Large Scale Machine Learning Over Networks

Francis Bach

INRIA - Ecole Normale Supérieure, Paris, France

Joint work with Kevin Scaman, Hadrien Hendrikx, Laurent Massoulié, Sébastien Bubeck, Yin-Tat Lee

PAISS Summer school - October 5, 2019

Scientific context

- **Proliferation of digital data**
  - Personal data
  - Industry
  - Scientific: from bioinformatics to humanities

- **Need for automated processing of massive data**

Scientific context

- **Proliferation of digital data**
  - Personal data
  - Industry
  - Scientific: from bioinformatics to humanities

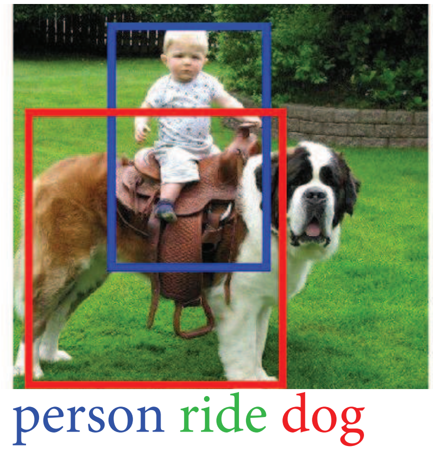
- **Need for automated processing of massive data**

- **Series of "hypes"**

  Big data $\rightarrow$ Data science $\rightarrow$ Machine Learning
  $\rightarrow$ Deep Learning $\rightarrow$ Artificial Intelligence

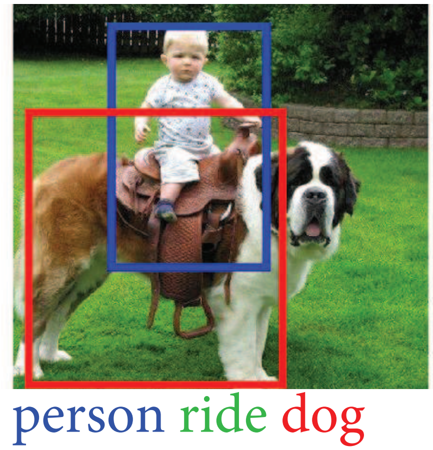
Recent progress in perception (vision, audio, text)

From translate.google.fr



From Peyré et al. (2017)

Recent progress in perception (vision, audio, text)



From translate.google.fr

From Peyré et al. (2017)

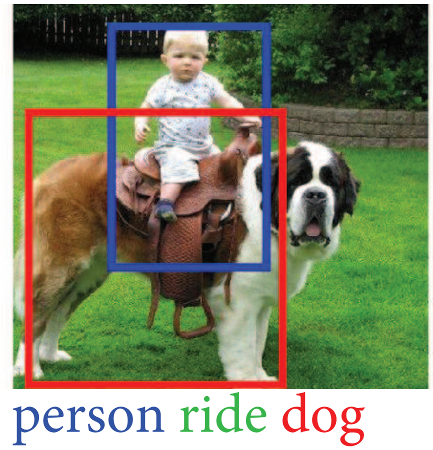
(1) **Massive data**

(2) **Computing power**

(3) **Methodological and scientific progress**

Recent progress in perception (vision, audio, text)

From translate.google.fr



From Peyré et al. (2017)

(1) **Massive data**

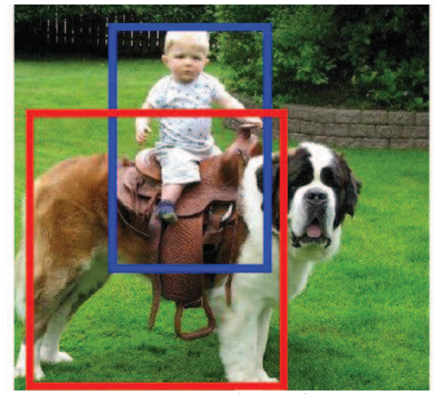
(2) **Computing power**

(3) **Methodological and scientific progress**

"Intelligence" = models + algorithms + data
+ computing power

Recent progress in perception (vision, audio, text)

From translate.google.fr



person ride dog

From Peyré et al. (2017)

(1) **Massive data**

(2) **Computing power**

(3) **Methodological and scientific progress**

"Intelligence" = models + algorithms + data
+ computing power

Outline

1. **Parametric supervised learning on a single machine**
   - Machine learning $\approx$ optimization of finite sums
   - From batch to stochastic gradient methods
   - Linearly-convergent stochastic methods for convex problems

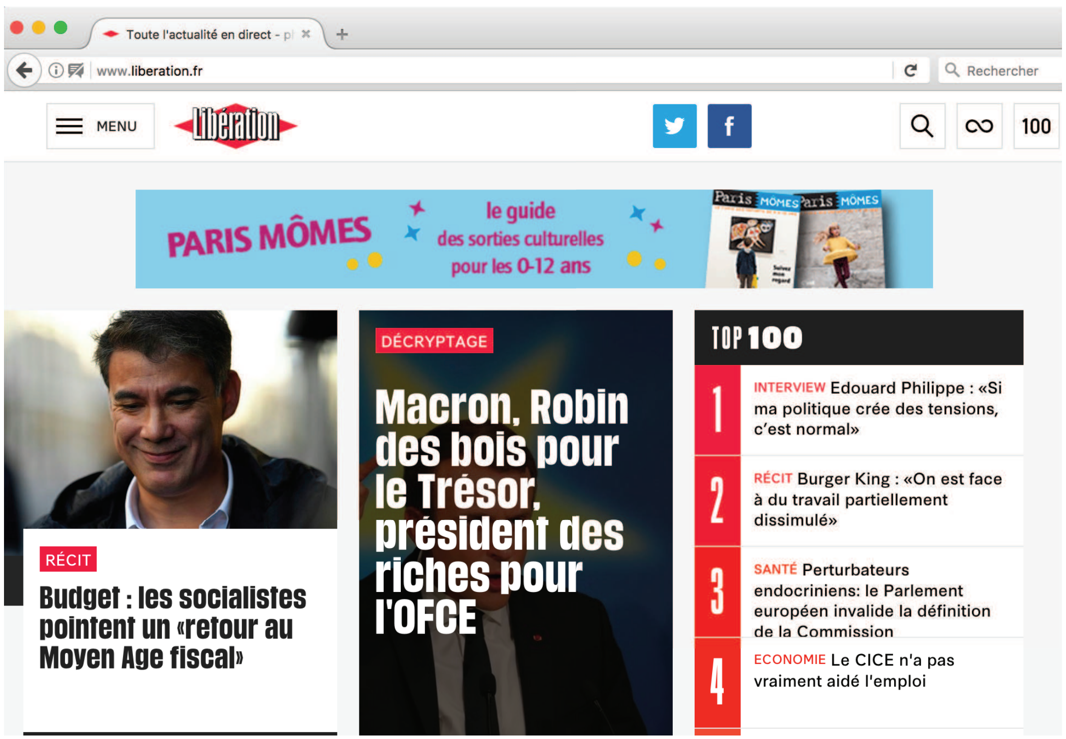
2. **Machine learning over networks**
   - Centralized and decentralized methods
   - From network averaging to optimization
   - Distributing the fastest single machine algorithms

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$

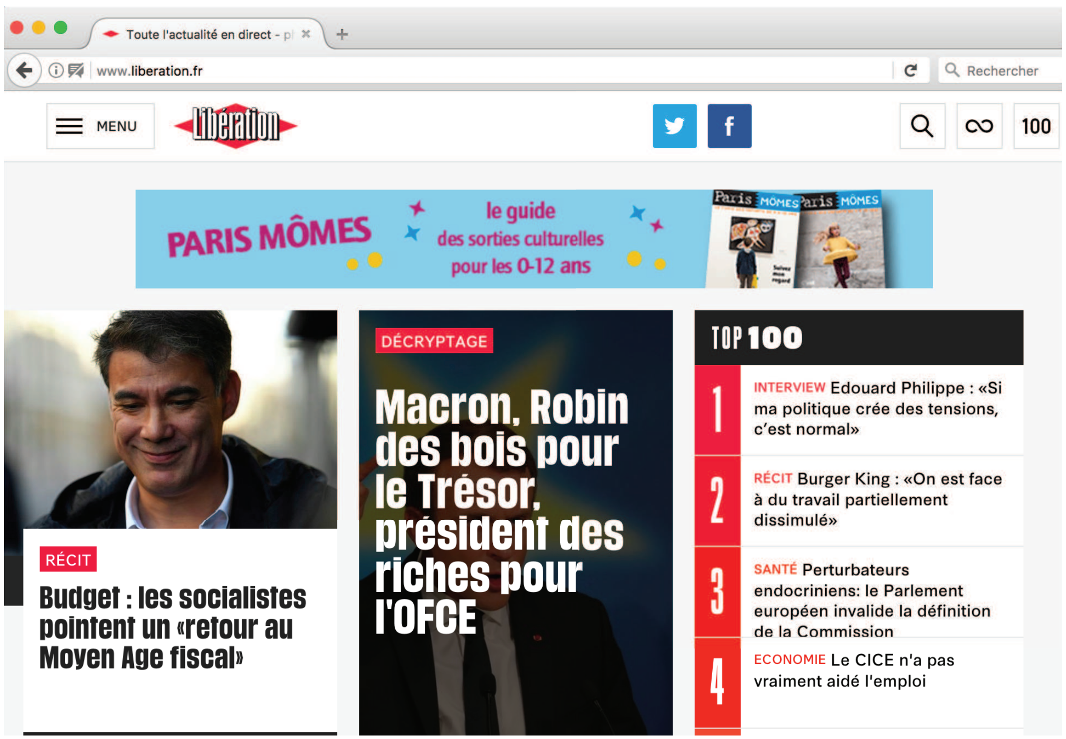


- **Advertising**: $n > 10^9$
  - $\Phi(x) \in \{0, 1\}^d$, $d > 10^9$
  - Navigation history + ad

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$

- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$



- **Advertising**: $n > 10^9$
  - $\Phi(x) \in \{0, 1\}^d$, $d > 10^9$
  - Navigation history + ad

- Linear predictions
  - $h(x, \theta) = \theta^\top \Phi(x)$

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$

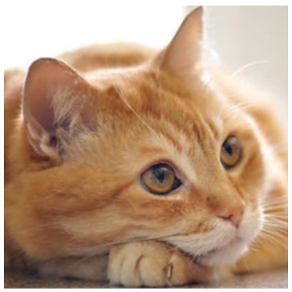
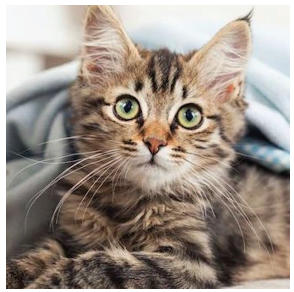
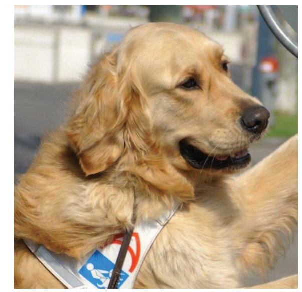
x_1 x_2 x_3 x_4 x_5 x_6

$y_1 = 1$ $y_2 = 1$ $y_3 = 1$ $y_4 = -1$ $y_5 = -1$ $y_6 = -1$

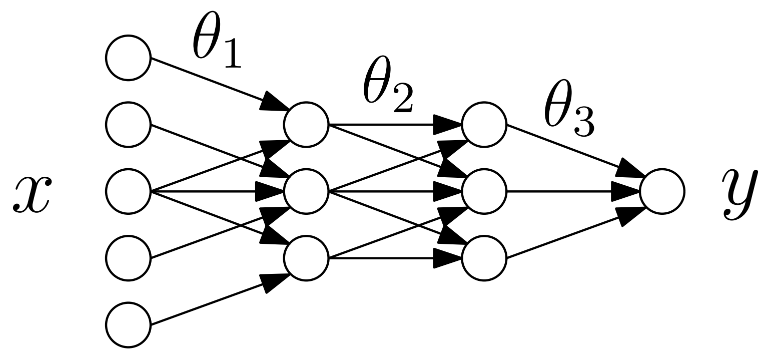
Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$

- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$

x_1	x_2	x_3	x_4	x_5	x_6
					
$y_1 = 1$	$y_2 = 1$	$y_3 = 1$	$y_4 = -1$	$y_5 = -1$	$y_6 = -1$

  - Neural networks ($n, d > 10^6$): $h(x, \theta) = \theta_m^\top \sigma(\theta_{m-1}^\top \sigma(\cdots \theta_2^\top \sigma(\theta_1^\top x))$



Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$
- **(regularized) empirical risk minimization**:

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \ \ell\big(y_i, h(x_i, \theta)\big) \quad + \quad \lambda \Omega(\theta)$$

data fitting term + regularizer

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$

- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$

- **(regularized) empirical risk minimization**:

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{2n} \sum_{i=1}^{n} \big(y_i - h(x_i, \theta)\big)^2 \quad + \quad \lambda \Omega(\theta)$$

(least-squares regression)

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$
- **(regularized) empirical risk minimization**:

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \log\big(1 + \exp(-y_i h(x_i, \theta))\big) \quad + \quad \lambda \Omega(\theta)$$

(logistic regression)

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$
- **(regularized) empirical risk minimization**:

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \quad \ell\big(y_i, h(x_i, \theta)\big) \quad + \quad \lambda \Omega(\theta)$$

data fitting term $+$ regularizer

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$
- **(regularized) empirical risk minimization**:

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \Big\{ \ell\big(y_i, h(x_i, \theta)\big) \quad + \quad \lambda \Omega(\theta) \Big\} \qquad = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

data fitting term + regularizer

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$
- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$
- **(regularized) empirical risk minimization**:

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \Big\{ \ell\big(y_i, h(x_i, \theta)\big) \quad + \quad \lambda \Omega(\theta) \Big\} \qquad = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

data fitting term $+$ regularizer

- Optimization: optimization of regularized risk training cost

Parametric supervised machine learning

- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$, **i.i.d.**

- **Prediction function** $h(x, \theta) \in \mathbb{R}$ parameterized by $\theta \in \mathbb{R}^d$

- **(regularized) empirical risk minimization**:

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \Big\{ \ell\big(y_i, h(x_i, \theta)\big) \quad + \quad \lambda \Omega(\theta) \Big\} \qquad = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

data fitting term $+$ regularizer

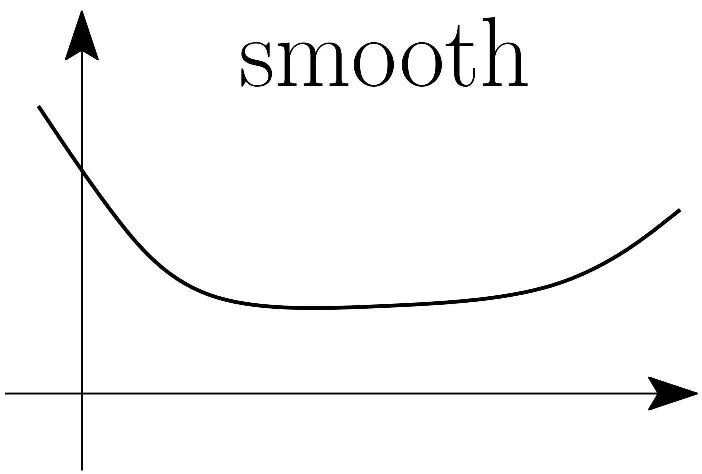
- Optimization: optimization of regularized risk — training cost

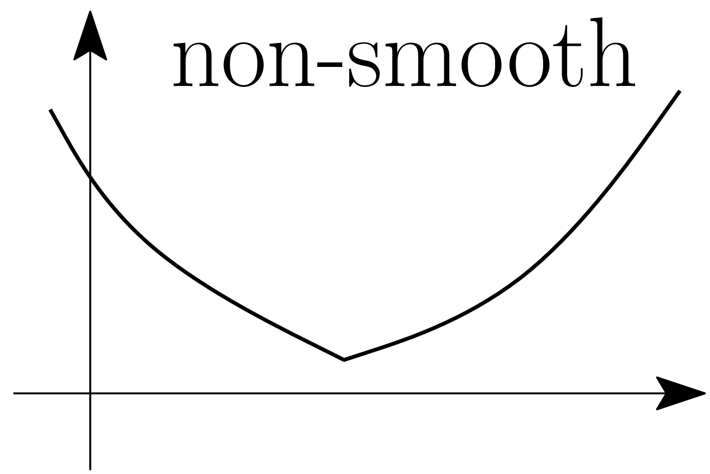
- Statistics: guarantees on $\mathbb{E}_{p(x,y)} \ell(y, h(x, \theta))$ — testing cost

Smoothness and (strong) convexity

- A function $g : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if and only if it is twice differentiable and

$$\forall \theta \in \mathbb{R}^d, \ \left| \text{eigenvalues}\big[g''(\theta)\big] \right| \leqslant L$$





Smoothness and (strong) convexity

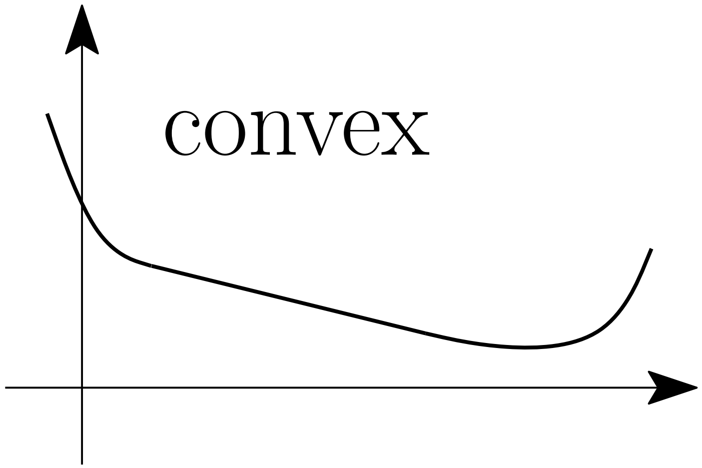
- A function $g : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if and only if it is twice differentiable and

$$\forall \theta \in \mathbb{R}^d, \ \big|\text{eigenvalues}\big[g''(\theta)\big]\big| \leqslant L$$

- **Machine learning**
  - with $g(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, h(x_i, \theta))$
  - Smooth prediction function $\theta \mapsto h(x_i, \theta)$ + smooth loss

Smoothness and (strong) convexity

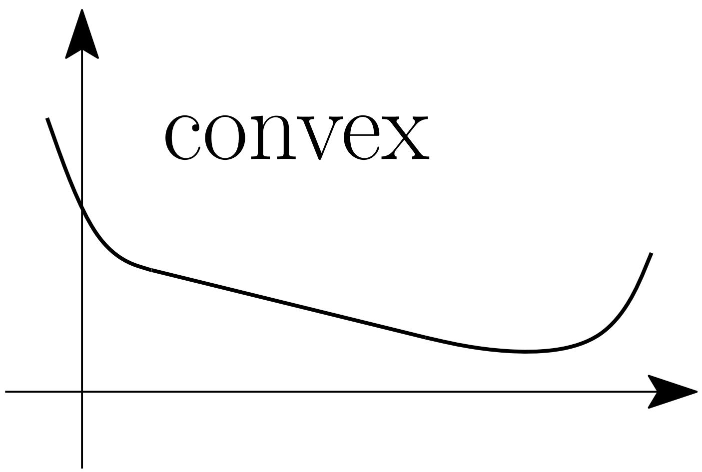
- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if
$$\forall \theta \in \mathbb{R}^d,\ \text{eigenvalues}\big[g''(\theta)\big] \geqslant 0$$

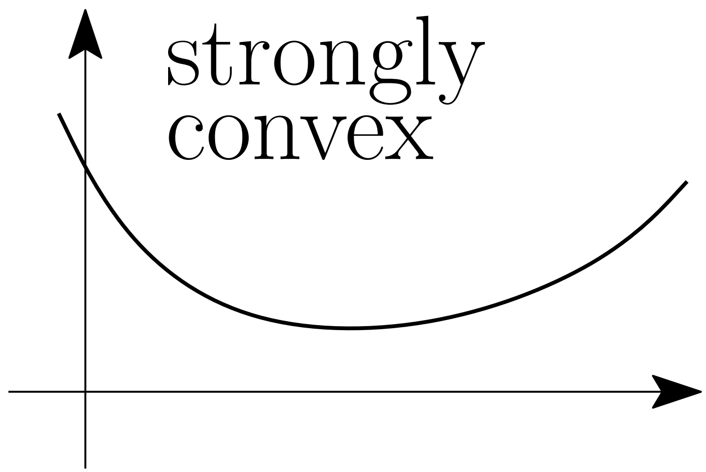


Smoothness and (strong) convexity

- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if

$$\forall \theta \in \mathbb{R}^d, \ \text{eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$



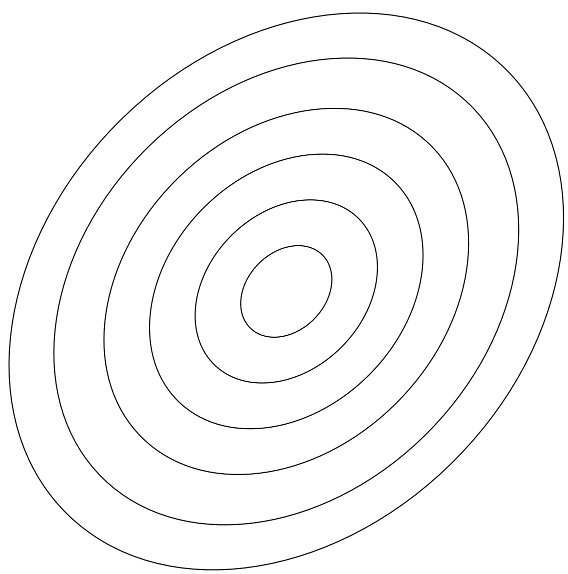


Smoothness and (strong) convexity

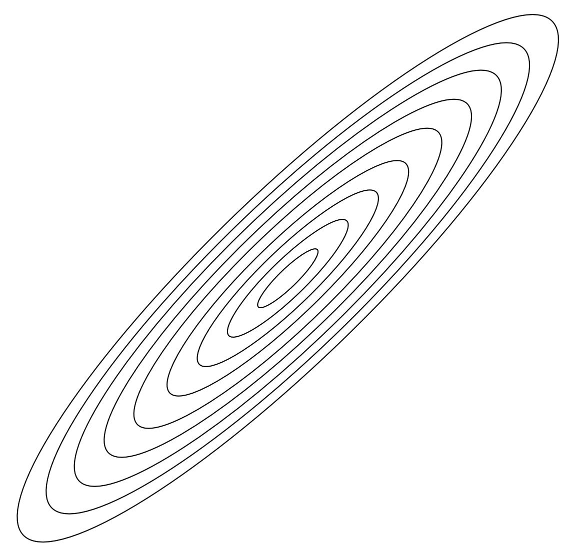
- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if

$$\forall \theta \in \mathbb{R}^d, \ \text{eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$

  - Condition number $\kappa = L/\mu \geqslant 1$



(small $\kappa = L/\mu$)



(large $\kappa = L/\mu$)

Smoothness and (strong) convexity

- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if
$$\forall \theta \in \mathbb{R}^d, \ \text{eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$

- **Convexity in machine learning**
  - With $g(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, h(x_i, \theta))$
  - Convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$

Smoothness and (strong) convexity

- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if

$$\forall \theta \in \mathbb{R}^d, \ \text{eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$

- **Convexity in machine learning**
  - With $g(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, h(x_i, \theta))$
  - Convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$

- **Relevance of convex optimization**
  - Easier design and analysis of algorithms
  - Global minimum vs. local minimum vs. stationary points
  - Gradient-based algorithms only need convexity for their analysis

Smoothness and (strong) convexity

- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if
$$\forall \theta \in \mathbb{R}^d, \ \text{eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$

- **Strong convexity in machine learning**
  - With $g(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, h(x_i, \theta))$
  - Strongly convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$

Smoothness and (strong) convexity

- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if

$$\forall \theta \in \mathbb{R}^d, \text{ eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$

- **Strong convexity in machine learning**
  - With $g(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, h(x_i, \theta))$
  - Strongly convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$
  - Invertible covariance matrix $\frac{1}{n}\sum_{i=1}^n \Phi(x_i)\Phi(x_i)^\top \Rightarrow n \geqslant d$
  - Even when $\mu > 0$, μ may be arbitrarily small!

Smoothness and (strong) convexity

- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if

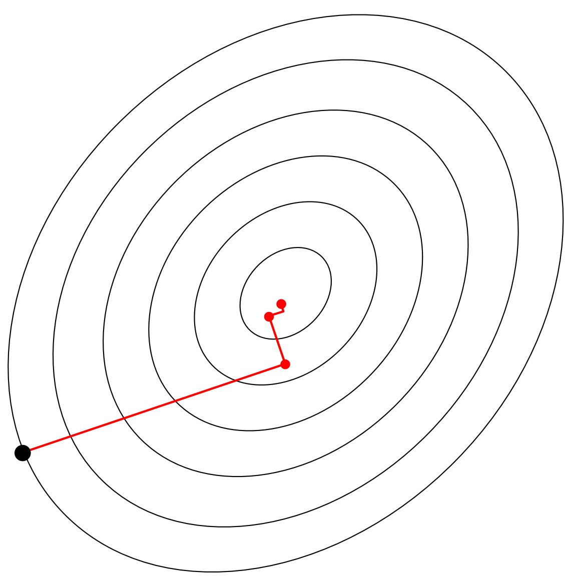
$$\forall \theta \in \mathbb{R}^d, \ \text{eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$

- **Strong convexity in machine learning**
  - With $g(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, h(x_i, \theta))$
  - Strongly convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$
  - Invertible covariance matrix $\frac{1}{n}\sum_{i=1}^n \Phi(x_i)\Phi(x_i)^\top \Rightarrow n \geqslant d$
  - Even when $\mu > 0$, μ may be arbitrarily small!

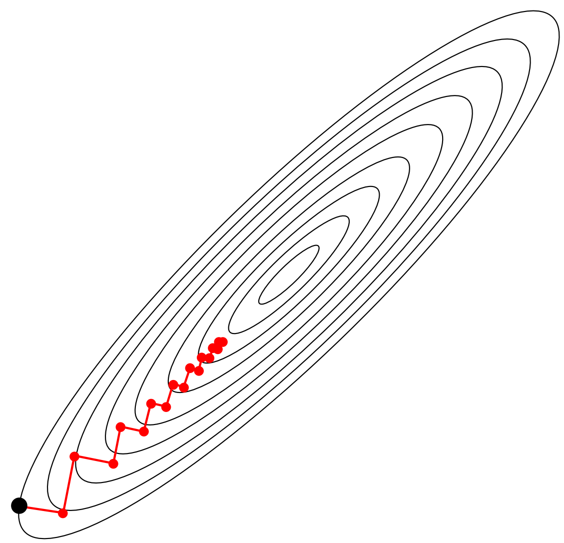
- **Adding regularization by $\frac{\mu}{2}\|\theta\|^2$**
  - creates additional bias unless μ is small, but reduces variance
  - Typically $L/\sqrt{n} \geqslant \mu \geqslant L/n \Rightarrow \kappa \in [\sqrt{n}, n]$

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and L-smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$ *(line search)*



(small $\kappa = L/\mu$)

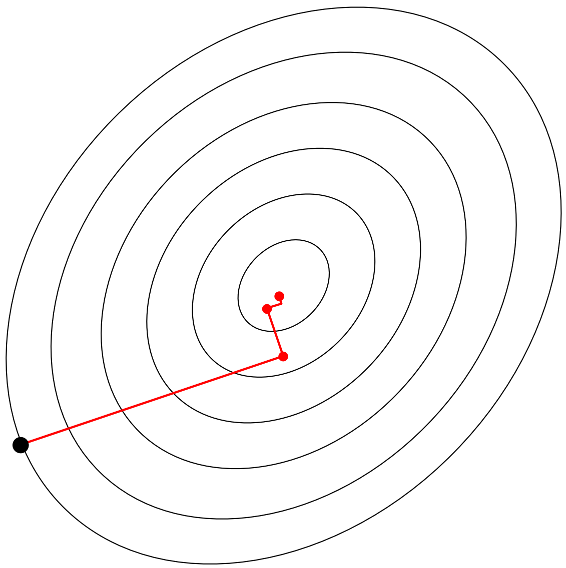


(large $\kappa = L/\mu$)

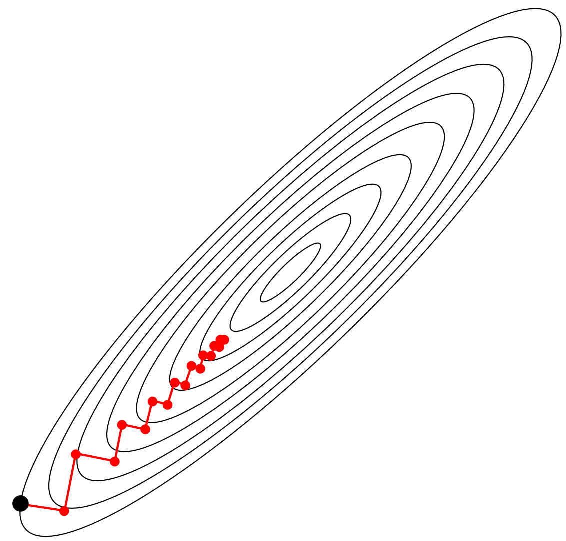
Iterative methods for minimizing smooth functions

- **Assumption**: g convex and L-smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$ *(line search)*

$$g(\theta_t) - g(\theta_*) \leqslant O(1/t)$$
$$g(\theta_t) - g(\theta_*) \leqslant O((1-1/\kappa)^t) = O(e^{-t/\kappa}) \text{ if } \mu\text{-strongly convex}$$



(small $\kappa = L/\mu$)



(large $\kappa = L/\mu$)

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and L-smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$

  $g(\theta_t) - g(\theta_*) \leqslant O(1/t)$
  $g(\theta_t) - g(\theta_*) \leqslant O((1-1/\kappa)^t) = O(e^{-t/\kappa})$ if μ-strongly convex

- **Acceleration** (Nesterov, 1983): second-order recursion

$$\theta_t = \eta_{t-1} - \gamma_t g'(\eta_{t-1}) \text{ and } \eta_t = \theta_t + \delta_t(\theta_t - \theta_{t-1})$$

  – Good choice of momentum term $\delta_t \in [0, 1)$

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and L-smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t\, g'(\theta_{t-1})$

  $g(\theta_t) - g(\theta_*) \leqslant O(1/t)$
  $g(\theta_t) - g(\theta_*) \leqslant O((1-1/\kappa)^t) = O(e^{-t/\kappa})$ if μ-strongly convex

- **Acceleration** (Nesterov, 1983): second-order recursion

$$\theta_t = \eta_{t-1} - \gamma_t g'(\eta_{t-1}) \text{ and } \eta_t = \theta_t + \delta_t(\theta_t - \theta_{t-1})$$

  - Good choice of momentum term $\delta_t \in [0,1)$
    $g(\theta_t) - g(\theta_*) \leqslant O(1/t^2)$
    $g(\theta_t) - g(\theta_*) \leqslant O((1-1/\sqrt{\kappa})^t) = O(e^{-t/\sqrt{\kappa}})$ if μ-strongly convex
  - Optimal rates after $t = O(d)$ iterations (Nesterov, 2004)

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$
  - $O(1/t)$ convergence rate for convex functions
  - $O(e^{-t/\kappa})$ *linear* if strongly-convex

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$
  - $O(1/t)$ convergence rate for convex functions
  - $O(e^{-t/\kappa})$ *linear* if strongly-convex
- **Newton method**: $\theta_t = \theta_{t-1} - g''(\theta_{t-1})^{-1} g'(\theta_{t-1})$
  - $O\big(e^{-\rho 2^t}\big)$ *quadratic* rate

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and smooth on $\mathbb{R}^d$

- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$
  - $O(1/t)$ convergence rate for convex functions
  - $O(e^{-t/\kappa})$ *linear* if strongly-convex $\Leftrightarrow$ $O(\kappa \log \frac{1}{\varepsilon})$ iterations

- **Newton method**: $\theta_t = \theta_{t-1} - g''(\theta_{t-1})^{-1} g'(\theta_{t-1})$
  - $O\big(e^{-\rho 2^t}\big)$ *quadratic* rate $\Leftrightarrow$ $O(\log \log \frac{1}{\varepsilon})$ iterations

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and smooth on $\mathbb{R}^d$

- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$
  - $O(1/t)$ convergence rate for convex functions
  - $O(e^{-t/\kappa})$ *linear* if strongly-convex $\Leftrightarrow$ complexity $= O(nd \cdot \kappa \log \frac{1}{\varepsilon})$

- **Newton method**: $\theta_t = \theta_{t-1} - g''(\theta_{t-1})^{-1} g'(\theta_{t-1})$
  - $O\big(e^{-\rho 2^t}\big)$ *quadratic* rate $\Leftrightarrow$ complexity $= O((nd^2 + d^3) \cdot \log \log \frac{1}{\varepsilon})$

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and smooth on $\mathbb{R}^d$

- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$
  - $O(1/t)$ convergence rate for convex functions
  - $O(e^{-t/\kappa})$ *linear* if strongly-convex $\Leftrightarrow$ complexity $= O(nd \cdot \kappa \log \frac{1}{\varepsilon})$

- **Newton method**: $\theta_t = \theta_{t-1} - g''(\theta_{t-1})^{-1} g'(\theta_{t-1})$
  - $O\big(e^{-\rho 2^t}\big)$ *quadratic* rate $\Leftrightarrow$ complexity $= O((nd^2 + d^3) \cdot \log\log \frac{1}{\varepsilon})$

- **Key insights for machine learning (Bottou and Bousquet, 2008)**
  1. No need to optimize below statistical error
  2. Cost functions are averages
  3. Testing error is more important than training error

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and smooth on $\mathbb{R}^d$

- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$
  - $O(1/t)$ convergence rate for convex functions
  - $O(e^{-t/\kappa})$ *linear* if strongly-convex $\Leftrightarrow$ complexity $= O(nd \cdot \kappa \log \frac{1}{\varepsilon})$

- **Newton method**: $\theta_t = \theta_{t-1} - g''(\theta_{t-1})^{-1} g'(\theta_{t-1})$
  - $O\big(e^{-\rho 2^t}\big)$ *quadratic* rate $\Leftrightarrow$ complexity $= O((nd^2 + d^3) \cdot \log \log \frac{1}{\varepsilon})$

- **Key insights for machine learning (Bottou and Bousquet, 2008)**
  1. No need to optimize below statistical error
  2. Cost functions are averages
  3. Testing error is more important than training error

Stochastic gradient descent (SGD) for finite sums

$$\min_{\theta\in\mathbb{R}^d} g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$$

- **Iteration**: $\theta_t = \theta_{t-1} - \gamma_t f'_{i(t)}(\theta_{t-1})$
  - Sampling with replacement: $i(t)$ random element of $\{1,\dots,n\}$
  - Polyak-Ruppert averaging: $\bar\theta_t = \frac{1}{t+1}\sum_{u=0}^{t}\theta_u$

Stochastic gradient descent (SGD) for finite sums

$$\min_{\theta \in \mathbb{R}^d} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

- **Iteration**: $\theta_t = \theta_{t-1} - \gamma_t f'_{i(t)}(\theta_{t-1})$
  - Sampling with replacement: $i(t)$ random element of $\{1, \dots, n\}$
  - Polyak-Ruppert averaging: $\bar{\theta}_t = \frac{1}{t+1} \sum_{u=0}^{t} \theta_u$

- **Convergence rate** if each f_i is convex L-smooth and g μ-strongly-convex:

$$\mathbb{E} g(\bar{\theta}_t) - g(\theta_*) \leqslant \begin{cases} O(1/\sqrt{t}) & \text{if } \gamma_t = 1/(L\sqrt{t}) \\ O(L/(\mu t)) = O(\kappa/t) & \text{if } \gamma_t = 1/(\mu t) \end{cases}$$

  - No adaptivity to strong-convexity in general
  - Running-time complexity: $O(d \cdot \kappa/\varepsilon)$

Stochastic gradient descent (SGD) for finite sums

$$\min_{\theta \in \mathbb{R}^d} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

- **Iteration**: $\theta_t = \theta_{t-1} - \gamma_t f'_{i(t)}(\theta_{t-1})$
  - Sampling with replacement: $i(t)$ random element of $\{1, \dots, n\}$
  - Polyak-Ruppert averaging: $\bar{\theta}_t = \frac{1}{t+1} \sum_{u=0}^{t} \theta_u$

- **Convergence rate** if each f_i is convex L-smooth and g μ-strongly-convex:

$$\mathbb{E} g(\bar{\theta}_t) - g(\theta_*) \leqslant \begin{cases} O(1/\sqrt{t}) & \text{if } \gamma_t = 1/(L\sqrt{t}) \\ O(L/(\mu t)) = O(\kappa/t) & \text{if } \gamma_t = 1/(\mu t) \end{cases}$$

  - No adaptivity to strong-convexity in general
  - Running-time complexity: $O(d \cdot \kappa/\varepsilon)$

- **NB**: single pass leads to bounds on testing error

Stochastic vs. deterministic methods

- Minimizing $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i, \theta)\big) + \lambda\Omega(\theta)$

Stochastic vs. deterministic methods

- Minimizing $g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i, \theta)\big) + \lambda \Omega(\theta)$

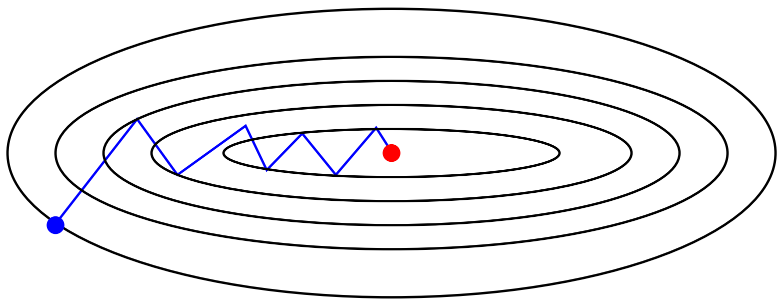
- Batch gradient descent: $\theta_t = \theta_{t-1} - \gamma \nabla g(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma}{n} \sum_{i=1}^{n} \nabla f_i(\theta_{t-1})$

  - Exponential convergence rate in $O(e^{-t/\kappa})$ for convex problems
  - Can be accelerated to $O(e^{-t/\sqrt{\kappa}})$ (Nesterov, 1983)
  - Iteration complexity is linear in n

Stochastic vs. deterministic methods

- Minimizing $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i,\theta)\big) + \lambda\Omega(\theta)$

- Batch gradient descent: $\theta_t = \theta_{t-1} - \gamma \nabla g(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma}{n}\sum_{i=1}^{n} \nabla f_i(\theta_{t-1})$



Stochastic vs. deterministic methods

- Minimizing $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i, \theta)\big) + \lambda\Omega(\theta)$

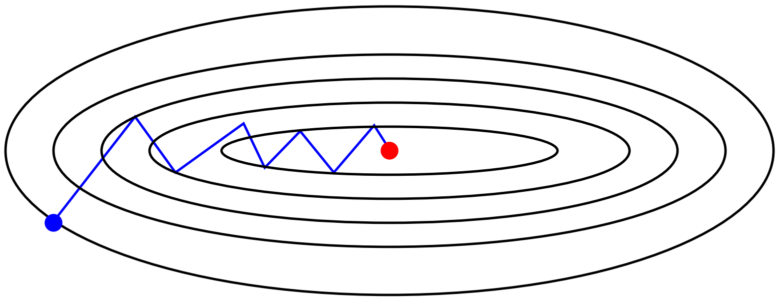
- Batch gradient descent: $\theta_t = \theta_{t-1} - \gamma \nabla g(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma}{n}\sum_{i=1}^{n} \nabla f_i(\theta_{t-1})$
  - Exponential convergence rate in $O(e^{-t/\kappa})$ for convex problems
  - Can be accelerated to $O(e^{-t/\sqrt{\kappa}})$ (Nesterov, 1983)
  - Iteration complexity is linear in n

- Stochastic gradient descent: $\theta_t = \theta_{t-1} - \gamma_t \nabla f_{i(t)}(\theta_{t-1})$
  - Sampling with replacement: $i(t)$ random element of $\{1, \dots, n\}$
  - Convergence rate in $O(\kappa/t)$
  - Iteration complexity is independent of n

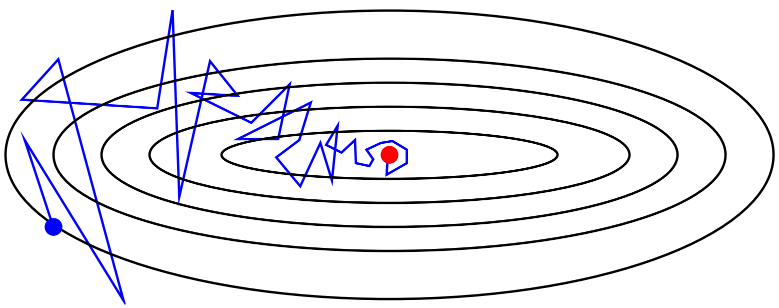
Stochastic vs. deterministic methods

- Minimizing $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i,\theta)\big) + \lambda\Omega(\theta)$

- Batch gradient descent: $\theta_t = \theta_{t-1} - \gamma\nabla g(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma}{n}\sum_{i=1}^{n}\nabla f_i(\theta_{t-1})$

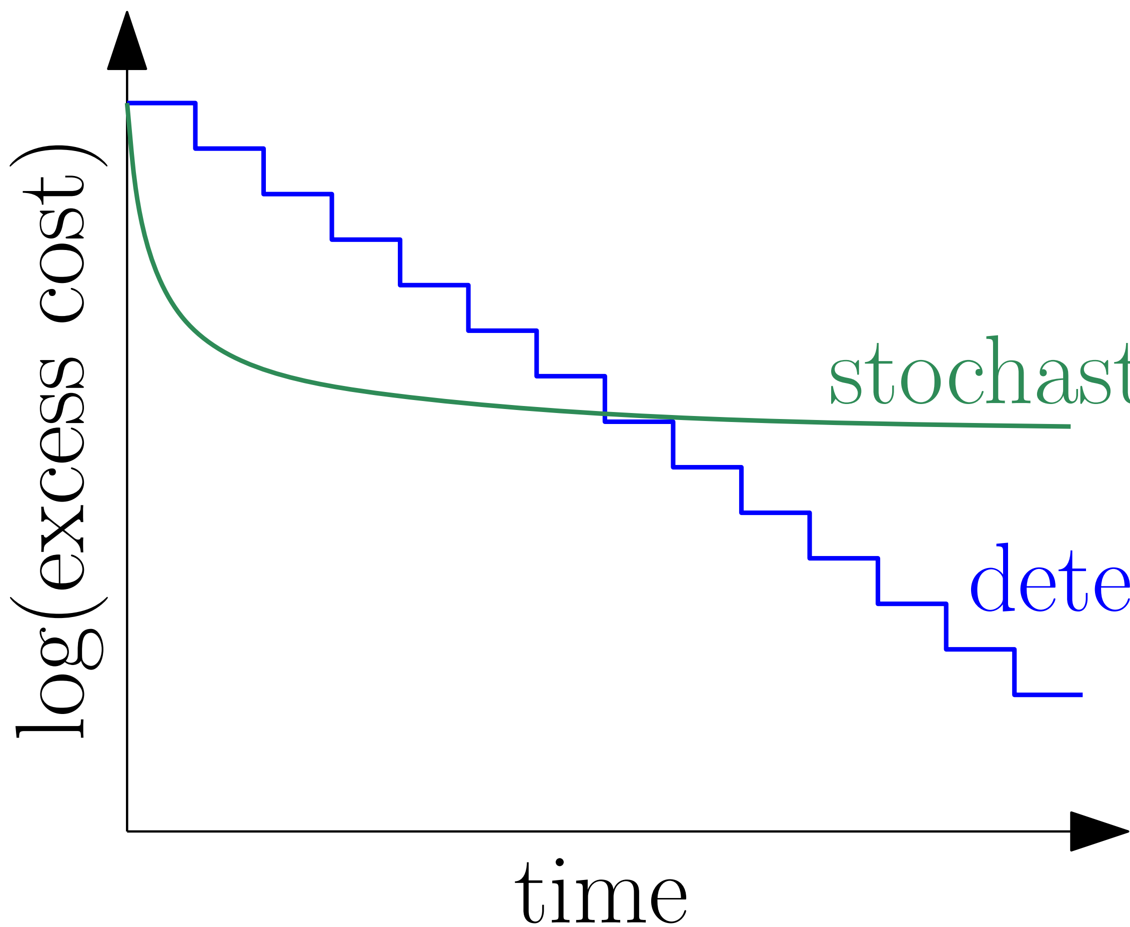


- Stochastic gradient descent: $\theta_t = \theta_{t-1} - \gamma_t \nabla f_{i(t)}(\theta_{t-1})$



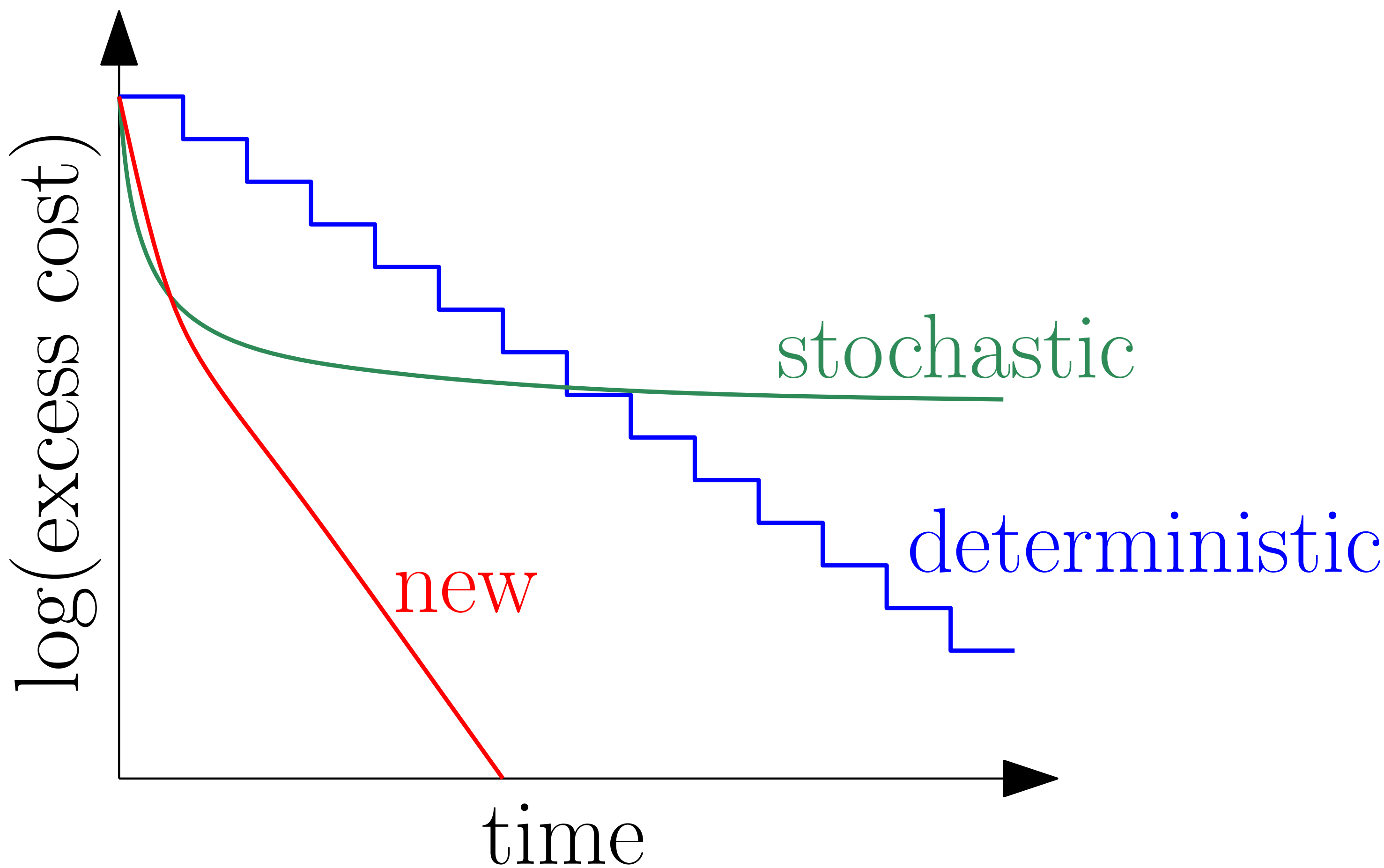
Stochastic vs. deterministic methods

- **Goal = best of both worlds**: Linear rate with $O(d)$ iteration cost
  Simple choice of step size



Stochastic vs. deterministic methods

- **Goal** = **best of both worlds**: Linear rate with $O(d)$ iteration cost
  Simple choice of step size



Recent progress in single machine optimization

- **Variance reduction**
  - Exponential convergence with $O(d)$ iteration cost
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014), etc...

$$\theta_t = \theta_{t-1} - \gamma \Big[\nabla f_{i(t)}(\theta_{t-1}) \qquad\qquad\qquad\qquad \Big]$$

Recent progress in single machine optimization

- **Variance reduction**
  - Exponential convergence with $O(d)$ iteration cost
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014), etc...

$$\theta_t = \theta_{t-1} - \gamma\Big[\nabla f_{i(t)}(\theta_{t-1}) + \frac{1}{n}\sum_{i=1}^{n} y_i^{t-1} - y_{i(t)}^{t-1}\Big]$$

(with y_i^t stored value at time t of gradient of the i-th function)

Recent progress in single machine optimization

- **Variance reduction**
  - Exponential convergence with $O(d)$ iteration cost
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014), etc...

- **Running-time to reach precision** ε (with $\kappa =$ condition number)

Stochastic gradient descent	$d\times$	κ	$\times$	$\frac{1}{\varepsilon}$
Gradient descent	$d\times$	$n\kappa$	$\times \log$	$\frac{1}{\varepsilon}$
Variance reduction	$d\times$	$(n+\kappa)$	$\times \log$	$\frac{1}{\varepsilon}$

  - Can be accelerated (e.g., Lan, 2015): $n+\kappa \Rightarrow n+\sqrt{n\kappa}$
  - Matching upper and lower bounds of complexity

Outline

1. **Parametric supervised learning on a single machine**
   - Machine learning $\approx$ optimization of finite sums
   - From batch to stochastic gradient methods
   - Linearly-convergent stochastic methods for convex problems

2. **Machine learning over networks**
   - Centralized and decentralized methods
   - From network averaging to optimization
   - Distributing the fastest single machine algorithms

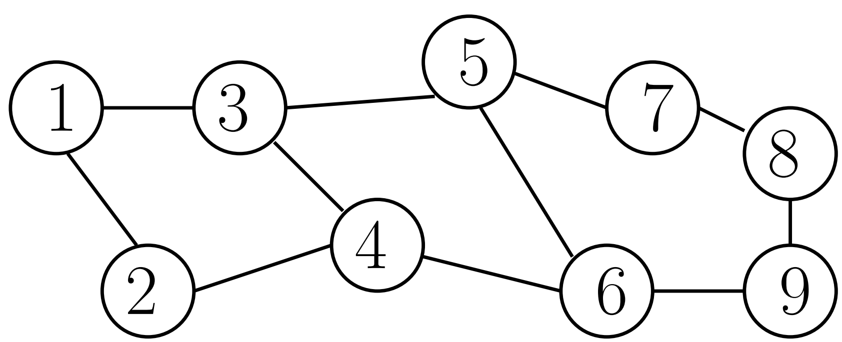
Distribution in machine learning (and beyond)

- **Machine learning through optimization**

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = g(\theta)$$

  - $f_i(\theta)$ error of model defined by θ on dataset indexed by i

- **Each dataset / function f_i only accessible by node i in a graph**



Distribution in machine learning (and beyond)

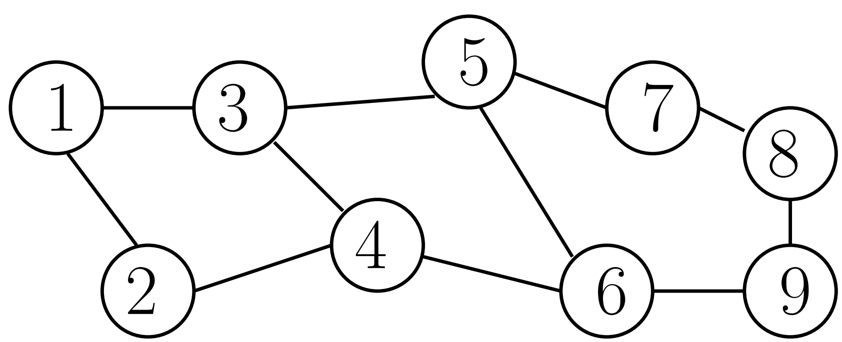
- **Machine learning through optimization**

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = g(\theta)$$

  – $f_i(\theta)$ error of model defined by θ on dataset indexed by i

  – Example: $f_i(\theta) = \dfrac{1}{m_i} \displaystyle\sum_{j=1}^{m_i} \ell(y_{ij}, \theta^\top \Phi(x_{ij}))$ if m_i observations

- **Each dataset / function f_i only accessible by node i in a graph**



Distribution in machine learning (and beyond)

- **Machine learning through optimization**

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = g(\theta)$$

  - $f_i(\theta)$ error of model defined by θ on dataset indexed by i
  - Example: $f_i(\theta) = \dfrac{1}{m_i} \displaystyle\sum_{j=1}^{m_i} \ell(y_{ij}, \theta^\top \Phi(x_{ij}))$ if m_i observations

- **Each dataset / function f_i only accessible by node i in a graph**
  - Massive datasets, multiple machines / cores
  - Communication / legal constraints

- **Goal: Minimize communication and local computation costs**

Distribution in machine learning (and beyond)

- **Machine learning through optimization**

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = g(\theta)$$

  – $f_i(\theta)$ error of model defined by θ on dataset indexed by i

- **Why not simply distributing a simple single machine algorithm?**

  – (accelerated) gradient descent (see, e.g., Nesterov, 2004)

$$\theta_t = \theta_{t-1} - \gamma \nabla g(\theta_{t-1})$$

  – Requires $\sqrt{\kappa} \log \frac{1}{\varepsilon}$ full gradient computations to reach precision ε

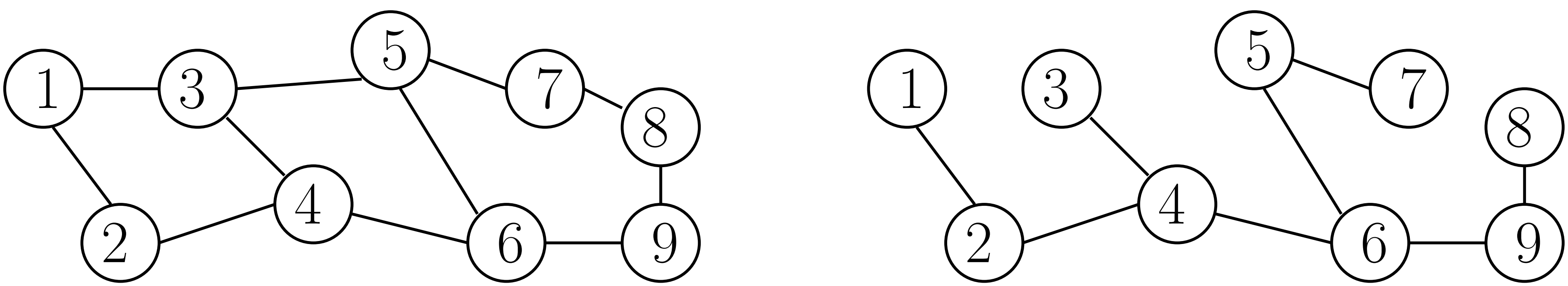
  – Need to perform distributed averaging over a network

Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \dfrac{1}{n}\sum_{i=1}^{n} \xi_i = \arg\min_{\theta \in \mathbb{R}} \ \dfrac{1}{n}\sum_{i=1}^{n}(\theta - \xi_i)^2$

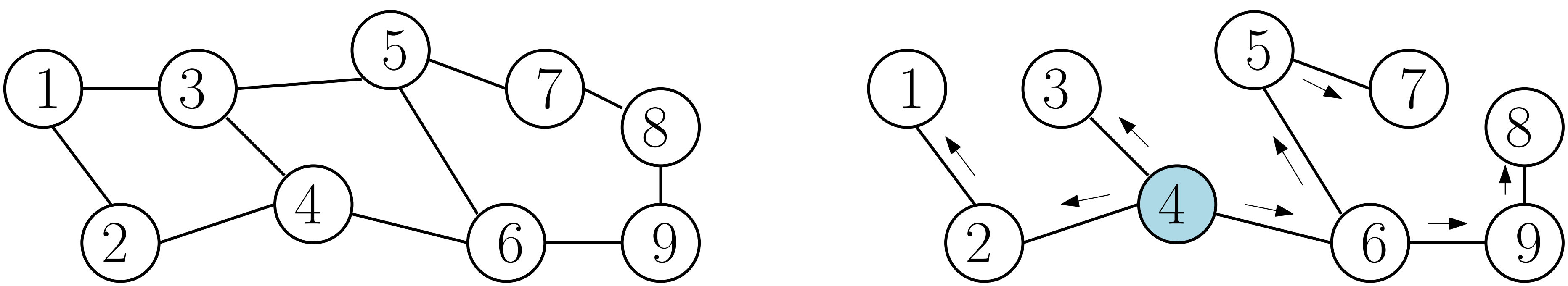
Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \frac{1}{n}\sum_{i=1}^{n} \xi_i = \arg\min_{\theta \in \mathbb{R}} \ \frac{1}{n}\sum_{i=1}^{n}(\theta - \xi_i)^2$
- **Centralized algorithms**
  - Compute a spanning tree with diameter $\leqslant 2\Delta$
  - Master/slave algorithm



Classical algorithms for distributed averaging

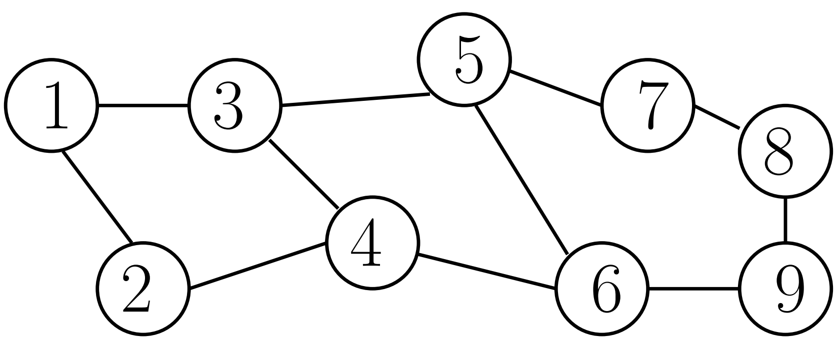
- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \dfrac{1}{n}\sum_{i=1}^{n} \xi_i = \arg\min_{\theta \in \mathbb{R}} \ \dfrac{1}{n}\sum_{i=1}^{n} (\theta - \xi_i)^2$
- **Centralized algorithms**
  - Compute a spanning tree with diameter $\leqslant 2\Delta$
  - Master/slave algorithm: Δ communication steps + no error

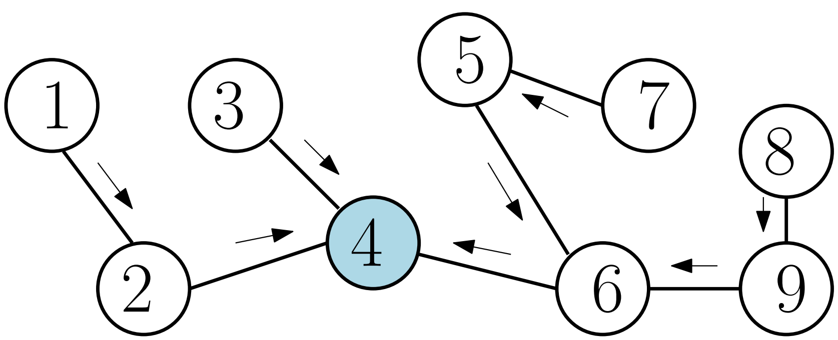


Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \frac{1}{n}\sum_{i=1}^{n} \xi_i = \arg\min_{\theta \in \mathbb{R}} \frac{1}{n}\sum_{i=1}^{n} (\theta - \xi_i)^2$

- **Centralized algorithms**
  - Compute a spanning tree with diameter $\leqslant 2\Delta$
  - Master/slave algorithm: Δ communication steps + no error





Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \frac{1}{n}\sum_{i=1}^n \xi_i = \arg\min_{\theta \in \mathbb{R}} \; \frac{1}{n}\sum_{i=1}^n (\theta - \xi_i)^2$
- **Centralized algorithms**
  - Compute a spanning tree with diameter $\leqslant 2\Delta$
  - Master/slave algorithm: Δ communication steps + no error
- **Application to centralized distributed optimization**
  - $\sqrt{\kappa}\log\frac{1}{\varepsilon}$ gradient steps and $\sqrt{\kappa}\Delta\log\frac{1}{\varepsilon}$ communication steps
  - "Optimal" (Scaman, Bach, Bubeck, Lee, and Massoulié, 2017)

Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \frac{1}{n}\sum_{i=1}^n \xi_i = \arg\min_{\theta \in \mathbb{R}} \ \frac{1}{n}\sum_{i=1}^n (\theta - \xi_i)^2$

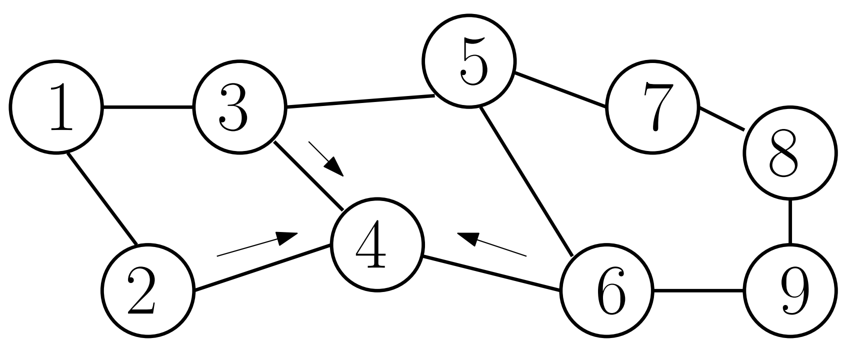
- **Centralized algorithms**
  - Compute a spanning tree with diameter $\leqslant 2\Delta$
  - Master/slave algorithm: Δ communication steps + no error

- **Application to centralized distributed optimization**
  - $\sqrt{\kappa}\log\frac{1}{\varepsilon}$ gradient steps and $\sqrt{\kappa}\Delta\log\frac{1}{\varepsilon}$ communication steps
  - "Optimal" (Scaman, Bach, Bubeck, Lee, and Massoulié, 2017)

- **Robustness?**

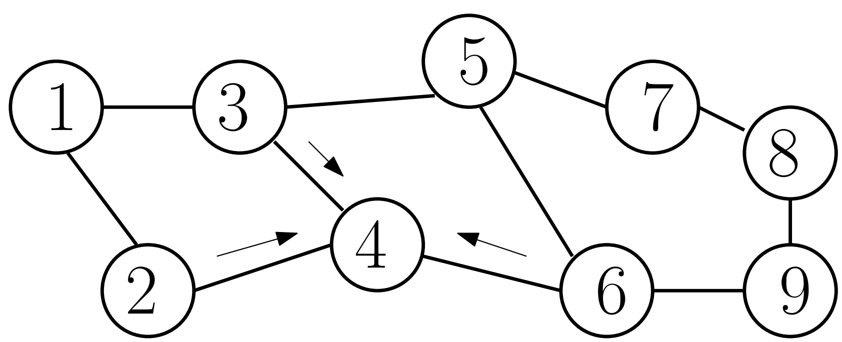
Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \dfrac{1}{n}\displaystyle\sum_{i=1}^n \xi_i = \arg\min_{\theta \in \mathbb{R}} \dfrac{1}{n}\sum_{i=1}^n (\theta - \xi_i)^2$
- **Decentralized algorithms** - **gossip** (Boyd et al., 2006)
  - Replace θ_i by a weighted average of its neighbors $\sum_{j=1}^n W_{ij}\theta_j$



Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \xi_i = \arg\min_{\theta \in \mathbb{R}} \; \dfrac{1}{n}\sum_{i=1}^{n} (\theta - \xi_i)^2$
- **Decentralized algorithms - gossip** (Boyd et al., 2006)
  - Replace θ_i by a weighted average of its neighbors $\sum_{j=1}^{n} W_{ij}\theta_j$
  - Potential asynchrony, changing network



Classical algorithms for distributed averaging

- **Goal**: Given n observations $\xi_1, \dots, \xi_n \in \mathbb{R}$
  - Compute $\theta_* = \dfrac{1}{n}\displaystyle\sum_{i=1}^n \xi_i = \arg\min_{\theta\in\mathbb{R}} \dfrac{1}{n}\sum_{i=1}^n (\theta - \xi_i)^2$

- **Decentralized algorithms - gossip** (Boyd et al., 2006)
  - Replace θ_i by a weighted average of its neighbors $\sum_{j=1}^n W_{ij}\theta_j$
  - Potential asynchrony, changing network

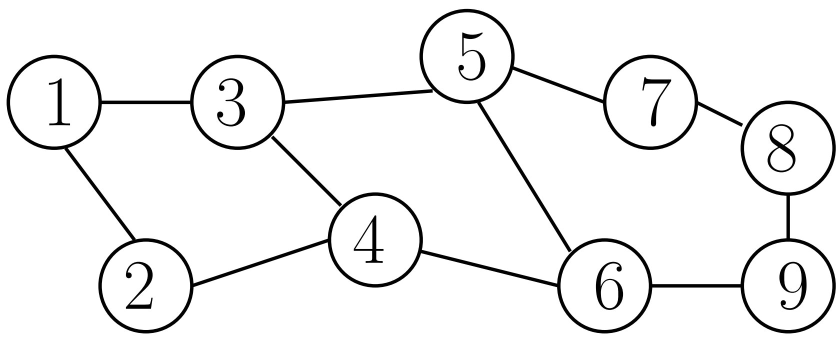
- **Synchronous gossip** (all nodes simultaneously)
  - Main iteration: $\boxed{\theta_t = W\theta_{t-1} = W^t\theta_0 = W^t\xi}$
  - Typical assumption: W symmetric doubly stochastic matrix

Convergence of synchronous gossip

- **Synchronous gossip** (all nodes simultaneously)
  - Main iteration: $\boxed{\theta_t = W\theta_{t-1} = W^t\theta_0 = W^t\xi}$
  - Typical assumption: W symmetric doubly stochastic matrix

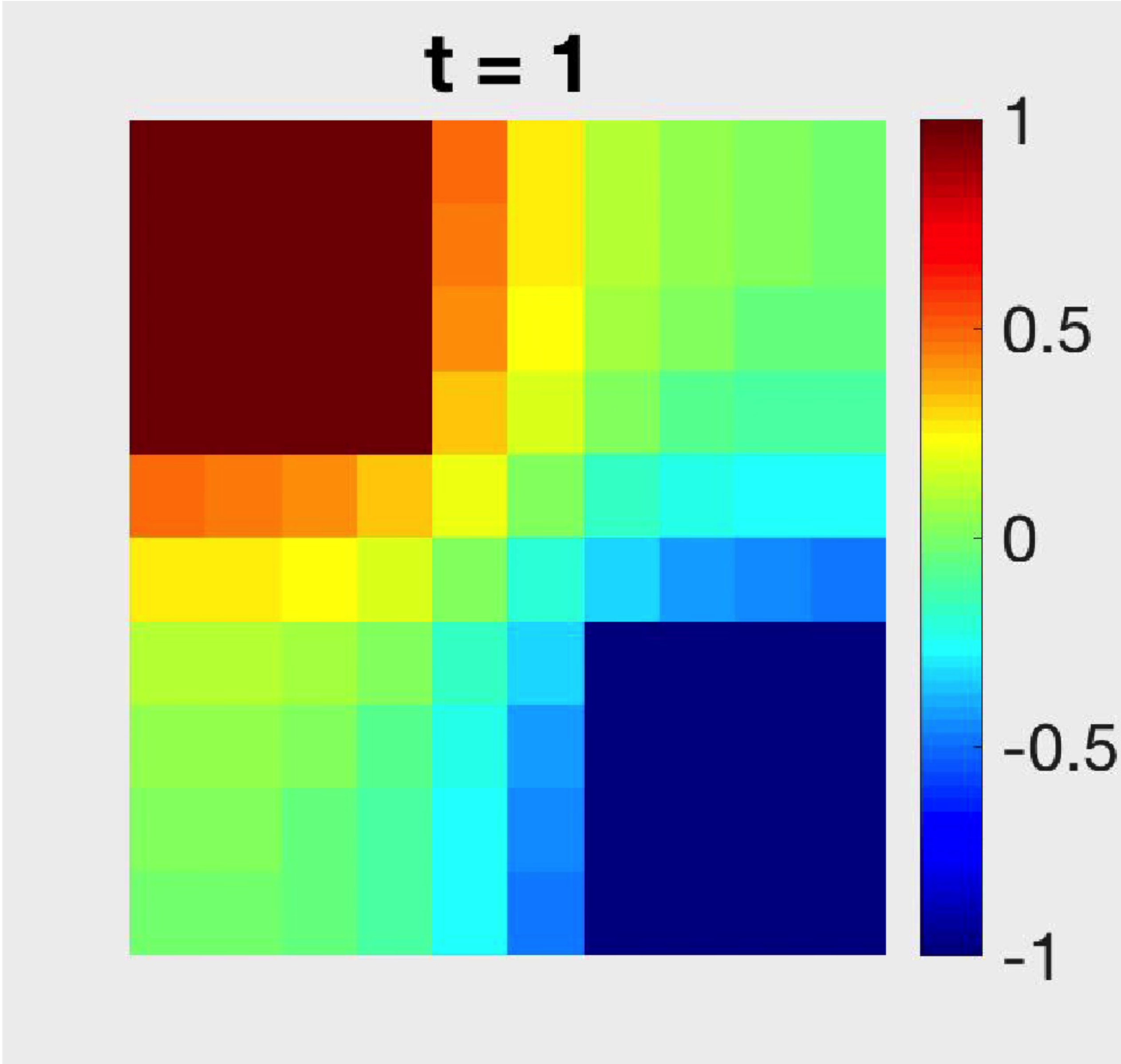
Convergence of synchronous gossip

- **Synchronous gossip** (all nodes simultaneously)
  - Main iteration: $\boxed{\theta_t = W\theta_{t-1} = W^t\theta_0 = W^t\xi}$
  - Typical assumption: W symmetric doubly stochastic matrix
  - Consequence: Eigenvalues$(W) \in [-1, 1]$
  - Eigengap $\gamma = \lambda_1(W) - \lambda_2(W) = 1 - \lambda_2(W)$
  - γ^{-1} = mixing time of the associated Markov chain



  - Need $\frac{1}{\gamma}\log\frac{1}{\varepsilon}$ iterations to reach precision ε (for classical averaging)

Illustration of synchronous gossip



Decentralized optimization

- **Mixing gossip and optimization**
  - Nedic and Ozdaglar (2009); Duchi et al. (2012); Wei and Ozdaglar (2012); Iutzeler et al. (2013); Shi et al. (2015); Jakovetić et al. (2015); Nedich et al. (2016); Mokhtari et al. (2016); Colin et al. (2016); Scaman et al. (2017), **etc.**

Decentralized optimization

- **Mixing gossip and optimization**

- **Lower bound on complexity** (Scaman et al., 2017)
  - $\sqrt{\kappa} \log \frac{1}{\varepsilon}$ gradient steps and $\sqrt{\kappa/\gamma} \log \frac{1}{\varepsilon}$ communication steps
  - Plain gossip not optimal!
    (need to gossip gradients with increasing precision)

Decentralized optimization

- **Mixing gossip and optimization**
- **Lower bound on complexity** (Scaman et al., 2017)
  - $\sqrt{\kappa}\log\frac{1}{\varepsilon}$ gradient steps and $\sqrt{\kappa/\gamma}\log\frac{1}{\varepsilon}$ communication steps
  - Plain gossip not optimal!
    (need to gossip gradients with increasing precision)
- **Is this lower bound achievable?**

Dual reformulation (Jakovetić et al., 2015)

$$\min_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(\theta) = \min_{\theta^{(1)},\ldots,\theta^{(n)} \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(\theta^{(i)}) \text{ such that } \forall i \sim j, \theta^{(i)} = \theta^{(j)}$$

Dual reformulation (Jakovetić et al., 2015)

$$\min_{\theta\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta) = \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) \text{ such that } \forall i\sim j, \theta^{(i)}=\theta^{(j)}$$

$$= \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\ \max_{\forall i\sim j, \lambda_{ij}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) + \sum_{i\sim j}\lambda_{ij}^\top(\theta^{(i)}-\theta^{(j)})$$

Dual reformulation (Jakovetić et al., 2015)

$$\min_{\theta\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta) = \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) \text{ such that } \forall i\sim j, \theta^{(i)}=\theta^{(j)}$$

$$= \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\ \max_{\forall i\sim j, \lambda_{ij}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) + \sum_{i\sim j}\lambda_{ij}^\top(\theta^{(i)}-\theta^{(j)})$$

$$= \max_{\forall i\sim j, \lambda_{ij}\in\mathbb{R}^d}\ \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) + \sum_{i=1}^n[\theta^{(i)}]^\top \text{linear}_i(\lambda)$$

Dual reformulation (Jakovetić et al., 2015)

$$\min_{\theta\in\mathbb{R}^d} \sum_{i=1}^n f_i(\theta) = \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d} \sum_{i=1}^n f_i(\theta^{(i)}) \text{ such that } \forall i \sim j, \theta^{(i)} = \theta^{(j)}$$

$$= \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d} \max_{\forall i\sim j, \lambda_{ij}\in\mathbb{R}^d} \sum_{i=1}^n f_i(\theta^{(i)}) + \sum_{i\sim j} \lambda_{ij}^\top(\theta^{(i)} - \theta^{(j)})$$

$$= \max_{\forall i\sim j, \lambda_{ij}\in\mathbb{R}^d} \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d} \sum_{i=1}^n f_i(\theta^{(i)}) + \sum_{i=1}^n [\theta^{(i)}]^\top \text{linear}_i(\lambda)$$

$$= \max_{\forall i\sim j, \lambda_{ij}\in\mathbb{R}^d} \sum_{i=1}^n \text{function}_i(\lambda) = \max_{\forall i\sim j, \lambda_{ij}\in\mathbb{R}^d} \text{function}(\lambda)$$

Dual reformulation (Jakovetić et al., 2015)

$$\min_{\theta\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta) = \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) \text{ such that } \forall i\sim j, \theta^{(i)}=\theta^{(j)}$$

$$= \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\ \max_{\forall i\sim j,\lambda_{ij}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) + \sum_{i\sim j}\lambda_{ij}^\top(\theta^{(i)}-\theta^{(j)})$$

$$= \max_{\forall i\sim j,\lambda_{ij}\in\mathbb{R}^d}\ \min_{\theta^{(1)},\dots,\theta^{(n)}\in\mathbb{R}^d}\sum_{i=1}^n f_i(\theta^{(i)}) + \sum_{i=1}^n[\theta^{(i)}]^\top \text{linear}_i(\lambda)$$

$$= \max_{\forall i\sim j,\lambda_{ij}\in\mathbb{R}^d}\sum_{i=1}^n \text{function}_i(\lambda) = \max_{\forall i\sim j,\lambda_{ij}\in\mathbb{R}^d}\text{function}(\lambda)$$

- **Accelerated gradient descent** (Scaman et al., 2017)
  $\Leftrightarrow$ alternating local gradient computations and a gossip step
  - $\sqrt{\kappa/\gamma}\log\frac{1}{\varepsilon}$ gradient steps and $\sqrt{\kappa/\gamma}\log\frac{1}{\varepsilon}$ communication steps
  - Not optimal $\Rightarrow$ need accelerated gossip

Accelerated gossip

- **Regular gossip**
  - Iterations: $\theta_t = W^t \theta_0$

- **Accelerated gossip**
  - Chebyshev acceleration (Auzinger, 2011; Arioli and Scott, 2014)
  - Shift-register gossip (Cao et al., 2006)
  - Linear combinations $\Leftrightarrow \eta_t = \sum_{k=0}^{t} \alpha_k \theta_k = \sum_{k=0}^{t} \alpha_k W^k \xi = P_t(W)\xi$
  - Optimal polynomial is the Chebyshev polynomial
  - Can be computed online with same cost as regular gossip, e.g.,

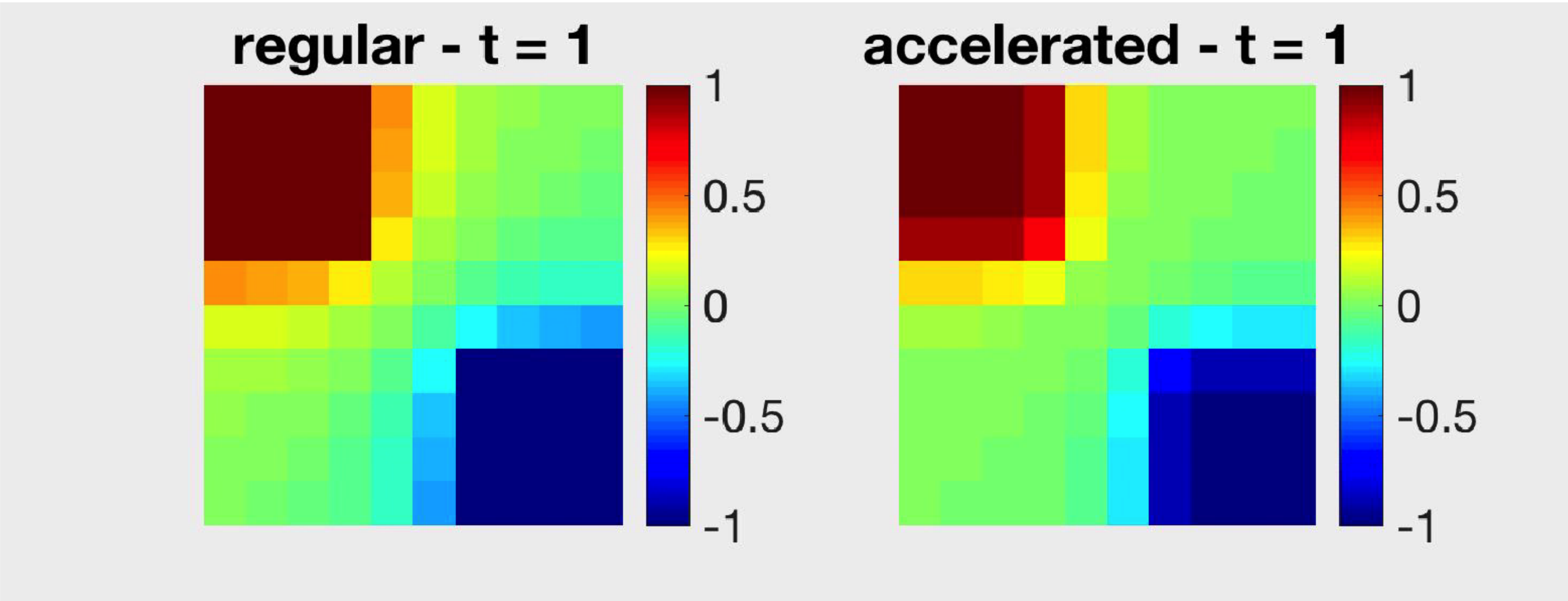
$$\theta_t = \omega_t W \theta_{t-1} + (1 - \omega_t)\theta_{t-1}$$

Accelerated gossip

- **Regular gossip**
  - Iterations: $\theta_t = W^t \theta_0$

- **Accelerated gossip**
  - Chebyshev acceleration (Auzinger, 2011; Arioli and Scott, 2014)
  - Shift-register gossip (Cao et al., 2006)
  - Linear combinations $\Leftrightarrow \eta_t = \sum_{k=0}^{t} \alpha_k \theta_k = \sum_{k=0}^{t} \alpha_k W^k \xi = P_t(W)\xi$
  - Optimal polynomial is the Chebyshev polynomial
  - Can be computed online with same cost as regular gossip, e.g.,
$$\theta_t = \omega_t W \theta_{t-1} + (1 - \omega_t)\theta_{t-1}$$
  - Replace γ^{-1} by $\gamma^{-1/2}$ in rates

Illustration of accelerated gossip



Accelerated gossip

- **Regular gossip**
  - Iterations: $\theta_t = W^t \theta_0$

- **Accelerated gossip**
  - Chebyshev acceleration (Auzinger, 2011; Arioli and Scott, 2014)
  - Shift-register gossip (Cao et al., 2006)
  - Linear combinations $\Leftrightarrow \eta_t = \sum_{k=0}^{t} \alpha_k \theta_k = \sum_{k=0}^{t} \alpha_k W^k \xi = P_t(W)\xi$
  - Optimal polynomial is the Chebyshev polynomial
  - Can be computed online with same cost as regular gossip, e.g.,
  $$\theta_t = \omega_t W \theta_{t-1} + (1 - \omega_t)\theta_{t-1}$$
  - Replace γ^{-1} by $\gamma^{-1/2}$ in rates

- $\Rightarrow$ **optimal complexity for optimization** (Scaman et al., 2017)

Distribution in machine learning (and beyond)

- **Machine learning through optimization**

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = g(\theta)$$

  – $f_i(\theta)$ error of model defined by θ on dataset indexed by i

  – Example: $f_i(\theta) = \dfrac{1}{m_i} \sum_{j=1}^{m_i} \ell(y_{ij}, \theta^\top \Phi(x_{ij}))$ if m_i observations

Distribution in machine learning (and beyond)

- **Machine learning through optimization**

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = g(\theta)$$

  - $f_i(\theta)$ error of model defined by θ on dataset indexed by i
  - Example: $f_i(\theta) = \dfrac{1}{m_i} \displaystyle\sum_{j=1}^{m_i} \ell(y_{ij}, \theta^\top \Phi(x_{ij}))$ if m_i observations

- **Single machine vs. "optimal" decentralized algorithm**

Algorithm	gradient steps	communication
Single machine algorithm	$nm + \sqrt{nm\kappa}$	0
MSDA (Scaman et al., 2017)	$m\sqrt{\kappa}$	$\sqrt{\kappa/\gamma}$

Distribution in machine learning (and beyond)

- **Machine learning through optimization**

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) = g(\theta)$$

  - $f_i(\theta)$ error of model defined by θ on dataset indexed by i
  - Example: $f_i(\theta) = \dfrac{1}{m_i} \displaystyle\sum_{j=1}^{m_i} \ell(y_{ij}, \theta^\top \Phi(x_{ij}))$ if m_i observations

- **MSDA (Scaman et al., 2017)**
  - $\sqrt{\kappa} \log \frac{1}{\varepsilon}$ gradient steps and $\sqrt{\kappa/\gamma} \log \frac{1}{\varepsilon}$ communication steps
  - "Optimal", but still not adapted to machine learning
  - Huge slow down when going from 1 to 2 machines
  - Only synchronous

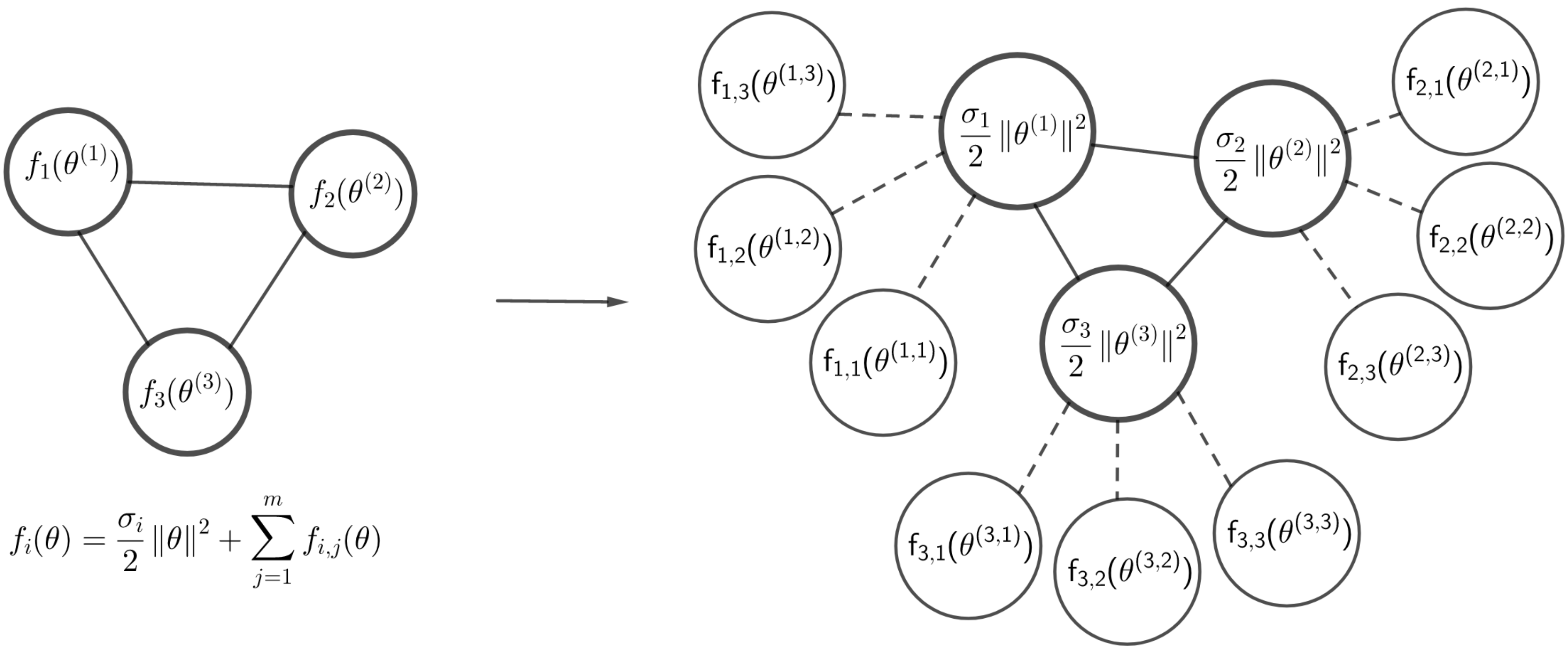
Decentralized algorithms for machine learning (Hendrikx, Bach, and Massoulié, 2019)

- **Trade-offs between gradient and communication steps**
  - Adapted to functions of the type $f_i(\theta) = \frac{1}{m}\sum_{j=1}^{m} \ell(y_{ij}, \theta^\top \Phi(x_{ij}))$
  - Allows for partial asynchrony
- n **computing nodes, with** m **observations each**

Algorithm	gradient steps	communication
Single machine algorithm	$nm + \sqrt{nm\kappa}$	0
MSDA (Scaman et al., 2017)	$m\sqrt{\kappa}$	$\sqrt{\kappa/\gamma}$
ADFS (Hendrikx et al., 2019)	$m + \sqrt{m\kappa}$	$\sqrt{\kappa/\gamma}$

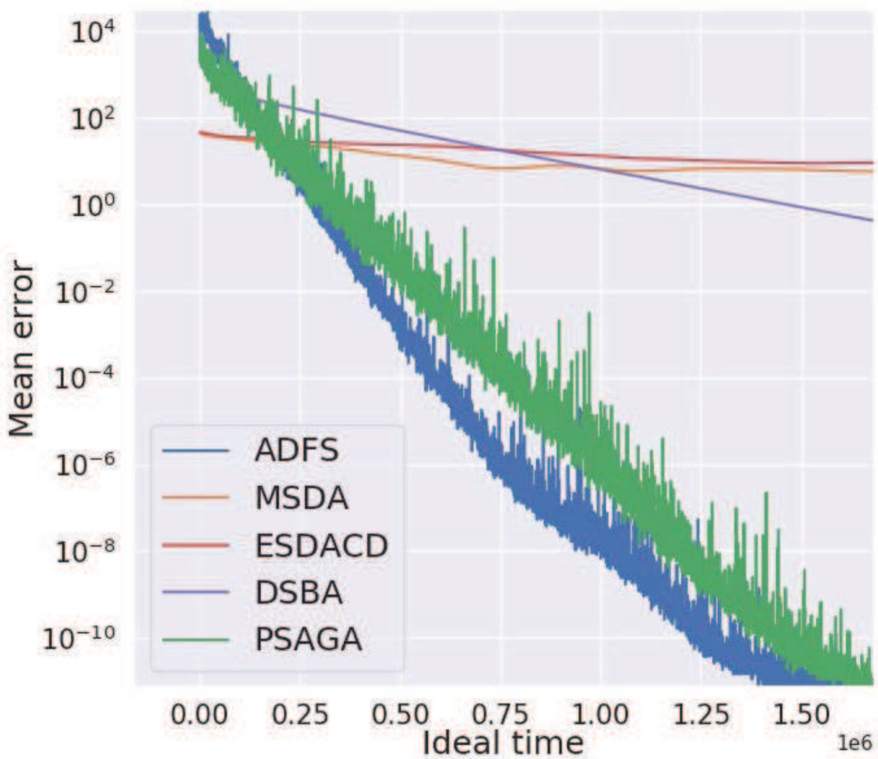
ADFS - Algorithm principle

- **Minimizing** $\sum_{i=1}^{n} \Big\{ \sum_{j=1}^{m} f_{i,j}(\theta) + \frac{\sigma_i}{2} \|\theta\|^2 \Big\}$
  - Create an equivalent graph
  - Dual randomized coordinate ascent (with non uniform sampling)
  - Decoupling of data and gossip steps

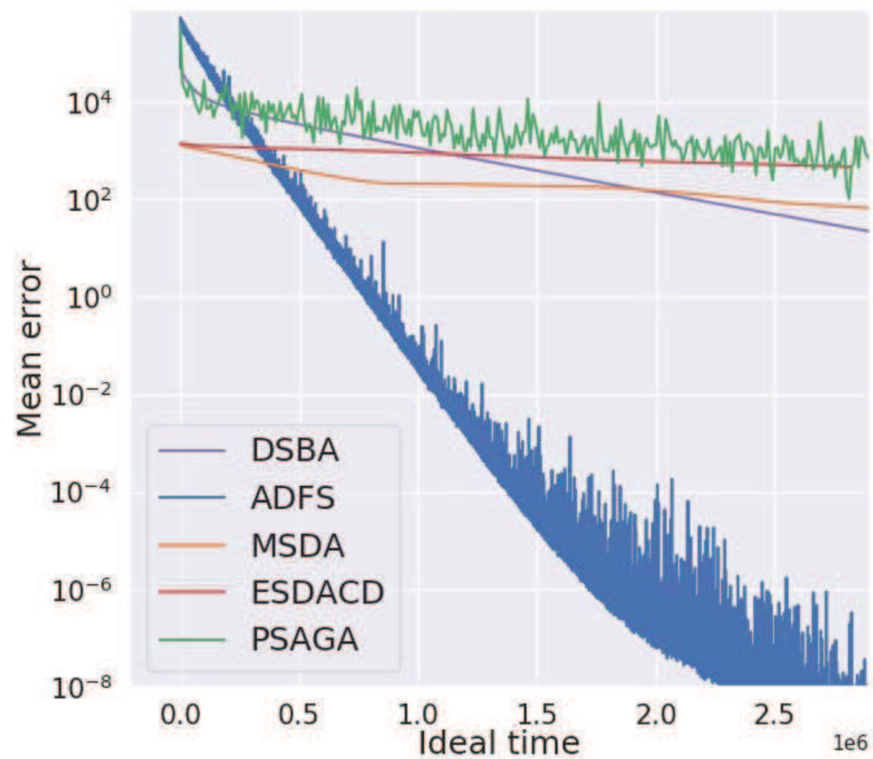


Decentralized algorithms for machine learning (Hendrikx, Bach, and Massoulié, 2019)

- **Running times on an actual cluster**
  - Logistic regression with $m = 10^4$ observations per node in $\mathbb{R}^{28}$
  - Two-dimensional grid network



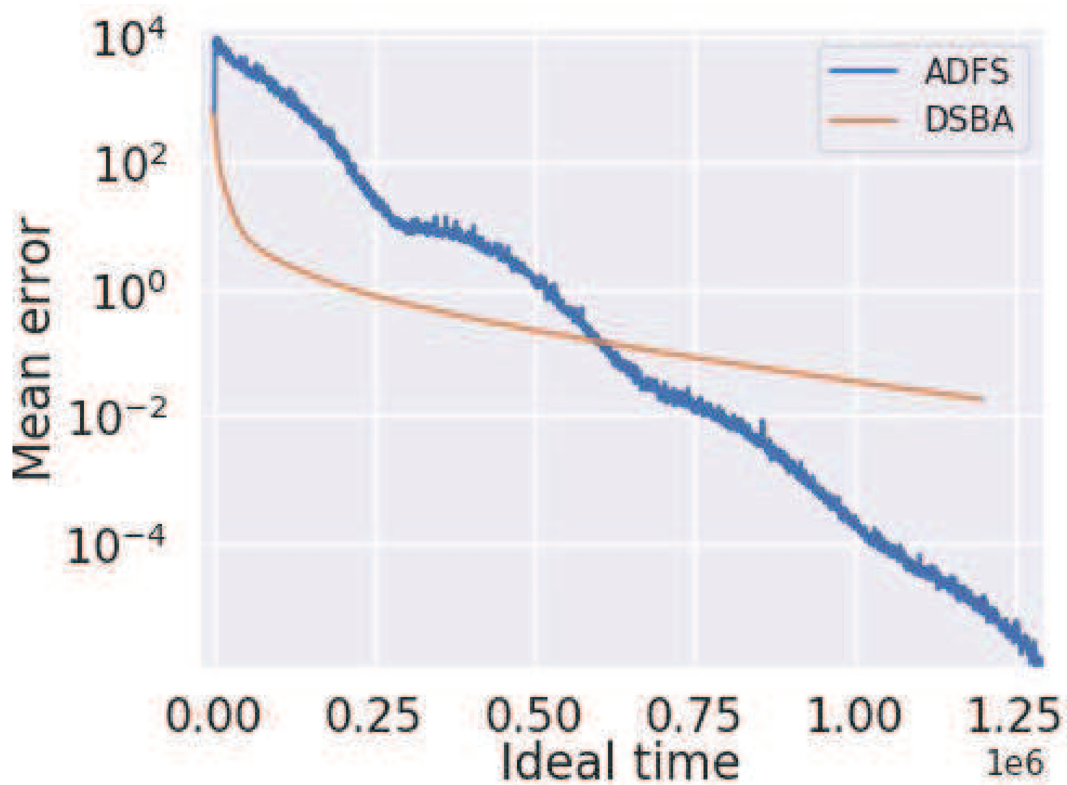
$n = 4$



$n = 100$

Decentralized algorithms for machine learning (Hendrikx, Bach, and Massoulié, 2019)

- **Running times on an actual cluster**
  - Logistic regression with $mn \approx 10^5$ observations in $\mathbb{R}^{47236}$
  - Two-dimensional grid network with $n = 100$ nodes



Conclusions

- **Distributed decentralized machine learning**
  - Distributing the fastest single machine algorithms!
  - n machines and m observations per machine
  - From $nm + \sqrt{nm\kappa}$ (single machine) to $m + \sqrt{m\kappa}$ gradient steps
  - Linear speed-ups for well-conditioned problems

Conclusions

- **Distributed decentralized machine learning**
  - Distributing the fastest single machine algorithms!
  - n machines and m observations per machine
  - From $nm + \sqrt{nm\kappa}$ (single machine) to $m + \sqrt{m\kappa}$ gradient steps
  - Linear speed-ups for well-conditioned problems
- **Extensions**
  - Beyond convex problems
  - Matching running time complexity lower bounds
  - Experiments on large-scale clouds

References

M. Arioli and J. Scott. Chebyshev acceleration of iterative refinement. *Numerical Algorithms*, 66(3): 591–608, 2014.

W. Auzinger. *Iterative Solution of Large Linear Systems*. Lecture notes, TU Wien, 2011.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Adv. NIPS*, 2008.

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

Ming Cao, Daniel A. Spielman, and Edmund M. Yeh. Accelerated gossip algorithms for distributed computation. In *44th Annual Allerton Conference on Communication, Control, and Computation*, pages 952–959, 2006.

Igor Colin, Aurelien Bellet, Joseph Salmon, and Stéphan Clémençon. Gossip dual averaging for decentralized optimization of pairwise functions. In *International Conference on Machine Learning*, pages 1388–1396, 2016.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Asynchronous accelerated proximal stochastic gradient for strongly convex distributed finite sums. Technical Report 1901.09865, arXiv, 2019.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 3671–3676. IEEE, 2013.

Dušan Jakovetić, José MF Moura, and Joao Xavier. Linear convergence rate of a class of distributed augmented lagrangian algorithms. *IEEE Transactions on Automatic Control*, 60(4):922–936, 2015.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 2013.

G. Lan. An optimal randomized incremental gradient method. Technical Report 1507.02000, arXiv, 2015.

N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro. A decentralized second-order method with exact linear convergence rate for consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):507–522, 2016.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

A. Nedich, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *ArXiv e-prints*, 2016.

Y. Nesterov. A method for solving a convex programming problem with rate of convergence $O(1/k^2)$. *Soviet Math. Doklady*, 269(3):543–547, 1983.

Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer, 2004.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, pages 3027–3036, 2017.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *51st Annual Conference on Decision and Control (CDC)*, pages 5445–5450. IEEE, 2012.

L. Zhang, M. Mahdavi, and R. Jin. Linear convergence with condition number independent access of full gradients. In *Advances in Neural Information Processing Systems*, 2013.