A Fast Distributed Asynchronous Newton-Based Optimization Algorithm

Ermin Wei

Joint work with Fatemeh (Samira) Mansoori

Electrical Engineering and Computer Science
Northwestern University

Midwest ML Symposium (MMLS)
Chicago
June 7, 2018
Sensor Network Example

- A network of 3 sensors, supervised passive learning.
- Data is collected at different sensors: input $t$, output $d$. 
Sensor Network Example

- A network of 3 sensors, supervised passive learning.
- Data is collected at different sensors: input $t$, output $d$. 

![Least square fit with polynomial max degree 3](image)
Sensor Network Example

- A network of 3 sensors, supervised passive learning.
- Data is collected at different sensors: input $t$, output $d$.

- System goal: a 3rd-degree polynomial model:

$$d(t) = x_3 t^3 + x_2 t^2 + x_1 t + x_0.$$  

- System objective:

$$\min_x \sum_{i=1}^3 \|A_i'x - d_i\|^2_2.$$  

where $A_i = [1, t_i, t_i^2, t_i^3]'$ at input data $t_i$.  

![Least square fit with polynomial max degree 3](image-url)
Regularized Empirical Loss Minimization Set-up

- System objective: train weight vector $x$ to

$$\min_x \sum_{i=1}^{n-1} L_i(x) + p(x),$$

for some loss function $L$ (on the prediction error) and penalty function $p$ (on the complexity of the model).

- **Example**: Least-Absolute Shrinkage and Selection Operator (LASSO):

$$\min_x \sum_{i=1}^{n-1} ||A_i'x - b_i||^2_2 + \lambda ||x||_1.$$

- Other examples from ML estimation, low rank matrix completion, image recovery [Schizas, Ribeiro, Giannakis 08], [Recht, Fazel, Parrilo 10], [Steidl, Teuber, 10]
Distributed Multi-agent Optimization

- Connected undirected network of $n$ cooperative agent to solve

\[
\min_x \sum_{i=1}^{n} f_i(x).
\]

- Each function $f_i$ is only locally available to agent $i$.

- **Distributed algorithm**: each agent performing computations locally and communicating only to neighbors.\(^1\)

- Introduce local copy $x_i$ for each agent and reformulation to distributed setup

\[
\min_x \sum_{i=1}^{n} f_i(x_i)
\]

\[s.t. \ x_i = x_j, \ \text{for} \ (i, j) \in E.\]

\(^1\)This talk will focus on the case where $x$ is in $\mathbb{R}$. The results generalize to $\mathbb{R}^n$. 
Other Motivations

- Many networks are large-scale, with agents with local information and heterogeneous preferences: call for distributed optimization.
- Most existing distributed optimization algorithms require a central clock.
- This motivated development of asynchronous distributed schemes for control and optimization of multi-agent networked systems.
Distributed Multi-agent Optimization

- Connected undirected network of \( n \) cooperative agent to solve

\[
\min_x \sum_{i=1}^n f_i(x).
\]

- Each function \( f_i \) is only locally available to agent \( i \).

- **Distributed algorithm**: each agent performing computations locally and communicating only to neighbors.\(^2\)

- Introduce local copy \( x_i \) for each agent and reformulation to distributed setup

\[
\min_x \sum_{i=1}^n f_i(x_i)
\]

\[\text{s.t. } x_i = x_j, \text{ for } (i, j) \in E.\]

\(^2\)This talk will focus on the case where \( x \) is in \( \mathbb{R} \). The results generalize to \( \mathbb{R}^n \).
Consensus Based Algorithms

- Each agent has a component of the system objective function and aims to minimize its local cost function.
- Each agent tries to keep its variable equal to those of neighboring agents.
- The basic distributed gradient descent algorithm: each iterate \((k)\), each agent takes a (sub)gradient step and averages with neighbors.

\[
x_i(k+1) = \sum_{j=1}^{n} W_{ij}(k)x_j(k) - s(k)d_i(k),
\]

where \(W_{ij} \geq 0\) weights, \(s(k) > 0\) stepsize, \(d_i(k)\) subgradient of \(f_i\) at \(x_i(k)\).
Standard ADMM solves a separable problem, where decision variable decomposes into two (linearly coupled) variables:

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

Consider an Augmented Lagrangian function:

$$L_\beta(x, y, p) = f(x) + g(y) - p'(Ax + By - c) + \frac{\beta}{2} \|Ax + By - c\|_2^2.$$ 

$$x^{k+1} = \operatorname{argmin}_x \quad L_\beta(x, y^k, p^k),$$
$$y^{k+1} = \operatorname{argmin}_y \quad L_\beta(x^{k+1}, y, p^k),$$
$$p^{k+1} = p^k - \beta(Ax^{k+1} - By^{k+1} - c).$$
Literature: Synchronous Distributed Optimization Algorithms

- **Primal Methods:**
  - Consensus and Distributed Gradient Descent Algorithms [Bertsekas, Tsitsiklis 89], [Jadbabaie, Lin, Morse 03], [Blondel, Hendrickx, Olshevsky, Tsitsiklis 05], [Nedić, Ozdaglar 09], [Yuan, Ling, Yin 13], [Jakovetic, Xavier, Moura 14], [Shi, Ling, Wu, Yin 14]

- **Dual Methods:**
  - Distributed dual averaging and ADMM Algorithms [Bertsekas, Tsitsiklis 89], [Boyd, Parikh, Chu, Peleato, Eckstein 11], [Duchi, Agarwal, Wainwright 12], [Wei, Ozdaglar 12]

- **Newton-based Methods:**
  - For network flow related problems [Jadbabaie, Ozdaglar, Zargham 09], [Wei, Ozdaglar, Jadbabaie 10], [Liu, Sherali 12]
  - For sum minimization problems: Network Newton method [Mokhtari, Ling, Ribeiro, 15]
Asynchronous vs. Synchronous Algorithms

- **Synchronous Algorithms**
  - Agents will need to have access to a central coordinator/clock.
  - They must wait for the slowest to finish before proceeding to the next iteration.

- **Asynchronous Algorithms**
  - Agents become active randomly in time and update using delayed/partial and local information.
  - There is no need for a central coordinator.
  - Partially asynchronous: bounded delay.
  - Totally asynchronous: infinitely often update for each agent.
Literature: Asynchronous Distributed Optimization Algorithms

- **Primal Methods:**
  - Gossip-Based Algorithms [Boyd, Ghosh, Prabhakar, Shah 06], [Ram, Nedić, Veeravalli 09]
  - Broadcast-Based Algorithms [Nedić 11]
  - Coordinate-Based Algorithms [Peng, Xu, Yan, Yin 16], [Hannah, Yin 16]

- **Primal-Dual Methods:**
  - Coordinate Descent Algorithms [Binachi, Hachem, Iutzeler 15]
  - Asynchronous Distributed ADMM [Wei, Ozdeğlar 13], [Chang, Hong, Liao, Wang 16]

- **Quasi Newton Methods:** [Eisen, Mokhtari, Ribeiro 16], [Bajovic, Jakovetic, Krejic, Jerinkic 17]

- These algorithms can be shown to achieve:
  - Totally asynchronous: sublinear linear rates (linear if similar delays).
We present an asynchronous distributed network Newton algorithm for multi-agent optimization with “nice” objective functions.

- **Asynchronous algorithm and distributed implementation**
  - Hessian approximation method
  - Algorithm and implementation

- **Convergence analysis**
  - Almost sure (global) convergence
  - Global linear convergence
  - Local *quadratic* convergence
Problem Formulation

We will consider minimizing the penalized version of the consensus problem over a connected undirected static network of \(n\) agents. For \(x = [x_i]_i\),

\[
\min_x F(x) = \frac{1}{2} x^T (I - W)x + \alpha \sum_{i=1}^{n} f_i(x_i).
\]

- **“Nice”**: The local objective functions \(f_i(x)\) are convex, twice continuously differentiable. Hessian matrices have bounded eigenvalues and are L-Lipschitz continuous.
- Each agent \(i\) knows \(f_i\) and updates \(x_i\).
- Matrix \(W\) is a symmetric doubly stochastic consensus/weight matrix: for all \(i\)

\[
W = W', \quad \sum_{j=1}^{n} W_{ij} = 1, \quad 0 < W_{ii} < 1.
\]

- Matrix \(W\) represents the network topology: \(W_{ij} \neq 0\) \iff \((i, j) \in E\) for \(i \neq j\).
- Each agent knows local positive weights \(W_{ij}\) for \(j\) in \(\mathcal{N}_i\) (neighbors of \(i\)).
Problem Formulation

We will consider minimizing the penalized version of the consensus problem over a connected undirected static network of $n$ agents. For $x = [x_i]_i$,

$$
\min_x F(x) = \frac{1}{2} x^T (I - W)x + \alpha \sum_{i=1}^{n} f_i(x_i).
$$

- **“Nice”:** The local objective functions $f_i(x)$ are convex, twice continuously differentiable. Hessian matrices have bounded eigenvalues and are L-Lipschitz continuous.
- Each agent $i$ knows $f_i$ and updates $x_i$.
- Matrix $W$ is a symmetric doubly stochastic consensus/weight matrix: for all $i$

$$
W = W', \quad \sum_{j=1}^{n} W_{ij} = 1, \quad 0 < W_{ii} < 1.
$$

- Matrix $W$ represents the network topology: $W_{ij} \neq 0 \iff (i, j) \in E$ for $i \neq j$.
- Each agent knows local positive weights $W_{ij}$ for $j$ in $\mathcal{N}_i$ (neighbors of $i$).
Problem Formulation

We will consider minimizing the penalized version of the consensus problem over a connected undirected static network of $n$ agents. For $x = [x_i]_i$,

$$\min_x F(x) = \frac{1}{2} x^T (I - W)x + \alpha \sum_{i=1}^n f_i(x_i).$$

- “Nice”: The local objective functions $f_i(x)$ are convex, twice continuously differentiable. Hessian matrices have bounded eigenvalues and are L-Lipschitz continuous.
- Each agent $i$ knows $f_i$ and updates $x_i$.
- Matrix $W$ is a symmetric doubly stochastic consensus/weight matrix: for all $i$

$$W = W', \quad \sum_{j=1}^n W_{ij} = 1, \quad 0 < W_{ii} < 1.$$

- Matrix $W$ represents the network topology: $W_{ij} \neq 0$ iff $(i,j) \in E$ for $i \neq j$.
- Each agent knows local positive weights $W_{ij}$ for $j$ in $\mathcal{N}_i$ (neighbors of $i$).
Problem Formulation

We will consider minimizing the penalized version of the consensus problem over a connected undirected static network of $n$ agents. For $x = [x_i]_i$,

$$
\min_x F(x) = \frac{1}{2} x^T (I - W)x + \alpha \sum_{i=1}^{n} f_i(x_i).
$$

- **“Nice”:** The local objective functions $f_i(x)$ are convex, twice continuously differentiable. Hessian matrices have bounded eigenvalues and are L-Lipschitz continuous.
- Each agent $i$ knows $f_i$ and updates $x_i$.
- Matrix $W$ is a symmetric doubly stochastic consensus/weight matrix: for all $i$

$$
W = W', \quad \sum_{j=1}^{n} W_{ij} = 1, \quad 0 < W_{ii} < 1.
$$

- Matrix $W$ represents the network topology: $W_{ij} \neq 0$ iff $(i, j) \in E$ for $i \neq j$.
- Each agent knows local positive weights $W_{ij}$ for $j$ in $\mathcal{N}_i$ (neighbors of $i$).
Problem Formulation

We will consider minimizing the penalized version of the consensus problem over a connected undirected static network of \(n\) agents. For \(x = [x_i]_i\),

\[
\min_x F(x) = \frac{1}{2} x^T (I - W)x + \alpha \sum_{i=1}^{n} f_i(x_i).
\]

- **“Nice”**: The local objective functions \(f_i(x)\) are convex, twice continuously differentiable. Hessian matrices have bounded eigenvalues and are L-Lipschitz continuous.
- Each agent \(i\) knows \(f_i\) and updates \(x_i\).
- Matrix \(W\) is a symmetric doubly stochastic consensus/weight matrix: for all \(i\)

\[
W = W', \quad \sum_{j=1}^{n} W_{ij} = 1, \quad 0 < W_{ii} < 1.
\]

- Matrix \(W\) represents the network topology: \(W_{ij} \neq 0 \iff (i, j) \in E\) for \(i \neq j\).
- Each agent knows local positive weights \(W_{ij}\) for \(j\) in \(N_i\) (neighbors of \(i\)).
Our asynchronous method is based on Newton’s algorithm for unconstrained problem with iteration

\[ x(t + 1) = x(t) + \varepsilon d(t), \]

- \( \varepsilon \) is some positive stepsize and \( d(t) \) is the Newton direction.

The Newton’s direction is \( d(t) = H(t)^{-1}g(t) \) with

\[ H(t) = \nabla^2 F(x(t)) = I - W + \alpha G(t), \quad \text{where} \quad G_{ii}(t) = \nabla^2 f_i(x_i(t)). \]

\[ g_i(t) = \nabla_i F(x(t)) = [(I - W)x(t)]_i + \alpha \nabla f_i(x_i(t)). \]

The Hessian inverse cannot be computed in a distributed way directly.

Asynchronous network Newton uses the matrix splitting techniques and truncated Taylor expansion to approximate the Hessian inverse in a distributed manner.
Our asynchronous method is based on Newton’s algorithm for unconstrained problem with iteration

\[ x(t + 1) = x(t) + \varepsilon d(t), \]

- \( \varepsilon \) is some positive stepsize and \( d(t) \) is the Newton direction.
- The Newton’s direction is \( d(t) = H(t)^{-1}g(t) \) with
  \[
  H(t) = \nabla^2 F(x(t)) = I - W + \alpha G(t), \quad \text{where} \quad G_{ii}(t) = \nabla^2 f_i(x_i(t)).
  \]
  \[
  g_i(t) = \nabla_i F(x(t)) = [(I - W)x(t)]_i + \alpha \nabla f_i(x_i(t)).
  \]

The Hessian inverse cannot be computed in a distributed way directly.

Asynchronous network Newton uses the matrix splitting techniques and truncated Taylor expansion to approximate the Hessian inverse in a distributed manner.
Asynchronous network Newton

Background on Hessian Approximation

- Hessian matrix can be splitted as $H(t) = I - W + \alpha G(t) = D(t) - B$ with
  $$D(t) = \alpha G(t) + 2(I - W_d), \quad B = I - 2W_d + W$$
  where $W_d$ is a diagonal matrix with $[W_d]_{ii} = W_{ii}$.
- $D(t)$ is positive definite and thus invertible.
- We can write $H(t)^{-1}$ as
  $$H(t)^{-1} = D(t)^{-1/2}(I - D(t)^{-1/2}BD(t)^{-1/2})^{-1}D(t)^{-1/2}$$
- We also have, if $\rho(A) < 1$, then $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$.
- From [Mokhtari, Ling, and Ribeiro 15] $\rho(D(t)^{-1/2}BD(t)^{-1/2}) < 1$. Then by finite truncation, we have Hessian inverse approximation
  $$\hat{H}(t)^{-1} = D(t)^{-1/2}\left[I + D(t)^{-1/2}BD(t)^{-1/2}\right]D(t)^{-1/2}.$$
Asynchronous network Newton

Background on Hessian Approximation

- Hessian matrix can be splitted as $H(t) = I - W + \alpha G(t) = D(t) - B$ with
  
  $D(t) = \alpha G(t) + 2(I - W_d), \quad B = I - 2W_d + W$

  where $W_d$ is a diagonal matrix with $[W_d]_{ii} = W_{ii}$.

- $D(t)$ is positive definite and thus invertible.

- We can write $H(t)^{-1}$ as
  
  $$H(t)^{-1} = D(t)^{-1/2}(I - D(t)^{-1/2}BD(t)^{-1/2})^{-1}D(t)^{-1/2}$$

- We also have, if $\rho(A) < 1$, then $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$.

- From [Mokhtari, Ling, and Ribeiro 15] $\rho(D(t)^{-1/2}BD(t)^{-1/2}) < 1$. Then by finite truncation, we have Hessian inverse approximation
  
  $$\hat{H}(t)^{-1} = D(t)^{-1/2}\left[I + D(t)^{-1/2}BD(t)^{-1/2}\right]D(t)^{-1/2}.$$
Distributed Computation of Approximate Hessian Inverse Matrix

\[
\hat{H}(t)^{-1} = D(t)^{-1/2} \left[ I + D(t)^{-1/2} BD(t)^{-1/2} \right] D(t)^{-1/2}.
\]

- Each agent \(i\) has local access to \(f_i\) and \(W_{ii}\) and to \(W_{ij}\) for all neighbors \(j\) in \(\mathcal{N}(i)\).
- At iteration \(t\)
  
  a. Each agent \(i\) computes the \(i'\)th diagonal element of \(D(t)\),
  \[
  D_{ii}(t) = \alpha \nabla^2 f_i(x_i(t)) + 2(1 - W_{ii}),
  \]
  
  b. Each agent \(i\) computes \(B_{ii}\) and \(B_{ij}\) for all neighbors \(j\).
  \[
  B_{ii} = 1 - 2W_{ii} + W_{ii} = 1 - W_{ii}, \quad B_{ij} = W_{ij},
  \]

Multiplication of matrix \(B\) corresponds to communicating with immediate neighbors, which can also be carried out locally. The Newton’s direction approximation would be

\[
d(t) = \hat{H}(t)^{-1} g(t).
\]
Asynchronous Network Newton

- Each agent $i$ is associated with a local clock that ticks with probability $p_i$, $(0 < \pi \leq p_i \leq \Pi < 1, \sum_{i=1}^{n} p_i = 1)$.
- Whenever the clock ticks, an agent is active and updates its local Newton direction.
- Each agent’s stepsize is inversely proportional to its activation probability.
- The active agent finishes updating before another activation happens.
- Whenever any agent is active the iteration counter is increased by 1.
Asynchronous Network Newton Algorithm

Initialization: For $i = 1, 2, \ldots, n$, each agent $i$ sets $x_i(0) = 0$, computes $D_{ii}(0)$, $g_i(0)$, $d^{(0)}_i(0)$, $B_{ii}$, and $B_{ij}$:

\[
D_{ii}(0) = \alpha \nabla^2 f_i(x_i(0)) + 2(1 - W_{ii}),
\]
\[
g_i(0) = (1 - W_{ii})x_i(0) + \alpha \nabla f_i(x_i(0)),
\]
\[
d^{(0)}_i(0) = -D_{ii}(0)^{-1}g_i(0),
\]
\[
B_{ii} = 1 - W_{ii}, \quad B_{ij} = W_{ij},
\]

and broadcasts $d^{(0)}_i(0)$ to all neighbors, stores received $d^{(0)}_j$, $x_j$ values from neighbors.
Asynchronous Network Newton Algorithm

For $i = 1, 2, \ldots, n$, each agent $i$ sets $x_i(0) = 0$, computes $D_{ii}(0)$, $g_i(0)$, $d_i^{(0)}(0)$, $B_{ii}$, and $B_{ij}$ and broadcasts $d_i^{(0)}(0)$ to all neighbors, stores received $d_j^{(0)}(0)$, $x_j$ values from neighbors.

For $t = 1, 2, \ldots$, an agent $i$ is active according to its local clock with probability $p_i$, computes its local Newton’s direction and updates its local iterate

$$g_i(t - 1) = (1 - W_{ii})x_i(t - 1) + \alpha \nabla f_i(x_i(t - 1)) - \sum_{j \in \mathcal{N}_i} W_{ij}x_j(t - 1)$$

$$d_i^{(0)}(t - 1) = -D_{ii}(t - 1)^{-1}g_i(t - 1)$$
Asynchronous Network Newton Algorithm

For $i = 1, 2, \ldots, n$, each agent $i$ sets $x_i(0) = 0$, computes $D_{ii}(0)$, $g_i(0)$, $d_i^{(0)}(0)$, $B_{ii}$, and $B_{ij}$ and broadcasts $d_i^{(0)}(0)$ to all neighbors, stores received $d_j^{(0)}$, $x_j$ values from neighbors.

For $t = 1, 2, \ldots$, an agent $i$ is active according to its local clock, computes its local Newton’s direction and updates its local iterate

$$d_i(t-1) = D_{ii}(t-1)^{-1}[B_{ii}d_i^{(0)}(t-1) - g_i(t-1) + \sum_{j \in \mathcal{N}_i} B_{ij}d_j^{(0)}(t-1)]$$

$$x_i(t) = x_i(t-1) + \frac{\varepsilon}{\rho_i}d_i(t-1)$$
Asynchronous Network Newton Algorithm

For $i = 1, 2, ..., n$, each agent $i$ sets $x_i(0) = 0$, computes $D_{ii}(0)$, $g_i(0)$, $d_i^{(0)}(0)$, $B_{ii}$, and $B_{ij}$ and broadcasts $d_i^{(0)}(0)$ to all neighbors, stores received $d_j^{(0)}$, $x_j$ values from neighbors.

a) For $t = 1, 2, ..., an agent $i$ is active according to its local clock, computes its local Newton's direction and updates its local iterate

b) Active agent updates $D_{ii}(t), g_i(t), d_i^{(0)}(t)$

$$D_{ii}(t) = \alpha \nabla^2 f_i(x_i(t)) + 2(1 - W_{ii}),$$

$$g_i(t) = (1 - W_{ii})x_i(t) + \alpha \nabla f_i(x_i(t)) - \sum_{j \in \mathcal{N}_i} W_{ij}x_j(t - 1),$$

$$d_i^{(0)}(t) = -D_{ii}(t)^{-1} g_i(t).$$

[Diagram of a network with nodes 1, 2, 3, 4, 5, with edges connecting them. Labels $g_2(t), D_{22}(t), d_2^{(0)}(t)$ are also present.]
Asynchronous Network Newton Algorithm

For $i = 1, 2, ..., n$, each agent $i$ sets $x_i(0) = 0$, computes $D_{ii}(0)$, $g_i(0)$, $d_i^{(0)}(0)$, $B_{ii}$, and $B_{ij}$ and broadcasts $d_i^{(0)}(0)$ to all neighbors, stores received $d_j^{(0)}$, $x_j$ values from neighbors.

a For $t = 1, 2, ...$, an agent $i$ is active according to its local clock, computes its local Newton's direction and updates its local iterate.

b Active agent updates $D_{ii}(t)$, $g_i(t)$, $d_i^{(0)}(t)$

c Active agent $i$ broadcasts $d_i^{(0)}(t)$, $x_i(t)$ to its neighbors who passively listen and store received most updated values.
Convergence Analysis

Almost Sure Convergence and Global Linear Rate

Theorem

When the stepsize parameter satisfies $0 < \varepsilon \leq 2\pi \left(\frac{\lambda}{\Lambda}\right)^2$, the sequence $\{F(x(t))\}$ converges to its optimal value $F^*$ almost surely.

If $0 < \varepsilon \leq \min \left\{ \frac{1}{2}, 2\pi \left(\frac{\lambda}{\Lambda}\right)^2 \right\}$, then $\{F(x(t))\}$ and $\{x(t)\}$ converge linearly in expectation, i.e.,

$$\mathbb{E}[F(x(t)) - F^*] \leq (1 - \beta)^t [F(x(0)) - F^*],$$

$$\mathbb{E}[\|x(t) - x^*\|] \leq \left(\frac{2(F(x(0)) - F^*)}{\alpha m}\right)^{1/2} \left((1 - \beta)^{1/2}\right)^t.$$
Almost Sure Convergence and Global Linear Rate

Theorem

When the stepsize parameter satisfies $0 < \varepsilon \leq 2\pi \left(\frac{\lambda}{\Lambda}\right)^2$, the sequence $\{F(x(t))\}$ converges to its optimal value $F^*$ almost surely.

If $0 < \varepsilon \leq \min \left\{\frac{1}{2}, 2\pi \left(\frac{\lambda}{\Lambda}\right)^2\right\}$, then $\{F(x(t))\}$ and $\{x(t)\}$ converge linearly in expectation, i.e.,

$$
\mathbb{E}[F(x(t)) - F^*] \leq (1 - \beta)^t [F(x(0)) - F^*],
$$

$$
\mathbb{E}[\|x(t) - x^*\|] \leq \left(\frac{2(F(x(0)) - F^*)}{\alpha m}\right)^{1/2} \left((1 - \beta)^{1/2}\right)^t.
$$

Constants: $\Lambda = \frac{1+\rho}{2(1-\Delta)+\alpha m}$, $\lambda = \frac{1}{2(1-\delta)+\alpha M}$, $0 < \beta = \frac{\alpha m \varepsilon (2\pi \lambda^2 - \varepsilon \Lambda^2)}{\lambda \pi} < 1$, $\pi = \min_i p_i$, $\delta = \min_i W_{ii}$, $\Delta = \max_i W_{ii}$, $ml \leq \nabla^2 f_i(x_i) \leq Ml$, $\rho = 2(1 - \delta)(2(1 - \delta) + \alpha m)$
Local Superlinear Convergence

Lemma

For stepsize with $0 < \varepsilon \leq \min \left\{ \frac{1}{2}, 2\pi \left( \frac{\lambda}{\Lambda} \right)^2 \right\}$, we have

$$\mathbb{E} \left[ \left\| D(t - 1)^{1/2} (x(t) - x^*) \right\| \right] \leq \Gamma_1 \left( \mathbb{E} \left[ \left\| D(t - 2)^{1/2} (x(t - 1) - x^*) \right\| \right] \right)^2$$

$$+ \Gamma(t) \mathbb{E} \left[ \left\| D(t - 2)^{1/2} (x(t - 1) - x^*) \right\| \right].$$
Local Superlinear Convergence

Lemma

For stepsize with $0 < \varepsilon \leq \min \left\{ \frac{1}{2}, 2\pi \left( \frac{\lambda}{\Lambda} \right)^2 \right\}$, we have

$$\mathbb{E} \left[ \left\| D(t - 1)^{1/2} (x(t) - x^*) \right\| \right] \leq \Gamma_1 \left( \mathbb{E} \left[ \left\| D(t - 2)^{1/2} (x(t - 1) - x^*) \right\| \right] \right)^2$$

$$+ \Gamma(t) \mathbb{E} \left[ \left\| D(t - 2)^{1/2} (x(t - 1) - x^*) \right\| \right] .$$

Constants: $\Gamma_1 = \frac{(2(1-\delta) + \alpha M)^{1/2}}{2\pi^2 \left( 2(1-\Delta) + \alpha m \right)} \frac{\alpha L \varepsilon \Lambda}{\lambda}$ and $\Gamma(t) = C_1 \left( 1 + C_3 (1 - \beta) \frac{t-2}{4} \right)$ with

$$C_1 = \left( 1 + \varepsilon \max \left\{ \frac{\varepsilon}{\pi} - 2, \frac{\varepsilon(1 - \rho^2)^2}{\pi} - 2(1 - \rho^2) \right\} \right)^{1/2} < 1,$$

$$C_2 = \left( \frac{\varepsilon \alpha L \Lambda}{\pi \left( 2(1-\Delta) + \alpha m \right)} \right)^{1/2},$$

and

$$C_3 = C_2 \left( \frac{2}{\lambda \pi^2} (F(x(0)) - F^*) \right)^{1/4} .$$
Local Superlinear Convergence

**Theorem**

For all $t$ with

$$t > \frac{4 \ln \frac{1-C_1}{C_3 C_1}}{\ln (1 - \beta)} + 2,$$

we have $\Gamma(t) < 1$ and there exists $0 < \theta < \frac{1-\Gamma(t)}{\Gamma_1 \Gamma(t)}$, such that the sequence

$$\mathbb{E} \left[ \| D(t-1)^{1/2} (x(t) - x^*) \| \right]$$

decreases with a quadratic rate in expectation in this interval.

This neighborhood is also characterized by

$$\theta \Gamma(t) \leq \mathbb{E} \left[ \| D(t-1)^{1/2} (x(t) - x^*) \| \right] < \frac{\theta}{\theta \Gamma_1 + 1},$$
Simulation Results

Asynchronous network Newton is compared against asynchronous ADMM [Wei, Ozdeglar 13] and asynchronous gossip [Ram, Nedić, Veeravalli 09] algorithms.

Tested on networks of 5 agents with complete and ring underlying graphs.

Tested on quadratic and non-quadratic objective functions.

Asynchronous network Newton outperforms the other two algorithms, which is expected due to the local quadratic rate.
Quadratic Objective Functions

- Objective function at agent $i$: $f_i(x_i) = (x - i)^2$
- Minimum activation probability: $\pi = \frac{2}{15}$
- Stepsize parameter for Async NN: $\epsilon = \frac{2}{15}$
Regularized Logistic Regression

- Data classification for $K$ training samples that are uniformly distributed over $n = 5$ agents in a network.
- Each agent $i$ has access to $k_i = \lfloor \frac{K}{n} \rfloor$ data points.
- $u_{ij}$ and $v_{ij}$, $j \in \{1, 2, \ldots, k_i\}$ are the feature vector and the label for the data point $j$ associated with agent $i$.

$$\min_x f(x) = \frac{\nu}{2} \|x\|^2 + \frac{1}{K} \sum_{i=1}^{n} \sum_{j=1}^{k_i} \log \left[ 1 + \exp(-v_{ij}u_{ij}x) \right],$$

$$f_i(x) = \frac{\nu}{2n} \|x\|^2 + \frac{1}{K} \sum_{j=1}^{k_i} \log \left[ 1 + \exp(-v_{ij}u_{ij}x) \right].$$

- Tested on "Pima Indian Diabetes" data set with 768 data points, feature vector of size 8, and a label which is either 1 or $-1$. 
Regularized Logistic Regression

\[ f_i(x) = \frac{v}{2n} ||x||^2 + \frac{1}{K} \sum_{j=1}^{k_i} \log \left[ 1 + \exp(-v_{ij}u_{ij}x) \right]. \]

- Minimum activation probability: \( \frac{2}{15} \)
- Stepsize parameter for Async NN: \( \varepsilon = 0.043 \)
Asynchronous distributed network Newton algorithm uses matrix splitting techniques to approximate the Hessian inverse and compute Newton step.

This algorithm converges almost surely with in expectation global linear rate of convergence.

Asynchronous network Newton achieves a local quadratic convergence rate (in expectation) to a neighborhood of the optimum.

Simulation results show the convergence speed improvement of the asynchronous network Newton compared to asynchronous ADMM and asynchronous gossip algorithm.

Future directions:
- Dynamic graph.
- Larger stepsize rules.