**MAEG4070 Engineering Optimization**

# Lecture 11 Distributed Optimization

Yue Chen
MAE, CUHK
email: yuechen@mae.cuhk.edu.hk
Nov 14, 2022

# *Content of this course (tentative)*

- Optimization — Lecture 1
  - Linear
    - Linear programming — Lecture 2
    - Dual Theory – Part I — Lecture 3
  - Linearization technique — Lecture 7
  - Non-Linear — Lecture 4
    - Unconstrained optimization — Lecture 5 & 6
    - Constrained optimization — Lecture 8 & 9
    - Dual Theory – Part II — Lecture 10
  - Distributed optimization — Lecture 11
  - Multi-objective optimization — Lecture 12
  - Robust optimization — Lecture 13

*Theory*

Programming — Lecture 14

Engineering examples — Lecture 15

# *Motivation*

In engineering, “Big Data” has had a significant impact in areas as varied as artificial intelligence, internet applications, medicine, finance, marketing, network analysis, and logistics.

However…

- The datasets are often extremely large
- The data is often very high dimensional
- The data is often stored or even collected in a distributed manner

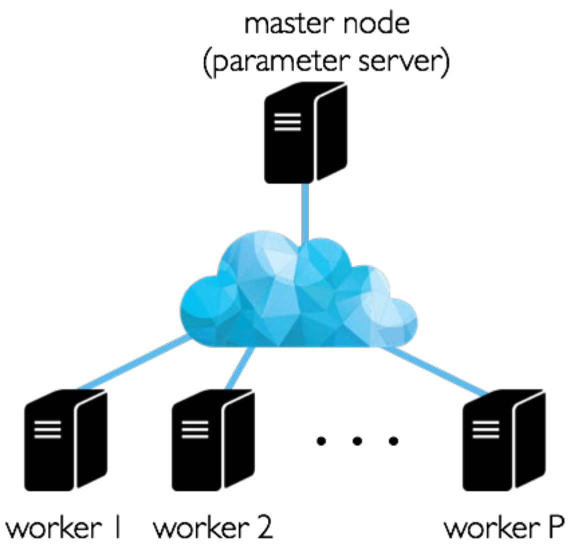
As a result, we want to develop algorithms:

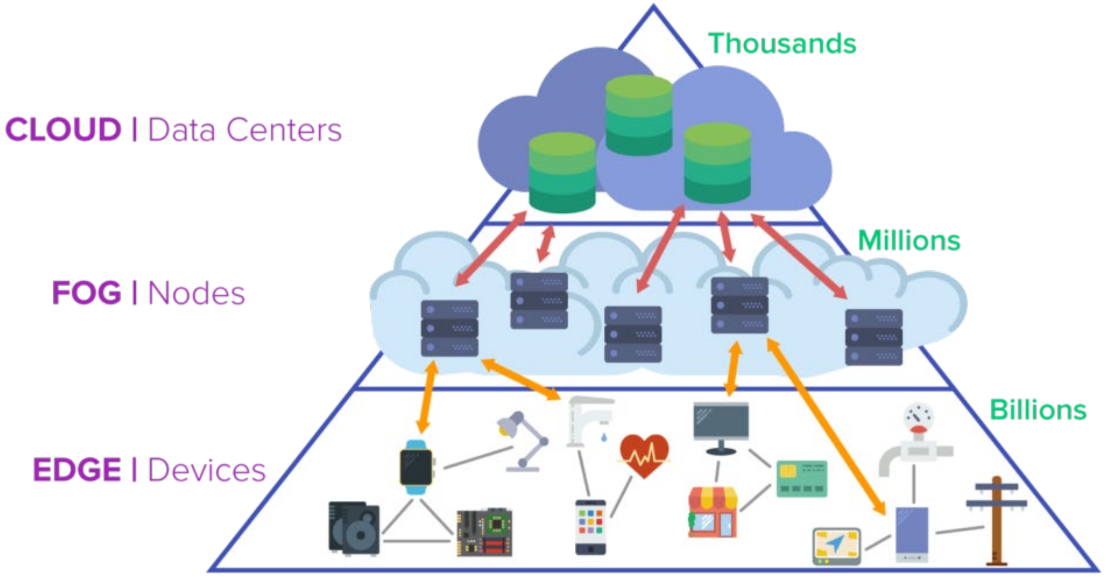
- rich enough to capture the complexity of modern data
- Scalable enough to process huge datasets in a parallelized or fully decentralized fashion.

# History

- *Dual decomposition* (early 1960s), similar ideas appear in well known work by Dantzig and Wolfe and Benders on large-scale linear programming
- *Augmented Lagrangians* and *the method of multipliers* for constrained optimization (late 1960s) by Hestenes and Powell.
- decentralized optimization, an active topic of research since the 1980s.

decomposition-coordination procedure






Pictures from Google image

# Dual problem

Consider the convex equality constrained optimization problem:

$$\min_x \; f(x)$$
$$\text{s.t. } Ax = b$$

The Lagrange function is

$$L(x, \lambda) = f(x) + \lambda^T (Ax - b)$$

The dual function is

$$g(\lambda) = \inf_x L(x, \lambda)$$

The dual problem is

$$\max_\lambda g(\lambda)$$

Finally, we can recover $x^*$ by

$$x^* = \text{argmin}_x L(x, \lambda^*)$$

# *Dual ascent*

The Lagrange function is $L(x,\lambda) = f(x) + \lambda^T(Ax - b)$.
The dual function is $g(\lambda) = \inf_x L(x,\lambda)$.
The dual problem is $\max_\lambda g(\lambda)$. $\Leftrightarrow$ $\boxed{\min_{\lambda} -g(\lambda)}$

We can apply the gradient method to the dual problem:

$$\lambda^{k+1} = \lambda^k + \alpha^k \nabla g(\lambda^k)$$

Note that $\nabla g(\lambda^k) = A\hat{x} - b$, where $\hat{x} = \text{argmin}_x L(x,\lambda^k)$.
Therefore, the *dual ascent* method can be summarized as

$$\begin{aligned} x^{k+1} &= \text{argmin}_x L(x,\lambda^k) && \textbf{x-minimization} \\ \lambda^{k+1} &= \lambda^k + \alpha^k(Ax^{k+1} - b) && \textbf{dual update} \end{aligned}$$

**This algorithm works with lots of strong assumptions**

# *Dual decomposition*

**Dual ascent is still centralized, how to turn it into distributed?**

Suppose $f$ is separable:

$$f(x) = f_1(x_1) + f_2(x_2) + \ldots + f_N(x_N), x = (x_1, x_2, \ldots, x_N)$$

Then the Lagrange function is also separable

$$\begin{aligned} L(x,\lambda) = \; & f(x) + \lambda(Ax - b) \\ = \; & f_1(x_1) + f_2(x_2) + \ldots + f_N(x_N) + \lambda^T(A_1x_1 + A_2x_2 + \ldots + A_Nx_N - b) \\ = \; & \underbrace{f_1(x_1) + \lambda^T(A_1x_1)}_{L_1(x_1,\lambda)} + \underbrace{f_2(x_2) + \lambda^T(A_2x_2)}_{L_2(x_2,\lambda)} + \ldots + \underbrace{f_N(x_N) + \lambda^T(A_Nx_N)}_{L_N(x_N,\lambda)} - \lambda^T b \end{aligned}$$

$$\boxed{\begin{aligned} x^{k+1} &= \text{argmin}_x L(x, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \alpha^k(Ax^{k+1} - b) \end{aligned}}$$

?

# Dual decomposition

First, for $x^{k+1} = \text{argmin}_x L(x, \lambda^k)$:

$$x_n^{k+1} = \text{argmin}_{x_n} L_n(x_n, \lambda^k) = \text{argmin}_{x_n} \left[ f_n(x_n) + (\lambda^k)^T A_n x_n \right], \forall n = 1, ..., N$$

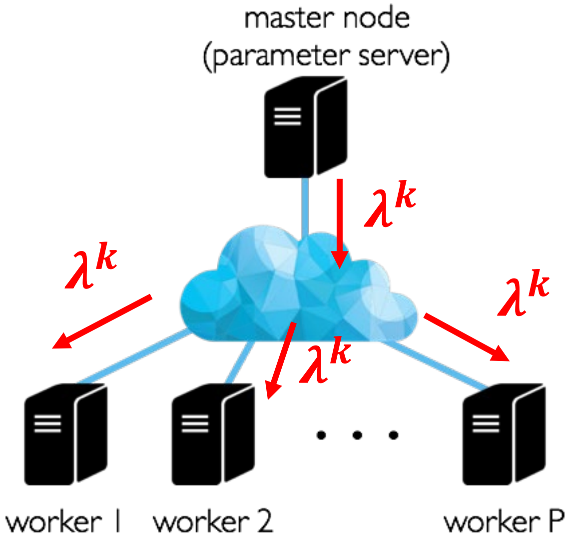
Then, for $\lambda^{k+1} = \lambda^k + \alpha^k (A x^{k+1} - b)$:

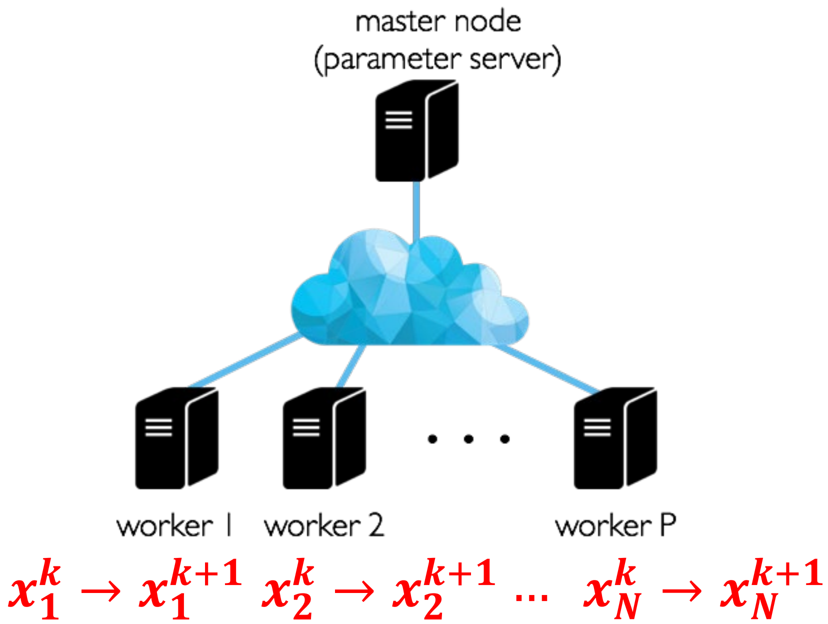
$$\lambda^{k+1} = \lambda^k + \alpha^k (\sum_{n=1}^{N} A_n x_n^{k+1} - b)$$

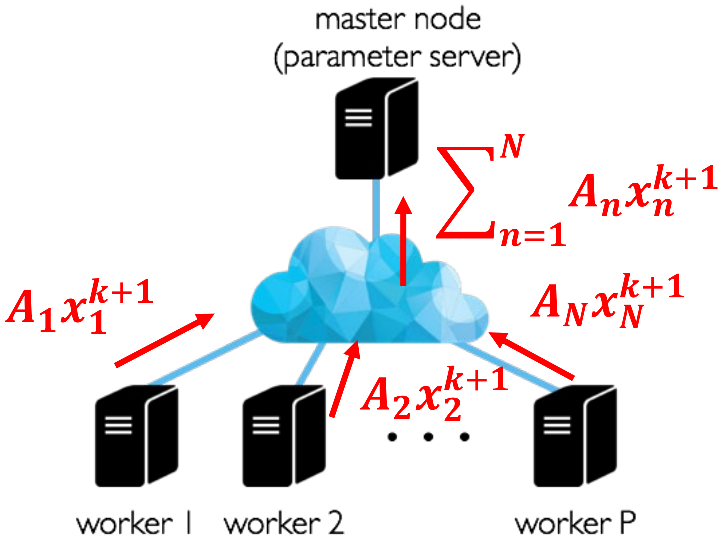
Thus, we can derive a distributed version of *dual ascent*:

$$\begin{aligned} x_n^{k+1} &= \ \text{argmin}_{x_n} L_n(x_n, \lambda^k), \forall n = 1, ..., N \\ \lambda^{k+1} &= \ \lambda^k + \alpha^k (\sum_{n=1}^{N} A_n x_n^{k+1} - b) \end{aligned}$$

# *Dual decomposition*




Solve a large problem

- By iteratively solving subproblems (in parallel)
- Dual variable update provides coordination

Works, with lots of assumptions; often slow



Pictures from Google image

# *Example-1*

For example, to solve problem:

$$\min_{x_1, x_2} \; x_1^2 + x_2^2$$
$$\text{s.t } x_1 + x_2 = 2$$

The Lagrange function is

$$\begin{aligned} L(x, \lambda) = & \; x_1^2 + x_2^2 + \lambda(x_1 + x_2 - 2) \\ = & \; \underbrace{x_1^2 + \lambda x_1}_{L_1(x_1, \lambda)} + \underbrace{x_2^2 + \lambda x_2}_{L_2(x_2, \lambda)} - 2\lambda \end{aligned}$$

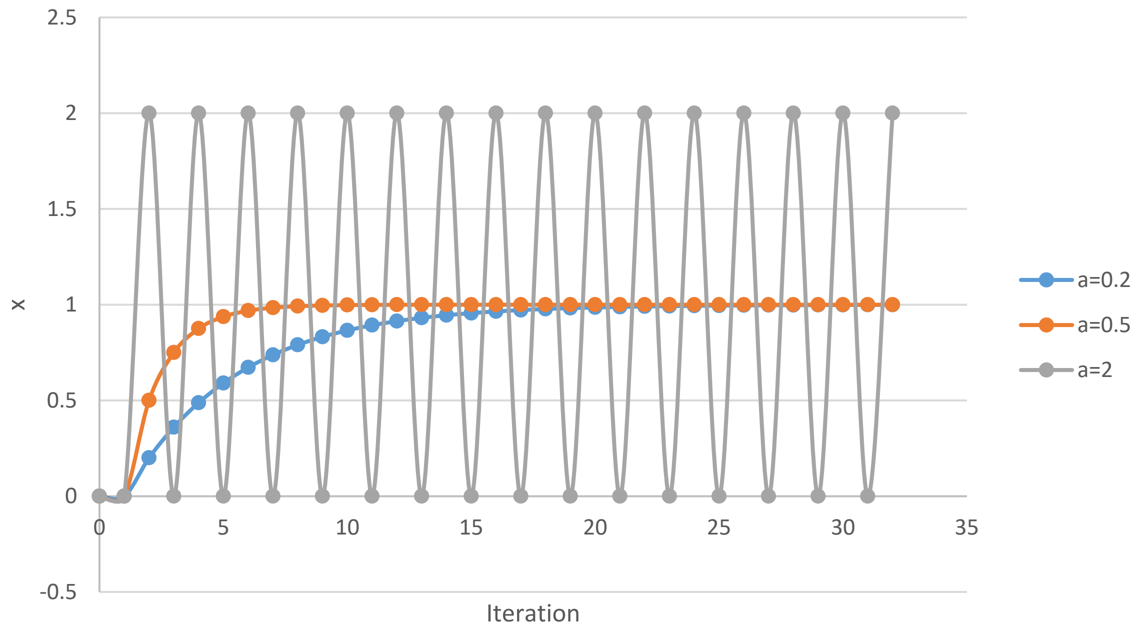
The updates are

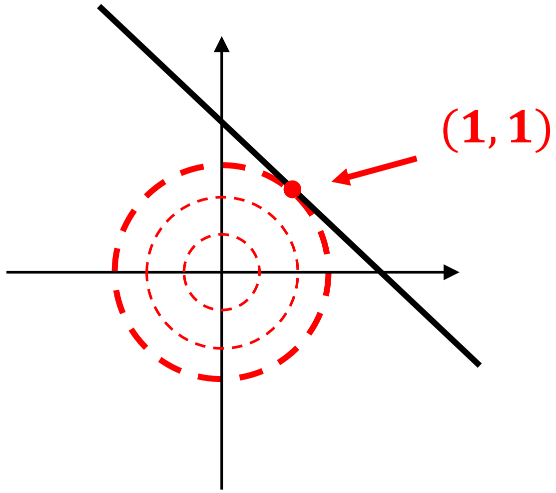
$$\begin{aligned} x_1^{k+1} = & \; \text{argmin}_{x_1} \left( x_1^2 + \lambda^k x_1 \right) = -\frac{\lambda^k}{2} \\ x_2^{k+1} = & \; \text{argmin}_{x_2} \left( x_2^2 + \lambda^k x_2 \right) = -\frac{\lambda^k}{2} \\ \lambda^{k+1} = & \; \lambda^k + \alpha^k (x_1^{k+1} + x_2^{k+1} - 2) \end{aligned}$$

# Example-1

The updates are

$$
\begin{aligned}
x_1^{k+1} &= \operatorname{argmin}_{x_1}\left(x_1^2 + \lambda^k x_1\right) = -\frac{\lambda^k}{2} \\
x_2^{k+1} &= \operatorname{argmin}_{x_2}\left(x_2^2 + \lambda^k x_2\right) = -\frac{\lambda^k}{2} \\
\lambda^{k+1} &= \lambda^k + \alpha^k (x_1^{k+1} + x_2^{k+1} - 2)
\end{aligned}
$$




A small $a$, converges slow
A large $a$, might not converge

# Example-2

To solve the optimization problem:

$$\min_{x_1,x_2} \; (x_1-2)^2 + (x_2-1)^2$$
$$\text{s.t. } x_1 + x_2 = 2$$

The Lagrangian function is

$$\begin{aligned} L(x,\lambda) &= (x_1-2)^2 + (x_2-1)^2 + \lambda(x_1+x_2-2) \\ &= \underbrace{(x_1-2)^2 + \lambda x_1}_{L_1(x_1,\lambda)} + \underbrace{(x_2-1)^2 + \lambda x_2}_{L_2(x_2,\lambda)} - 2\lambda \end{aligned}$$

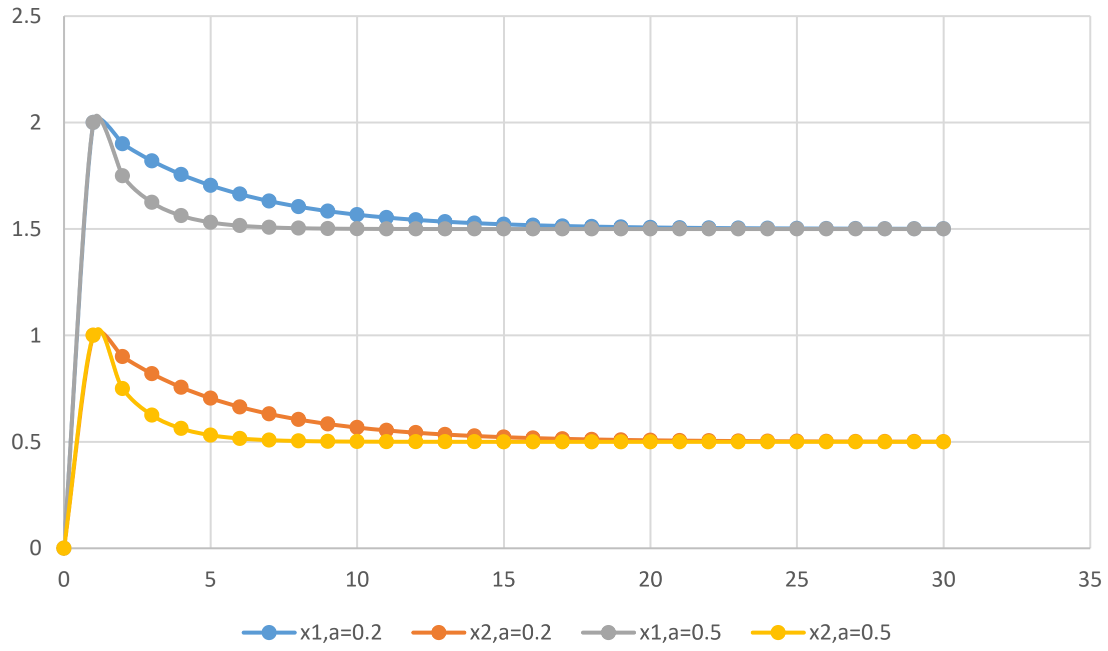
The updates are

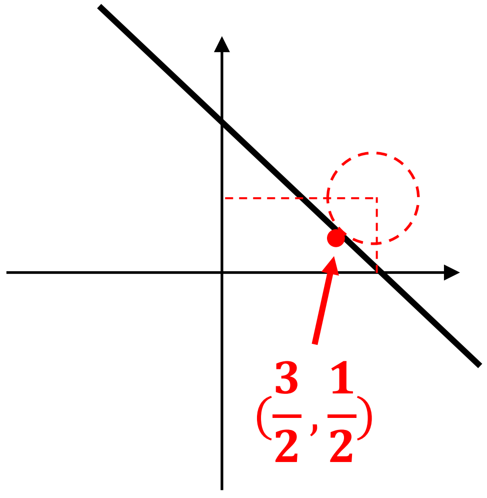
$$\begin{aligned} x_1^{k+1} &= \operatorname{argmin}_{x_1}(x_1-2)^2 + \lambda^k x_1 = 2 - \frac{\lambda^k}{2} \\ x_2^{k+1} &= \operatorname{argmin}_{x_2}(x_2-1)^2 + \lambda^k x_2 = 1 - \frac{\lambda^k}{2} \\ \lambda^{k+1} &= \lambda^k + \alpha^k(x_1^{k+1} + x_2^{k+1} - 2) \end{aligned}$$

# *Example-2*

The updates are

$$
\begin{aligned}
x_1^{k+1} &= \operatorname{argmin}_{x_1}(x_1-2)^2+\lambda^k x_1 = 2-\frac{\lambda^k}{2}\\
x_2^{k+1} &= \operatorname{argmin}_{x_2}(x_2-1)^2+\lambda^k x_2 = 1-\frac{\lambda^k}{2}\\
\lambda^{k+1} &= \lambda^k+\alpha^k(x_1^{k+1}+x_2^{k+1}-2)
\end{aligned}
$$




A small $a$, converges slow

A large $a$, might not converge

# *Method of multipliers**

Augmented Lagrangian methods

- Bring robustness to the dual ascent method
- Yield convergence without assumptions like strict convexity or finiteness of *f*.

The augmented Lagrangian (Hestenes, Powell 1969) is

$$L_\rho(x, \lambda) = f(x) + \lambda^T(Ax - b) + \boxed{(\rho/2)||Ax - b||_2^2}$$

Similarly, the updates (method of multipliers) are

$$\begin{aligned} x^{k+1} &= \operatorname{argmin}_x L_\rho(x, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} - b) \end{aligned}$$

- Good news: converges under much more relaxed conditions
- Bad news: quadratic penalty destroys splitting of the *x*-update, so can't be decomposed

# *Alternating Direction Method of Multipliers (ADMM)**

ADMM method (Gabay, Mercier, Glowinski, Marrocco, 1976)

- With good robustness of method of multipliers
- Can support decomposition

ADMM method deals with problem with form of:

Convex, closed, proper

$$\min_{x,y} \ f(x) + g(y)$$
$$\text{s.t. } Ax + By = c$$

The Lagrangian is:

$$L_\rho(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - c) + (\rho/2)||Ax + By - c||_2^2$$

The updates are

$$\begin{aligned} x^{k+1} &= \text{argmin}_x L_\rho(x, y^k, \lambda^k) \\ y^{k+1} &= \text{argmin}_y L_\rho(x^{k+1}, y, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} + By^{k+1} - b) \end{aligned}$$

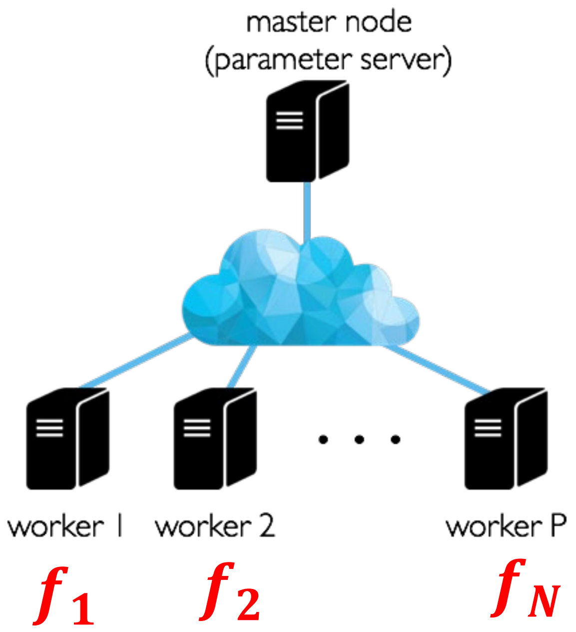
# *Example – Consensus Problem**

The consensus problem is modeled as

$$\min_{x_n, \forall n} \sum_{n=1}^{N} f_n(x_n)$$
$$\text{s.t. } x_n = y, \forall n = 1, ..., N$$

Each block only knows part of the objective function

For example, In model fitting, $x$ represents the parameters in a model and $f_n$ represents the loss function associated with the *n*-th block of data or measurements. In this case, we would say that $x$ is found by *collaborative filtering*, since the data sources are 'collaborating' to develop a global model.






Pictures from Google image

# *Example – Consensus Problem**

The consensus problem is modeled as

$$\min_{x_n, \forall n} \sum_{n=1}^{N} f_n(x_n)$$
$$\text{s.t. } x_n = y, \forall n = 1, ..., N$$

The augmented Lagrangian is

$$L_\rho(x, y, \lambda) = \sum_{n=1}^{N} \underbrace{\left(f_n(x_n) + \lambda_n^T (x_n - y) + (\rho/2)||x_n - y||_2^2\right)}_{L_{\rho,n}(x_n, y, \lambda)}$$

is decomposible. Then the ADMM updates are

$$\begin{aligned}
x_n^{k+1} &= \ \operatorname{argmin}_{x_n} L_{\rho,n}(x_n, y^k, \lambda^k), \forall n = 1, ..., N \\
y^{k+1} &= \ \operatorname{argmin}_y L_\rho(x^{k+1}, y, \lambda^k) \\
\lambda_n^{k+1} &= \ \lambda_n^k + \rho(x_n^{k+1} - y^{k+1}), \forall n = 1, ..., N
\end{aligned}$$

# *Example – Consensus Problem**

Consider this problem:

$$\min_{x_1,x_2} \; (x_1^2 - 2x_1 + 2) + (x_2^2 - 4x_2 + 3)$$
$$\text{s.t. } x_1 = y, x_2 = y$$

This problem is equivalent to

$$\min_{x} \; 2x^2 - 6x + 5$$

So the optimal solution is $x^* = 1.5$.
The augmented Lagrangian is

$$\begin{aligned} L_\rho(x, y, \lambda) = \; & (x_1^2 - 2x_1 + 2) + \lambda_1(x_1 - y) + (\rho/2)(x_1 - y)^2 \\ & + (x_2^2 - 4x_2 + 3) + \lambda_2(x_2 - y) + (\rho/2)(x_2 - y)^2 \end{aligned}$$

# *Example – Consensus Problem**

The augmented Lagrangian is

$$L_\rho(x, y, \lambda) = (x_1^2 - 2x_1 + 2) + \lambda_1(x_1 - y) + (\rho/2)(x_1 - y)^2 \\ + (x_2^2 - 4x_2 + 3) + \lambda_2(x_2 - y) + (\rho/2)(x_2 - y)^2$$

We then calculate $\operatorname{argmin}_{x_1} L_{\rho,1}(x_1, y^k, \lambda^k)$:

$$\frac{\partial L_{\rho,1}}{\partial x_1} = 2x_1 - 2 + \lambda_1^k + \rho(x_1 - y^k) = (2+\rho)x_1 - (2 - \lambda_1^k + \rho y^k)$$

We have

$$x_1^{k+1} = \frac{1}{2+\rho}(2 - \lambda_1^k + \rho y^k)$$

Similarly

$$x_2^{k+1} = \frac{1}{2+\rho}(4 - \lambda_2^k + \rho y^k)$$

# Example – Consensus Problem*

We then calculate $\text{argmin}_y L_\rho(x^{k+1}, y, \rho^{k+1})$:

$$\frac{\partial L_\rho}{\partial y} = -\lambda_1^k - \lambda_2^k - \rho(x_1^{k+1} - y) - \rho(x_2^{k+1} - y) = 0$$

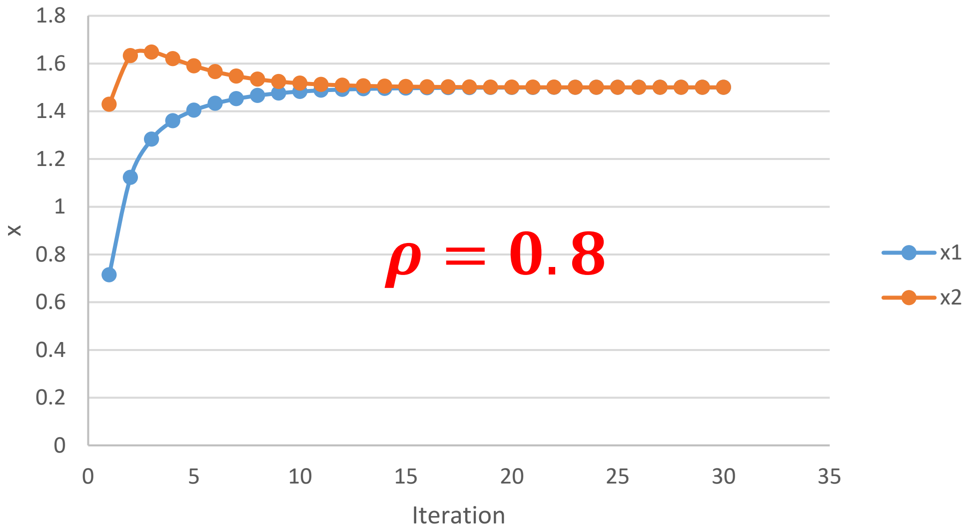
We have

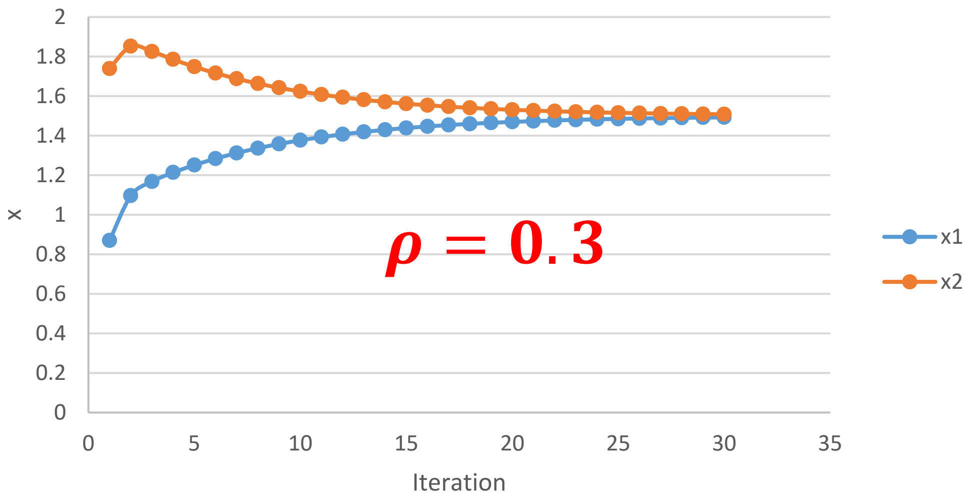
$$y^{k+1} = \frac{\lambda_1^k + \lambda_2^k + \rho x_1^{k+1} + \rho x_2^{k+1}}{2\rho}$$

Therefore, the ADMM updates are

$$\begin{aligned}
x_1^{k+1} &= \frac{1}{2+\rho}(2 - \lambda_1^k + \rho y^k) \\
x_2^{k+1} &= \frac{1}{2+\rho}(4 - \lambda_2^k + \rho y^k) \\
y^{k+1} &= \frac{\lambda_1^k + \lambda_2^k + \rho x_1^{k+1} + \rho x_2^{k+1}}{2\rho} \\
\lambda_1^{k+1} &= \lambda_1^k + \rho(x_1^{k+1} - y^{k+1}) \\
\lambda_2^{k+1} &= \lambda_2^k + \rho(x_2^{k+1} - y^{k+1})
\end{aligned}$$

# *Example – Consensus Problem**




$$
\begin{aligned}
x_1^{k+1} &= \frac{1}{2+\rho}(2 - \lambda_1^k + \rho y^k) \\
x_2^{k+1} &= \frac{1}{2+\rho}(4 - \lambda_2^k + \rho y^k) \\
y^{k+1} &= \frac{\lambda_1^k + \lambda_2^k + \rho x_1^{k+1} + \rho x_2^{k+1}}{2\rho} \\
\lambda_1^{k+1} &= \lambda_1^k + \rho(x_1^{k+1} - y^{k+1}) \\
\lambda_2^{k+1} &= \lambda_2^k + \rho(x_2^{k+1} - y^{k+1})
\end{aligned}
$$

**A smaller $\rho$ takes longer to converge**

# *Example – Optimal Exchange**

The optimal exchange problem is

$$\min_{x_n, \forall n} \sum_{n=1}^{N} f_n(x_n)$$
$$\text{s.t.} \sum_{n=1}^{N} x_n = 0$$

- Components of $x_n$*:* quantities of commodities that are exchanged among *N* agents.
- When $(x_n)_j \geq 0$: the amount of commodity *j received* by subsystem *n* from the exchange.
- When $(x_n)_j \leq 0$: $-(x_n)_j$ is the amount of commodity *j contributed* by subsystem *n* to the exchange.
- The *equilibrium constraint* that each commodity clears
- have a long history in economics, particularly in the theories of market exchange, resource allocation, and general equilibrium (Walras, Arrow and Debreu, and Uzawa)




Pictures from Google image

# Example – Optimal Exchange*

$$\min_{x_n, \forall n} \sum_{n=1}^{N} f_n(x_n)$$
$$\text{s.t.} \sum_{n=1}^{N} x_n = 0$$

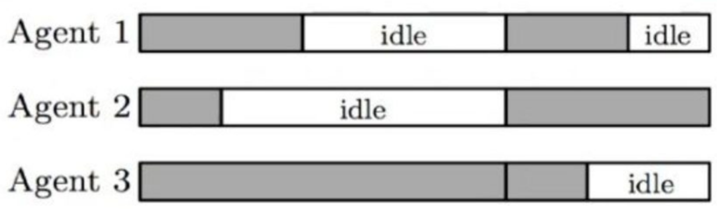
$$\Updownarrow$$

$$\min_{x_n, y_n, \forall n} \sum_{n=1}^{N} f_n(x_n) + g(\sum_{n=1}^{N} y_n)$$
$$\text{s.t. } x_n = y_n, \forall n = 1, ..., N$$

where $g(.)$ is the indicator function:

$$g(\sum_{n=1}^{N} y_n) = \begin{cases} \text{a large constant}, & \forall \sum_{n=1}^{N} y_n = 0 \\ 0, & \forall \sum_{n=1}^{N} y_n \neq 0 \end{cases}$$

# *Syncrhonous v.s. Asynchronous**




**Synchronous**
(wait for the slowest)


**Asynchronous**
(non-stop, no wait)

Synchronous parallel algorithm:
- Easy to implement; easy to analyze
- Unevenly job distribution: more idle time

Asynchronous parallel algorithm:
- Hard to implement; hard to analyze
- Unevenly job distribution: less idle time



Pictures from Google image

Thanks！