Convergence Rates in Decentralized Optimization

Alex Olshevsky
Department of Electrical and Computer Engineering
Boston University
Distributed and Multi-agent Control

- Strong need for protocols to coordinate multiple agents.
- Such protocols need to be distributed in the sense of involving only local interactions among agents.

Image credit: CubeSat, TCLabs, Kmel Robotics
**Challenges**

- Decentralized methods.
- Unreliable links.
- Node failures.
- Too much data.
- Too much local information.
- Malicious nodes.
- **Fast & scalable performance.**
- Interaction of cyber & physical components.

Image credit: UW Center for Demography
Problems of Interest

- Formation control
- Target Localization
- Cooperative Estimation
- Distributed Learning
- Leader-following
- Coverage control

- Load balancing
- Clock synchronization in sensor networks
- Resource allocation
- Dynamics in social networks
- Distributed Optimization
This presentation

1. Major concerns in multi-agent control (3 slides)
2. **Three problems (4 slides)**
   a) Distributed learning
   b) Localization from distance measurements
   c) Distributed optimization
3. A common theme: average consensus protocols (10 slides)
   a) Introduction
   b) Main result
   c) Intuition
4. Revisiting the three problems from part 2 (21 slides)
5. Conclusion (1 slide)
Distributed learning

- There is a true state of the world $\theta^*$ that belongs to a finite set of hypotheses $\Theta$.
- At time $t$, agent $i$ receives i.i.d. random variables $s_i(t)$, lying in some finite set. These measurements have distributions $P_i(\cdot|\theta)$, which are known to node $i$.
- Want to cooperate and identify the true state of the world. Can only interact with neighbors in some graph(s).
- A variation: no true state of the world, some hypotheses just explain things better than others.
- Will focus on source localization as a particular example.
Each agent (imprecisely) measures distance to source; these give rise to beliefs, which need to be fused in order to decide a hypotheses on the location of the source.
Decentralized optimization

- There are $n$ agents. Only agent $i$ knows the convex function $f_i(x)$.
- Agents want to cooperate to compute a minimizer of
  \[ F(x) = \frac{1}{n} \sum_i f_i(x) \]
- As always, agents can only interact with neighbors in an undirected graph -- or a time-varying sequence of graphs.
- Too expensive to share all the functions with everyone.
- But: everyone can compute their own function values and (sub)gradients.
Distributed regression -- an example

- Users with feature vectors $a_i$ are shown an ad.
- $y_i$ is a binary variable measuring whether they `liked it.``
- One usually looks for vectors $z$ corresponding to predictors $\text{sign}(z^t a_i + b)$
- Some relaxations considered in the literature:
  \[
  \sum_i 1 - y_i(z^t a_i + b) + \lambda \|z\|_1 \\
  \sum_i \max(0,1 - y_i(z^t a_i + b)) + \lambda \|z\|_1 \\
  \sum_i \log (1 + e^{-y_i(z^t a_i + b)}) + \lambda \|z\|_1
  \]
  Want to find $z$ & $b$ that minimize the above.
- If the $k$’th cluster has data $(y_i, a_i, i \in S_k)$, then setting
  \[
  f_k(z,b) = \sum_{i \in S_k} 1 - y_i(z^t a_i + b) + \lambda \|z\|_1
  \]
  recovers the problem of finding a minimizer of $\sum_k f_k$
This presentation

1. Major concerns in multi-agent control (3 slides)
2. Three problems (4 slides)
   a) Distributed learning
   b) Localization from distance measurements
   c) Distributed optimization & distributed regression
3. **Average consensus protocols (10 slides)**
   a) Introduction
   b) Main result
   c) Intuition
4. Revisiting the three problems from part 2 (15 slides)
5. Conclusion (2 slides)
The Consensus Problem - I

- There are \( n \) agents, which we will label \( 1, \ldots, n \)
- Agent \( i \) begins with a real number \( x_i(0) \) stored in memory
- Goal is to compute the average
  \[
  \frac{1}{n} \sum_{i} x_i(0)
  \]
- Nodes are limited to interacting with neighbors in an undirected graph or a sequence of undirected graphs.
The Consensus Problem - II

- Protocols need to be fully distributed, based only on local information and interaction between neighbors. Some kind of connectivity assumption will be needed.

- Want protocols inherently robust to failing links, failing or malicious nodes, don’t suffer from a “data curse” by storing everything.
- Want to avoid protocols based on flooding or leader election.
- Preview: this seems like a toy problem, but plays a key role in all the problems previously described.
Consensus Algorithms: Gossip

Nodes break up into a matching

...and update as

\[ x_i(t+1), x_j(t+1) = \frac{1}{2} \left( x_i(t) + x_j(t) \right) \]

First studied by [Cybenko, 1989] in the context of load balancing (processors want to equalize work along a network).
Consensus Algorithms: Equal-neighbor

\[ x_i(t+1) = x_i(t) + c \sum_{j \in N(i,t)} x_j(t) - x_i(t) \]

- Here \( N(i,t) \) is the set of neighbors of node \( i \) at time \( t \).
- Works if \( c \) is small enough (on a fixed graph, \( c \) should be smaller than the inverse of the largest degree).
- First proposed by [Mehyar, Spanos, Pongsajapan, Low, Murray, 2007].
Consensus Algorithms: Metropolis

\[ x_i(t+1) = x_i(t) + \sum_{j \in N(i,t)} w_{ij}(t) (x_j(t) - x_i(t)) \]

- First proposed in this context by [Xiao, Boyd, 2004].
- Here \( w_{ij}(t) \) are the Metropolis weights

\[ w_{ij}(t) = \min(1 + d_i(t), 1 + d_j(t))^{-1} \]

where \( d_i(t) \) is the degree of node \( i \) at time \( t \).  
- Avoids the hassle of choosing the constant \( c \) before.
Consensus Algorithms: others

- All of the above protocols are linear:
  \[ x(t+1) = A(t) \cdot x(t) \]
  where \( A(t) = [a_{ij}(t)] \) is a stochastic matrix. Note that \( A(t) \) is always compatible with the graph in the sense of \( a_{ij}(t) = 0 \) whenever there is no edge between \( i \) and \( j \).

- Can design nonlinear protocols [Chapman and Mesbahi, 2012], [Krause 2000], [Hui and Haddad, 2008], [Srivastava, Moehlis, Bullo, 2011], many others….

- Most prominent is the so-called push-sum protocol [Dobra, Kempe, Gehrke 2003] which takes the ratio of two linear updates.
Our Focus: Designing Good Protocols

- **Our goal**: simple and robust protocols that work quickly...even in the worst case.
- What does ``worst-case”’ mean?
- Look at time until the measure of disagreement
  \[
  S(t) = \max_i x_i(t) - \min_i x_i(t)
  \]
  is shrunk by a factor of \( \varepsilon \).
  Call this \( T(n, \varepsilon) \).
- We can take worst-case over either all fixed connected graphs or all time-varying graph sequence (satisfying some long-term connectivity conditions).
## Previous Work and Our Result

<table>
<thead>
<tr>
<th>Authors</th>
<th>Bound for $T(n, \varepsilon)$</th>
<th>Worst-case over</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Tsitsiklis, Bertsekas, Athans, 1986]</td>
<td>$O(n^n \log (1/\varepsilon))$</td>
<td>Time-varying directed graphs</td>
</tr>
<tr>
<td>[Jadbabaie, Lin, Morse, 2003]</td>
<td>$O(n^n \log (1/\varepsilon))$</td>
<td>Time-varying directed graphs</td>
</tr>
<tr>
<td>[O., Tsitsiklis, 2009]</td>
<td>$O(n^3 \log (n/\varepsilon))$</td>
<td>Time-varying undirected graphs</td>
</tr>
<tr>
<td>[Nedic, O., Ozdaglar, Tsitsiklis, 2011]</td>
<td>$O(n^2 \log (n/\varepsilon))$</td>
<td>Time-varying undirected graphs</td>
</tr>
<tr>
<td>[O., 2015], this presentation</td>
<td>$O(n \log (n/\varepsilon))$</td>
<td>Fixed undirected graphs</td>
</tr>
</tbody>
</table>
The Accelerated Metropolis Protocol - I

\[ y_i(t+1) = \sum_j a_{ij} x_j(t) \]
\[ x_i(t+1) = y_i(t+1) + (1-(9n)^{-1}) (y_i(t+1) - y_i(t)) \]

- Here \( a_{ij} \) is half of the Metropolis weight whenever \( i,j \) are neighbors. \( A(t)=[a_{ij}] \) is a stochastic matrix.
- Must be initialized as \( x(0)=y(0) \).
- **Theorem [O., 2015]**: If each node of an undirected connected graph uses the AM method, then each \( x_i(t) \) converges to the average of the initial values. Furthermore, \( S(t) \leq \varepsilon S(0) \) after \( O(n \log (n/\varepsilon)) \) updates.
The Accelerated Metropolis Protocol - II

\[ y_i(t+1) = \sum_j a_{ij} x_j(t) \]

\[ x_i(t+1) = y_i(t+1) + \left(1-(9n)^{-1}\right)(y_i(t+1) - y_i(t)) \]

- The idea that iterative methods for linear systems can benefit from extrapolation is very old (~1950s). Used in consensus by [Cao, Spielman, Yeh 2006], [Johansson, Johansson 2008], [Kokiopoulou, Frossard, 2009], [Oreshkin, Coates, Rabbat 2010], [Chen, Tron, Terzis, Vidal 2011], [Liu, Anderson, Cao, Morse 2013], ...

- As written, requires knowledge of the number of nodes by each node. This can be relaxed: each node only needs to know an upper bound correct within a constant factor.
Proof idea

- The natural update $x(t+1) = A \cdot x(t)$ with stochastic $A$ corresponds to asking about the speed at which a Markov chain converges to a stationary distribution.
- Main insight 1: Metropolis chain mixes well because it decreases the centrality of high-degree vertices.
- In particular: whereas the ordinary random walk takes $O(n^3)$ to mix, the Metropolis walk takes $O(n^2)$
- Main insight 2: can think of Markov chain mixing as gradient descent, and use Nesterov acceleration to take square root of running time.
- This argument can give $O(\text{diameter})$ convergence (up to log factors) on geometric random graphs or 2D grids.
This presentation

1. Major concerns in multi-agent control (3 slides)
2. Three problems (4 slides)
   a) Distributed learning
   b) Localization from distance measurements
   c) Distributed optimization & distributed regression
3. A common theme: consensus protocols (10 slides)
   a) Introduction
   b) Main result
   c) Intuition
4. **Revisiting the three problems from part 2** (15 slides)
5. Conclusion (2 slides)
There are $n$ agents. Agent $i$ knows the convex function $f_i(x)$. Agents want to cooperate to compute a minimizer of

$$F(x) = \frac{1}{n} \sum_i f_i(x)$$

This contains the consensus problem as a special case.

In the centralized setup, assuming each $f_i(x)$ has subgradient bounded by $L$, the subgradient method on the function $F(x)$ results in

$$F(x_a(t)) - F(x^*) = O\left(\frac{1}{\sqrt{t}}\right)$$

This means that the time until the objective is within epsilon of the optimal value is $O(1/\epsilon^2)$.
Previous work

- [Nedic, Ozdaglar 2009] proposed that node $i$ maintain the variable $x_i(t)$ which is updated as

$$x_i(t+1) = \sum_j a_{ij}(t) x_j(t) - \alpha g_i(t)$$

where $g_i(t)$ is the subgradient of $f_i(x)$ at $x_i(t)$ and $[a_{ij}(t)]$ is any of the consensus matrices above.

- [Nedic, Ozdaglar, 2009] showed that each averaged $x_i(t)$ converges to a small neighborhood of the same minimizer of $F(\cdot)$.
Intuition
There is a natural algorithm inspired by the AM Method:

\[ y_i(t+1) = \sum_j a_{ij} x_j(t) - a g_i(t) \]

\[ z_i(t+1) = y_i(t) - a g_i(t) \]

\[ x_i(t+1) = y_i(t+1) + (1 - 1/(9n)) (y_i(t+1) - z_i(t+1)) \]

...where \( g_i(t) \) is the subgradient of \( f_i \) at \( x_i(t) \), \( L \) is an upper bound on the norm of \( g_i(t) \), \( a = 1/(L\sqrt{n\sqrt{T}}) \), and \( a_{ij} \) are half-Metropolis weights.

**Main idea:** this interleaves gradient descent with an averaging scheme.
Linear Time Decentralized Optimization - II

- **Theorem [O., 2015]:** on any undirected connected graph, we have that all $x_i(t)$ approach the same minimizer of $F$ and $F(x_a(t))-F(x^*) < \epsilon$ after $O(n/\epsilon^2)$ iterations.

- Initial paper [Nedic, Ozdaglar 2009] had a bound of $O(n^{2n}/\epsilon^2)$ to get within $\epsilon$

- Later improved by [Ram, Nedic, Veeravalli 2011] to $O(n^4/\epsilon^2)$ time to get within $\epsilon$

- In simulations, the linear convergence time still holds on time-varying graphs.
What have we accomplished?

We have proposed an algorithm that:

- Every agent stores three numbers.
- Always works in linear time on fixed graphs (this is optimal).
- Automatically robust to failing nodes.
- Simulations show it is robust to link failures.
- Simulations show it works in linear time on time-varying graphs.
Distributed (non)Bayesian Learning

- There is a finite set of hypotheses $\Theta$.
- At time $t$, agent $i$ receives i.i.d. measurements $s_i(t)$, lying in some finite set, having a distribution $q_i$.
- Under hypothesis $\theta$, the measurements $s_i(t)$ have distribution $P_i(.|\theta)$.
- Nodes want to cooperate and identify the state of the world which best explains the observations.
- Call that state of the world $\theta^*$.
- Formally: $\theta^* = \arg\min_{\theta} \sum_i D_{KL}(q_i, P_i(.|\theta))$
Here $\theta_2$ is $\theta^*$ and is the true state of the world.

Here $\theta_2$ could be $\theta^*$ although it is not the best in terms of the observations of any individual agent.
Distributed Bayesian Learning

- Agent $i$ maintains a stochastic vector over $\Theta$, which we will denote $b_i(t, \theta)$, initialized to be uniform. Stack these up into $b_i(t)$.

- For a nonnegative vector $x$, define $N(x)$ to be $x/\|x\|_1$.

- Bayes rule may be written as

  $$b_{i_{\text{temp}}}(t+1) = b_i(t) \cdot P(s_i(t)|\theta))$$

  $$b_i(t+1) = N(b_{i_{\text{temp}}}(t+1))$$

where $\cdot$ is elementwise multiplication of vectors.
The Independent Bayes Update

Let $\Omega^i$ be the set of hypotheses best for agent $i$. Well-known: if agents use above rule (i.e., ignore each other) then all $b_i(t, \Theta)$ concentrate on $\Omega^i$ as $t \to +\infty$. 

\[ \begin{align*} 
\Omega^1 & \quad \bullet \theta_1 \\
\Omega^2 & \quad \bullet \theta_2 \quad \bullet \theta_3 \quad \bullet \theta_4 \\
\Omega^3 & \quad \bullet \theta_5 \quad \bullet \theta_6 
\end{align*} \]
First attempt at an algorithm:

\[
\begin{align*}
    b_{i, \text{temp}}(t+1) &= b_i(t) \cdot P(s_i(t)|\theta)) \cdot \prod_{j \in N(i, t)} b_j(t)^{a_{ij}} \\
    b_i(t+1) &= \mathcal{N}(b_{i, \text{temp}}(t+1))
\end{align*}
\]

Essentially proposed by [Alanyali, Saligrama, Savas, Aeron 2004]. Each node performs a weighted Bayes update treating the beliefs of neighbors as observations and ignoring dependencies.

Theorem [Nedic, O., Uribe 2015], [Shahrampour, Rakhlin, Jadbabaie 2015], [Lalitha, Sarwate, Javidi 2015]: if \( [a_{ij}] \) is any of the stochastic consensus matrices from before, and the graph is undirected and connected, then almost surely all \( b_i(t, \theta) \) geometrically approach \( 1(\theta^*) \) (i.e., indicator of \( \theta^* \)).
The update
\[
b_{i, \text{temp}}(t+1) = b_i(t) \cdot P(s_i(t)|\Theta)) \cdot \prod_{j \in N(i,t)} b_j(t)^{a_{ij}}
\]
\[
b_i(t+1) = N(b_{i, \text{temp}}(t+1))
\]
is very similar to a consensus update after the nonlinear change of variables \(y_i(t) = \log b_i(t)\).

Similar idea to distributed optimization: each node "pulls" in favor of the explanations that favor its data and these pulls are reconciled through a consensus scheme.
Well if that is the case, then how about:

\[ b_{i, \text{temp}}(t+1) = b_i(t) .* P_i(s_i(t)|\theta)) .* \prod_{j \in N(i)} b_j(t)^{(1+\sigma)a_{ij}} \]

\[ v_{i, \text{temp}}(t+1) = \prod_{j \in N(i)} b_j(t-1) .* P_j(s_j(t)|\theta)) \]

\[ b_i(t+1) = N(b_{i, \text{temp}}(t+1) ./ v_{i, \text{temp}}(t+1)) \]

where \( a_{ij} \) are the lazy Metropolis weights and \( \sigma = 1-(18n)^{-1} \).

Intuition: each node pulls in favor its own beliefs, and these pulls are reconciled now using the AM method.
Distributed (non)Bayesian Learning - V

**Theorem** [Nedic, O., Uribe 2015]: Suppose that under $\theta^*$ all events occur with probability at least $p_{\text{min}}$.

Then, for all $\theta \neq \theta^*$ and all $t$, we have with probability $1 - \rho$ the bound

$$b_i(t, \theta) \leq e^{-(a/2)t+c}$$

...holds for all $t \geq N(\rho)$ where

$$a = (1/n) \min_{\theta \neq \theta^*} \left[ \sum_j D_{KL}(q_j || P_j(s_j(t)|\theta)) - D_{KL}(q_j || P_j(s_j(t)|\theta^*)) \right]$$

$$c = O(n \log n \log (1/p_{\text{min}}))$$

$$N(\rho) = O([\log (1/p_{\text{min}}) \log (1/\rho)] / a^2)$$
Learning for Target Localization

- Fixed target position.
- 15 sensors performing random motion.
- Gaussian noise
- Time-varying graph, often disconnected.
- Learning is very quick.
Learning for Target Tracking

- Target performs random motion.
- 10 sensors performing random motion.
- Gaussian noise
- Time-varying graph, often disconnected.
Following a target

- Target performs random motion.
- 10 sensors:
  -- attracted to estimates of target position
  -- repulsed from each other
- Gaussian noise
Following a faster target: failure

- Target performs random motion.
- 10 sensors:
  -- attracted to estimates of target position
  -- repulsed from each other
- Much faster target than before
Following a faster target: success

- Target performs random motion.
- 12 sensors:
  - 8 are:
    -- attracted to estimates of target position
    -- repulsed from each other
  - 4 perform random motions.
Tracking with incorrect measurements

- Both target and sensors perform random motion.
- Red sensors have random bias in addition to noise.
- Blue sensors are just noisy.
- Time-varying graph.
- Now takes longer for estimates to resolve.
Conclusion

● *One* (very simple) result: a consensus protocol with convergence time \( O(n \log (n/\varepsilon)) \).

● *This talk*: linear-time algorithms for distributed optimization and distributed learning.

● *Main take-away*: every multi-agent problem that can be solved by coupling local objectives via consensus terms can be linearly scalable in network size with this method.