Distributed adaptive filters
Part I – Consensus algorithms

Institute of Telecommunications
TU Wien
Motivation

Let us have many

1. independent working units,
Motivation

Let us have many

1. independent working units,

2. which can see only their neighbors,
Motivation

Let us have many

1. independent working units,
2. which can see only their neighbors,
3. and are “simple-minded”
Motivation

Let us have many

1. independent working units,
2. which can see only their neighbors,
3. and are “simple-minded”

Is it possible to find an agreement (consensus) and work together on something ”bigger”?
Yes, it is!

Nature:  

Man-made:
Reaching a consensus $\rightarrow$ distributed algorithms

Each node contains a different value.
Reaching a consensus $\rightarrow$ distributed algorithms

converging...
Reaching a consensus $\rightarrow$ distributed algorithms

converging...
Reaching a consensus → distributed algorithms

converging...
Reaching a consensus $\rightarrow$ distributed algorithms

converging...
Reaching a consensus $\rightarrow$ distributed algorithms

All nodes have the same value.
Distributed Algorithms
Distributed Algorithms

Distributed computing environments, e.g. sensor networks, P2P networks, have many **advantages**:

- large distributed computation power
- fault-tolerance/robustness
Distributed Algorithms

Distributed computing environments, e.g. sensor networks, P2P networks, have many **advantages**:

- large distributed computation power
- fault-tolerance/robustness

but also **disadvantages**:

- limited capabilities of nodes
- limited transmission capacity
- often heterogeneous, dynamic networks
- complicated implementation and analysis of algorithms
From the \textbf{network topology} view:
From the **network topology** view:

- **centralized** – a central node collects data, controls and manages the load balance
  - easier implementation and flow control
  - fully dependent on serving node

(client ←→ server approach)
From the **network topology** view:

- **centralized** – a central node collects data, controls and manages the load balance
  - easier implementation and flow control
  - fully dependent on serving node

  ![client ↔ server approach](image)

- **decentralized** – no central node, all nodes behave as independent units
  - more fault tolerant
  - possibly economically more feasible
  - harder to implement reliably (needs for synchronization, etc.)

  ![peer ↔ peer approach](image)
From information spread view:
From information spread view:

**Gossip** (or "*epidemic*"") protocols:
- each node contacts one or few nodes in each round
- asynchronous behaviour
- possibly robust to errors
From information spread view:

**Gossip** (or ”*epidemic*”) protocols:
  - each node contacts one or few nodes in each round
  - asynchronous behaviour
  - possibly robust to errors

**Consensus** algorithms:
  - nodes broadcast messages
  - synchronous approach
  - all nodes in the networks converge to a consensus (average, maximum, etc.)
Summary

- Distributed algorithms are composed of small building ”blocks” each computing a small part of a bigger problem.
Summary

- Distributed algorithms are composed of small building "blocks" each computing a small part of a bigger problem.
- Blocks are equivalent and interchangeable among the nodes.
Summary

- Distributed algorithms are composed of small building ”blocks” each computing a small part of a bigger problem.
- Blocks are equivalent and interchangeable among the nodes.
- In contrast to parallel algorithms, distributed algorithms
Distributed algorithms are composed of small building "blocks" each computing a small part of a bigger problem.

Blocks are equivalent and interchangeable among the nodes.

In contrast to parallel algorithms, distributed algorithms typically work on an underlying graph.
Distributed algorithms are composed of small building "blocks" each computing a small part of a bigger problem.

Blocks are equivalent and interchangeable among the nodes.

In contrast to parallel algorithms, distributed algorithms typically work on an underlying graph ⇒ blocks cannot exchange data with each other directly ⇒ data "flow" in the network
Summary

- Distributed algorithms are composed of small building "blocks" each computing a small part of a bigger problem.
- Blocks are equivalent and interchangeable among the nodes.
- In contrast to parallel algorithms, distributed algorithms
  - Typically work on an underlying graph ⇒
  - blocks cannot exchange data with each other directly ⇒ data "flow" in the network

Using simple building blocks we obtain more sophisticated algorithms and applications:

- distributed FFT, RLS, PAST, QR, etc.
- distributed target tracking, sensor localizations, etc.
- heterogenous peer-to-peer computing applications (seti@home, BOINC)
Tools for designing and analysis of the distributed algorithms

1. Graph theory
2. Linear algebra and matrix theory

\[ A = \begin{pmatrix}
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0
\end{pmatrix} \]
Distributed consensus algorithms
Consensus Algorithm

- **Linear Consensus Algorithm** – at time $k$ each node receive data from its neighbors and combine them with its own, i.e.

  $$x_i(k + 1) = w_{ii}x_i(k) + \sum_{j \in N_i} w_{ij}x_j(k)$$

  $$\mathbf{x}(k + 1) = \mathbf{W}\mathbf{x}(k)$$

  where $\mathbf{W}^{N \times N}$ is a weight matrix which represents connections in the network.

- If for any $\mathbf{x}(0) \in \mathcal{R}^N$, there exists a scalar $\alpha$ such that

  $$\lim_{k \to \infty} \mathbf{x}(k) = \alpha\mathbf{1}$$

  we say that we have reached a **consensus**.
Example of convergence of linear average consensus algorithm
Linear consensus algorithm

- Assumptions:
  - synchronous functioning,
  - each node $i$ communicates only with its neighbors $\mathcal{N}_i$,
  - static, strongly connected network (no link or node errors).
Linear consensus algorithm

- Assumptions:
  - synchronous functioning,
  - each node $i$ communicates only with its neighbors $\mathcal{N}_i$,
  - static, strongly connected network (no link or node errors).

- Problems:
  - Conditions on convergence.
  - Proper selection of weights.

- derivation...
Conditions on convergence → Perron-Frobenius theorem

“A real square matrix with positive (non-negative irreducible) elements has a unique largest real eigenvalue with corresponding eigenvector with strictly positive components.”

▶ Important terms:

▶ irreducible matrix = strongly connected
▶ period of a matrix
▶ primitive matrix
▶ derivation and examples...
Typical weight models

- **Constant weights:**
  
  \[
  [W]_{i,j} = \begin{cases} 
  \varepsilon & \text{if } (i, j) \in \mathcal{E} \\
  1 - \varepsilon d_i & \text{if } i = j \\
  0 & \text{otherwise},
  \end{cases}
  \]
  
  with \( 0 < \varepsilon < \frac{1}{\Delta} \), \( \Delta = \max\{d_i\} \).

- **Metropolis-Hastings weights:**
  
  \[
  [W]_{i,j} = \begin{cases} 
  \frac{1}{1 + \max\{d_i, d_j\}} & \text{if } (i, j) \in \mathcal{E} \\
  1 - \sum_{j'} [W]_{ij'} & \text{if } i = j, \\
  0 & \text{otherwise}.
  \end{cases}
  \]

- **“Billion-dollar” weights** (Google matrix used at PageRank):

  \[
  W = \alpha D_{\text{out}} A + (1 - \alpha) \frac{1}{N}
  \]

  \(- D_{\text{out}} - \text{inverse out-degree matrix, } \alpha \approx 0.85\)
Google example

Figure: Graph of hyperlinked webpages

\[ A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad W = \begin{pmatrix} 0.03 & 0.455 & 0.03 & 0.455 & 0.03 \\ 0.2425 & 0.03 & 0.2425 & 0.2425 & 0.2425 \\ 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \\ 0.455 & 0.455 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \end{pmatrix} \]

- Stationary eigenvector: \( v = (0.195, 0.386, 0.112, 0.195, 0.112) \)
- Webpages (nodes) with biggest relevance: \( (2, 1, 4, 3, 5) \)
Speed of convergence

Let 

\[ 1 = \lambda_1 > \lambda_2 \geq \lambda_3 \ldots \lambda_N, \text{ such that } \lambda_2 = \max_{i=2,\ldots,N} |\lambda_i|, \text{ then:} \]

\[
\|x(k) - \frac{1}{N} \mathbf{1}^T \mathbf{1} x(0)\|_2 = \|W^k x(0) - \frac{1}{N} \mathbf{1} \mathbf{1}^T x(0)\|_2 = \|(W^k - \frac{1}{N} \mathbf{1} \mathbf{1}^T) x(0)\|_2 \\
\leq \left\| U \begin{pmatrix} 1 \\ \lambda_2^k \\ \vdots \\ \vdots \end{pmatrix} U^T - U \begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix} U^T \right\|_2 \|x(0)\|_2 \\
= \lambda_2^k \|x(0)\|_2
\]
Speed of convergence

Let

\[ 1 = \lambda_1 > \lambda_2 \geq \lambda_3 \ldots \lambda_N, \text{ such that } \lambda_2 = \max_{i=2,\ldots,N} |\lambda_i|, \text{ then:} \]

\[
\| x(k) - \frac{1}{N} 1^T 1 x(0) \|_2 = \| W^k x(0) - \frac{1}{N} 1 1^T x(0) \|_2 = \| (W^k - \frac{1}{N} 1 1^T) x(0) \|_2 \\
\leq \left\| U \begin{pmatrix} 1 \\ \lambda_2^k \\ & \ddots \end{pmatrix} U^T - U \begin{pmatrix} 1 & & \\ & 0 & \\ & & \ddots \end{pmatrix} U^T \right\|_2 \| x(0) \|_2 \\
= \lambda_2^k \| x(0) \|_2
\]

- The smaller \( \lambda_2 \) the faster the convergence.
- Spectral gap: \( 1 - \lambda_2 \)
- \( \rightarrow \) known only to specific topologies.
Physical Interpretation I

Let’s have an equation of diffusion (on a 2D lattice)

\[
\frac{\partial \phi(x, y; t)}{\partial t} = D \nabla^2_{x,y} \phi(x, y; t)
\]

after discretizing the differential equation in time and space, we obtain

\[
\phi_k(n + 1) = \phi_k(n) + \mu \sum_{m \in \mathcal{N}_k} (\phi_m(n) - \phi_k(n))
\]
leading to:

\[
\Phi(n + 1) = \Phi(n) + \eta (A \Phi(n) - D \Phi(n)) = \\
\Phi(n) - \mu \left( D - A \right) \Phi(n) \\
\]

\[
\Phi(n + 1) = (I - \eta L) \Phi(n) \\
\]

And we have the equation of average consensus:

\[
\Phi(n + 1) = W \Phi(n) 
\]
Summary

- Linear consensus algorithm $\Rightarrow$ only by interacting with neighbors we reach a **global** agreement (average) $\rightarrow$ no routing protocol necessary
Summary

- Linear consensus algorithm ⇒ only by interacting with neighbors we reach a **global** agreement (average) → no routing protocol necessary

- Linear consensus algorithm ⇒ physically it is a **diffusion** process (like a syrup in water)
Summary

- Linear consensus algorithm ⇒ only by interacting with neighbors we reach a global agreement (average) → no routing protocol necessary

- Linear consensus algorithm ⇒ physically it is a diffusion process (like a syrup in water)

- Speed of ”diffusion” is influenced by the density (eigenvalues) of the ”water”

- There exist many other consensus algorithms – linear, nonlinear, synchronous, asynchronous, with state-dependent weights, time-varying, etc.
To be continued