Distributed Optimization

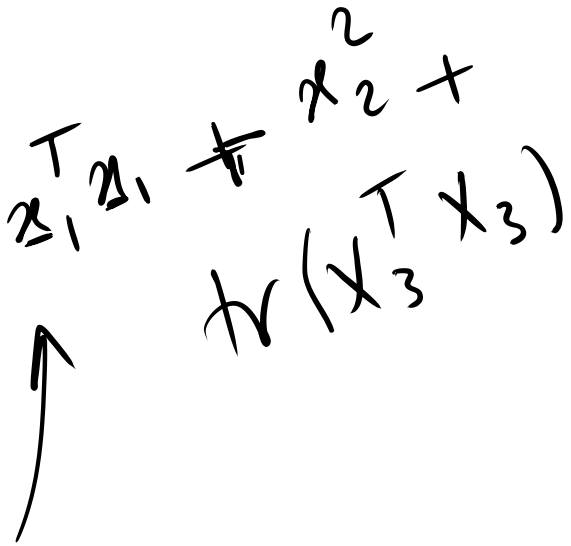
Optimization Techniques (ENGG*6140)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Winter 2023

Problem Statement

Problem Statement



- When we have several optimization variables, we have:

$$\underset{\{\boldsymbol{x}_i\}_{i=1}^{m}}{\text{minimize}} \quad f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m), \tag{1}$$

  where m is the number of optimization variables.
- In this case, we can have **distributed optimization** because we can work on the optimization variables in a distributed manner.

Alternating Optimization

Alternating Optimization

- Consider the following multivariate optimization problem:

$$\underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} \quad f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m),$$

  where the objective function depends on m variables.
- When we have several optimization variables, we can alternate between optimizing over each of these variables. This technique is called **alternating optimization** in the literature [1] (also see [2, Chapter 4]).
- *Alternating optimization* alternates between updating every variable while assuming other variables are constant, set to their last updated value. After random feasible initialization, it updates solutions as [1]:

$$\begin{aligned}
\boldsymbol{x}_1^{(k+1)} &:= \arg\min_{\boldsymbol{x}_1} f(\boldsymbol{x}_1, \boldsymbol{x}_2^{(k)}, \ldots, \boldsymbol{x}_{m-1}^{(k)}, \boldsymbol{x}_m^{(k)}), \\
\boldsymbol{x}_2^{(k+1)} &:= \arg\min_{\boldsymbol{x}_2} f(\boldsymbol{x}_1^{(k+1)}, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{m-1}^{(k)}, \boldsymbol{x}_m^{(k)}), \\
&\vdots \\
\boldsymbol{x}_m^{(k+1)} &:= \arg\min_{\boldsymbol{x}_m} f(\boldsymbol{x}_1^{(k+1)}, \boldsymbol{x}_2^{(k+1)}, \ldots, \boldsymbol{x}_{m-1}^{(k+1)}, \boldsymbol{x}_m),
\end{aligned}$$

  until convergence.
- Any optimization methods, including first-order and second-order methods, can be used for each of the optimization lines above.
- In most cases, alternating optimization is robust to changing the order of updates of variables.

Alternating Optimization for Decomposable Function in terms of Variables

- If the function $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ is decomposable in terms of variables, i.e., if we have:

$$f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) = \sum_{i=1}^{m} f_i(\boldsymbol{x}_i),$$

the alternating optimization can be simplified to:

$$\begin{cases} \boldsymbol{x}_1^{(k+1)} := \arg\min_{\boldsymbol{x}_1} f_1(\boldsymbol{x}_1), \\ \boldsymbol{x}_2^{(k+1)} := \arg\min_{\boldsymbol{x}_2} f_2(\boldsymbol{x}_2), \\ \vdots \\ \boldsymbol{x}_m^{(k+1)} := \arg\min_{\boldsymbol{x}_m} f_m(\boldsymbol{x}_m), \end{cases}$$

because other terms become constant in optimization.

- The above updates mean that if the function is completely decomposable in terms of variables, the updates of variables are independent and can be done independently.
- Hence, in that case, alternating optimization is reduced to m independent optimization problems, each of which can be solved by any optimization method such as the first-order and second-order methods.

Proximal Alternating Optimization

- **Proximal alternating optimization** uses proximal operator:

$$\mathbf{prox}_{\lambda g}(\boldsymbol{x}) := \arg\min_{\boldsymbol{u}} \left(g(\boldsymbol{u}) + \frac{1}{2\lambda} \|\boldsymbol{u} - \boldsymbol{x}\|_2^2 \right), \tag{2}$$

for minimization to keep the updated solution close to the solution of previous iteration [1]:

$$\begin{aligned}
\boldsymbol{x}_1^{(k+1)} &:= \arg\min_{\boldsymbol{x}_1} \left(f(\boldsymbol{x}_1, \boldsymbol{x}_2^{(k)}, \ldots, \boldsymbol{x}_{m-1}^{(k)}, \boldsymbol{x}_m^{(k)}) + \frac{1}{2\lambda} \|\boldsymbol{x}_1 - \boldsymbol{x}_1^{(k)}\|_2^2 \right), \\
\boldsymbol{x}_2^{(k+1)} &:= \arg\min_{\boldsymbol{x}_2} \left(f(\boldsymbol{x}_1^{(k+1)}, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{m-1}^{(k)}, \boldsymbol{x}_m^{(k)}) + \frac{1}{2\lambda} \|\boldsymbol{x}_2 - \boldsymbol{x}_2^{(k)}\|_2^2 \right), \\
&\vdots \\
\boldsymbol{x}_m^{(k+1)} &:= \arg\min_{\boldsymbol{x}_m} \left(f(\boldsymbol{x}_1^{(k+1)}, \boldsymbol{x}_2^{(k+1)}, \ldots, \boldsymbol{x}_{m-1}^{(k+1)}, \boldsymbol{x}_m) + \frac{1}{2\lambda} \|\boldsymbol{x}_m - \boldsymbol{x}_m^{(k)}\|_2^2 \right).
\end{aligned}$$

Alternating Optimization for Constrained Problems

- The alternating optimization methods can also be used for constrained problems:

$$\begin{aligned} &\underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} && f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \\ &\text{subject to} && \boldsymbol{x}_i \in \mathcal{S}_i, \quad \forall i \in \{1, \ldots, m\}. \end{aligned} \tag{3}$$

- In this case, every line of the optimization is a constrained problem:

$$\begin{aligned} \boldsymbol{x}_1^{(k+1)} &:= \arg\min_{\boldsymbol{x}_1} \left(f(\boldsymbol{x}_1, \boldsymbol{x}_2^{(k)}, \ldots, \boldsymbol{x}_{m-1}^{(k)}, \boldsymbol{x}_m^{(k)}), \text{ s.t. } \boldsymbol{x}_1 \in \mathcal{S}_1 \right), \\ \boldsymbol{x}_2^{(k+1)} &:= \arg\min_{\boldsymbol{x}_2} \left(f(\boldsymbol{x}_1^{(k+1)}, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{m-1}^{(k)}, \boldsymbol{x}_m^{(k)}), \text{ s.t. } \boldsymbol{x}_2 \in \mathcal{S}_2 \right), \\ &\vdots \\ \boldsymbol{x}_m^{(k+1)} &:= \arg\min_{\boldsymbol{x}_m} \left(f(\boldsymbol{x}_1^{(k+1)}, \boldsymbol{x}_2^{(k+1)}, \ldots, \boldsymbol{x}_{m-1}^{(k+1)}, \boldsymbol{x}_m), \text{ s.t. } \boldsymbol{x}_m \in \mathcal{S}_m \right). \end{aligned}$$

- Any constrained optimization methods can be used for each of the optimization lines above. Some examples are projected gradient method, proximal methods, interior-point methods, etc.
- Practical experiments have shown there is usually no need to use a complete optimization until convergence for every step in the alternating optimization, either unconstrained or constrained. Often, a single step of updating, such as a step of gradient descent or projected gradient method, is enough for the whole algorithm to work.

Dual Ascent and Dual Decomposition Methods

Dual Ascent Method

- Consider the following problem:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{aligned} \tag{4}$$

- We follow the method of Lagrange multipliers discussed before.
- The Lagrangian is:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \boldsymbol{\nu}^\top (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}).$$

- The dual function is:

$$g(\boldsymbol{\nu}) = \inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}). \tag{5}$$

- The optimal dual problem maximizes $g(\boldsymbol{\nu})$:

$$\boldsymbol{\nu}^* = \arg\max_{\boldsymbol{\nu}} g(\boldsymbol{\nu}), \tag{6}$$

  so the optimal primal variable is:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^*). \tag{7}$$

- For solving Eq. (6), we should take the derivative of the dual function w.r.t. the dual variable:

$$\nabla_{\boldsymbol{\nu}} g(\boldsymbol{\nu}) \overset{(5)}{=} \nabla_{\boldsymbol{\nu}} (\inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu})) \overset{(7)}{=} \nabla_{\boldsymbol{\nu}} (f(\boldsymbol{x}^*) + \boldsymbol{\nu}^\top (\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b})) = \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}.$$

Dual Ascent Method

- We found:

$$\begin{cases} \boldsymbol{\nu}^* = \arg\max_{\boldsymbol{\nu}} g(\boldsymbol{\nu}), \\ \boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^*). \end{cases}$$

- The dual problem is a maximization problem so we can use gradient ascent for iteratively updating the dual variable with this gradient. We can alternate between updating the optimal primal and dual variables:

$$\boldsymbol{x}^{(k+1)} := \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}), \tag{8}$$

$$\boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \eta^{(k)}(\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b}), \tag{9}$$

  where k is the iteration index and $\eta^{(k)}$ is the step size (also called the learning rate) at iteration k.
- Eq. (8) can be performed by any optimization method. We compute the gradient of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)})$ w.r.t. $\boldsymbol{x}$. If setting this gradient to zero does not give $\boldsymbol{x}$ in closed form, we can use gradient descent to perform Eq. (8).
- Some papers approximate Eq. (8) by one step or few steps of gradient descent rather than a complete gradient descent until convergence. If using one step, we can write Eq. (8) as:

$$\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \gamma \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}), \tag{10}$$

  where $\gamma > 0$ is the step size. It has been shown empirically that even one step of gradient descent for Eq. (8) works properly for the whole alternating algorithm.

Dual Ascent Method

- We had:

$$\begin{cases} \boldsymbol{x}^{(k+1)} := \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}), \\ \boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \eta^{(k)} (\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b}), \end{cases}$$

- We continue the iterations until convergence of the primal and dual variables to stable values. When we get closer to convergence, we will have $(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{b}) \to 0$ so that we will not have update of dual variable according to Eq. (9). This means that after convergence, we have $(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{b}) \approx 0$ so that the constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in Eq. (4) is getting satisfied. In other words, the update of dual variable in Eq. (9) is taking care of satisfying the constraint.
- This method is known as the **dual ascent** method because it uses gradient ascent for updating the dual variable.

Dual Decomposition Method

- Again, consider the following problem:

$$\begin{bmatrix} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{bmatrix}$$

- If the objective function can be distributed and decomposed on b blocks $\{\boldsymbol{x}_i\}_{i=1}^{b}$, i.e.:

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}_1) + \cdots + f_b(\boldsymbol{x}_b),$$

we can have b Lagrangian functions where the total Lagrangian is the summation of these functions:

$$\mathcal{L}_i(\boldsymbol{x}_i, \boldsymbol{\nu}) = f(\boldsymbol{x}_i) + \boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{b}),$$

$$\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{\nu}) = \sum_{i=1}^{b} \big(f(\boldsymbol{x}_i) + \boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{b})\big).$$

- We can divide the Eq. (8), $\boldsymbol{x}^{(k+1)} := \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)})$, into b updates, each for one of the blocks.

$$\boldsymbol{x}_i^{(k+1)} := \arg\min_{\boldsymbol{x}_i} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}), \quad \forall i \in \{1, \ldots, b\}, \tag{11}$$

$$\boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \eta^{(k)}(\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b}). \tag{12}$$

Dual Decomposition Method

- We found:

$$\begin{cases} \boldsymbol{x}_i^{(k+1)} := \arg\min_{\boldsymbol{x}_i} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}), & \forall i \in \{1, \dots, b\}, \\ \boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \eta^{(k)}(\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b}). \end{cases}$$

- This is called **dual decomposition** developed by decomposition techniques such as the **Dantzig-Wolfe decomposition** (1960) [3], **Bender's decomposition** (1962) [4], and **Lagrangian decomposition** (1963) [5].
- The dual decomposition methods can divide a problem into sub-problems and solve them in parallel. Hence, it can be used for big data but they are usually slow to converge.

Augmented Lagrangian Method (Method of Multipliers)

Augmented Lagrangian Method (Method of Multipliers)

- Recall Eq. (4):

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{aligned}$$

- Assume we regularize the objective function in Eq. (4) by a penalty on not satisfying the constraint:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{13}$$

where $\rho > 0$ is the regularization parameter.

Definition (Augmented Lagrangian (1969) [6, 7])

The Lagrangian for problem (13) is:

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\nu}) := f(\boldsymbol{x}) + \boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2. \tag{14}$$

This Lagrangian is called the **augmented Lagrangian** for problem (4).

Augmented Lagrangian Method (Method of Multipliers)

- The augmented Lagrangian:

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\nu}) := f(\boldsymbol{x}) + \boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2.$$

- Recall Eqs. (8) and (9):

$$\begin{cases} \boldsymbol{x}^{(k+1)} := \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}), \\ \boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \eta^{(k)}(\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b}), \end{cases}$$

- We can use this augmented Lagrangian in Eqs. (8) and (9):

$$\begin{cases} \boldsymbol{x}^{(k+1)} := \arg\min_{\boldsymbol{x}} \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\nu}^{(k)}), & (15) \\ \boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \rho(\boldsymbol{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b}), & (16) \end{cases}$$

where we use ρ for the step size of updating the dual variable. This method is called the *augmented Lagrangian method* or the **method of multipliers** (1969) [6, 7, 8].

Alternating Direction Method of Multipliers (ADMM)

Alternating Direction Method of Multipliers (ADMM)

- **Alternating Direction Method of Multipliers (ADMM)**, proposed in 1976 [9, 10, 11], has been used in many recent machine learning and signal processing papers.
- The usefulness and goal for using ADMM (and other distributed methods) are two-fold:
  - it makes the problem distributed and parallelizable on several servers,
  - it makes it possible to solve an optimization problem with multiple variables.
- Consider the following problem:

$$\begin{aligned} \underset{\boldsymbol{x}_1, \boldsymbol{x}_2}{\text{minimize}} \quad & f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{x}_2 = \boldsymbol{c}, \end{aligned} \tag{17}$$

  which is an optimization over two variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.
- The augmented Lagrangian for this problem is:

$$\mathcal{L}_\rho(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{\nu}) = f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2) + \boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{x}_2 - \boldsymbol{c}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{x}_2 - \boldsymbol{c}\|_2^2. \tag{18}$$

- We can alternate between updating the primal variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and the dual variable $\boldsymbol{\nu}$ until convergence of these variables:

$$\boldsymbol{x}_1^{(k+1)} := \arg\min_{\boldsymbol{x}_1} \mathcal{L}_\rho(\boldsymbol{x}_1, \boldsymbol{x}_2^{(k)}, \boldsymbol{\nu}^{(k)}), \tag{19}$$

$$\boldsymbol{x}_2^{(k+1)} := \arg\min_{\boldsymbol{x}_2} \mathcal{L}_\rho(\boldsymbol{x}_1^{(k+1)}, \boldsymbol{x}_2, \boldsymbol{\nu}^{(k)}), \tag{20}$$

$$\boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \rho(\boldsymbol{A}\boldsymbol{x}_1^{(k+1)} + \boldsymbol{B}\boldsymbol{x}_2^{(k+1)} - \boldsymbol{c}). \tag{21}$$

Alternating Direction Method of Multipliers (ADMM)

- Note that the order of updating primal and dual variables is important and the dual variable should be updated after the primal variables but the order of updating primal variables is not important.
- A good survey/tutorial on ADMM is by Boyd in 2011 [11].
- As was explained before, Eqs. (19) and (20) can be performed by any optimization method such as calculating the gradient of augmented Lagrangian w.r.t. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, and using a few (or even one) iterations of gradient descent for each of these equations.

Simplifying Equations in ADMM

- The last term in the augmented Lagrangian, Eq. (18), can be restated as:

$$
\begin{aligned}
&\boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c})+\frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c}\|_2^2\\
&=\boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c})+\frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c}\|_2^2+\frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2+\frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2\\
&=\frac{\rho}{2}\Big(\|\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c}\|_2^2+\frac{1}{\rho^2}\|\boldsymbol{\nu}\|_2^2+\frac{2}{\rho}\boldsymbol{\nu}^\top(\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c})\Big)-\frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2\\
&\overset{(a)}{=}\frac{\rho}{2}\Big\|\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c}+\frac{1}{\rho}\boldsymbol{\nu}\Big\|_2^2-\frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2\\
&\overset{(b)}{=}\frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c}+\boldsymbol{u}\|_2^2-\frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2.
\end{aligned}
$$

$(a+b)^2=a^2+b^2+2ab$

where (a) is because of the square of summation of two terms and (b) is because we define $\boldsymbol{u}:=(1/\rho)\boldsymbol{\nu}$.

- The last term $-(1/(2\rho))\|\boldsymbol{\nu}\|_2^2$ is constant w.r.t. the primal variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ so we can drop that term from Lagrangian when updating the primal variables.
- Hence, the augmented Lagrangian can be restated as:

$$
\mathcal{L}_\rho(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{u})=f_1(\boldsymbol{x}_1)+f_2(\boldsymbol{x}_2)+\frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}_1+\boldsymbol{B}\boldsymbol{x}_2-\boldsymbol{c}+\boldsymbol{u}\|_2^2+\text{constant}. \tag{22}
$$

Simplifying Equations in ADMM

- The augmented Lagrangian:

$$\mathcal{L}_\rho(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{u}) = f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2) + \frac{\rho}{2}\left\|\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{x}_2 - \boldsymbol{c} + \boldsymbol{u}\right\|_2^2 + \text{constant}.$$

- For updating $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the terms $f_2(\boldsymbol{x}_2)$ and $f_1(\boldsymbol{x}_1)$ are constant, respectively, and can be dropped (because here arg min is important and not the minimum value). Hence, Eqs. (19), (20), and (21) can be restated as:

$$\boldsymbol{x}_1^{(k+1)} := \arg\min_{\boldsymbol{x}_1} \left(f_1(\boldsymbol{x}_1) + \frac{\rho}{2}\left\|\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{x}_2^{(k)} - \boldsymbol{c} + \boldsymbol{u}^{(k)}\right\|_2^2 \right), \tag{23}$$

$$\boldsymbol{x}_2^{(k+1)} := \arg\min_{\boldsymbol{x}_2} \left(f_2(\boldsymbol{x}_2) + \frac{\rho}{2}\left\|\boldsymbol{A}\boldsymbol{x}_1^{(k+1)} + \boldsymbol{B}\boldsymbol{x}_2 - \boldsymbol{c} + \boldsymbol{u}^{(k)}\right\|_2^2 \right), \tag{24}$$

$$\boldsymbol{u}^{(k+1)} := \boldsymbol{u}^{(k)} + \rho(\boldsymbol{A}\boldsymbol{x}_1^{(k+1)} + \boldsymbol{B}\boldsymbol{x}_2^{(k+1)} - \boldsymbol{c}). \tag{25}$$

- Again, Eqs. (23) and (24) can be performed by one or few steps of gradient descent or any other optimization method.
- The convergence of ADMM for non-convex and non-smooth functions has been analyzed in [12].

ADMM Algorithm for General Optimization Problems and Any Number of Variables

ADMM Algorithm for General Optimization Problems and Any Number of Variables

- ADMM can be extended to several equality and inequality constraints for several optimization variables [13, 14].
- Consider the following optimization problem with m optimization variables and an equality and inequality constraint for every variable:

$$\begin{aligned} \underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} \quad & \sum_{i=1}^{m} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & y_i(\boldsymbol{x}_i) \leq 0, \ i \in \{1, \ldots, m\}, \\ & h_i(\boldsymbol{x}_i) = 0, \ i \in \{1, \ldots, m\}. \end{aligned} \tag{26}$$

- We can convert every inequality constraint to equality constraints by this technique [13, 14]:

$$y_i(\boldsymbol{x}_i) \leq 0 \quad \equiv \quad y_i'(\boldsymbol{x}_i) := \left(\max(0, y_i(\boldsymbol{x}_i))\right)^2 = 0.$$

- Hence, the problem becomes:

$$\begin{aligned} \underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} \quad & \sum_{i=1}^{m} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & y_i'(\boldsymbol{x}_i) = 0, \ i \in \{1, \ldots, m\}, \\ & h_i(\boldsymbol{x}_i) = 0, \ i \in \{1, \ldots, m\}. \end{aligned}$$

ADMM Algorithm for General Optimization Problems and Any Number of Variables

- We found:

$$\begin{aligned} \underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} \quad & \sum_{i=1}^m f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & y_i'(\boldsymbol{x}_i) = 0, \ i \in \{1, \dots, m\}, \\ & h_i(\boldsymbol{x}_i) = 0, \ i \in \{1, \dots, m\}. \end{aligned}$$

- Having dual variables $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_m]^\top$ and $\boldsymbol{\nu} = [\nu_1, \dots, \nu_m]^\top$ and regularization parameter $\rho > 0$, the augmented Lagrangian for this problem is:

$$\begin{aligned} & \mathcal{L}_\rho(\{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{\lambda}, \boldsymbol{\nu}) \\ &= \sum_{i=1}^m f_i(\boldsymbol{x}_i) + \sum_{i=1}^m \lambda_i y_i'(\boldsymbol{x}_i) + \sum_{i=1}^m \nu_i h_i(\boldsymbol{x}_i) + \frac{\eta}{2} \sum_{i=1}^m (y_i'(\boldsymbol{x}_i))^2 + \frac{\rho}{2} \sum_{i=1}^m (h_i(\boldsymbol{x}_i))^2 \\ &= \sum_{i=1}^m f_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}^\top \boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{\nu}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{\rho}{2} \|\boldsymbol{y}'(\boldsymbol{x})\|_2^2 + \frac{\rho}{2} \|\boldsymbol{h}(\boldsymbol{x})\|_2^2, \end{aligned} \tag{27}$$

where $\mathbb{R}^m \ni \boldsymbol{y}'(\boldsymbol{x}) := [y_1'(\boldsymbol{x}_1), \dots, y_m'(\boldsymbol{x}_m)]^\top$ and $\mathbb{R}^m \ni \boldsymbol{h}(\boldsymbol{x}) := [h_1(\boldsymbol{x}_1), \dots, h_m(\boldsymbol{x}_m)]^\top$.

ADMM Algorithm for General Optimization Problems and Any Number of Variables

- We found:

$$\mathcal{L}_\rho(\{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{\nu}', \boldsymbol{\nu}) = \sum_{i=1}^{m} f_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}^\top \boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{\nu}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{y}'(\boldsymbol{x})\|_2^2 + \frac{\rho}{2}\|\boldsymbol{h}(\boldsymbol{x})\|_2^2.$$

- Updating the primal and dual variables are performed as [13, 14]:

$$\boldsymbol{x}_i^{(k+1)} := \arg\min_{\boldsymbol{x}_i} \mathcal{L}_\rho(\boldsymbol{x}_i, \lambda_i^{(k)}, \nu_i^{(k)}),\ \forall i \in \{1, \ldots, m\},$$

$$\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} + \rho\, \boldsymbol{y}'(\boldsymbol{x}^{(k+1)}),$$

$$\boldsymbol{\nu}^{(k+1)} := \boldsymbol{\nu}^{(k)} + \rho\, \boldsymbol{h}(\boldsymbol{x}^{(k+1)}).$$

- Note that as the Lagrangian is completely decomposable by the i indices, the optimization for every i-th primal or dual variable does not depend on other indices; in other words, the terms of other indices become constant for every index.

ADMM Algorithm for General Optimization Problems and Any Number of Variables

- The last terms in the augmented Lagrangian, Eq. (27), can be restated as:

$$
\begin{aligned}
&\boldsymbol{\lambda}^\top \boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{\nu}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{y}'(\boldsymbol{x})\|_2^2 + \frac{\rho}{2}\|\boldsymbol{h}(\boldsymbol{x})\|_2^2 \\
&= \boldsymbol{\lambda}^\top \boldsymbol{y}'(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{y}'(\boldsymbol{x})\|_2^2 + \frac{1}{2\rho}\|\boldsymbol{\lambda}\|_2^2 - \frac{1}{2\rho}\|\boldsymbol{\lambda}\|_2^2 \\
&\qquad + \boldsymbol{\nu}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{h}(\boldsymbol{x})\|_2^2 + \frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2 - \frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2 \\
&= \frac{\rho}{2}\Big(\|\boldsymbol{y}'(\boldsymbol{x})\|_2^2 + \frac{1}{\rho^2}\|\boldsymbol{\lambda}\|_2^2 + \frac{2}{\rho}\boldsymbol{\lambda}^\top \boldsymbol{y}'(\boldsymbol{x})\Big) - \frac{1}{2\rho}\|\boldsymbol{\lambda}\|_2^2 \\
&\qquad + \frac{\rho}{2}\Big(\|\boldsymbol{h}(\boldsymbol{x})\|_2^2 + \frac{1}{\rho^2}\|\boldsymbol{\nu}\|_2^2 + \frac{2}{\rho}\boldsymbol{\nu}^\top \boldsymbol{h}(\boldsymbol{x})\Big) - \frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2 \\
&= \frac{\rho}{2}\|\boldsymbol{y}'(\boldsymbol{x}) + \frac{1}{\rho}\boldsymbol{\lambda}\|_2^2 - \frac{1}{2\rho}\|\boldsymbol{\lambda}\|_2^2 + \frac{\rho}{2}\|\boldsymbol{h}(\boldsymbol{x}) + \frac{1}{\rho}\boldsymbol{\nu}\|_2^2 - \frac{1}{2\rho}\|\boldsymbol{\nu}\|_2^2 \\
&\overset{(a)}{=} \frac{\rho}{2}\|\boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{u}_\lambda\|_2^2 + \frac{\rho}{2}\|\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{u}_\nu\|_2^2 - \text{constant},
\end{aligned}
$$

where (a) is because we define $\boldsymbol{u}_\lambda := (1/\rho)\boldsymbol{\lambda}$ and $\boldsymbol{u}_\nu := (1/\rho)\boldsymbol{\nu}$.

ADMM Algorithm for General Optimization Problems and Any Number of Variables

- The augmented Lagrangian was:

$$\mathcal{L}_\rho(\{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{\nu}', \boldsymbol{\nu}) = \sum_{i=1}^m f_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}^\top \boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{\nu}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{y}'(\boldsymbol{x})\|_2^2 + \frac{\rho}{2}\|\boldsymbol{h}(\boldsymbol{x})\|_2^2.$$

- We simplified the last terms in the augmented Lagrangian:

$$\begin{aligned}\boldsymbol{\lambda}^\top \boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{\nu}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{y}'(\boldsymbol{x})\|_2^2 + \frac{\rho}{2}\|\boldsymbol{h}(\boldsymbol{x})\|_2^2 \\ = \frac{\rho}{2}\left\|\boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{u}_\lambda\right\|_2^2 + \frac{\rho}{2}\left\|\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{u}_\nu\right\|_2^2 - \text{constant}.\end{aligned}$$

- Hence, the Lagrangian can be restated as:

$$\begin{aligned}\mathcal{L}_\rho(\{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{u}_\lambda, \boldsymbol{u}_\nu) &= \sum_{i=1}^m f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\left\|\boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{u}_\lambda\right\|_2^2 + \frac{\rho}{2}\left\|\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{u}_\nu\right\|_2^2 + \text{constant} \\ &= \sum_{i=1}^m f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\sum_{i=1}^m \left[(y_i'(\boldsymbol{x}_i) + u_{\lambda,i})^2 + (h_i(\boldsymbol{x}_i) + u_{\nu,i})^2\right] + \text{constant},\end{aligned}$$

where $u_{\lambda,i} = (1/\rho)\lambda_i$ and $u_{\nu,i} = (1/\rho)\nu_i$ are the i-th elements of $\boldsymbol{u}_\lambda$ and $\boldsymbol{u}_\nu$, respectively.

ADMM Algorithm for General Optimization Problems and Any Number of Variables

- The Lagrangian was restated as:

$$\mathcal{L}_\rho(\{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{u}_\lambda, \boldsymbol{u}_\nu) = \sum_{i=1}^m f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\left\|\boldsymbol{y}'(\boldsymbol{x}) + \boldsymbol{u}_\lambda\right\|_2^2 + \frac{\rho}{2}\left\|\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{u}_\nu\right\|_2^2 + \text{constant}$$

$$= \sum_{i=1}^m f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\sum_{i=1}^m \left[(y_i'(\boldsymbol{x}_i) + u_{\lambda,i})^2 + (h_i(\boldsymbol{x}_i) + u_{\nu,i})^2\right] + \text{constant}.$$

- Hence, updating variables can be restated as:

$$\boldsymbol{x}_i^{(k+1)} := \arg\min_{\boldsymbol{x}_i}\left(f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\left[(y_i'(\boldsymbol{x}_i) + u_{\lambda,i}^{(k)})^2 + (h_i(\boldsymbol{x}_i) + u_{\nu,i}^{(k)})^2\right]\right), \forall i \in \{1, \ldots, m\}, \tag{28}$$

$$u_{\lambda,i}^{(k+1)} := u_{\lambda,i}^{(k)} + \rho\, y_i'(\boldsymbol{x}_i^{(k+1)}), \ \forall i \in \{1, \ldots, m\} \tag{29}$$

$$u_{\nu,i}^{(k+1)} := u_{\nu,i}^{(k)} + \rho\, h_i(\boldsymbol{x}_i^{(k+1)}), \ \forall i \in \{1, \ldots, m\}. \tag{30}$$

Use of ADMM for Distributed Optimization

Use of ADMM for Distributed Optimization

- **ADMM** is one of the most well-known algorithms for **distributed optimization**.
- If the problem can be divided into several disjoint blocks (i.e., several primal variables), we can solve the optimization for each primal variable on a separate core or server (see Eq. (28) for every i):

$$\boldsymbol{x}_i^{(k+1)} := \arg\min_{\boldsymbol{x}_i} \left(f_i(\boldsymbol{x}_i) + \frac{\rho}{2} \left[(y_i'(\boldsymbol{x}_i) + u_{\lambda,i}^{(k)})^2 + (h_i(\boldsymbol{x}_i) + u_{\nu,i}^{(k)})^2 \right] \right), \forall i \in \{1, \ldots, m\}.$$

  Hence, in every iteration of ADMM, the update of primal variables can be performed in parallel by distributed servers.
- At the end of each iteration, the updated primal variables are gathered in a central server so that the update of dual variable(s) is performed (see Eqs. (29) and (30)):

$$\begin{cases} u_{\lambda,i}^{(k+1)} := u_{\lambda,i}^{(k)} + \rho\, y_i'(\boldsymbol{x}_i^{(k+1)}), \ \forall i \in \{1, \ldots, m\}, \\ u_{\nu,i}^{(k+1)} := u_{\nu,i}^{(k)} + \rho\, h_i(\boldsymbol{x}_i^{(k+1)}), \ \forall i \in \{1, \ldots, m\}. \end{cases}$$

- Then, the updated dual variable(s) is sent to the distributed servers so they update their primal variables. This procedure is repeated until convergence of primal and dual variables.
- In this sense, ADMM is performed similar to the approach of **federated learning** [15, 16].

Making Optimization Problem Distributed

Making Optimization Problem Distributed

- We can convert a non-distributed optimization problem to a distributed optimization problem to solve it using ADMM. Many recent machine learning and signal processing papers are using this technique.
- **Univariate optimization problem:** Consider a regular non-distributed problem with one optimization variable $\boldsymbol{x}$:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{m} f_i(\boldsymbol{x}) \\ \text{subject to} \quad & y_i(\boldsymbol{x}) \leq 0, \ i \in \{1, \ldots, m\}, \\ & h_i(\boldsymbol{x}) = 0, \ i \in \{1, \ldots, m\}. \end{aligned} \tag{31}$$

- This problem can be stated as:

$$\begin{aligned} \underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} \quad & \sum_{i=1}^{m} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & y_i(\boldsymbol{x}_i) \leq 0, \ i \in \{1, \ldots, m\}, \\ & h_i(\boldsymbol{x}_i) = 0, \ i \in \{1, \ldots, m\}, \\ & \boldsymbol{x}_i = \boldsymbol{z}, \ i \in \{1, \ldots, m\}, \end{aligned} \tag{32}$$

where we introduce m variables $\{\boldsymbol{x}_i\}_{i=1}^m$ and use the trick $\boldsymbol{x}_i = \boldsymbol{z}, \forall i$ to make them equal to one variable.

Making Optimization Problem Distributed

- Eq. (32) is similar to Eq. (26) except that it has $2m$ equality constraints rather than m equality constraints.
- Hence, we can use ADMM updates similarly to Eqs. (28), (29), and (30) but with slight change because of the additional m constraints.
- We introduce m new dual variables for constraints $\boldsymbol{x}_i = \boldsymbol{z}, \forall i$ and update those dual variables as well as other variables. The augmented Lagrangian also has some additional terms for the new constraints.
- The Lagrangian and ADMM updates of this are not stated here because of its similarity to the previous equations.
- This is a good technique to make a problem distributed, use ADMM for solving it, and solving it in parallel servers.

Making Optimization Problem Distributed

- **Multivariate optimization problem:** Consider a regular non-distributed problem with multiple optimization variables $\{\boldsymbol{x}_i\}_{i=1}^m$:

$$\begin{aligned} \underset{\{\boldsymbol{x}\}_{i=1}^m}{\text{minimize}} \quad & \sum_{i=1}^m f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \boldsymbol{x}_i \in \mathcal{S}_i, \ i \in \{1, \ldots, m\}, \end{aligned} \tag{33}$$

where $\boldsymbol{x}_i \in \mathcal{S}_i$ can be any constraint such as belonging to a set $\mathcal{S}_i$, an equality constraint, or an inequality constraint.

- We can embed the constraint in the objective function using an indicator function:

$$\underset{\{\boldsymbol{x}\}_{i=1}^m}{\text{minimize}} \quad \sum_{i=1}^m \big(f_i(\boldsymbol{x}_i) + \phi_i(\boldsymbol{x}_i)\big),$$

where $\phi_i(\boldsymbol{x}_i) := \mathbb{I}(\boldsymbol{x}_i \in \mathcal{S}_i)$ is zero if $\boldsymbol{x}_i \in \mathcal{S}_i$ and is infinity otherwise.

- This problem can be stated as:

$$\begin{aligned} \underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} \quad & \sum_{i=1}^m \big(f_i(\boldsymbol{x}_i) + \phi_i(\boldsymbol{z}_i)\big) \\ \text{subject to} \quad & \boldsymbol{x}_i = \boldsymbol{z}_i, \ i \in \{1, \ldots, m\}, \end{aligned} \tag{34}$$

where we introduce a variable $\boldsymbol{z}_i$ for every $\boldsymbol{x}_i$, use the introduced variable for the second term in the objective function, and we equate them in the constraint.

Making Optimization Problem Distributed

- We found:

$$\begin{aligned} \underset{\{\boldsymbol{x}_i\}_{i=1}^m}{\text{minimize}} \quad & \sum_{i=1}^{m} \big(f_i(\boldsymbol{x}_i) + \phi_i(\boldsymbol{z}_i)\big) \\ \text{subject to} \quad & \boldsymbol{x}_i = \boldsymbol{z}_i, i \in \{1, \ldots, m\}. \end{aligned}$$

- As the constraints $\boldsymbol{x}_i - \boldsymbol{z}_i = \boldsymbol{0}, \forall i$ are equality constraints, we can use Eqs. (23), (24), and (25) as ADMM updates for this problem:

$$\boldsymbol{x}_i^{(k+1)} := \arg\min_{\boldsymbol{x}_i} \left(f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\|\boldsymbol{x}_i - \boldsymbol{z}_i^{(k)} + \boldsymbol{u}_i^{(k)}\|_2^2\right), \quad \forall i \in \{1, \ldots, m\}, \tag{35}$$

$$\boldsymbol{z}_i^{(k+1)} := \arg\min_{\boldsymbol{z}_i} \left(\phi_i(\boldsymbol{z}_i) + \frac{\rho}{2}\|\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}_i + \boldsymbol{u}_i^{(k)}\|_2^2\right), \quad \forall i \in \{1, \ldots, m\}, \tag{36}$$

$$\boldsymbol{u}_i^{(k+1)} := \boldsymbol{u}_i^{(k)} + \rho(\boldsymbol{x}_i^{(k+1)} + \boldsymbol{z}_i^{(k+1)}), \forall i \in \{1, \ldots, m\}.$$

- Comparing Eqs. (35) and (36) with the proximal operator, $\mathbf{prox}_{\lambda g}(\boldsymbol{x}) := \arg\min_{\boldsymbol{u}} \left(g(\boldsymbol{u}) + \frac{1}{2\lambda}\|\boldsymbol{u} - \boldsymbol{x}\|_2^2\right)$ shows that these ADMM updates can be written as proximal mappings:

$$\begin{aligned} \boldsymbol{x}_i^{(k+1)} &:= \mathbf{prox}_{\frac{1}{\rho} f_i}(\boldsymbol{z}_i^{(k)} - \boldsymbol{u}_i^{(k)}), \forall i \in \{1, \ldots, m\}, \\ \boldsymbol{z}_i^{(k+1)} &:= \mathbf{prox}_{\frac{1}{\rho}\phi_i}(\boldsymbol{x}_i^{(k+1)} + \boldsymbol{u}_i^{(k)}), \forall i \in \{1, \ldots, m\}, \\ \boldsymbol{u}_i^{(k+1)} &:= \boldsymbol{u}_i^{(k)} + \rho(\boldsymbol{x}_i^{(k+1)} + \boldsymbol{z}_i^{(k+1)}), \forall i \in \{1, \ldots, m\}, \end{aligned} \tag{37}$$

if we notice that $\|\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}_i + \boldsymbol{u}_i^{(k)}\|_2^2 = \|\boldsymbol{z}_i - \boldsymbol{x}_i^{(k+1)} - \boldsymbol{u}_i^{(k)}\|_2^2$.

Making Optimization Problem Distributed

- Note that in many papers, such as [17], we only have $m = 1$. In that case, we only have two primal variables $\boldsymbol{x}$ and $\boldsymbol{z}$.
- According to the lemma we had (that the proximal operator of indicator function is projection), as the function $\phi_i(.)$ is an indicator function, Eq. (37):

$$\boldsymbol{z}_i^{(k+1)} := \mathbf{prox}_{\frac{1}{\rho}\phi_i}(\boldsymbol{x}_i^{(k+1)} + \boldsymbol{u}_i^{(k)}), \forall i \in \{1, \ldots, m\},$$

  can be implemented by projection onto the set $\mathcal{S}_i$:

$$\boldsymbol{z}_i^{(k+1)} := \Pi_{\mathcal{S}_i}(\boldsymbol{x}_i^{(k+1)} + \boldsymbol{u}_i^{(k)}), \forall i \in \{1, \ldots, m\}.$$

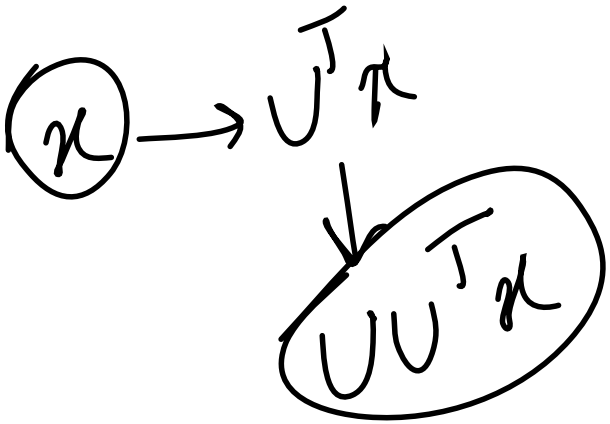
- As an example, assume the variables are all matrices so we have $\boldsymbol{X}_i$, $\boldsymbol{Z}_i$, and $\boldsymbol{U}_i$. If the set $\mathcal{S}_i$ is the cone of orthogonal matrices, the constraint $\boldsymbol{X}_i \in \mathcal{S}_i$ would be $\boldsymbol{X}_i^\top \boldsymbol{X}_i = \boldsymbol{I}$. In this case, the update of matrix variable $\boldsymbol{Z}_i$ would be done by setting the singular values of $(\boldsymbol{x}_i^{(k+1)} + \boldsymbol{u}_i^{(k)})$ to one (recall projection onto the cone of the orthogonal matrices).

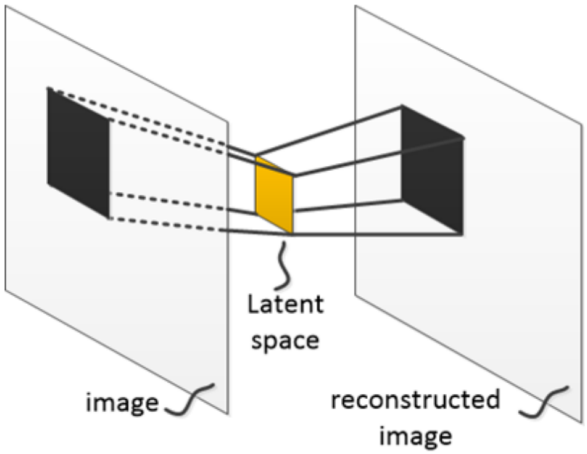
$$\min \quad f(X) = \operatorname{tr}(X^T A X) \quad \Big\} \longrightarrow \text{PCA}, \cdots$$
$$\text{s.t.} \quad X^T X = I$$

Example of ADMM

Example of ADMM

- This example is in my own paper (Image Structural Component Analysis) [18].
- Let image be divided into b blocks. Our goal is to find a p-dimensional subspace for reconstruction of every image block (each block as q pixels and we have $p \ll q$).





- Considering all the b blocks in an image, the problem is:

$$\star \begin{cases} \underset{\boldsymbol{U}_i \in \mathbb{R}^{q \times p}}{\text{minimize}} & \sum_{i=1}^{b} ||\check{\boldsymbol{x}}_i - \boldsymbol{U}_i \boldsymbol{U}_i^\top \check{\boldsymbol{x}}_i||_2^2, \\ \text{subject to} & \boldsymbol{U}_i^\top \boldsymbol{U}_i = \boldsymbol{I}, \quad \forall i \in \{1, \ldots, b\}, \end{cases} \tag{38}$$

where $\check{\boldsymbol{x}}_i \in \mathbb{R}^q$ and $\boldsymbol{U}_i \in \mathbb{R}^{q \times p}$ are the i-th block and the bases of its subspace, respectively.

Example of ADMM

- We had:

$$\underset{\boldsymbol{U}_i \in \mathbb{R}^{q\times p}}{\text{minimize}} \quad \sum_{i=1}^{b} ||\check{\boldsymbol{x}}_i - \boldsymbol{U}_i \boldsymbol{U}_i^\top \check{\boldsymbol{x}}_i||_S,$$
$$\text{subject to} \quad \boldsymbol{U}_i^\top \boldsymbol{U}_i = \boldsymbol{I}, \quad \forall i \in \{1, \ldots, b\},$$

- We convert it to:

$$\underset{\boldsymbol{U}_i, \boldsymbol{V}_i \in \mathbb{R}^{q\times p}}{\text{minimize}} \quad \sum_{i=1}^{b} \big(f(\boldsymbol{U}_i) + h(\boldsymbol{V}_i)\big),$$
$$\text{subject to} \quad \boldsymbol{U} - \boldsymbol{V} = \boldsymbol{0}, \tag{39}$$

where $f(\boldsymbol{U}_i) := ||\check{\boldsymbol{x}}_i - \boldsymbol{U}_i \boldsymbol{U}_i^\top \check{\boldsymbol{x}}_i||_S$ and $h(\boldsymbol{V}_i) := \mathbb{I}(\boldsymbol{V}_i^\top \boldsymbol{V}_i = \boldsymbol{I})$.

- The (squared) SSIM distance, which we denote by $||.||_S$, is [19]:

$$\mathbb{R} \ni ||\check{\boldsymbol{x}}_1 - \check{\boldsymbol{x}}_2||_S := 1 - \text{SSIM}(\check{\boldsymbol{x}}_1, \check{\boldsymbol{x}}_2) = \frac{||\check{\boldsymbol{x}}_1 - \check{\boldsymbol{x}}_2||_2^2}{||\check{\boldsymbol{x}}_1||_2^2 + ||\check{\boldsymbol{x}}_2||_2^2 + c}, \tag{40}$$

- The $\mathbb{I}(.)$ denotes the indicator function which is zero if its condition is satisfied and is infinite otherwise.
- The $\boldsymbol{U}$ and $\boldsymbol{V}$ are defined as union of partitions to form an image-form array, i.e., $\boldsymbol{U} := \cup_{i=1}^{b} \boldsymbol{U}_i$ and $\boldsymbol{V} := \cup_{i=1}^{b} \boldsymbol{V}_i$ [17].

Example of ADMM

- Eq. (39) was:

$$\begin{aligned} \underset{\boldsymbol{U}_i, \boldsymbol{V}_i \in \mathbb{R}^{q \times p}}{\text{minimize}} \quad & \sum_{i=1}^{b} \big(f(\boldsymbol{U}_i) + h(\boldsymbol{V}_i)\big), \\ \text{subject to} \quad & \boldsymbol{U} - \boldsymbol{V} = \boldsymbol{0}. \end{aligned}$$

- The augmented Lagrangian for Eq. (39) is:

$$\begin{aligned} \mathcal{L}_\rho &= \sum_{i=1}^{b} \big(f(\boldsymbol{U}_i) + h(\boldsymbol{V}_i)\big) + \mathbf{tr}\big(\boldsymbol{\Lambda}^\top (\boldsymbol{U} - \boldsymbol{V})\big) + (\rho/2)\, ||\boldsymbol{U} - \boldsymbol{V}||_F^2 \\ &= \sum_{i=1}^{b} \big(f(\boldsymbol{U}_i) + h(\boldsymbol{V}_i)\big) + (\rho/2)\, ||\boldsymbol{U} - \boldsymbol{V} + \boldsymbol{J}||_F^2 - (\rho/2)\, ||\boldsymbol{\Lambda}||_F^2, \end{aligned}$$

where $\|.\|_F$ is the Frobenius norm, $\boldsymbol{\Lambda} := \cup_{i=1}^{b} \boldsymbol{\Lambda}_i$ is the Lagrange multiplier, $\rho > 0$ is a parameter, and $\boldsymbol{J} := (1/\rho)\boldsymbol{\Lambda} = (1/\rho) \cup_{i=1}^{b} \boldsymbol{\Lambda}_i = \cup_{i=1}^{b} \boldsymbol{J}_i$.

- Note that the term $(\rho/2)\, ||\boldsymbol{\Lambda}||_F^2$ is a constant with respect to $\boldsymbol{U}$ and $\boldsymbol{V}$ and can be dropped. The updates of $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{J}$ are done as [11, 17]:

$$\boldsymbol{U}_i^{(k+1)} := \arg\min_{\boldsymbol{U}_i} \Big(f(\boldsymbol{U}_i) + (\rho/2)\, ||\boldsymbol{U}_i - \boldsymbol{V}_i^{(k)} + \boldsymbol{J}_i^{(k)}||_F^2\Big), \tag{41}$$

$$\boldsymbol{V}_i^{(k+1)} := \arg\min_{\boldsymbol{V}_i} \Big(h(\boldsymbol{V}_i) + (\rho/2)\, ||\boldsymbol{U}_i^{(k+1)} - \boldsymbol{V}_i + \boldsymbol{J}_i^{(k)}||_F^2\Big), \tag{42}$$

$$\boldsymbol{J}^{(k+1)} := \boldsymbol{J}^{(k)} + \boldsymbol{U}^{(k+1)} - \boldsymbol{V}^{(k+1)}. \tag{43}$$

Example of ADMM

Fig. 1. Examples from the training dataset: (a) original image, (b) contrast stretched, (c) Gaussian noise, (d) luminance enhanced, (e) Gaussian blurring, (f) salt & pepper impulse noise, and (g) JPEG distortion.

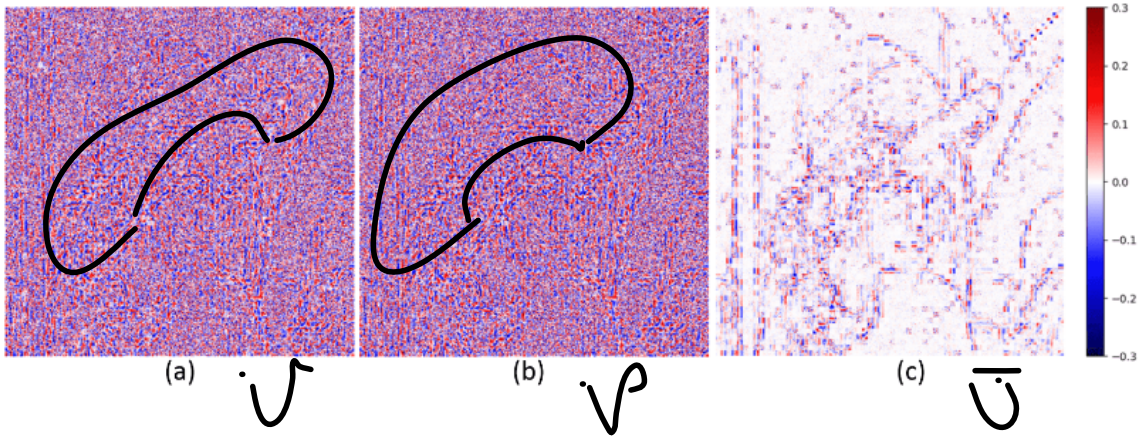


Fig. 2. The first dimension of the trained (a) $\boldsymbol{U}$, (b) $\boldsymbol{V}$, and (c) $\boldsymbol{J}$ for ISCA.

Acknowledgement

- Some slides of this slide deck are inspired by the lectures of Prof. Stephen Boyd at the Stanford University.
- Our tutorial also has the materials of this slide deck: [20]

References

[1] Q. Li, Z. Zhu, and G. Tang, "Alternating minimizations converge to second-order optimal solutions," in *International Conference on Machine Learning*, pp. 3935–3943, 2019.

[2] P. Jain and P. Kar, "Non-convex optimization for machine learning," *arXiv preprint arXiv:1712.07897*, 2017.

[3] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations research*, vol. 8, no. 1, pp. 101–111, 1960.

[4] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[5] H. Everett III, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations research*, vol. 11, no. 3, pp. 399–417, 1963.

[6] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

[7] M. J. Powell, "A method for nonlinear constraints in minimization problems," *Optimization*, pp. 283–298, 1969.

[8] D. P. Bertsekas, "The method of multipliers for equality constrained problems," *Constrained optimization and Lagrange multiplier methods*, pp. 96–157, 1982.

References (cont.)

[9] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & mathematics with applications*, vol. 2, no. 1, pp. 17–40, 1976.

[10] R. Glowinski and A. Marrocco, "Finite element approximation and iterative methods of solution for 2-D nonlinear magnetostatic problems," in *Proceeding of International Conference on the Computation of Electromagnetic Fields (COMPUMAG)*, 1976.

[11] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*.
Now Publishers Inc, 2011.

[12] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[13] J. Giesen and S. Laue, "Distributed convex optimization with many convex constraints," *arXiv preprint arXiv:1610.02967*, 2016.

[14] J. Giesen and S. Laue, "Combining ADMM and the augmented Lagrangian method for efficiently handling many constraints," in *International Joint Conference on Artificial Intelligence*, pp. 4525–4531, 2019.

[15] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

References (cont.)

[16] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[17] D. Otero, D. La Torre, O. V. Michailovich, and E. R. Vrscay, "Alternate direction method of multipliers for unconstrained structural similarity-based optimization," in *International Conference Image Analysis and Recognition*, pp. 20–29, Springer, 2018.

[18] B. Ghojogh, F. Karray, and M. Crowley, "Principal component analysis using structural similarity index for images," in *Image Analysis and Recognition: 16th International Conference, ICIAR 2019, Waterloo, ON, Canada, August 27–29, 2019, Proceedings, Part I 16*, pp. 77–88, Springer, 2019.

[19] D. Brunet, E. R. Vrscay, and Z. Wang, "On the mathematical properties of the structural similarity index," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1488–1499, 2012.

[20] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "KKT conditions, first-order and second-order optimization, and distributed optimization: Tutorial and survey," *arXiv preprint arXiv:2110.01858*, 2021.