + 2\rho I\right)^{-1}\left(2\rho\mathbf{z} - 2\boldsymbol{\beta}_i + \mathbf{x}_i^T\mathbf{y}_i\right)$$

$$\mathbf{z}^* = \frac{\sum_{i=1}^{n_b}\boldsymbol{\beta}_i + \rho\sum_{i=1}^{n_b}\mathbf{w}_i}{\lambda + 4\rho}$$

$$0 = \mathbf{w}_i - \mathbf{z}$$

We can directly translate them to the ADMM algorithm. Notebook `Example 7.4` shows an example for $n_b = 4$.

Example 7.4


Distributed ADMM. Fully connected network
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
5
10
15
20
25
Iteration
Node_1_Coordinate 1
Node_1_Coordinate_2
Node_2_Coordinate 1
Node_2_Coordinate_2
Node_3_Coordinate 1
Node_3_Coordinate_2
Node_4_Coordinate 1
Node_4_Coordinate_2
Block optimum coord. 1
Block optimum coord. 2

Case_study_7_3

Implement the LASSO algorithm using distributed ADMM. In this case, ADMM looks like:

$$
\begin{aligned}
\mathbf{w}_{i,k+1} &= \arg\min_{\mathbf{w}_i} \left(\frac{1}{n} \left\| \mathbf{X}_i \mathbf{w}_i - \mathbf{y}_i \right\|_2^2 + \frac{\rho}{2} \left\| \mathbf{w}_i - \mathbf{z}_k + \mathbf{u}_{i,k} \right\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \text{Soft}\left(\bar{\mathbf{w}}_{k+1} + \bar{\mathbf{u}}_k, \lambda/\rho n \right) \\
\mathbf{u}_{i,k+1} &= \mathbf{u}_{i,k} + \mathbf{w}_{i,k+1} - \mathbf{z}_{k+1}
\end{aligned}
$$

Case_study_7_3

Implement the LASSO algorithm using distributed ADMM. In this case, ADMM looks like:

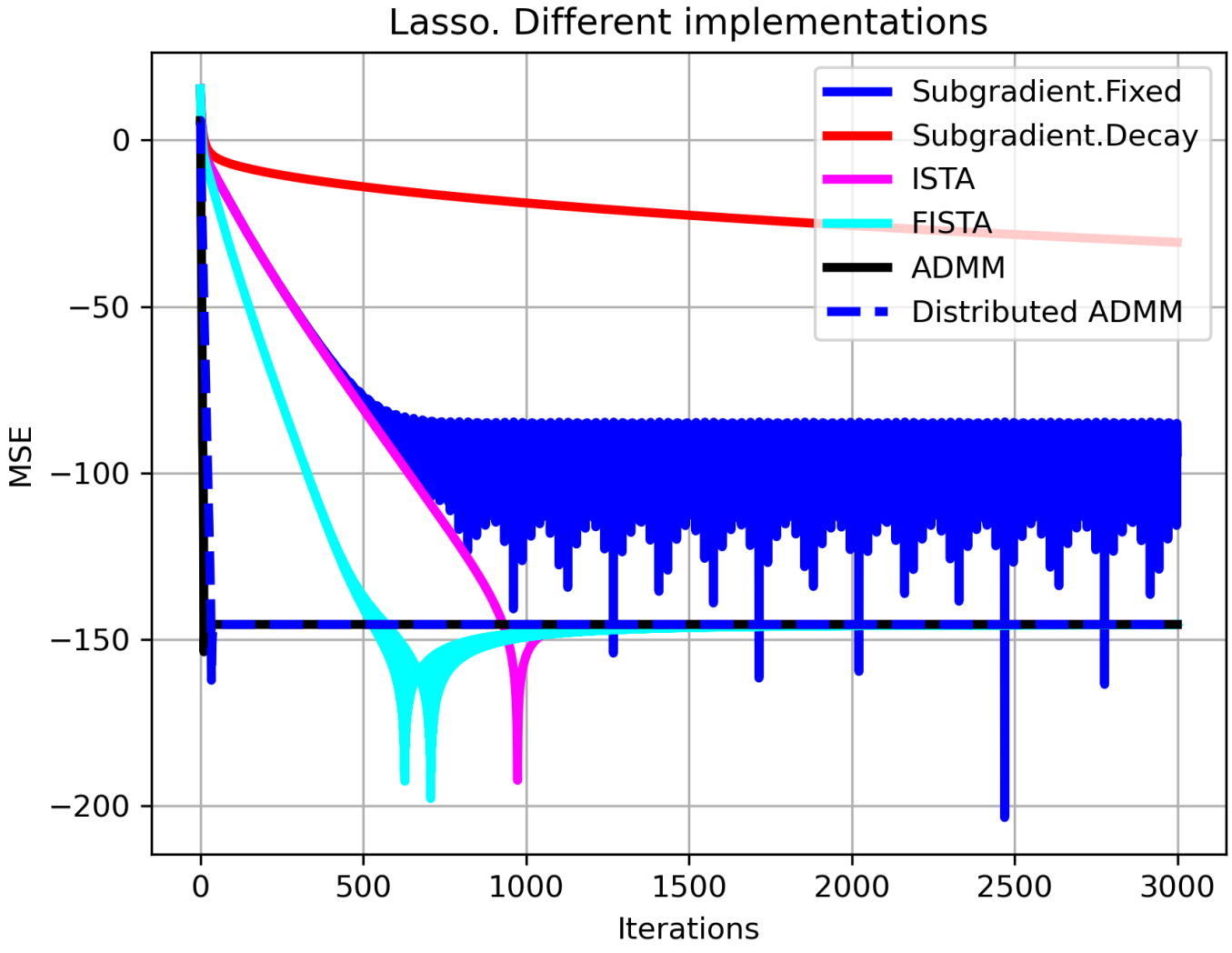
$$\begin{aligned}
\mathbf{w}_{i,k+1} &= \arg\min_{\mathbf{w}_i} \left(\frac{1}{n} \|\mathbf{X}_i \mathbf{w}_i - \mathbf{y}_i\|_2^2 + \frac{\rho}{2} \|\mathbf{w}_i - \mathbf{z}_k + \mathbf{u}_{i,k}\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \text{Soft}\left(\bar{\mathbf{w}}_{k+1} + \bar{\mathbf{u}}_k, \lambda/\rho n\right) \\
\mathbf{u}_{i,k+1} &= \mathbf{u}_{i,k} + \mathbf{w}_{i,k+1} - \mathbf{z}_{k+1}
\end{aligned}$$

The $\mathbf{w}-$update is very simple in this case:

$$\mathbf{w}_{i,k+1} = \left(\frac{2}{n} \mathbf{X}_i^T \mathbf{X}_i + \rho I \right)^{-1} \left(\frac{2}{n} \mathbf{X}_i^T \mathbf{y}_i + \rho \left(\mathbf{z}_k - \mathbf{u}_{i,k} \right) \right)$$

It can be implemented using the `prox_quadratic` function used before.

Case_study_7_3

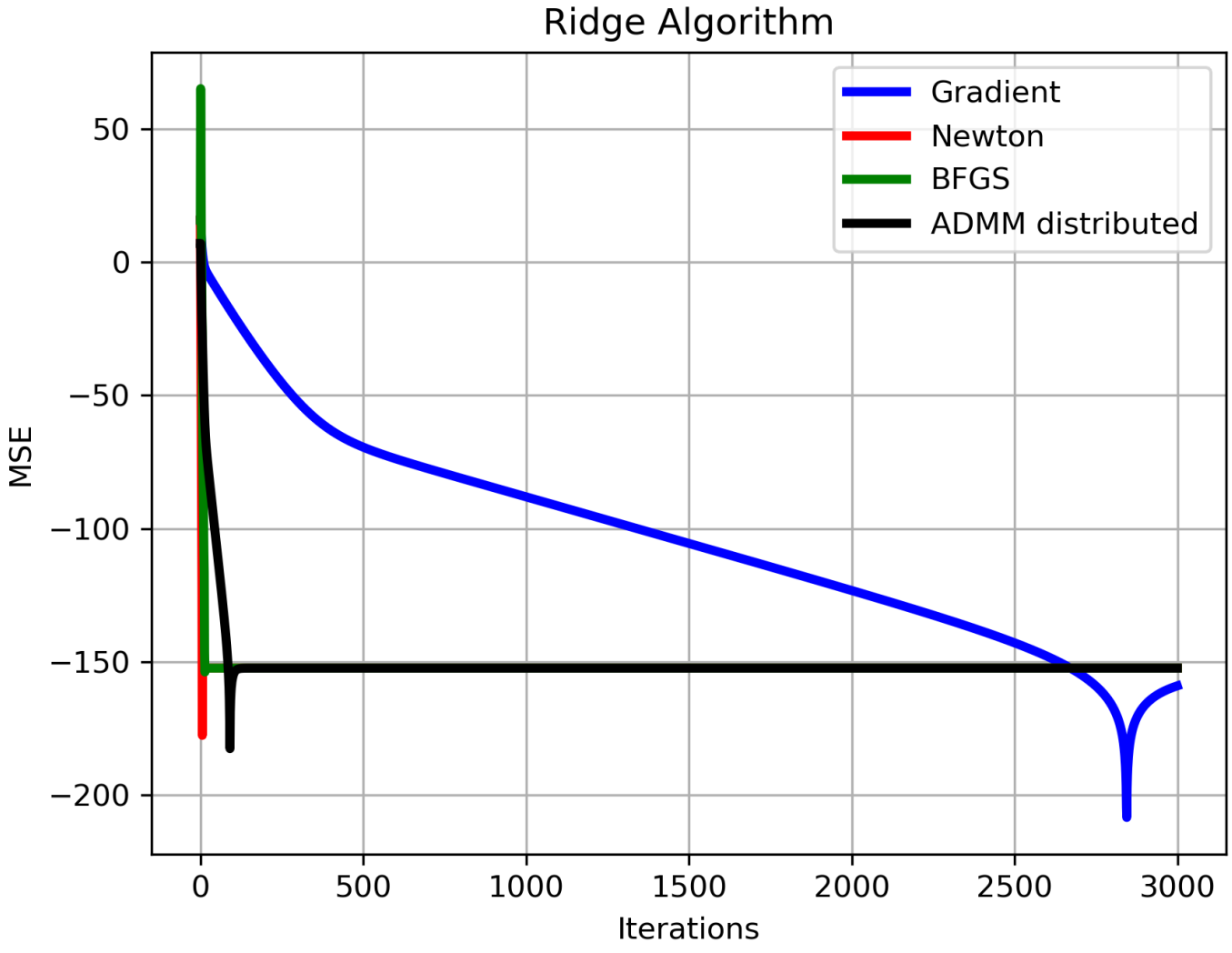


Case_study_7_4

The goal of this case study is to implement the Ridge algorithm using the distributed ADMM. It is very similar to the LASSO problem but with a different second equation.

$$
\begin{aligned}
\mathbf{w}_{i,k+1} &= \arg\min_{x_i} \left(\frac{1}{n} \|\mathbf{X}_i \mathbf{w}_i - \mathbf{y}_i\|_2^2 + \frac{\rho}{2} \|\mathbf{w}_i - \mathbf{z}_k + \mathbf{u}_{i,k}\|_2^2 \right) \\
\mathbf{z}_{k+1} &= \arg\min_{z} \left(\frac{\lambda}{2} \|\mathbf{z}\|_2^2 + \frac{n\rho}{2} \|\mathbf{z} - \bar{\mathbf{w}}_{k+1} - \bar{\mathbf{u}}_k\|_2^2 \right) \\
\mathbf{u}_{i,k+1} &= \mathbf{u}_{i,k} + \mathbf{w}_{i,k+1} - \mathbf{z}_{k+1}
\end{aligned}
$$

Case_study_7_4



Questions?

References

[1] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[2] Jean Gallier and Jocelyn Quaintance. "Fundamentals of optimization theory with applications to machine learning". In: *University of Pennsylvania Philadelphia, PA* 19104 (2019).

Thank You

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