Distributed Consensus

CS289

Distributed Consensus

Average Consensus

- The Average Height Problem
- The Equal Candy Problem

Distributed Consensus in “Real” Distributed Systems

- Estimation in distributed sensors (avg, median, product)
- Load-balancing in computer networks
- Natural Phenomena (diffusion, quorum-sensing)
- Synchronization (heartbeat, distributed antennas, wireless)
- Flocking and formation control (fish and birds, UAVs)
- Environmentally-adaptive robotic systems

Why recognizing “similarity” matters

Distributed robotics
- Formation control and coordination
  - In unmanned vehicles
  - Self-adapting modular robots

Natural Systems
- Collective Synchronization
  - Bird Flocking
  - Diffusion

Theoretical Advances
- Analyze correctness and performance (without knowing application details)
- Analyze for different and complex networks (e.g., time-varying networks)
- Techniques: Control theory, graph theory
Outline

• Part I
  – We will look at the distributed consensus problem from the readings, and go through the mathematical analysis.

• Part II
  – I will show how ideas from distributed consensus have been used recently to show analytically why/how synchronization and flocking work

How do we solve the Problem?

Problem:
  – Given a Graph G=(V,E) undirected, connected
  – Each node i in V has some initial value x(i)(0)
  – Each node i has some neighbors nbrs(i)
  – Nodes must cooperate to compute the average of initial values.

Answer:

intuition: look at how you differ from your local neighbors, and move in the right direction to reduce your disagreement...

\[ x_i(t+1) = x_i(t) + \alpha \Delta x_i \]
where
\[ \Delta x_i = \sum_{k=\text{nbrs}(i)} [x_k(t) - x_i(t)] \]
and
\[ k = \text{nbrs}(i) \]

Notice that it’s NOT OBVIOUS that this locally greedy (myopic) should work...

Globally, we can see that the average is 7 (i.e. 28/4)

...But locally, for node C, its own value will first go down.

\[ x_i(t+1) = x_i(t) + \alpha \Delta x_i \]
where
\[ \Delta x_i = \sum_{k=\text{nbrs}(i)} [x_k(t) - x_i(t)] \]
and
\[ k = \text{nbrs}(i) \]

Think of a line graph (continuous set of nodes) Information must travel, but it can also “slosh” around. How do we know it will ever settle?

In fact, requires \( \alpha < \frac{1}{d_{max}} \)
If \( \alpha = 1/2 \) or \( \alpha = 1/3 \), this example will work
Distributed Consensus

- $x_i(0) =$ initial value
- $x_i(t+1) = x_i(t) + \alpha \Delta x_i$
  
  where $\Delta x_i = \sum (x_k(t) - x_i(t))$ and $k = \text{nbrs}(i)$

Interesting Properties of this Algorithm
- Simple node behavior (Anonymous, leaderless, no params)
- Self-maintaining (provides inherent robustness)
- It works! (provably so if $\alpha < 1/d_{\text{max}}$)

Provably Correct
- Will converge to average, on any undirected connected graphs
- Time depends on (a) distance to answer (b) network topology
  
  How do we prove this?

Graph Laplacian

- From a local point of view (node)
  
  $x_i(t+1) = x_i(t) + \alpha \Delta x_i$
  
  $\Delta x_i = \sum (x_k(t) - x_i(t))$ where $k = \text{nbrs}(i)$
  
  $\Delta x_i = (\sum x_j(t) - N_i x_i(t))$ where $N_i =$ number of ners

- From a global point of view (state matrix)

In matrix form:

\[
\Delta X = -L X(t)
\]
Graph Laplacian

Turns out "L" is a famous matrix!!!

\[ L = D - A \]

(Degree matrix - Adj matrix)

Definition: Spectral properties of a matrix A
eigenvalues (scalar) = \( v_1, v_2, v_3, \ldots, v_n \) (scalars)
eigenvectors (vector) = \( e_1, e_2, e_3, \ldots, e_n \)

For matrix A, \( A.e_1 = v_1.e_1 \) (eigen decomposition)

If G is a undirected connected graph, then for \( L(G) \):
\( v_1 = 0 \) and \( e_1 = [a \ a \ a \ a \ a \ldots] \) is unique
(For undirected/connected)
\( v_2 = \text{algebraic connectivity} \) and is > 0
(how "dense" the graph is)
the other \( v_s \) and \( e_s \) are also "magical"

\[ \Delta X = -L X(t) \]

Back to Distributed Consensus

• From a local point of view (node)

\[ x_i(t+1) = x_i(t) + \alpha \Delta x_i \]
\[ \Delta x_i = \frac{1}{N_i} \sum_{j \in \text{nbrs}(i)} \left( x_j(t) - x_i(t) \right) \]
where \( \alpha \) = number of nrs

• From a global point of view (state matrix)

\[ \begin{bmatrix}
\Delta x_0 \\
\Delta x_1 \\
\Delta x_2 \\
\Delta x_3
\end{bmatrix} =
\begin{bmatrix}
-0 & 1 & 1 & 0 \\
1 & -0 & 1 & 1 \\
1 & 1 & -0 & 2 \\
0 & 1 & 0 & -0
\end{bmatrix}
\begin{bmatrix}
x_0(t) \\
x_1(t) \\
x_2(t) \\
x_3(t)
\end{bmatrix} \]

Captures the decentralized process: \( \Delta X = -L X(t) \)

Proving the algorithm works

\[ X(t+1) = X(t) + \alpha \Delta X \]

where \( \Delta X = -L X(t) \)

• Prove Correctness:
  – When it stops, the answer must be the average
  – It always stops, from any initial condition
Proving the algorithm works

\[ X(t+1) = X(t) + \alpha \Delta X \]
where \( \Delta X = -LX(t) \)

- **Prove Correctness:**
  - When it stops, the answer must be the average
  - It always stops, from any initial condition

- **If G is undirected and connected**
  1. Consensus is a unique fixed point
  2. The Consensus is the average of initial values
  3. This is a stable fixed point

\[ X(t+1) = X(t) + \alpha \Delta X \]
where \( \Delta X = -LX(t) \)

- **Prove Correctness:**
  - When it stops, the answer must be the average
  - It always stops, from any initial condition

- **If G is undirected and connected**
  1. Consensus is a unique fixed point
     - Stops when \( \Delta X = -LX(t) = 0 \)
     - As we saw earlier, \( v_1 = 0, e_1 = [a a a a a a a a a] \) (and \( v_2 > 0 \))
  2. The Consensus is the average of initial values
  3. This is a stable fixed point

\[ X(t+1) = X(t) + \alpha \Delta X \]
where \( \Delta X = -LX(t) \)

- **Prove Correctness:**
  - When it stops, the answer must be the average
  - It always stops, from any initial condition

- **If G is undirected and connected**
  1. Consensus is a unique fixed point
     - Stops when \( \Delta X = -LX(t) = 0 \)
     - As we saw earlier, \( v_1 = 0, e_1 = [a a a a a a a a a] \) (and \( v_2 > 0 \))
  2. The Consensus is the average of initial values
  3. This is a stable fixed point

### Proving Stability

**Metric of “disagreement”**
(at time t, what’s the system error?)

\[ M(t) = \sum (x_i(t) - \text{avg})^2 \] sum of squared error

- **Prove that with each step, the dynamics of this system will cause this disagreement to be reduced**
  - At each step, I reduce the disagreement by a fraction that depends on topology...

\[ M(t+1) <= M(t) - 2.v2.M(t) \]

- While initial convergence may be slow, reaction to perturbations is extremely fast!
Beyond Simple Consensus

Generalizable
• Directed graphs (strongly connected) [OS, T]
• Time-varying graphs [T, FL, OS]
• Gossip graphs [G]
• Distributed homeostasis (constraints) [F]
• Applications: Flocking, Synchronization, Vehicle formations, Sensor fusion, Self-adaptive robotic systems.

Citations
- [OS] Olfati-Saber, Murray, 2003
- [FL] Tanner, Jadbabaie, Pappas, 2003
- [G] Kempe et al 03, Xiao & Boyd 2004, Xiao et al 06
- [T] Luc Moreau, CDC 2003
- [F] Fax and Murray, 2004

Outline

• Part I
  - We will look at the distributed consensus problem from the readings, and go through the math.

• Part II
  - I will show how ideas from distributed consensus have been used recently to show analytically why/how synchronization and flocking work

PART II

• Synchronization
  - Mirollo and Strogatz, SIAM 1990.
  - Izhikevich, IEEE Trans on Neural Networks, 1999

• Flocking
  - Tanner, Jadbabaie, Pappas, CDC, 2003 (2)
  - Olfati-Saber, Murray, CDC 2003
  - Review: Olfati-Saber, Fax, Murray, 2007

• Both can be seen as a form of collective consensus

Mirollo and Strogatz Sync (1990)

How does a firefly (node) behave?

\[ o_i(t+1) = o_i(t) + \Delta o_i \]
\[ \Delta o_i = \frac{1}{T} + \text{jump}(o_i) \cdot p_i(t) \]

Where \( p_i(t) = 1 \) if some neighbor fired ("pulse")
A simple jump function is \( \text{jump}(o) = c \cdot o \)
One can understand how this behaves for 2 oscillators
Lucarelli and Wang, 2004

**Local Point of View** (slightly modified)

\[ \Delta o_i = 1/T + \{c_0 o_i\} \sum p_k(t) \]

where \( p_k(t) = 1 \) if nbr \( k \) fired

If \( c \) is very small, then

Can applying Theorem by Izhikevich (1999)

can transform a pulse system to a continuous system

\[ \Delta o_i = e(1/T) \sum (O_k(t) - o_i(t)) \]

**Global Point of View**

\[ \Delta o = -\alpha L o(t) \]

Laplacian => Consensus!!

Speed of synchronization is affected by \( v^2 \)

(L&W proved a transformation for all jump functions that satisfy M&S criteria)

---

**Flocking**

- Reynolds’ Rules
  - Nearest neighbor behavior
  - Combination: cohesion, repulsion, alignment

  What do these rules guarantee?

- Tanner et al: What defines a Flock?
  - All flock members align their heading
  - All flock members achieve desired spacing
  - A connected flock remains connected (not proved)

- Alignment is like consensus
  - Problem is that the network changes at each step
  - Need to prove Consensus over time-varying topologies!!

---

**Flocking Mathematically**

- \( r_i \) and \( v_i \) = position and velocity of node \( i \)
- \( v_i(t+1) = v_i(t) + \Delta v_i \)
- \( \Delta v_i = \text{align-with-nbrs (consensus)} + \text{maintain "good" distance with nbrs} \)
- \( \Delta v_i = \sum [v_k(t) - v_i(t)] + \sum \text{gradient } f(r_{ik}) \)

  \( f(r_{ik}) = \text{infinity if too close, 0 if perfect, high if too far} \)

- Globally
  \[ \Delta v = -Lv(t) + \text{other term} \]

- Problem is, the topology changes at every step!
- Old world: \( v(t) = A^t v(0) \)
- New world: \( v(t) = A(t)A(t-1).....A(1)A(0) v(0) \)
  - But it still works!!!!!
Swarm Intelligence

- Clustering
  - Stigmergy
  - Data Sorting & Clustering
- Foraging
  - Stigmergy
  - Search
  - Optimization
  - Routing
- Task Allocation
  - Stigmergy
  - Threshold-based division of labor
- Construction
  - Self-Assembly
  - (Self-assembly/Construction Stigmergy)
- Collective Transport
  - Distributed
  - Consensus (Physics)
- Flocking & Synchrony
  - Distributed
  - Consensus (Spatial/Time)
- Library of Decentralized Algorithms
- House Hunting & Quorum sensing
  - Distributed Consensus
  - Threshold-based division of labor
  - Symmetry-breaking
  - (Spatial/Time)