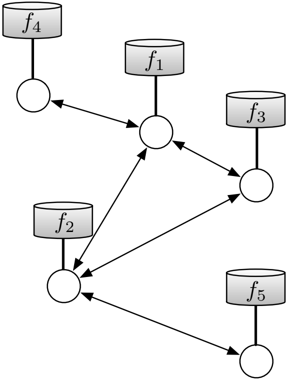
Network Science
Models and Distributed Algorithms

IST-CMU Phd course
João Xavier
TA: João Martins

October 27, 2016

Optimization in static undirected networks



- n agents; agent i holds function $f_i : \mathbf{R}^d \to \mathbf{R} \cup \{\infty\}$
- communication network is static and undirected
- communication happens in discrete time $t = 0, 1, 2, 3, \ldots$
- goal: compute

$$x^\star \in \arg\min_{x \in \mathbf{R}^d} f(x) := \frac{f_1(x) + \cdots + f_n(x)}{n}$$

Example: consensus

- we can view the arithmetic mean

$$\overline{\theta} = \frac{\theta_1 + \cdots + \theta_n}{n}$$

as the solution of

$$\underset{x \in \mathbf{R}}{\text{minimize}} \quad \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \underbrace{\frac{1}{2}\left(x - \theta_i\right)^2}_{f_i(x)}$$

Example: distributed logistic regression

- parametric model linking feature $A \in \mathbf{R}^d$ to outcome $B \in \{0, 1\}$:

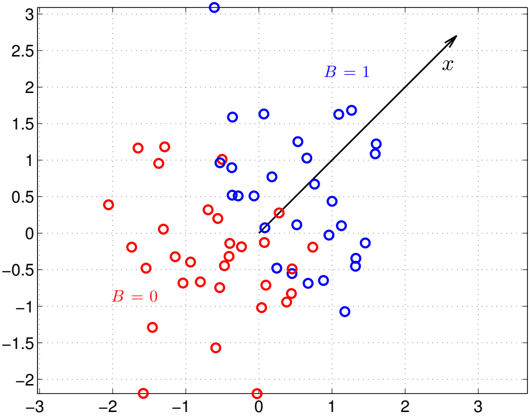
$$\log \frac{\mathbf{P}\left(B = 1 \,|\, A = a; x\right)}{\mathbf{P}\left(B = 0 \,|\, A = a; x\right)} = a^T x$$

- equivalent to

$$\mathbf{P}(B = 0 \,|\, A = a; x) = \frac{1}{1 + e^{a^T x}} \quad \mathbf{P}(B = 1 \,|\, X = x; w) = \frac{e^{a^T x}}{1 + e^{a^T x}}$$

- $x \in \mathbf{R}^d$ is the model parameter

- example:



- we are given the dataset $\left\{(a_i, b_i) \in \mathbf{R}^d \times \{0,1\} \,:\, i = 1, \ldots, n\right\}$

- how do we learn x from the training dataset?

- maximum likelihood (ML) formulation:

$$\underset{x \in \mathbf{R}^d}{\text{maximize}} \quad \mathbf{P}\left(B_1 = b_1, \ldots, B_n = b_n \,|\, A_1 = a_1, \ldots, A_n = a_n; x\right)$$

- boils down to solving

$$\underset{x \in \mathbf{R}^d}{\text{minimize}} \quad \textstyle\sum_{i=1}^n -b_i a_i^T x + \log\left(1 + e^{a_i^T x}\right)$$

- adding a regularizer ($\rho > 0$):

$$\underset{x \in \mathbf{R}^d}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^{n} \underbrace{-b_i a_i^T x + \log\left(1 + e^{a_i^T x}\right) + \frac{\rho}{2} \|x\|^2}_{f_i(x)}$$

- agent i holds training point (a_i, b_i) (for simplicity; it can hold more)

Example: target localization

- target at unknown position $p \in \mathbf{R}^m$ ($m = 2$ or 3)

- agent i at known position $q_i \in \mathbf{R}^m$, $i = 1, \ldots, n$

- agent i measures

$$d_i = \|p - q_i\| + \text{noise}$$

- how to find the target position p from the network data $d_1, \ldots, d_n$?

- assuming measurement noise is small:

$$\|p\|^2 - 2q_i^T p + \|q_i\|^2 \simeq d_i^2$$

or

$$\underbrace{\begin{bmatrix} 1 & -2q_i^T \end{bmatrix}}_{a_i^T} \underbrace{\begin{bmatrix} \|p\|^2 \\ p \end{bmatrix}}_{x} \simeq \underbrace{d_i^2 - \|q_i\|^2}_{b_i}$$

- find $x = \left(\|p\|^2, p\right)$ by solving a distributed least-squares problem:

$$\underset{x \in \mathbf{R}^{m+1}}{\text{minimize}} \quad \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \underbrace{\frac{1}{2}\left(a_i^T x - b_i\right)^2}_{f_i(x)}$$

- suboptimal approach, but exact for typical agents' configurations with noiseless measurements

The optimization class $C^2(m, M)$

- notation: let

$$C^2(m, M) = \big\{\phi \in \mathbf{R}^d \to \mathbf{R} : \phi \text{ is continuously twice-differentiable} \\ \text{and } mI \preceq \nabla^2\phi(x) \preceq MI, \text{ for all } x \in \mathbf{R}^d\big\}$$

- assume each $f_i \in C^2(0, M_i)$ and $f \in C^2(m, M)$ with $m > 0$
(we can always take $M = \frac{M_1 + \cdots + M_n}{n}$)

- examples:
  - consensus
$$M_i = 1, \quad M = 1, \quad m = 1$$
  - regularized logistic regression
$$M_i = \|a_i\|^2 + \rho, \quad M = \frac{\|a_1\|^2 + \cdots + \|a_n\|^2}{n} + \rho, \quad m = \rho$$
  - target localization
$$M_i = \|a_i\|^2, \quad M = \frac{\|a_1\|^2 + \cdots + \|a_n\|^2}{n}, \quad m = \frac{\sigma_{\min}^2\left(\begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}\right)}{n}$$
    ($m > 0$ if $\{q_1, \ldots, q_n\}$ is an affine independent set)

- if $\phi \in C^2(m, M)$ then
  - $\phi(y) + \nabla\phi(y)^T(x-y) + \frac{m}{2}\|x-y\|^2 \leq \phi(x)$
  - $\phi(x) \leq \phi(y) + \nabla\phi(y)^T(x-y) + \frac{M}{2}\|x-y\|^2$
    (ϕ is sandwiched between two quadratics)
  - $m\|x-y\|^2 \leq (\nabla\phi(x) - \nabla\phi(y))^T(x-y) \leq M\|x-y\|^2$
  - $m\|x-y\| \leq \|\nabla\phi(x) - \nabla\phi(y)\| \leq M\|x-y\|$

- if $\phi \in C^2(m, M)$ with $m > 0$, then optimization problem

$$\underset{x \in \mathbf{R}^d}{\text{minimize}} \quad \phi(x)$$

has unique minimizer $x^\star$

- consider simple gradient method: $x^0 \in \mathbf{R}^d$ and

$$x^{k+1} = x^k - \alpha \nabla \phi\left(x^k\right), \quad k = 0, 1, 2, \ldots$$

- converges linearly for $0 < \alpha < \frac{2m}{M^2}$:

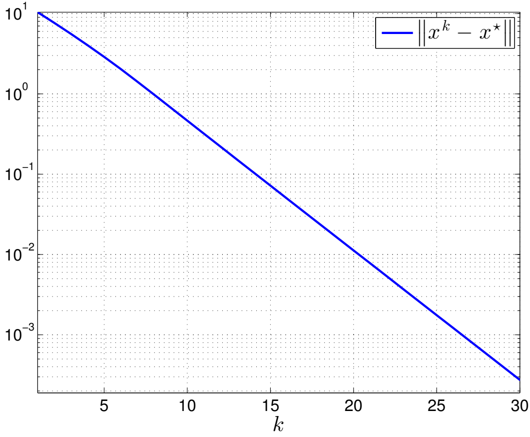
$$\left\|x^k - x^\star\right\| \leq \left(\sqrt{1 + \alpha^2 M^2 - 2\alpha m}\right)^k \left\|x^0 - x^\star\right\|$$

- with optimum $\alpha = \frac{m}{M^2}$:

$$\left\|x^k - x^\star\right\| \leq \left(\sqrt{1 - \frac{1}{\frac{M}{m}}}\right)^k \left\|x^0 - x^\star\right\|$$

- example:

$$f(x) = \log\left(1 + e^{x}\right) + \frac{1}{2}x^2$$



- how to apply the gradient method in distributed settings?

- for simplicity, take $d = 1$

- naive approach: each agent
  - does a (local) gradient step and
  - averages the result with neighbors

- in matrix notation:

$$x(t+1) = W\left(x(t) - \alpha \nabla F\left(x(t)\right)\right), \quad t = 0, 1, 2, \ldots,$$

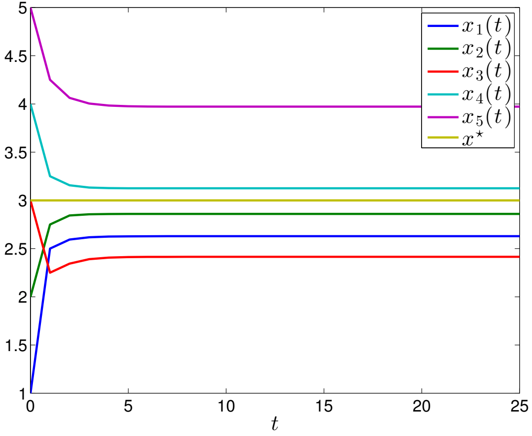
where $F : \mathbf{R} \times \cdots \times \mathbf{R} \to \mathbf{R}$,

$$F(x_1, \ldots, x_n) = f_1(x_1) + \cdots + f_n(x_n),$$

and W is a primitive matrix, $W\mathbf{1} = \mathbf{1}$ and $W_{ij} = 0$ whenever $i \not\sim j$

- let's try on consensus problem with metropolis W and $0 < \alpha < 2$

- naive scheme doesn't work:



- how can we fix this?

- for consensus: $F(x_1, \ldots, x_n) = \frac{1}{2}(x - \theta_1)^2 + \cdots + \frac{1}{2}(x_n - \theta_n)^2$

- in matrix notation:

$$F(x) = \frac{1}{2}\|x - \theta\|^2, \quad \nabla F(x) = x - \theta$$

- algorithm is

$$x(t+1) = W\left(x(t) - \alpha\left(x(t) - \theta\right)\right)$$

- we will change coordinates to analyze the algorithm

- the EVD of W is

$$W = \begin{bmatrix} \frac{1}{\sqrt{n}}\mathbf{1} & U \end{bmatrix} \begin{bmatrix} 1 & \\ & \Lambda \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{n}}\mathbf{1}^T \\ U^T \end{bmatrix}$$

- $U \in \mathbf{R}^{n \times n-1}$ spans the orthogonal complement of span$(\mathbf{1})$:

$$U^T U = I, \quad U^T \mathbf{1} = 0$$

- in

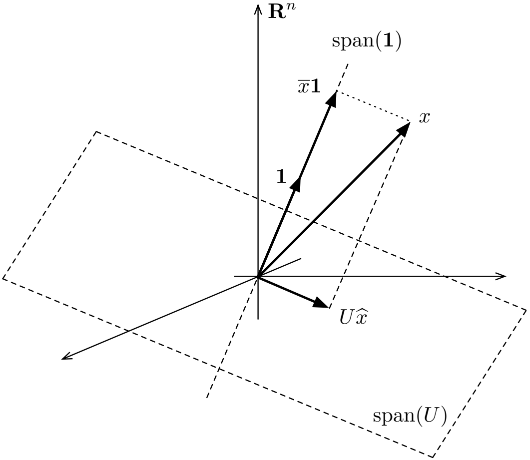
$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_{n-1} \end{bmatrix},$$

all $|\lambda_i| < 1$ (since W is a primitive matrix)

- any $x \in \mathbf{R}^n$ can be uniquely decomposed as

$$x = \overline{x}\mathbf{1} + U\widehat{x}$$

where $\overline{x} = \frac{1}{n}\mathbf{1}^T x$ and $\widehat{x} = U^T x$



- in the $(\overline{x}, \widehat{x})$ coordinates, algorithm is:

$$\begin{aligned}\overline{x}(t+1) &= \overline{x}(t) - \alpha(\overline{x}(t) - \overline{\theta}) \\ \widehat{x}(t+1) &= \Lambda\left(\widehat{x}(t) - \alpha\left(\widehat{x}(t) - \widehat{\theta}\right)\right)\end{aligned}$$

- we would like:

$$\overline{x}(t) \underset{t\to\infty}{\rightarrow} \overline{\theta} \quad \text{and} \quad \widehat{x}(t) \underset{t\to\infty}{\rightarrow} 0$$

- on one hand, since $\overline{x}(0) = \overline{\theta}$:

$$\overline{x}(t) \equiv \overline{\theta}, \quad \text{for all } t$$

- on the other hand:

$$\widehat{x}_i(t) \underset{t\to\infty}{\rightarrow} \frac{\alpha\lambda_i\widehat{\theta}_i}{1-\lambda_i(1-\alpha)} \neq 0$$

- this is what we need to fix

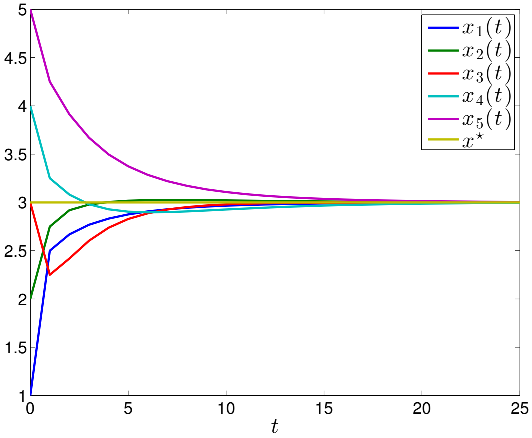
- how?

- by making the stepsize time-variant and diminishing to zero:

$$x(t+1) = W\left(x(t) - \alpha(t)\nabla F\left(x(t)\right)\right)$$

with $\alpha(t) \downarrow 0$

- back to example on page 16 with $\alpha(t) = (0.1)^t$:



- we fixed the problem

- is this the end of the story?

- let's try on the optimization problem

$$\underset{x \in \mathbf{R}}{\text{minimize}} \quad \tfrac{1}{n} \sum_{i=1}^{n} \underbrace{\frac{1}{2\sigma_i^2} \left(x - \theta_i\right)^2}_{f_i(x)}$$

- problema data is

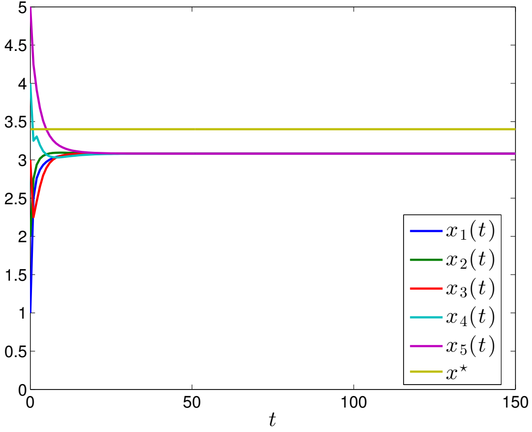
$$\theta_1, \ldots, \theta_5 = 1, 2, 3, 4, 5 \qquad \sigma_1^2, \ldots, \sigma_5^2 = 1, 1, 0.5, 0.2, 1$$

- algorithm is: $x(0) = \theta$ and

$$x(t+1) = W \left(x(t) - \alpha(t) \nabla F\left(x(t)\right)\right),$$

with $\alpha(t) = (0.1)^t$

- algorithm doesn't work:



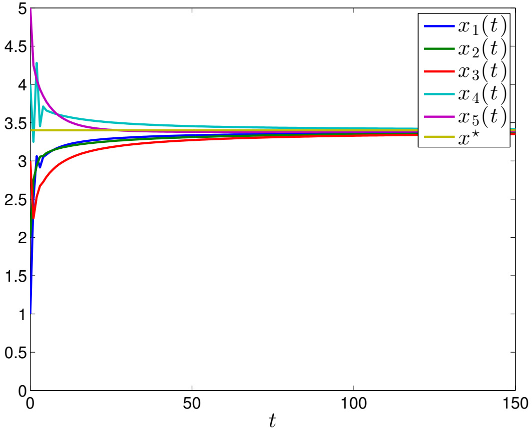
- can we fix the problem?

- yes, if the stepsize sequence satisfies:

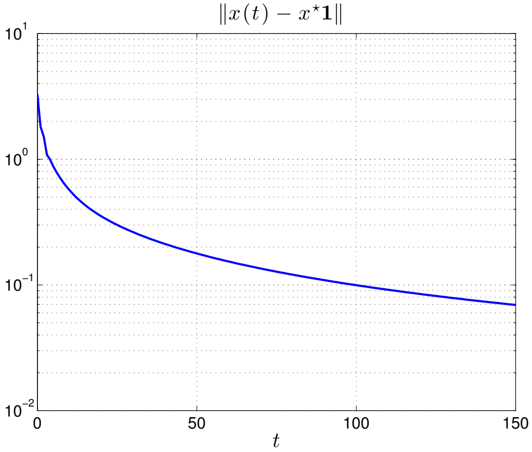
$$\alpha(t) > 0, \quad \sum_t \alpha(t) = \infty, \quad \sum_t \alpha(t)^2 < \infty$$

- see proof in K. Kvaternik and L. Pavel, "Lyapunov analysis of a distributed optimization scheme," *5th Int. Conf. on Network Games, Control and Opt.*, 2011.

- back to example on page 24 with $\alpha(t) = \frac{1}{t+1}$:



- unfortunately, we have lost linear convergence...



EXTRA algorithm

- EXTRA[1] algorithm is a constant stepsize gradient algorithm:

$$\begin{aligned} x(0) &= \text{initialization} \\ x(1) &= Wx(0) - \alpha \nabla F\left(x(0)\right) \\ x(t+1) &= (I+W)x(t) - \alpha \nabla F\left(x(t)\right) - \widetilde{W} x(t-1) + \alpha \nabla F\left(x(t-1)\right) \end{aligned}$$

  with $\widetilde{W} = \frac{I+W}{2}$ (other choices for $\widetilde{W}$ are possible)

- equivalently:

$$x(t+1) = Wx(t) - \alpha \nabla F\left(x(t)\right) - (\widetilde{W} - W) \sum_{s=0}^{t-1} x(s)$$

  for $t \geq 0$

- algorithm form is not obvious! We will offer an intuitive path

[1] W. Shi *et al.*, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," 25(2), *SIAM Journal on Opt.*, 2015.

- recall our goal: to compute

$$x^\star \in \arg\min_{x\in\mathbf{R}} f(x) := \frac{f_1(x) + \cdots + f_n(x)}{n}$$

- let's go back to naive idea[2] on page 15:

$$x(t+1) = Wx(t) - \alpha\nabla F\left(x(t)\right), \quad t = 0, 1, 2, \ldots,$$

where $F : \mathbf{R} \times \cdots \times \mathbf{R} \to \mathbf{R}$,

$$F(x_1, \ldots, x_n) = f_1(x_1) + \cdots + f_n(x_n),$$

and W is a primitive matrix, $W\mathbf{1} = \mathbf{1}$ and $W_{ij} = 0$ whenever $i \not\sim j$

[2] with a slight change: W acts only on $x(t)$, not on $\nabla F(x(t))$.

- for consensus, we have $f_i(x) = \frac{1}{2}\left(x - \theta_i\right)^2$ and

$$x(t+1) = Wx(t) - \alpha\left(x(t) - \theta\right) \quad t = 0, 1, 2, \ldots$$

with $x(0) = \theta$

- using the general decomposition $x = \overline{x}\mathbf{1} + U\widehat{x}$ on pages 18-19:

$$\begin{aligned} \overline{x}(t+1) &= \overline{x}(t) - \alpha\left(\overline{x}(t) - \overline{\theta}\right) \\ \widehat{x}(t+1) &= \Lambda\widehat{x}(t) - \alpha\left(\widehat{x}(t) - \widehat{\theta}\right) \end{aligned}$$

- we need $\overline{x}(t) \underset{t\to\infty}{\to} \overline{\theta}$ and $\widehat{x}(t) \underset{t\to\infty}{\to} 0$

- $\overline{x}(t) \equiv \overline{\theta}$ but

$$\widehat{x}_i(t+1) = (\lambda_i - \alpha)\widehat{x}_i(t) + \alpha\widehat{\theta}_i \quad \Rightarrow \quad \widehat{x}_i(t) \underset{t\to\infty}{\to} \frac{\alpha\widehat{\theta}_i}{1-(\lambda_i - \alpha)}$$

(assuming $0 < \alpha < 2$)

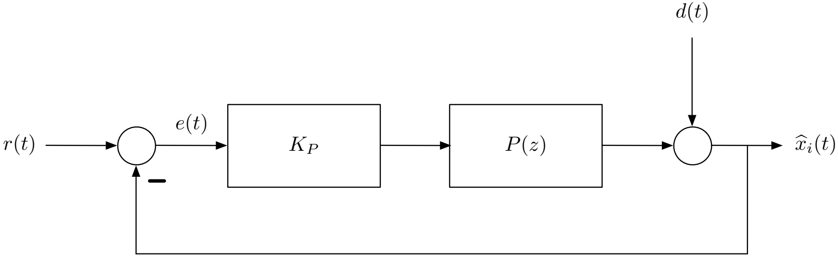
- shrinking the stepsize $\alpha(t) \downarrow 0$ solves the problem but kills linear convergence

- can we make $\widehat{x}_i(t) \to 0$ with a constant α?

- an insight from control theory: view the recursion

$$\widehat{x}_i(t+1) = (\lambda_i - \alpha)\widehat{x}_i(t) + \alpha\widehat{\theta}_i$$

as the feedback proportional controller



- $r(t) \equiv 0$ is the reference
- $K_P = -(\lambda_i - \alpha)$ is the controller gain
- $P(z) = z^{-1}$ is the z-transform of the plant
- $d(t) \equiv \alpha\widehat{\theta}_i$ is the disturbance
- $e(t) = r(t) - \widehat{x}_i(t)$ is the mismatch between $r(t)$ and $\widehat{x}_i(t)$

- transfer function from disturbance to error is

$$\frac{E(z)}{D(z)} = \frac{1}{1 + K_P z^{-1}}$$

- for $d(t) \equiv \alpha\widehat{\theta}_i$ for $t \geq 0$, we have $D(z) = \frac{\alpha\widehat{\theta}_i}{1-z^{-1}}$ and

$$E(z) = \frac{\alpha\widehat{\theta}_i}{\left(1 - (\lambda_i - \alpha)z^{-1}\right)\left(1 - z^{-1}\right)}$$

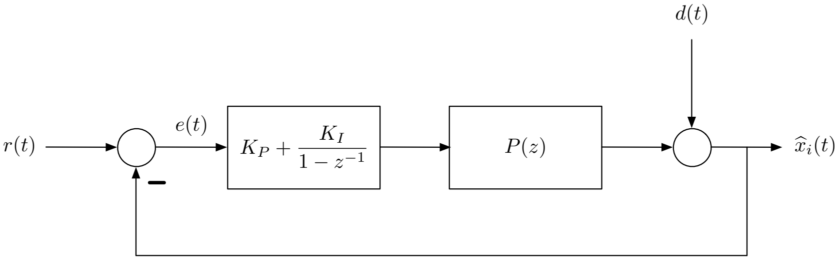
- from the Final Value Theorem,

$$\begin{aligned} \lim_{t\to\infty} e(t) &= \lim_{z\to 1}(z-1)E(z) \\ &= \frac{\alpha\widehat{\theta}_i}{1 - (\lambda_i - \alpha)} \end{aligned}$$

(confirms the steady-state error that we already knew)

- how can we suppress a steady-state error?

- standard control trick: add an integral controller $(K_i \neq 0)$



- transfer function becomes

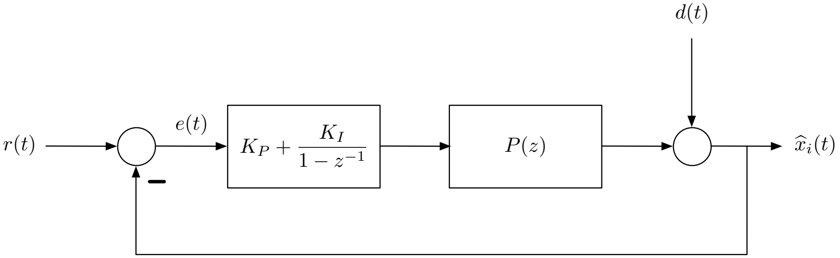
$$\frac{E(z)}{D(z)} = \frac{1}{1 + \left(K_P + \frac{K_I}{1-z^{-1}}\right) z^{-1}}$$

- plugging $D(z) = \frac{\alpha\widehat{\theta}_i}{1-z^{-1}}$ gives

$$E(z) = \frac{\alpha\widehat{\theta}_i}{\left(1 + \left(-(\lambda_i - \alpha) + \frac{K_I}{1-z^{-1}}\right) z^{-1}\right)(1 - z^{-1})}$$

- Final Value Theorem gives

$$\begin{aligned} \lim_{t\to\infty} e(t) &= \lim_{z\to 1}(z-1)E(z) \\ &= 0 \end{aligned}$$



- corresponds to time-dynamics:

$$\widehat{x}_i(t+1) = (\lambda_i - \alpha)\widehat{x}_i(t) + \alpha\widehat{\theta}_i - K_I \sum_{s=0}^{t-1} \widehat{x}_i(s)$$

- last equation is in $(\overline{x}, \widehat{x})$ coordinates

- can we backtrack to natural coordinates x?

- let's try the obvious idea:

$$x(t+1) = Wx(t) - \alpha\left(x(t) - \theta\right) - K_I \sum_{s=0}^{t-1} x(s)$$

- gives

$$\overline{x}(t+1) = \overline{x}(t) - \alpha\left(\overline{x}(t) - \overline{\theta}\right) - K_I \sum_{s=0}^{t-1} \overline{x}_i(s)$$

  and

$$\overline{x}(t) \underset{t\to\infty}{\to} 0!$$

- we need $K_I = 0$ for the coordinate $\overline{x}$...

- possible approach:

$$x(t+1) = Wx(t) - \alpha\left(x(t) - \theta\right) - \frac{I-W}{2}\sum_{s=0}^{t-1} x(s)$$

- in $(\overline{x}, \widehat{x})$ coordinates:

$$\begin{aligned}\overline{x}(t+1) &= \overline{x}(t) - \alpha\left(\overline{x}(t) - \overline{\theta}\right)\\ \widehat{x}_i(t+1) &= \lambda_i \widehat{x}(t) - \alpha\left(\widehat{x}_i(t) - \widehat{\theta}_i\right) - K_i \sum_{s=0}^{t-1} \widehat{x}_i(s)\end{aligned}$$

with $K_i := \frac{1-\lambda_i}{2} \neq 0$ (recall that $|\lambda_i| < 1$)

- our path led us to the recursion:

$$x(t+1) = Wx(t) - \alpha\left(x(t) - \theta\right) - \frac{I-W}{2}\sum_{s=0}^{t-1} x(s)$$

- equivalent form:

$$x(t+1) = Wx(t) - \alpha\nabla F\left(x(t)\right) - \left(\widetilde{W} - W\right)\sum_{s=0}^{t-1} x(s)$$

because $\nabla F(x) = x - \theta$ and $\widetilde{W} := \frac{I+W}{2}$,

- compare with EXTRA algorithm:

$$x(t+1) = Wx(t) - \alpha\nabla F\left(x(t)\right) - (\widetilde{W} - W)\sum_{s=0}^{t-1} x(s)$$

for generic F

Brief analysis of EXTRA

- EXTRA has the right "fixed-point" property: if $x(t) \to x$ then $x = x^\star \mathbf{1}$ with

$$x^\star \in \arg\min_{x \in \mathbf{R}} f(x) := \frac{f_1(x) + \cdots + f_n(x)}{n}$$

- EXTRA converges linearly for consensus problem: from

$$\widehat{x}_i(t+1) = \lambda_i \widehat{x}(t) - \alpha \left(\widehat{x}_i(t) - \widehat{\theta}_i \right) - K_i \sum_{s=0}^{t-1} \widehat{x}_i(s)$$

with $K_i = \frac{1-\lambda_i}{2}$, we get

$$\widehat{x}_i(t+1) = (1 - \alpha + \lambda_i)\widehat{x}_i(t) + \left(\alpha - \frac{1+\lambda_i}{2} \right) \widehat{x}_i(t-1)$$

- in vector form:

$$\begin{bmatrix} \widehat{x}_i(t+1) \\ \widehat{x}_i(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 1-\alpha+\lambda_i & \alpha - \frac{1+\lambda_i}{2} \\ 1 & 0 \end{bmatrix}}_{A(\alpha)} \begin{bmatrix} \widehat{x}_i(t) \\ \widehat{x}_i(t-1) \end{bmatrix}$$

- we conclude:

$$\begin{bmatrix} \widehat{x}_i(t+1) \\ \widehat{x}_i(t) \end{bmatrix} = A(\alpha)^t \begin{bmatrix} \widehat{x}_i(1) \\ \widehat{x}_i(0) \end{bmatrix}$$

- linear convergence occurs if $\rho\left(A(\alpha)\right) < 1$

- we will show that $\rho\left(A(\alpha)\right) < 1$ for $0 < \alpha < \frac{1}{2}$

- characteristic polynomial of $A(\alpha)$ is

$$p(s) = s^2 - (1 + \lambda_i - \alpha)s + \left(\frac{1 + \lambda_i}{2} - \alpha\right)$$

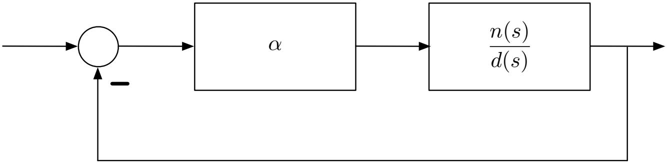
- we need to know how the roots of $p(s)$ vary with α

- idea: re-arrange

$$p(s) = \underbrace{s^2 - (1 + \lambda_i)s + \frac{1 + \lambda_i}{2}}_{d(s)} + \alpha \underbrace{(s - 1)}_{n(s)}$$

and apply well-known root locus techniques from basic control

- proportional controller structure:



- for $\alpha = 0$, the roots of $p(s)$ are those of $d(s)$:

$$\frac{1+\lambda_i}{2} \pm i\frac{1}{2}\sqrt{(1+\lambda_i)(1-\lambda_i)}$$

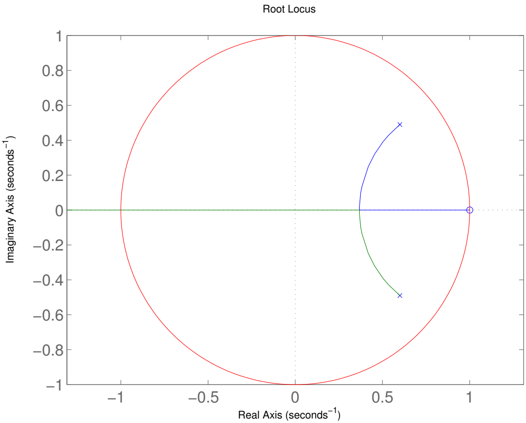
with absolute value

$$\sqrt{\frac{1+\lambda_i}{2}} \in \,]0,1[$$

(recall that $|\lambda_i| < 1$)

- as $\alpha \to \infty$, one root of $p(s)$ goes to ∞ and the other goes to 1

- example with $\lambda_i = 0.2$:



- we see that $\rho\left(A(\alpha)\right) < 1$ until $s = -1$ becomes a root of $p(s)$:

$$\begin{aligned} p(-1) = 0 \quad &\Leftrightarrow \quad 1 + (1+\lambda_i) + \frac{1+\lambda_i}{2} - 2\alpha = 0 \\ &\Leftrightarrow \quad \alpha = \frac{1}{2} + \frac{3}{4}(\lambda_i + 1) \end{aligned}$$

- we conclude that $\rho\left(A(\alpha)\right) < 1$ for $0 < \alpha < \frac{1}{2}$

- **Theorem**. Assume
  - $f_i \in C^2(0, M_i)$
  - $f \in C^2(m, M)$ with $m > 0$
  - $W \succeq 0$.

  Then, EXTRA converges linearly for

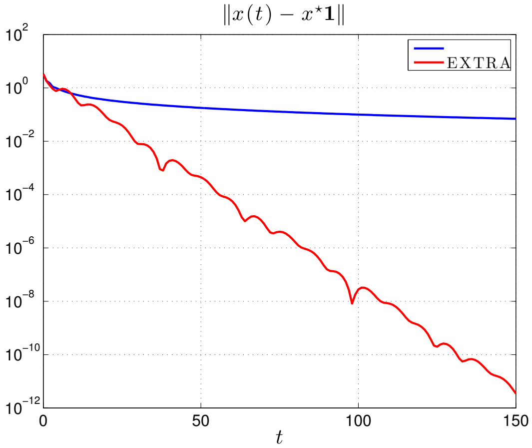
$$0 < \alpha < \frac{m}{\max\{M_1^2, \ldots, M_n^2\}}.$$

- design of stepsize is independent from the network topology

- see proof of theorem 3.7 in W. Shi *et al.*, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," 25(2), *SIAM Journal on Opt.*, 2015.

- theorem 3.7 shows that the condition $f \in C^2(m, M)$ can be weakened (f only needs to be restricted strongly convex)

- comparing EXTRA with algorithm on page 27:



Another gradient approach with constant stepsize

- G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," `https://arxiv.org/abs/1605.07112`, 2016.

- algorithm:

$$\begin{aligned} x_i(0) &= \text{initialization}, \quad i = 1, \ldots, n \\ s_i(0) &= \nabla f_i\left(x_i(0)\right), \quad i = 1, \ldots, n \\ x(t+1) &= Wx(t) - \alpha s(t) \\ s(t+1) &= Ws(t) + \nabla F\left(x(t+1)\right) - \nabla F\left(x(t)\right) \end{aligned}$$

- we can see $s(t)$ as tracking

$$\frac{1}{n} \sum_{i=1}^{n} \nabla f_i\left(x_i(t)\right)$$

(recall problem 3 from homework 1)

Brief analysis

- The algorithm has the right "fixed-point" property: if $(x(t), s(t)) \to (x, s)$ then $x = x^\star \mathbf{1}$ with

$$x^\star \in \arg\min_{x \in \mathbf{R}} f(x) := \frac{f_1(x) + \cdots + f_n(x)}{n},$$

  and $s = 0$

- The algorithm converges linearly for consensus problem: initialization $x(0) = \theta$, $s = 0$, and

$$\begin{aligned} x(t+1) &= Wx(t) - \alpha s(t) \\ s(t+1) &= Ws(t) + x(t+1) - x(t) \end{aligned}$$

  imply

$$\overline{x}(t) \equiv \overline{\theta} \quad \overline{s}(t) \equiv 0$$

- on the other hand:

$$\begin{aligned}
\widehat{x}(t+1) &= \Lambda\widehat{x}(t) - \alpha\widehat{s}(t) \\
\widehat{s}(t+1) &= \Lambda\widehat{s}(t) + \widehat{x}(t+1) - \widehat{x}(t) \\
&= (\Lambda - \alpha I)\,\widehat{s}(t) + (\Lambda - I)\,\widehat{x}(t)
\end{aligned}$$

- in vector form:

$$\begin{bmatrix} \widehat{x}(t+1) \\ \widehat{s}(t+1) \end{bmatrix} = \underbrace{\begin{bmatrix} \Lambda & -\alpha I \\ \Lambda - I & \Lambda - \alpha I \end{bmatrix}}_{A(\alpha)} \begin{bmatrix} \widehat{x}(t) \\ \widehat{s}(t) \end{bmatrix}$$

- we want to show that $\rho\left(A(\alpha)\right) < 1$ for some interval $\alpha \in]0, \overline{\alpha}[$ with $\overline{\alpha} > 0$

- $A(\alpha)$ is similar to a block-diagonal matrix:

$$A(\alpha) \sim \begin{bmatrix} A_1(\alpha) & & & \\ & A_2(\alpha) & & \\ & & \ddots & \\ & & & A_{n-1}(\alpha) \end{bmatrix}$$

  where

$$A_i(\alpha) = \begin{bmatrix} \lambda_i & -\alpha \\ \lambda_i - 1 & \lambda_i - \alpha \end{bmatrix}$$

- it suffices to show that $\rho\left(A_i(\alpha)\right) < 1$ for $\alpha \in]0, \overline{\alpha}[$

- characteristic polynomial of $A_i(\alpha)$ is

$$p(s) = \underbrace{(s - \lambda_i)^2}_{d(s)} + \alpha \underbrace{(s - 1)}_{n(s)}$$

- for $\alpha = 0$, the roots of $p(s)$ are those of $d(s)$:

$$\lambda_i$$

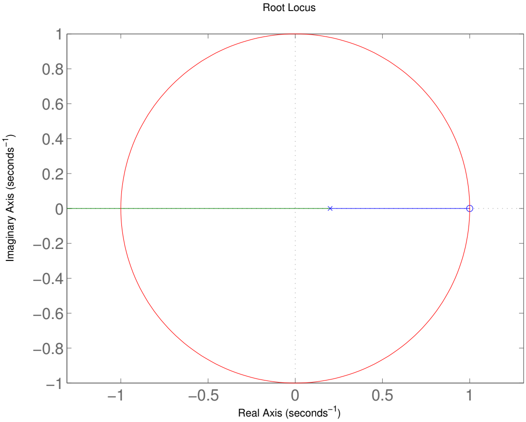
with absolute value

$$|\lambda_i| \in [0, 1[$$

(recall that $|\lambda_i| < 1$)

- as $\alpha \to \infty$, one root of $p(s)$ goes to ∞ and the other goes to 1

- example with $\lambda_i = 0.2$:



- we see that $\rho(A(\alpha)) < 1$ until $s = -1$ becomes a root of $p(s)$:

$$\begin{aligned} p(-1) = 0 \quad &\Leftrightarrow \quad (\lambda_i + 1)^2 - 2\alpha = 0 \\ &\Leftrightarrow \quad \alpha = \frac{(\lambda_i + 1)^2}{2} \end{aligned}$$

- if all $\lambda_i \geq 0$, we conclude that $\rho(A(\alpha)) < 1$ for $0 < \alpha < \overline{\alpha} := \frac{1}{2}$

- **Theorem**. Assume $f_i \in C^2(m, M)$ with $m > 0$. Then, the algorithm converges linearly for

$$0 < \alpha < \overline{\alpha}.$$

- $\overline{\alpha}$ depends on the network topology

- see proof of theorem 1 in G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," `https://arxiv.org/abs/1605.07112`, 2016.

The optimization class $C^1(M)$

- notation:

$$C^1(M) = \{\phi \in \mathbf{R}^d \to \mathbf{R} : \phi \text{ is continuously-differentiable and } \|\nabla\phi(x) - \nabla\phi(y)\| \leq M \|x - y\|, \text{ for all } x, y \in \mathbf{R}^d\}$$

- $C^2(m, M)$ is contained in $C^1(M)$

- some papers that only assume f_i are convex and in $C^1(M)$:
  - D. Jakovetić et al., "Fast distributed gradient methods," *IEEE Trans, on Aut. Control*, 59(5), 2014. **Rate**: $\mathcal{O}\left(1/t^2\right)$ (with further assumption of bounded gradients: $\|\nabla f_i(x)\| \leq C$ for all x)
  - W. Shi *et al.*, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," 25(2), *SIAM Journal on Opt.*, 2015. **Rate**: $\mathcal{O}(1/t)$
  - G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," `https://arxiv.org/abs/1605.07112`, 2016. **Rate**: $\mathcal{O}(1/t)$

ADMM

- ADMM = Alternate Direction Method of Multipliers

- "old" optimization method for

$$\begin{array}{ll} \underset{x,z}{\text{minimize}} & g(y) + h(z) \\ \text{subject to} & Ay + Bz = c \end{array}$$

  where g and h are convex functions

- applied to distributed optimization in I. Schizas *et al.*, "Consensus in *ad hoc* WSNs with noisy links," *IEEE Trans. on Sig. Proc*, 56(1), 2008

- ADMM is based on the augmented Lagrangian function

$$L(y, z; \lambda) = g(y) + h(z) + \lambda^T (Ay + Bz - c) + \frac{\rho}{2} \|Ay + Bz - c\|^2$$

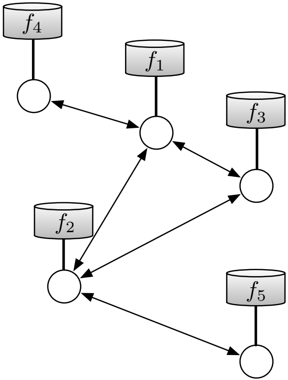
  where $\rho > 0$ is chosen by the user

- $\lambda = (\ldots, \lambda_i, \ldots)$ is lagrange multiplier: λ_i is associated with ith constraint in $Ay + Bz = c$

- ADMM:

$$\begin{aligned}
z(0) &= \text{initialization} \\
\lambda(0) &= \text{initialization} \\
y(t+1) &= \arg\min_y L\left(y, z(t); \lambda(t)\right) \\
z(t+1) &= \arg\min_z L\left(y(t+1), z; \lambda(t)\right) \\
\lambda(t+1) &= \lambda(t) + \rho\left(Ay(t+1) + Bz(t+1) - c\right)
\end{aligned}$$

  for $t = 0, 1, 2, \ldots$

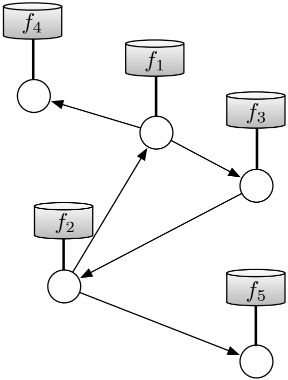


- we will now see how ADMM can generate a distributed algorithm for

$$\underset{x}{\text{minimize}}\, f(x) := \frac{f_1(x) + \cdots + f_n(x)}{n}$$

step 1: choose a direction for each edge in the network

- example:



  - vertex set is $\mathcal{V} = \{1, 2, \ldots, 5\}$
  - arc set is $\mathcal{A} = \{(1,4), (1,3), (2,1), (3,2), (2,5)\}$
  - for an arc $a \in \mathcal{A}$: $S(a)$:= source of arc a, $T(a)$:= sink or arc a ($S(1,4) = 1$, $T(1,4) = 4$, $S(1,3) = 1$, $T(1,3) = 3$, $\ldots$)

step 2: clone variables

- we want to solve

$$\underset{x}{\text{minimize}} \sum_v f_v(x)$$

- reformulate as

$$\begin{array}{ll} \underset{y_v, z_a}{\text{minimize}} & \sum_v f_v(y_v) \\ \text{subject to} & y_{S(a)} = z_a, \quad a \in \mathcal{A} \\ & y_{T(a)} = z_a, \quad a \in \mathcal{A} \end{array}$$

- because network is connected, constraints make all y_v's the same:

$$y_v = y_u, \quad \text{for all } v, u \in \mathcal{V}$$

step 3: apply ADMM to reformulated problem

$$\begin{array}{ll} \underset{y_v, z_a}{\text{minimize}} & \underbrace{\sum_v f_v(y_v)}_{g(y)} + \underbrace{0}_{h(z)} \\ \text{subject to} & y_{S(a)} = z_a, \quad a \in \mathcal{A} \\ & y_{T(a)} = z_a, \quad a \in \mathcal{A} \end{array}$$

- (primal) variables are $y = \{y_v\}_{v \in \mathcal{V}}$ and $z = \{z_a\}_{a \in \mathcal{A}}$
- associate lagrange multiplier s_a with constraint $y_{S(a)} = z_a$
- associate lagrange multiplier t_a with constraint $y_{T(a)} = z_a$

- augmented lagrangian function is

$$\begin{aligned} L\left(y_v, z_a; s_a, t_a\right) &= \sum_v f_v(y_v) + \\ &\sum_a s_a^T \left(y_{S(a)} - z_a\right) + \frac{\rho}{2} \sum_a \left\|y_{S(a)} - z_a\right\|^2 + \\ &\sum_a t_a^T \left(y_{T(a)} - z_a\right) + \frac{\rho}{2} \sum_a \left\|y_{T(a)} - z_a\right\|^2 \end{aligned}$$

- the ADMM iterations are

$$\begin{aligned} y(t+1) &= \arg\min_y L\left(y_v, z_a(t); s_a(t), t_a(t)\right) && (1) \\ z(t+1) &= \arg\min_z L\left(y_v(t+1), z; s_a(t), t_a(t)\right) && (2) \\ s_a(t+1) &= s_a(t) + \rho\left(y_{S(a)}(t+1) - z_a(t+1)\right) && (3) \\ t_a(t+1) &= t_a(t) + \rho\left(y_{T(a)}(t+1) - z_a(t+1)\right) && (4) \end{aligned}$$

step 4: simplify the iterations

- from (2), we get

$$z_a(t+1) = \frac{y_{S(a)}(t+1) + y_{T(a)}(t+1)}{2} - \frac{s_a(t) + t_a(t)}{2\rho} \qquad (5)$$

- plugging (5) into (3) and (4) gives

$$s_a(t+1) = s_a(t) + \rho\left(\frac{y_{S(a)}(t+1) - y_{T(a)}(t+1)}{2} + \frac{s_a(t) + t_a(t)}{2\rho}\right) \qquad (6)$$

$$t_a(t+1) = t_a(t) + \rho\left(\frac{y_{T(a)}(t+1) - y_{S(a)}(t+1)}{2} + \frac{s_a(t) + t_a(t)}{2\rho}\right) \qquad (7)$$

- trick: if $s_a(0) = 0$ and $t_a(0) = 0$ for $a \in \mathcal{A}$, then (6) and (7) imply

$$s_a(t) = -t_a(t), \quad \text{for } t \geq 0 \tag{8}$$

- plugging (8) into (5)–(7) gives

$$\begin{aligned} z_a(t+1) &= \frac{y_{S(a)}(t+1) + y_{T(a)}(t+1)}{2} && (9) \\ s_a(t+1) &= s_a(t) + \rho\frac{y_{S(a)}(t+1) - y_{T(a)}(t+1)}{2} && (10) \\ t_a(t+1) &= t_a(t) + \rho\frac{y_{T(a)}(t+1) - y_{S(a)}(t+1)}{2} && (11) \end{aligned}$$

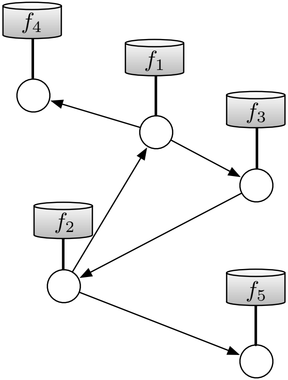
- rewrite (1) as a separable problem across agents:

$$y(t+1) = \arg\min_y \sum_v f_v(y_v) + \Bigg(\underbrace{\sum_{a \in \mathcal{S}(v)} s_a(t) + \sum_{a \in \mathcal{T}(v)} t_a(t)}_{\lambda_v(t)} \Bigg)^T y_v + \frac{\rho}{2} \sum_{a \in \mathcal{S}(v)} \|y_v - z_a(t)\|^2 + \frac{\rho}{2} \sum_{a \in \mathcal{T}(v)} \|y_v - z_a(t)\|^2$$

where

- $\mathcal{S}(v)$ = set of arcs that leave v
- $\mathcal{T}(v)$ = set of arcs that arrive at v

- example:



  - $\mathcal{S}(1) = \{(1,4),(1,3)\}$
  - $\mathcal{T}(1) = \{(2,1)\}$
  - $\mathcal{S}(2) = \{(2,1),(2,5)\}$
  - $\mathcal{T}(2) = \{(3,2)\}$
  - $\mathcal{S}(4) = \emptyset$
  - ...

- equivalently:

$$y_v(t+1) = \arg\min_{y_v} f_v\,(y_v) + \lambda_v(t)^T y_v + \frac{\rho}{2} \sum_{u \sim v} \left\| y_v - \frac{y_v(t) + y_u(t)}{2} \right\|^2 \tag{12}$$

- update (12) does not depend on $s_a(t)$ or $t_a(t)$; only on $\lambda_v(t)$

- we can find a recursion for $\lambda_v(t)$:

$$\begin{aligned} \lambda_v(t+1) &= \sum_{a \in \mathcal{S}(v)} s_a(t+1) + \sum_{a \in \mathcal{T}(v)} t_a(t+1) \\ &= \lambda_v(t) + \rho \sum_{u \sim v} y_v(t+1) - y_u(t+1) \end{aligned} \tag{13}$$

(we used (10) and (11))

- final algorithm:

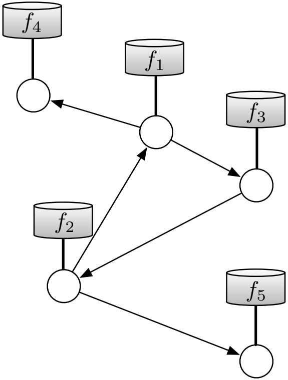
$$\begin{aligned} y_v(0) &= \text{initialization} \\ \lambda_v(0) &= 0 \\ y_v(t+1) &= \arg\min_{y_v} f_v\left(y_v\right) + \lambda_v(t)^T y_v + \frac{\rho}{2} \sum_{u \sim v} \left\| y_v - \frac{y_v(t) + y_u(t)}{2} \right\|^2 \\ \lambda_v(t+1) &= \lambda_v(t) + \rho \sum_{u \sim v} y_v(t+1) - y_u(t+1) \end{aligned}$$

for $t = 0, 1, 2, \ldots$

- algorithm is distributed

- agent v manages $y_v(t)$ and $\lambda_v(t)$

Example: distributed logistic regression



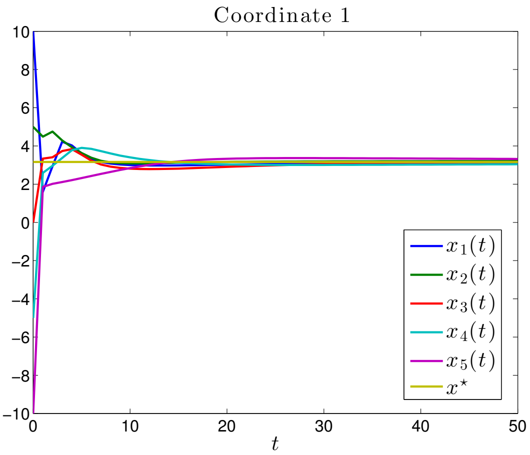
- dataset of agent i:

$$\left\{(a_i(k), b_i(k)) \in \mathbf{R}^2 \times \{0,1\} \;:\; k = 1, \ldots, 10\right\}$$

- private function of agent i: $f_i \;:\; \mathbf{R}^2 \to \mathbf{R}$

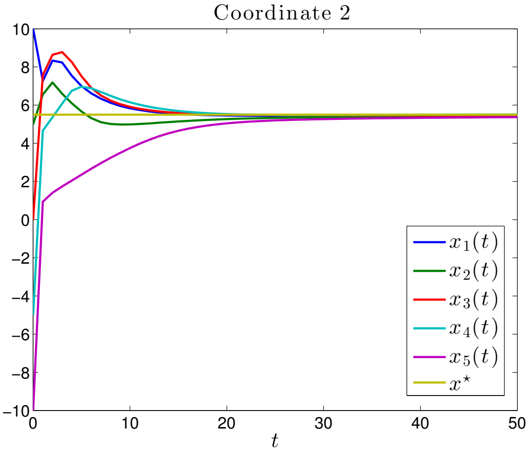
$$f_i(x) = \sum_{k=1}^{10} -b_k(i) a_k(i)^T x + \log\left(1 + e^{a_k(i)^T x}\right)$$

$\rho = 0.01$

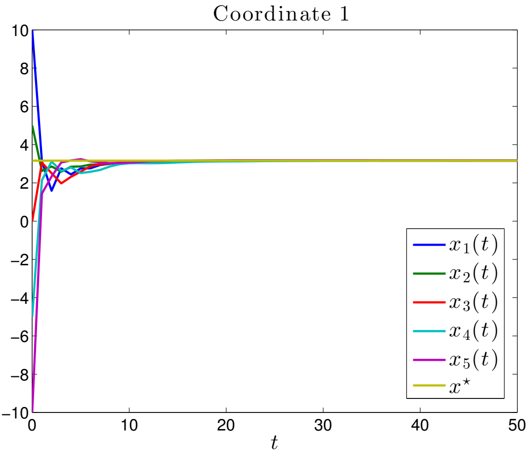


Coordinate 1
10
8
6
4
2
0
−2
−4
−6
−8
−10
0
10
20
30
40
50
t
$x_1(t)$
$x_2(t)$
$x_3(t)$
$x_4(t)$
$x_5(t)$
$x^\star$

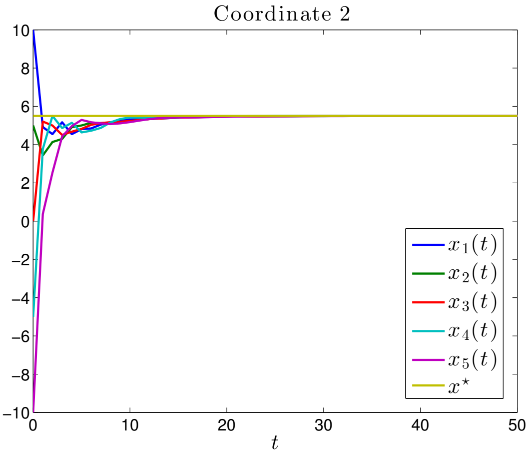
$\rho = 0.01$


Coordinate 2
10
8
6
4
2
0
−2
−4
−6
−8
−10
0
10
20
30
40
50
t
$x_1(t)$
$x_2(t)$
$x_3(t)$
$x_4(t)$
$x_5(t)$
$x^\star$

$\rho = 0.1$

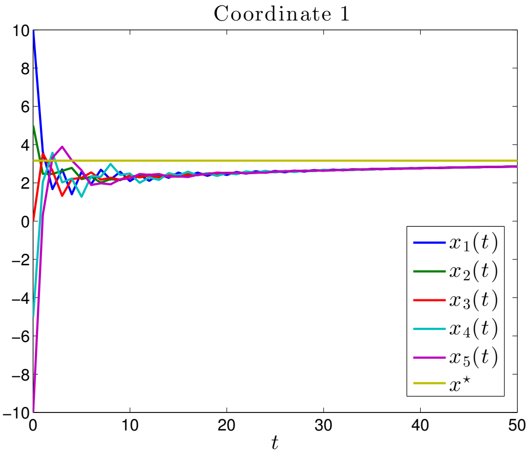

Coordinate 1
10
8
6
4
2
0
−2
−4
−6
−8
−10
0
10
20
30
40
50
t
$x_1(t)$
$x_2(t)$
$x_3(t)$
$x_4(t)$
$x_5(t)$
$x^\star$

$$\rho = 0.1$$



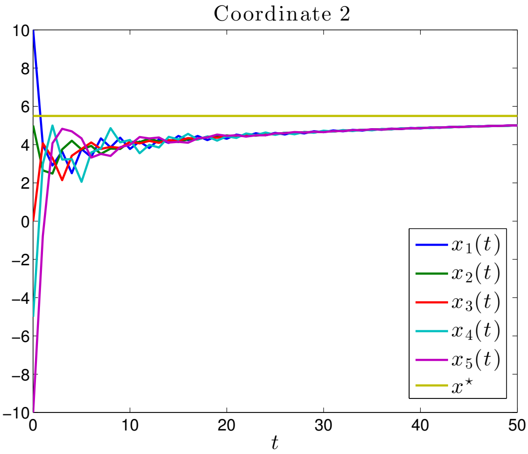
Coordinate 2
10
8
6
4
2
0
–2
–4
–6
–8
–10
0
10
20
30
40
50
t
$x_1(t)$
$x_2(t)$
$x_3(t)$
$x_4(t)$
$x_5(t)$
$x^\star$

$\rho = 1$



Coordinate 1
10
8
6
4
2
0
–2
–4
–6
–8
–10
0
10
20
30
40
50
t
$x_1(t)$
$x_2(t)$
$x_3(t)$
$x_4(t)$
$x_5(t)$
$x^\star$

$\rho = 1$


Coordinate 2
10
8
6
4
2
0
−2
−4
−6
−8
−10
0
10
20
30
40
50
t
$x_1(t)$
$x_2(t)$
$x_3(t)$
$x_4(t)$
$x_5(t)$
$x^\star$

To know more (a tiny slice of available work)

- some (sub)gradient methods with shrinking stepsize:
  - A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. on Aut. Control*, 54(1), 2009.
  - K. Kvaternik and L. Pavel, "Lyapunov analysis of a distributed optimization scheme," *5th Int. Conf. on Network Games, Control and Opt., 2011.*

- some gradient algorithms for $C^2(m, M)$ with constant stepsize:
  - D. Jakovetić *et al.*, "Linear convergence rate of a class of distributed augmented lagrangian algorithms," *IEEE Trans. on Aut. Control*, 60(4), 2015.
  - W. Shi *et al.*, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," 25(2), *SIAM Journal on Opt.*, 2015.
  - G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," `https://arxiv.org/abs/1605.07112`, 2016.

- some gradient algorithms for $C^1(m, M)$:
  - D. Jakovetić et al., "Fast distributed gradient methods," *IEEE Trans, on Aut. Control*, 59(5), 2014.
  - W. Shi *et al.*, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," 25(2), *SIAM Journal on Opt.*, 2015.
  - G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," `https://arxiv.org/abs/1605.07112`, 2016.

- some papers on ADMM:
  - I. Schizas *et al.*, "Consensus in *ad hoc* WSNs with noisy links," *IEEE Trans. on Sig. Proc*, 56(1), 2008.
  - J. Bazerque and G. Giannakis,"Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. on Sig. Proc.*, 58(3), 2010.
  - S. Boyd *et al.*, *Distributed optimization and statistical learning via the ADMM*, Foundations and Trends in Machine Learning, 3(1), 2011.