Distributed Algorithms for Optimization in Networks

Angelia Nedić

January 27, 2022

Angelia.Nedich@asu.edu

Distributed (Large-Scale) Optimization Problems: Sources

- Automatic Control Systems (robot networks)
  - Energy Systems
  - Envisioned Smart Grids and Smart Cities
- Signal and Image Processing (Image Reconstruction, Pattern Recognition)
- Data Science (Learning from Data)

Machine Learning Problem

- Consider a prototype problem arising in the supervised learning, where a machine (or neural net) is trained from a large data set.
- The problem typically consists of minimizing some objective cost subject to a large number of constraints of the following form:

$$\begin{array}{ll} \text{minimize} & \rho(x) \\ \text{subject to} & g(x; y_i, z_i) \leq 0, \qquad i = 1, \ldots, m, \quad x \in \mathbb{R}^n, \end{array} \tag{1}$$

  where p is the number of data points ($m \gg 1$), $x \in \mathbb{R}^n$ is a decision vector (the vector of weights in neural-nets), and the function $\rho(\cdot)$ is used to promote certain properties of the solutions, such as sparsity or robustness.
- The function $g(x; y_i, z_i)$ represents a constraint imposed by the data point $(y_i, z_i) \in \mathbb{R}^{n+1}$, where y_i is a measurement and z_i is the label associated with the measurement.
- For example, for linear classifiers, each data constraint is linear, i.e.,

$$g(x; y_i, z_i) = 1 - z_i \langle y_i, x \rangle$$

  while the labels z_i are binary.
- The difficulty in solving problem (1) lies in **the large number m of constraints**.

Strategies

- The existing methods developed prior to the emergence of such large problems could not cope with such a large scale.

- To cope with the large number of constraints, there are two main conceptual approaches related to problem (1)
  - **Penalty-Based Reformulation**, which essentially replaces problem (1) with an unconstrained problem obtained by penalizing the constraints to form a new objective function. *The resulting unconstrained problem is not necessarily equivalent to the original constrained problem* (1).
  - **Sampled-Constraint Approximation**, where the problem is either approximated or addressed directly by sampling the constraints "on-the-go" (within an algorithm).

Penalty-Based Reformulation

- Original constrained problem

$$\text{minimize } \rho(x) \qquad \text{subject to} \quad g(x; y_i, z_i) \leq 0, \qquad i = 1, \ldots, m, \quad x \in \mathbb{R}^n,$$

- Introducing a loss function $\ell(\cdot)$ (associated with the quality of data-fitting) and a regularization parameter $r > 0$, the problem is re-formulated as an unconstrained problem:

$$\text{minimize} \quad r\rho(x) + \frac{1}{m}\sum_{i=1}^{m} \ell(x; y_i, z_i), \tag{2}$$

  where the loss function penalizes the violation of constraints $g(x; y_i, z_i) \leq 0$, $i = 1, \ldots, m$.
- For example, for linear classifiers, common choices include:
  - The *logistic regression* loss given by $\ell(x; y, z) = \log\left(1 + e^{-z\langle x, y\rangle}\right)$
  - The *hinge loss* $\ell(x; y, z) = \max\{0, 1 - z\langle x, y\rangle\}$.
- By scaling the objective function in (2) with a regularization parameter $r > 0$, we can interpret $1/r$ as the penalty parameter.
- The resulting penalized problem balances the regularizing function $\rho(\cdot)$ and the average sum of the loss functions, where the balance is controlled by the parameter $r > 0$.

Minimizing the Average Sum of Loss-Functions

- We will now consider a general form of the problem in (2):

$$\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} f_i(x), \tag{3}$$

- There is a vast body of work that offers various gradient methods for solving such an unconstrained problems with an additive-type objective function.
- The random incremental gradient method, often referred to as *stochastic gradient descent* in some of the machine learning community, has been the most successful due to its simplicity, and it has a long tradition starting with Kibardin 1980* (see Bertsekas 2012† for an in-depth survey on these methods).
- A renewed interest driven by a desire to improve its convergence rate, which can be unfavorable due to the stochastic errors induced by the sampling of the objective function gradients.
- The development of several efficient variance-reduction methods, such as stochastic variance reduced gradient (SVRG), SAG, SAGA, Katyusha.

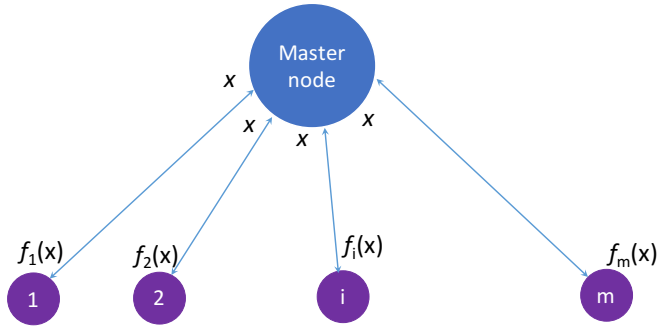
*V. M. Kibardin *Decomposition into Functions in the Minimization Problem*, Automation and Remote Control, 40 (9) 1311–1323, 1980

†D. P. Bertsekas *Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey*, in a book on Optimization for Machine Learning, pp. 85–119, MIT Press, Cambridge, MA, 2012

Distributed but Centralized Computational Architecture

- The existing the random incremental gradient methods (aka stochastic gradient descent) can be distributed within a master-slave architecture

  Solving $\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} f_i(x)$ in a master-slave architecture with a master node and m workers. The master node



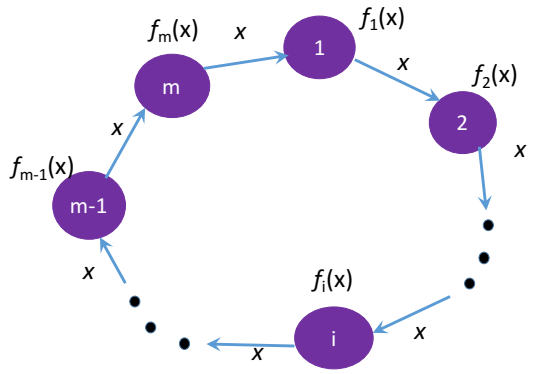
  is responsible for maintaining the decision vector x, Each worker i is responsible for processing the function f_i given the state x (typically computes the gradient $\nabla f(x)$).

- Such an architecture is not **fully distributed** (i.e., decentralized) as **it requires a central entity to coordinate the computations** of the slaves (workers).
- This architecture inherently requires the knowledge of the number m of workers.
- Communication with the central entity (master node) is intense when m is large.
- Fast methods (SVRG, SAG, SAGA, etc.) also require master-node with memory of the size $m \times n$ to store past gradients for each $f_i, i = 1, \ldots, m$

Distributed & Decentralized Computational Architecture

- The information processing (iterate updates) of a cyclic incremental gradient method can be interpreted as computations in a cyclic directed graph (see D.P. Bertsekas' webpage for papers on incremental methods)

  Solving $\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} f_i(x)$ with a cyclic incremental method. Each iteration consists of an update of x along a



  cyclic directed graph over the nodes $1, 2, \ldots, m$. A node i receives x from its up-stream neighbor $i-1$, updates x based on $\nabla f_i(x)$, and sends the updated x to its down-stream neighbor $i+1$.

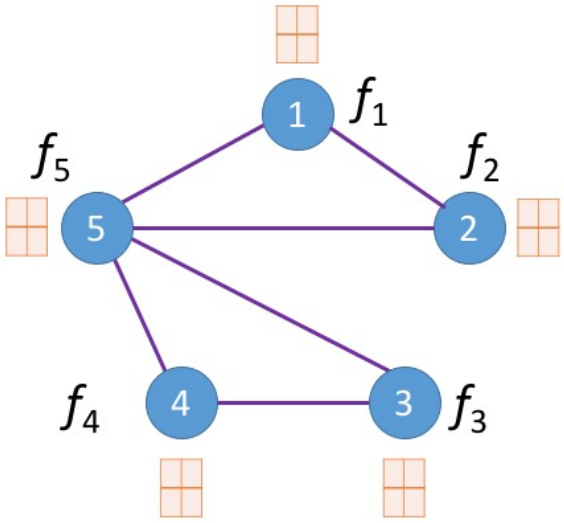
- Information processing (algorithm) along such a cycle has two shortcomings:
  - Takes long time for a full iteration update when m is large
  - Failure of one node, or a link, breaks the computations.

General Distributed & Decentralized Model

We consider a problem

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} f_i(x)$$

in a system consisting of m agents that are embedded in a communication network.



- Function f_i is *privately known* only to agent i, not shared with any other agent.
- Agents communicate some limited information with their immediate neighbors only
- The problem is to be solved *distributedly* i.e., **without a central entity**
- Every agent i has only local knowledge of the graph, i.e., it only knows its neighbors
- No agent knows even the total number m of the agents in the system
- **Note:** The problems $\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} f_i(x)$ and $\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} f_i(x)$ are equivalent in the sense that their sets of optimal solutions coincide

- The lack of central authority is compensated by agent collaboration and communication
  - DeGroot consensus model [DeGroot 1974] - also referred to as agreement model
  - A variant of this problem, using consensus model, has been studied in the 80's: Borkar & Varaya 1982, Tsitsiklis 1984, Tsitsiklis, Bertsekas & Athens 1986, Bertsekas & Tsitsiklis book
    "Parallel and Distributed Computations: Numerical Methods" 1989

- We will discuss distributed gradient methods that employ DeGroot (or Push-sum) consensus protocols, which utilize:
  - Basic graph concepts (Laplacian and their spectral properties)
  - Row-stochastic and column stochastic matrices (properties of averaging, convergence) (tools from Nonnegative matrix and Markov Chain theory)
  - Optimization theory and techniques (gradient methods)

Graphs

- A graph over $m \geq 2$ nodes is denoted by $\mathbb{G} = ([m], \mathcal{E})$, where $[m] = \{1, 2, \ldots, m\}$ and $\mathcal{E} \subseteq [m] \times [m]$ is the set of edges.
- When a graph $\mathbb{G} = ([m], \mathcal{E})$ is undirected (bidirectional), the graph edges are specified by unordered pairs of distinct nodes $\{i, j\} \in \mathcal{E}$.
- When a graph $\mathbb{G} = ([m], \mathcal{E})$ is directed, the graph edges are specified by ordered pair of distinct nodes $(i, j) \in \mathcal{E}$.
- In what follows, graphs will be used to represent **the information flow** among a set of m agents (also referred to as nodes) communicating over a network with following interpretation of the graph edges:
  - An undirected edge (or a link) $\{i, j\}$ indicates that i can receive from and send information to j, and j can receive from and send the information to i;
  - A directed edge (or a link) (i, j) indicates that i can send information to agent j.
- An undirected graph $\mathbb{G}$ is **connected** if there is a path connecting every two distinct nodes in the graph.
- A directed graph $\mathbb{G}$ is **strongly connected** if there is a directed path connecting each node to every other node in the graph.

Neighbors in a Graph

- Given an undirected graph $\mathbb{G} = ([m], \mathcal{E})$, for each agent i, we let N_i be the set of neighbors of i,

$$N_i = \{j \in [m] \mid \{i, j\} \in \mathcal{E}\}$$

- Given a directed graph $\mathbb{G} = ([m], \mathcal{E})$, for each agent i, we identify two sets of neighbors:
  - The set of in-neighbors N_i^{in} of agent i

$$N_i^{\text{in}} = \{j \in [m] \mid (j, i) \in \mathcal{E}\}$$

  - The set of out-neighbors N_i^{out} of agent i

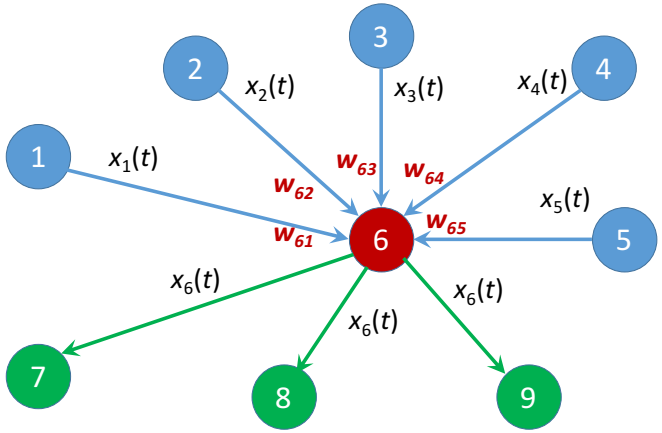
$$N_i^{\text{out}} = \{j \in [m] \mid (i, j) \in \mathcal{E}\}$$

- We will assume that **the graph $\mathbb{G}$ always contains self-loops at every node** (meaning that every agent i always has access to its own information)

- Under this assumption we have that i is always in its neighbor set/sets: i.e., for all i

DeGroot Consensus/Agreement Model

- Consider a set of m agents where every agent i has a value $x_i(0) \in \mathbb{R}$ (opinion).
- The graph representing who-knows-whom is a strongly connected directed graph $\mathbb{G} = ([m], \mathcal{E})$
- Agent i chooses positive trust weights $w_{ij} > 0$ for its in-neighbors $j \in N_i^{\text{in}}$; these weights sum to 1, $\sum_{j \in N_i^{\text{in}}} w_{ij} = 1$
- Over time, the agents communicate with their neighbors and share their values Specifically, at each time t, each agent $i \in [m]$
  - Receives values $x_j(t)$ from its in-neighbors N_i^{in} and
  - Sends its own value $x_i(t)$ to its out-neighbors N_i^{out}



- The *agents obtain no other information based on which they can update their values*
- Upon sharing their values, the agents update using the trust weights they have selected,

$$x_i(t+1) = \sum_{j \in N_i^{\text{in}}} w_{ij} x_j(t) \qquad \text{for all } i \in [m]$$

- To compactly write the evolution of opinions, define

$$w_{ij} = 0 \qquad \text{for all } j \notin N_i^{\text{in}} \text{ and for all } i \in [m]$$

and let $W = [w_{ij}]$. Define $x(t)$ as the column vector with entries $x_i(t)$, $i \in [m]$. Then, we have

$$x(t+1) = Wx(t) \qquad \text{for all } t \geq 0$$

- Thus, the evolution of $x(t)$ is linear

$$x(t) = W^t x(0) \qquad \text{for all} t \geq 0$$

- The trust matrix W is stochastic, i.e., it is a non-negative matrix and the entries sum to 1 in each row

$$W \geq 0, \qquad W\mathbf{1} = \mathbf{1}$$

where $\mathbf{1}$ is the m-dimensional vector with all entries equal to 1.

Existence and Characterization of the Limit

- The trust matrix W **is compliant with the directed graph** $\mathbb{G}$: there is an edge from j to i if and only if $W_{ij} > 0$.
- **We assume that the graph** $\mathbb{G}$ **is strongly connected**
- Analysis using Markov Chain theory
  - If we view W as a transition matrix of a homogeneous Markov Chain, then the chain is ergodic, meaning that

$$\lim_{t\to\infty} W^t = \mathbf{1}\pi',$$

    where $\pi = [\pi_1, \ldots, \pi_m]'$ is a positive stochastic vector, i.e., $\pi > 0$ and $\mathbf{1}'\pi = 1$.
  - The vector π is the vector of steady-state distributions of the chain.
  - Using this limit in the evolution of $x(t)$, we conclude that

$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} W^t x(0) = 1\pi' x(0)$$

    which implies

$$\lim_{t\to\infty} x_i(t) = \pi' x(0) \qquad \text{for all } i \in [m]$$

- Thus, agents reach a consensus asymptotically
- The consensus value is $\pi' x(0)$

- The convergence rate can be assessed using reversible Markov Chains[‡]
- Alternatively, use the non-negative matrix theory[§]
- When the graph $\mathbb{G}$ is strongly connected, by Perron-Frobenius Theorem:
  - The vector $\mathbf{1}$ is the unique right-eigenvector of W associated with the eigenvalue 1; all the other eigenvalues of W are less than 1 in modulus
  - There exists a unique stochastic vector $\pi > 0$ which is the left-eigenvector of W associated with the eigenvalue 1: $\pi' W = \pi', \qquad \pi > 0, \qquad \mathbf{1}'\pi = 1$
- The fact that 1 is the largest in modulus of all eigenvalues of W leads to

$$\sum_{i=1}^{m} \pi_i (x_i(t+1) - \langle \pi, x(0) \rangle)^2 \leq \rho_W \sum_{i=1}^{m} \pi_i (x_i(t) - \langle \pi, x(0) \rangle)^2$$

  where $\rho_W \in (0, 1)$ is the second largest (in modulus) eigenvalue of W[¶]
- Recently, using the properties of weighted-averaging and the graph $\mathbb{G}$ structure, we have shown alternative bound that is explicit in terms of the graph structure[‖]

[‡] P. Brémaud *Gibbs Fields, Monte Carlo Simulation, and Queues* New York, USA: Springer-Verlag, 1999

[§] E. Seneta *Nonnegative Matrices*, M. Fiedler *Special Matrices and their Applications in Numerical Mathematics*

[¶] R. Xin, K. Sahu, U.A. Khan, S. Kar, Distributed stochastic optimization with gradient tracking over strongly-connected networks, IEEE CDC 2019

[‖] see https://arxiv.org/abs/2201.02323, Lemma 6

Consensus Protocol: Optimization Point of View

- Reaching agreement means that decisions of all agents are the same: *Feasibility problem of finding an* $x \in \mathbb{R}^n$ *such that*: $x_i = x$ for all $i \in [m]$
- **When an underlying information-flow (directed strongly connected) graph** $\mathbb{G} = ([m], \mathcal{E})$ is given, the above problem is equivalent to**

$$x_j = x_i \qquad \text{for all } j \in N_i^{\text{in}} \text{ and all } i \in [m]$$

- Using a non-negative matrix W (compliant with the graph structure), where agent i decides on i-row of W translates to the following equivalent problem††

$$\sum_{j=1}^{m} w_{ij} x_j = \left(\sum_{j=1}^{m} w_{ij} \right) x_i \qquad \text{for all } i \in [m]$$

- When the weights sum to 1 (W is row stochastic), the problem is equivalent to

$$\sum_{j=1}^{m} w_{ij} x_j = x_i \qquad \text{for all } i \in [m]$$

- Introducing the matrix $\mathbf{x}$ with rows given by x_i', we have following equivalent feasibility problem: $W\mathbf{x} = \mathbf{x}$

**We can also have an equivalent formulation by using the out-neighbors N_i^{out}

††Recall that $i \in N_i^{\text{in}}$ for all $i \in [m]$ (self-loops) and $w_{ij} > 0$ only when $j \in N_i^{\text{in}}$

- **Consensus protocol solves the above feasibility problem distributedly!**
- **When the underlying graph $\mathbb{G}$ is directed and strongly connected**, the solutions of the preceding feasibility problem are of the form $x^* = a\mathbf{1}$ for some scalar $a \in \mathbb{R}^m$
- **Alternative approach: cast the consensus problem as an equivalent feasibility problem for a system of equations using the Laplacian of the graph or a weighted-graph Laplacian** (the same as above except w_{ii} is defined differently)
- Time-Varying Graphs
  - At communication round t, the connectivity graph is $\mathbb{G}_t = ([m], \mathcal{E}_t)$
  - The agents use time-varying row-stochastic matrices W_t

$$[W_t]_{ij} > 0 \qquad \text{when } j \in N_{it}^{\text{in}} \text{ (the set of in-neighbors of } i \text{ in } \mathbb{G}_t)$$

  - At time t, the agents face feasibility problem $W_t\mathbf{x} = \mathbf{x}$
- The consensus method evolution equation (non-stationary process):

$$\mathbf{x}(t+1) = W_t W_{t-1} \cdots W_0 \mathbf{x}(0) \qquad \text{for all } t \geq 0$$

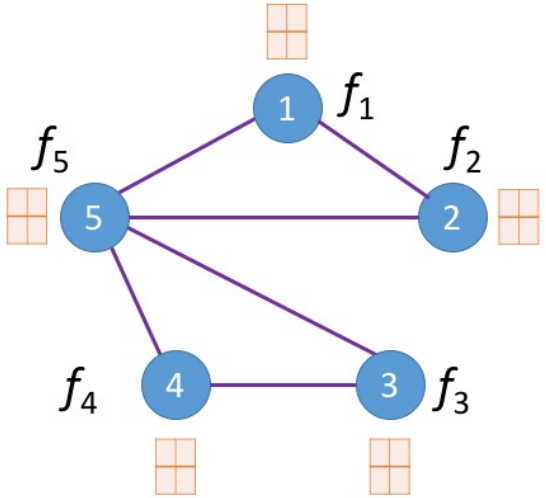
- We can resort to theory on inhomogeneous Markov Chains or backward products of non-negative matrices
- Assuming that each graph is strongly connected, will this work?
- Yes! The key observation is that **each feasibility problem $W_t\mathbf{x} = \mathbf{x}$ has the same solution set $\{\alpha\mathbf{1} \mid a \in \mathbb{R}\}$ for all $t \geq 0$**.

Using DeGroot Consensus In Distributed Optimization

We consider our problem

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} f_i(x)$$

in a system consisting of m agents that are embedded in a communication network.



- Function $f_i(\cdot)$ is *privately known* only to agent i.
- Agents communicate some limited information with their immediate neighbors only
- Agents do not share their functions $f_i(\cdot)$'s.
- The problem is to be solved *distributedly* i.e., **without a central entity**
- The lack of central authority is compensated by using *DeGroot consensus model to act as a virtual coordinator*.
- Lets assume that the underlying graph $\mathbb{G} = ([m], \mathcal{E})$ is undirected and connected

Consensus-Based Method for Optimization

- The agents communicate over a graph, and N_i is the set of neighbors of agent i
- At time t, every agent i sends $x_i(t)$ to its neighbors $j \in N_i$, and receives $x_j(t)$ from them; then, every agent updates (AN and A. Ozdaglar 2009)

$$x_i(t+1) = \underbrace{\sum_{j=1}^{m} w_{ij} x_j(t)}_{\text{consensus}} - \alpha_t \nabla f_i(x_i(t)) \quad \text{where } \alpha_t > 0 \text{ is a stepsize}$$

- It can be viewed as an extension of the DeGroot model where agents have additional side information that guides their consensus point
- Assuming that the problem has a solution and some other conditions, each agent decision $x_i(t)$ converges to a common optimal solution x^* of the system problem,

$$\lim_{t \to \infty} x_i(t) = x^* \qquad \text{for all } i,$$

where x^* is a minimizer of $\sum_{j=1}^{m} f_j(x)$ over $x \in \mathbb{R}^n$.

- *The method can work correctly as long as*: *every agent is equally influential*
- Recall that DeGroot protocol leads to consensus vector $\pi'\mathbf{x}(0)$ where $\pi > 0$, $\pi' W = \pi'$
- The analysis of the optimization reveals that the agents will solve the problem $\min_{x\in\mathbb{R}^n} \sum_{i=1}^m \pi_i f_i(x)$
- Thus, to control the agent influence vector π, the trust matrix W is often assumed to be doubly stochastic resulting in equal agent influence, $\pi_i = \frac{1}{m}$ for all i.
- Such a matrix can be constructed using Metropolis-Hastings weights

$$w_{ij} = \begin{cases} \frac{1}{1+\max\{d_i,d_j\}} & \text{if } \{i,j\} \in \mathcal{E},\ j \neq i \\ 1 - \sum_{j\in N_i, i\neq j} w_{ij} & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

where d_i is the degree of the node i in the graph $\mathbb{G}$ (not counting the self-loop). Extends to time-varying case: L. Xiao, S. Boyd, S. Lall, Distributed Average Consensus with Time-Varying Metropolis Weights, 2006

- There are other approaches that similarly require extra information exchange at each round to create a symmetric row-stochastic matrix W_t (AN, A. Ozdaglar, Distributed subgradient methods for multi-agent optimization, 2009)
- It works as long as the graphs are undirected

Eliminating Influential Bias: Alternative Consensus

- **The algorithm cannot be efficiently implemented in directed time-varying graphs**, i.e., the construction of doubly-stochastic W in a directed graph is time consuming: Gharesifard and Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," European Journal of Control, 18 (6), 539-557, 2012

- An alternative via $push\text{-}sum\ algorithm$ for consensus:
  D. Kempe, A. Dobra, and J. Gehrke Gossip-based computation of aggregate information, In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pages 482–491, 2003
  F. Benezit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli Weighted gossip: distributed averaging using non-doubly stochastic matrices, In Proceedings of the 2010 IEEE International Symposium on Information Theory, 2010

Push-sum and Optimization methods

- Dominguez-Garcia and Hadjicostis. Distributed strategies for average consensus in directed graphs. In Proceedings of the IEEE Conference on Decision and Control, Dec 2011.
- Hadjicostis, Dominguez-Garcia, and Vaidya, "Resilient Average Consensus in the Presence of Heterogeneous Packet Dropping Links" CDC, 2012
- Tsianos and Rabbat. Distributed consensus and optimization under communication delays. In Proc. of Allerton Conference on Communication, Control, and Computing, 2011.
- Tsianos, Lawlor, and Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In Proceedings of the 50th Allerton Conference on Communication, Control, and Computing, 2012.
- Tsianos, Lawlor, and Rabbat. Push-sum distributed dual averaging for convex optimization. In Proceedings of the IEEE Conference on Decision and Control, 2012.
- Tsianos. The role of the Network in Distributed Optimization Algorithms: Convergence Rates, Scalability, Communication / Computation Tradeoffs and Communication Delays. PhD thesis, McGill University, Dept. of Electrical and Computer Engineering, 2013.

Push-sum: Column Stochastic Matrix

- Given a directed and strongly connected graph $\mathbb{G} = ([m], \mathcal{E})$, let C be a matrix compatible with the graph

$$C_{ij} > 0 \qquad \text{when } (j,i) \in \mathcal{E}, \qquad\qquad C_{ij} = 0 \qquad \text{when } (j,i) \notin \mathcal{E}$$

- Assume that C has positive diagonal entries

- Also, let C be a column-stochastic matrix

$$\mathbf{1}'C = \mathbf{1}'$$

- Then $\lim_{t\to\infty} C^t = \phi\mathbf{1}'$ where ϕ is a stochastic vector with $\phi_i > 0$ for all i

- Consider a process

$$x(t) = Cx(t-1) \qquad \text{for } t \geq 1$$

with an arbitrary $x(0) \in \mathbb{R}^n$

- Then

$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} C^t x(0) = \phi \mathbf{1}' x(0) = \langle \mathbf{1}, x(0) \rangle \, \phi$$

- Repeating this process with a different initial point $y(0)$, we obtain

$$y(t) = Cy(t-1) \qquad \text{for } t \geq 1$$
$$\lim_{t\to\infty} y(t) = \langle \mathbf{1}, y(0) \rangle \, \phi$$

- Look at the coordinate-wise ratio

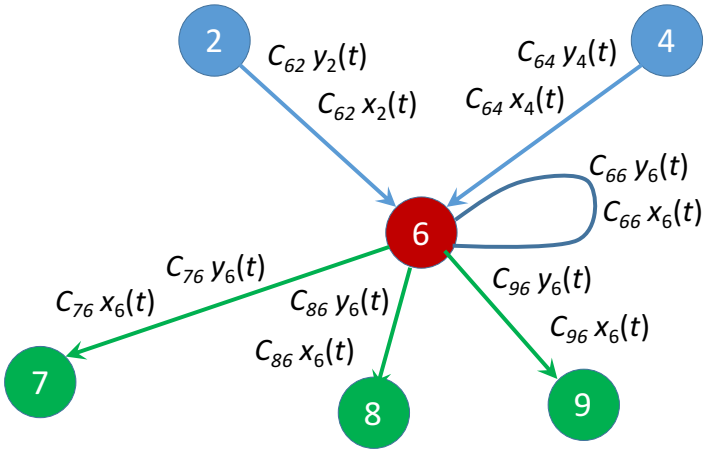
$$z_i(t) = \frac{x_i(t)}{y_i(t)}, \qquad \lim_{t\to\infty} z_i(t) = \frac{\langle \mathbf{1}, x(0) \rangle \, \phi_i}{\langle \mathbf{1}, y(0) \rangle \, \phi_i} = \frac{\langle \mathbf{1}, x(0) \rangle}{\langle \mathbf{1}, y(0) \rangle}$$

- If we want

$$\lim_{t\to\infty} z_i(t) = \frac{1}{m} \langle \mathbf{1}, x(0) \rangle$$

it can be done by choosing the initial values $y_i(0) = 1$ for all $i \in [m]$

Push-sum Protocol Illustration



- Agent i decides on the values C_{ji} for its out-neighbors $j \in N_i^{\text{out}}$ (see the plot) and sends $C_{ji}x_i(t)$ and $C_{ji}y_i(t)$ (the ith column of C sums to 1)

- Agent i receives such values from its in-neighbors and updates:

$$x_i(t+1) = \sum_{j \in N_i^{\text{in}} \cup \{i\}} C_{ij} x_j(t),$$

$$y_i(t+1) = \sum_{j \in N_i^{\text{in}} \cup \{i\}} C_{ij} y_j(t)$$

$$z_i(t+1) = \frac{x_i(t+1)}{y_i(t+1)}$$

- The x-variables can be vectors, while y-variables are always scalars

Optimization

- **The gradient-push method can be used for minimizing $\sum_{i=1}^{m} f_i(x)$ over $x \in \mathbb{R}^n$**
- Every node i maintains vectors $x_i(t), w_i(t)$ in $\mathbb{R}^n$, and an auxiliary scalar variable $y_i(t)$, initialized with $y_i(0) = 1$ for all i.
- At tome $t+1$:
  - **Communication**: Each node j sends $C_{\ell j}x_j(t)$, $C_{\ell j}y_j(t)$ to out-neighbors $\ell \in N_j^{\text{out}}$
  - **Computation**: Upon receiving these quantities, every node updates:

$$
\begin{aligned}
w_i(t+1) &= \sum_{j \in N_i^{\text{in}}(t) \cup \{i\}} C_{ij} x_j(t) \\
y_i(t+1) &= \sum_{j \in N_i^{\text{in}}(t) \cup \{i\}} C_{ij} y_j(t), \\
z_i(t+1) &= \frac{w_i(t+1)}{y_i(t+1)}, \\
x_i(t+1) &= w_i(t+1) - \alpha(t+1)\nabla f_i(z_i(t+1)), \qquad (4)
\end{aligned}
$$

- The method is initiated with an arbitrary $x_i(0)$ and $y_i(0) = 1$ for all i.
  The stepsize $\alpha(t+1) > 0$ satisfies the following decay conditions $\sum_{t=1}^{\infty} \alpha(t) = \infty$ and $\sum_{t=1}^{\infty} \alpha^2(t) < \infty$

- Under this stepsize (and B-uniform strong connectivity), the algorithm produces the iterates that converge to a **consensual** minimizer of $\sum_{i=1}^{m} f_i(z)$ over $z \in \mathbb{R}^n$.
  - Convergence rate is of the order of $O(1/\sqrt{t})$ for convex functions and $O(1/t)$ for strongly convex functions (AN and Olshevsky *Distributed Optimization over Time-varying Directed Graphs* IEEE TAC, 2015; AN and Olshevsky *Stochastic Gradient-Push for Strongly Convex Functions on Time-Varying Directed Graphs* 2017 Tatarenko and Touri 2015 –Non-Convex Distributed Optimization)
  - For the protocol to work, every agent must know its out-neighbors - may not be realistic in time-varying case
  - Numerical instabilities may occur when $y_i(t)$ is too small

- **Neither De-Groot nor Push-sum based gradient methods can achieve geometric (linear) convergence rate!**

Achieving Geometric Rate: Gradient Tracking

- Lets get back to undirected graph $\mathbb{G} = ([m], \mathcal{E})$

- In weighted-average consensus-based distributed method, the agents were selfish (applies to the push-sum-based method as well)

$$x_i(t+1) = \underbrace{\sum_{j=1}^{m} w_{ij} x_j(t)}_{\text{collaborative}} - \alpha \underbrace{\nabla f_i(x_i(t))}_{\text{selfish}}$$

  where we use a fixed stepsize

- In the models with gradient tracking, the agents are "aware" that there is a system objective and they collaborate on both the decisions and the directions

- **Basic Idea**: In DeGroot consensus model, with W doubly stochastic agent i iterate $x_i(t+1) = \sum_{j=1}^{m} w_{ij} x_j(t)$ tracks the average of the agents' iterates $x_j(t)$, $j \in [m]$

- The iterate $\sum_{j=1}^{m} w_{ij} x_j(t)$ is sufficient to properly track the averages $(1/m) \sum_{j=1}^{m} x_j(t)$ since the agent use no additional information (no other inputs in the system)

- Apply the same idea to gradients: **DIGing – Distributed Inexact Gradient track-ing**
  Each agent uses an estimate $g_i(t)$ to track the gradient averages of all the agents

$$x_i(t+1) = \sum_{j=1}^{m} w_{ij} x_j(t) - \alpha g_i(t)$$

$$g_i(t+1) = \sum_{j=1}^{m} w_{ij} g_j(t) + \underbrace{\nabla f_i(x_i(t+1)) - \nabla f_i(x_i(t))}_{\text{innovation/new input}}$$

- Agents exchange both decision estimates $x_j(t)$ and the gradient estimates $g_j(t)$ with their neighbors

- The updates are reminiscent of "tracking/filtering":
  predicted state + the innovation term

- The innovation term is needed to "track gradients" since the gradient difference is a "new information/new input" to the system from agent i.

- Through the exchange of $g_i(t)$ and the consensus step $\sum_{j=1}^{m} w_{ij} g_j(t)$, these local agent inputs (from times prior to t) are eventually spread to all agents in the graph

Gradient-Tracking Literature

- Tracking technique used in (not for gradients)
  M. Zhu and S. Martínez, *Discrete-Time Dynamic Average Consensus*, Automatica, 46 (2010),

- A method using gradient tracking proposed in
  J. Xu, S. Zhu, Y. Soh, and L. Xie, *Augmented Distributed Gradient Methods for Multi-Agent Optimization Under Uncoordinated Constant Stepsizes*, in Proceedings of the 54th IEEE Conference on Decision and Control (CDC), 2015, pp. 2055–2060.

- A part of Xu's thesis work
  J. Xu, *Augmented Distributed Optimization for Networked Systems*, PhD thesis, Nanyang Technological University, 2016.

- G. Qu and N. Li, *Harnessing Smoothness to Accelerate Distributed Optimization*, IEEE Transactions on Control of Network Systems 5 (3) 1245–1260, 2018.

Algorithms NEXT and SONATA

- NEXT by Lorenzo and Scutari - considers general non-convex (objective) problems and a class of algorithms

  P. Di Lorenzo and G. Scutari *Distributed nonconvex optimization over networks*, in IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015, pp. 229–232.

  P. Di Lorenzo and G. Scutari, *NEXT: In-Network Nonconvex Optimization*, IEEE Transactions on Signal and Information Processing over Networks, 2016.

  P. Di Lorenzo and G. Scutari *Distributed nonconvex optimization over time-varying networks*, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 4124–4128.

- SONATA and its asynchronous variants; convex and nonconvex problems

  Y.Sun, G. Scutari, D.Palomar *Distributed Nonconvex Multiagent Optimization Over Time-Varying Networks* https://arxiv.org/abs/1607.00249, 2016

  Y. Tian, Y. Sun, B. Du, G. Scutari *ASY-SONATA: Achieving Geometric Convergence for Distributed Asynchronous Optimization* Allerton Conference on Communication,

Control, and Computing (Allerton) 2018
Y. Sun, A. Daneshmand, G. Scutari *Convergence Rate of Distributed Optimization Algorithms Based on Gradient Tracking* https://arxiv.org/abs/1905.02637, 2019

- Our motivation was to have a distributed algorithm with a geometric convergence rate

  $\|x_i(t) - x^*\| \leq q^t M, \qquad$ for some $M > 0$, $q \in (0, 1)$, and for all agents $i \in [m]$.

- It was developed in: A.N., A. Olshevsky and W. Shi, Achieving Linear Convergence For Distributed Optimization Over Deterministic Time-Varying Graphs," SIAM Journal on Optimization 27 (4) 2597–2633, 2017

- Works for undirected graphs :(

Push-Pull Method[‡‡]

- Works on both undirected and directed graphs, but **static** i.e., $\mathbb{G} = ([m], \mathcal{E})$.
- It is a variant of DIGing that uses different matrices for mixing the decisions and the directions
- **Exchange**: (from an agent's perspective
  - (**Pull**) Every agent i receives $x_j(k) - \alpha g_j(k)$ from its in-neighbors $j \in N_i^{\text{in}}$
  - (**Push**) Every agent i sends $C_{\ell i} g_i(k)$ to all its out-neighbors $\ell \in N_i^{\text{out}}$
- **Update**: Every agent i updates its decision x and direction g as follows

$$x_i(k+1) = \sum_{j=1}^m R_{ij} \left(x_j(k) - \alpha g_j(k)\right);$$
$$g_i(k+1) = \sum_{j=1}^m C_{ij} g_j(k) + \nabla f_i(x_i(k+1)) - \nabla f_i(x_i(k)).$$

The matrix R is row-stochastic, while C is a column stochastic!!!

$r_{ij} = 0$ if $j \notin \mathcal{N}_i^{\text{in}}$ and $c_{ij} = 0$ if $j \notin \mathcal{N}_i^{\text{out}}$.

The method is initialized with arbitrary $x_i(0) \in \mathbb{R}^n$ and $g_i(0) = \nabla f_i(x_i(0))$ for all i.

The stepsize α can be agent dependent.

[‡‡] S. Pu, W. Shi, J. Xu, A. N. "Push-Pull Gradient Methods for Distributed Optimization in Networks," IEEE TAC 2021

Related Work

- S. Pu, W. Shi, J. Xu, and A. Nedić, *A push-pull gradient method for distributed optimization in networks*, Proceedings of the 54th IEEE Conference on Decision and Control (CDC), 2018; journal version on arxiv: https://arxiv.org/abs/1810.06653
- C. Xi, V. S. Mai, R. Xin, E. H. Abed, and U. A. Khan, *Linear convergence in optimization over directed graphs with row-stochastic matrices*, IEEE Transactions on Automatic Control, 2018.
- R. Xin, C. Xi, and U. A. Khan, *Frost–fast row-stochastic optimization with uncoordinated step-sizes*, EURASIP Journal on Advances in Signal Processing, 2019.
- R. Xin and U. A. Khan, *A linear algorithm for optimization over directed graphs with geometric convergence*, arXiv preprint arXiv:1803.02503, 2018; IEEE Control Systems Letters 2 (3) 315 – 320, 2018.
- The method has been recently extended to time-varying graphs: F. Saadatniaki, R. Xin, and U. A. Khan, "Decentralized optimization over time-varying directed graphs with row and column-stochastic matrices," IEEE Transactions on Automatic Control, 2020.

Landscape

- Fast distributed gradient methods are developed that can match the best performance of centralized gradient methods
- New directions
  - Nonconvex problems (T. Tatarenko & B. Touri 2017, Gesualdo Scutari's group at Purdue, Khan's group at Tufts U)
  - Asynchronous implementations (S. Pu, Scutari's group)
  - Impact of network topology (N. Neglia at INRIA, A. Olshevsky at BU)
  - Impact of delays (M. Johansson at KTH, M.G. Rabbat at Facebook/McGill)
  - Privacy (Y. Wang at Clemson University)
  - Presence of malicious agents (S. Sundaram, N. Vaidya, A. Scaglione, W.U. Bajwa, S. Gil, A. Goldsmith, AN)