Optimization for Large Scale Machine Learning

Francis Bach

INRIA - Ecole Normale Supérieure, Paris, France

Machine Learning Summer School - Tübingen, 2020

Slides available at `www.di.ens.fr/~fbach/mlss2020.pdf`

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

Scientific context

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

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

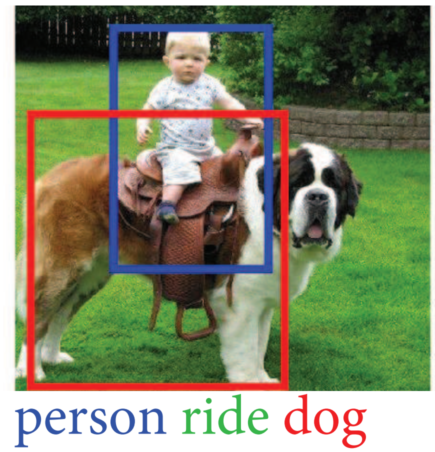
- **Series of "hypes"**

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

- **Healthy interactions between theory, applications, and hype?**

Recent progress in perception (vision, audio, text)

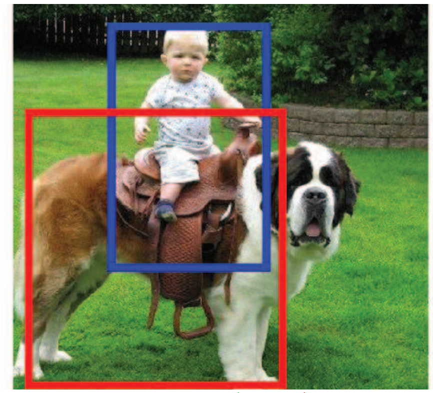
From translate.google.fr



From Peyré et al. (2017)

Recent progress in perception (vision, audio, text)

From translate.google.fr



person ride dog

From Peyré et al. (2017)

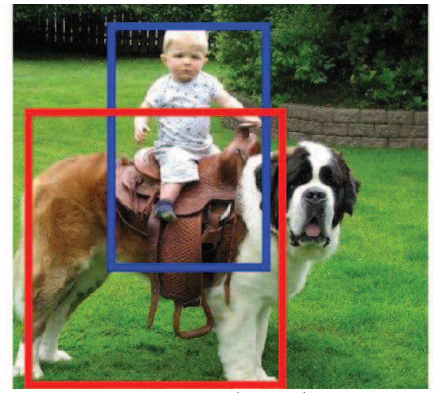
(1) **Massive data**

(2) **Computing power**

(3) **Methodological and scientific progress**

Recent progress in perception (vision, audio, text)

From translate.google.fr



person ride dog

From Peyré et al. (2017)

(1) **Massive data**

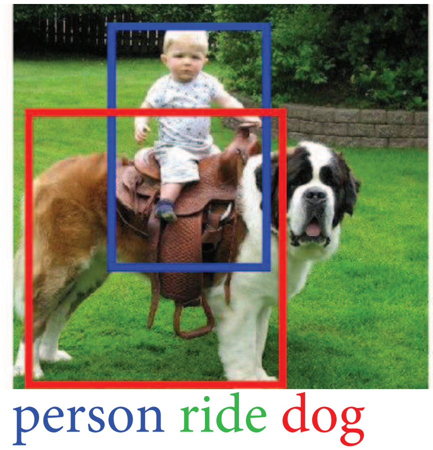
(2) **Computing power**

(3) **Methodological and scientific progress**

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

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**

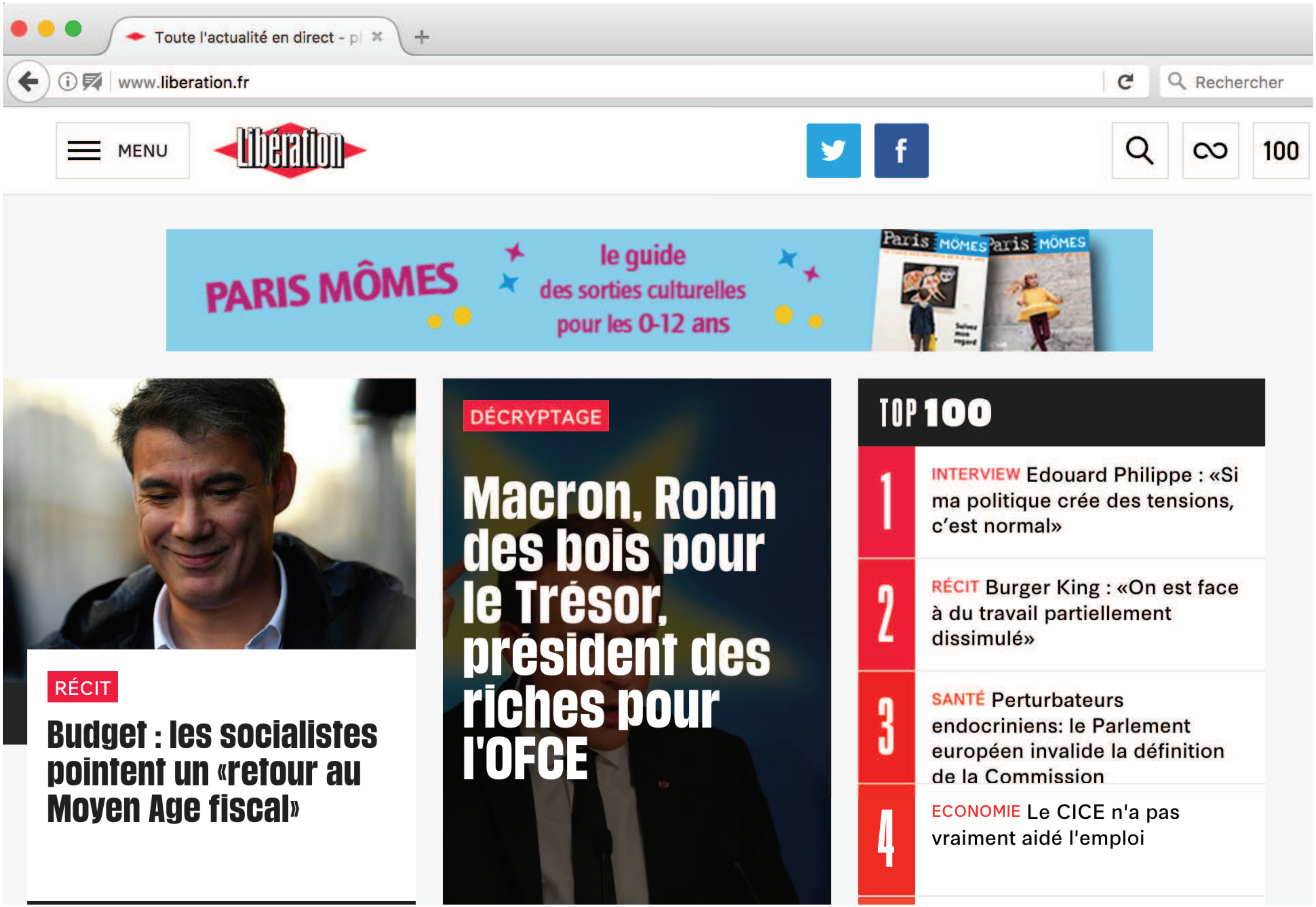
"Intelligence" = models + algorithms + data + computing power

Machine learning for large-scale data

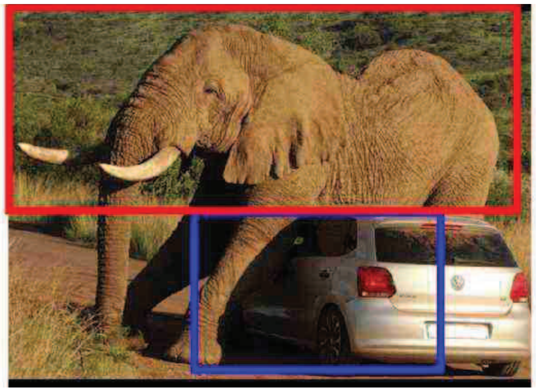
- **Large-scale supervised machine learning**: **large d, large n**
  - d : dimension of each observation (input) or number of parameters
  - n : number of observations

- **Examples**: computer vision, advertising, bioinformatics, etc.

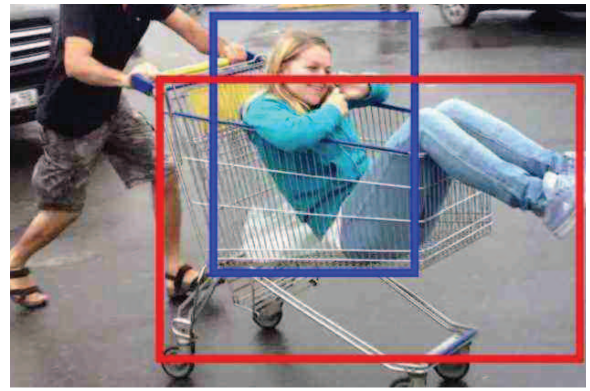
Advertising



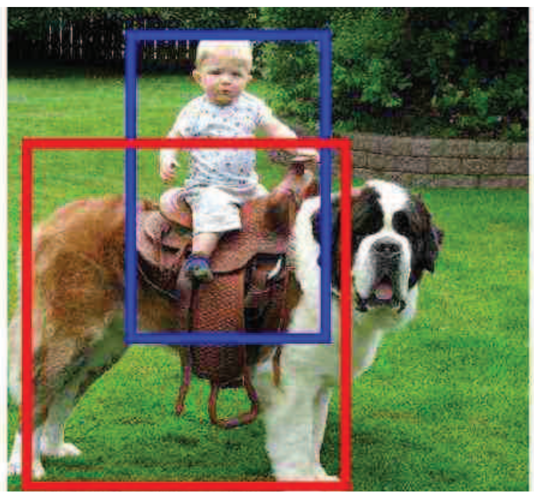
Object / action recognition in images



car under elephant



person in cart

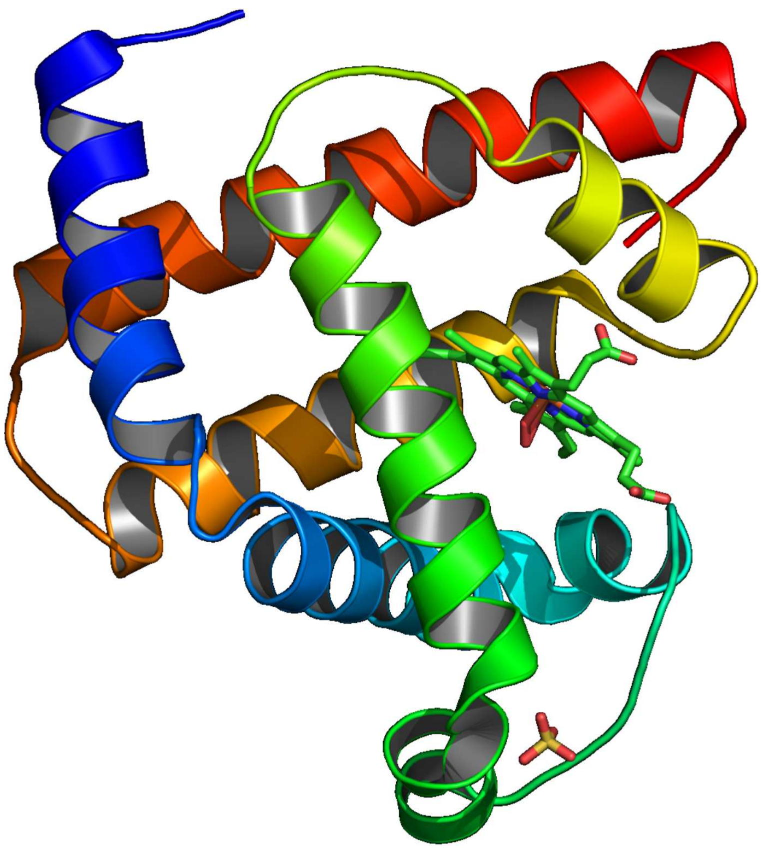


person ride dog

person on top of traffic light

From Peyré, Laptev, Schmid and Sivic (2017)

Bioinformatics



- Predicting multiple functions and interactions of **proteins**
- **Massive data**: up to 1 millions for humans!
- **Complex data**
  - Amino-acid sequence
  - Link with DNA
  - Tri-dimensional molecule

Machine learning for large-scale data

- **Large-scale supervised machine learning**: **large d, large n**
  - d : dimension of each observation (input), or number of parameters
  - n : number of observations

- **Examples**: computer vision, advertising, bioinformatics, etc.

- **Ideal running-time complexity**: $O(dn)$

Machine learning for large-scale data

- **Large-scale supervised machine learning**: **large d, large n**
  - d : dimension of each observation (input), or number of parameters
  - n : number of observations
- **Examples**: computer vision, advertising, bioinformatics, etc.
- **Ideal running-time complexity**: $O(dn)$
- **Going back to simple methods**
  - Stochastic gradient methods (Robbins and Monro, 1951)
- **Goal: Present classical algorithms and some recent progress**

Outline

1. **Introduction/motivation: Supervised machine learning**
   - Machine learning $\approx$ optimization of finite sums
   - Batch optimization methods

2. **Fast stochastic gradient methods for convex problems**
   - Variance reduction: for *training* error
   - Constant step-sizes: for *testing* error

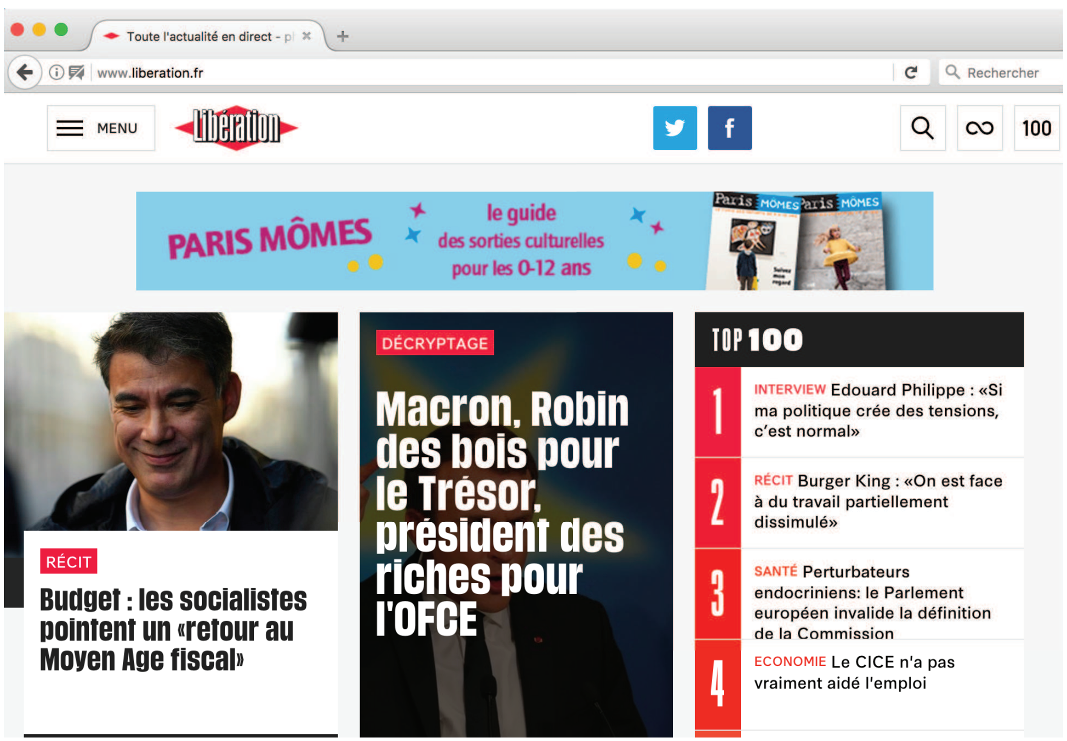
3. **Beyond convex problems**
   - Generic algorithms with generic "guarantees"
   - Global convergence for over-parameterized neural networks

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 $\theta \in \mathbb{R}^d$

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 $\theta \in \mathbb{R}^d$

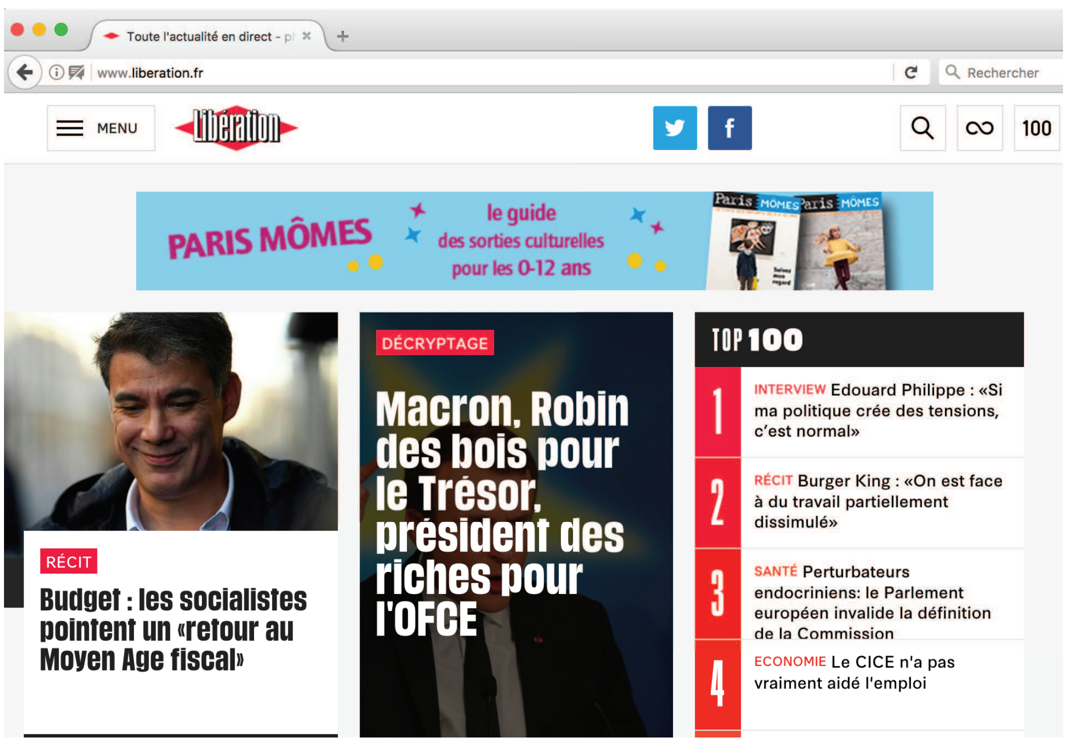


- **Advertising**: $n > 10^9$
  - $\Phi(x) \in \{0, 1\}^d$, $d > 10^9$
  - Navigation history + ad

Parametric supervised machine learning

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

- **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

- 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

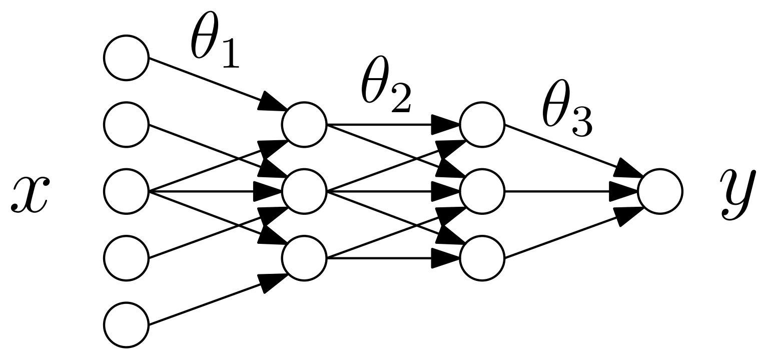
- **Data**: n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$

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

x_1 $\quad$ x_2 $\quad$ x_3 $\quad$ x_4 $\quad$ x_5 $\quad$ x_6

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

  - 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**: find $\hat{\theta}$ solution of

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

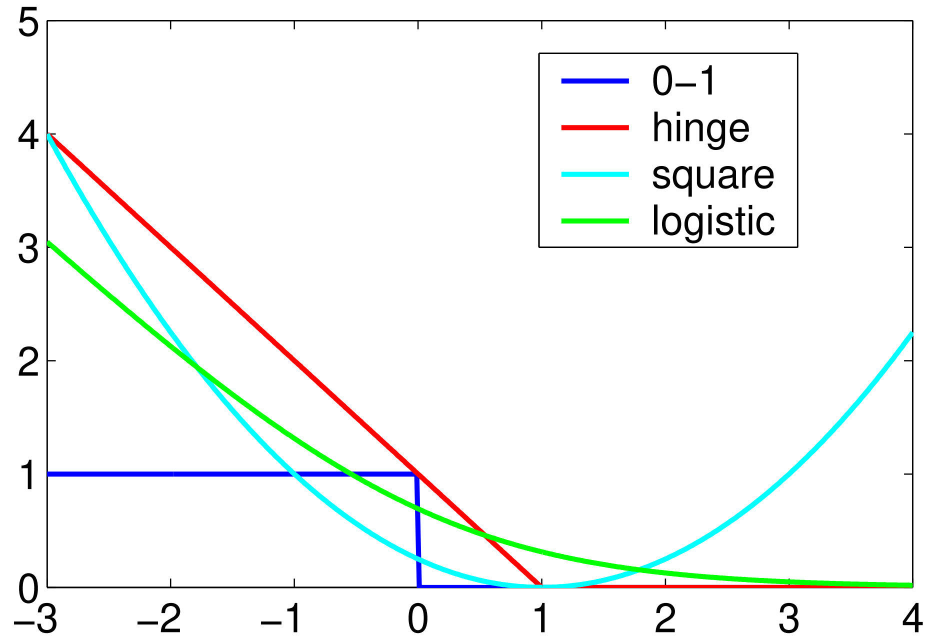
data fitting term $+$ regularizer

Usual losses

- **Regression**: $y \in \mathbb{R}$, prediction $\hat{y} = h(x, \theta)$
  - quadratic loss $\frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - h(x, \theta))^2$

Usual losses

- **Regression**: $y \in \mathbb{R}$, prediction $\hat{y} = h(x,\theta)$
  - quadratic loss $\frac{1}{2}(y-\hat{y})^2 = \frac{1}{2}(y-h(x,\theta))^2$
- **Classification** : $y \in \{-1,1\}$, prediction $\hat{y} = \text{sign}(h(x,\theta))$
  - loss of the form $\ell(y\,h(x,\theta))$
  - "True" 0-1 loss: $\ell(y\,h(x,\theta)) = 1_{y\,h(x,\theta)<0}$
  - Usual convex losses:



Main motivating examples

- **Support vector machine** (hinge loss): non-smooth

$$\ell(Y, h(X\theta)) = \max\{1 - Yh(X, \theta), 0\}$$

- **Logistic regression**: smooth

$$\ell(Y, h(X\theta)) = \log(1 + \exp(-Yh(X, \theta)))$$

- **Least-squares regression**

$$\ell(Y, h(X\theta)) = \frac{1}{2}(Y - h(X, \theta))^2$$

- **Structured output regression**
  - See Tsochantaridis et al. (2005); Lacoste-Julien et al. (2013)

Usual regularizers

- **Main goal**: avoid overfitting

- **(squared) Euclidean norm**: $\|\theta\|_2^2 = \sum_{j=1}^{d} |\theta_j|^2$
  - Numerically well-behaved if $h(x,\theta) = \theta^\top \Phi(x)$
  - Representer theorem and kernel methods : $\theta = \sum_{i=1}^{n} \alpha_i \Phi(x_i)$
  - See, e.g., Schölkopf and Smola (2001); Shawe-Taylor and Cristianini (2004)

Usual regularizers

- **Main goal**: avoid overfitting

- **(squared) Euclidean norm**: $\|\theta\|_2^2 = \sum_{j=1}^d |\theta_j|^2$
  - Numerically well-behaved if $h(x, \theta) = \theta^\top \Phi(x)$
  - Representer theorem and kernel methods : $\theta = \sum_{i=1}^n \alpha_i \Phi(x_i)$
  - See, e.g., Schölkopf and Smola (2001); Shawe-Taylor and Cristianini (2004)

- **Sparsity-inducing norms**
  - Main example: ℓ_1-norm $\|\theta\|_1 = \sum_{j=1}^d |\theta_j|$
  - Perform model selection as well as regularization
  - Non-smooth optimization and structured sparsity
  - See, e.g., Bach, Jenatton, Mairal, and Obozinski (2012a,b)

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**: find $\hat{\theta}$ solution of

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

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**: find $\hat{\theta}$ solution of

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

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**: find $\hat{\theta}$ solution of

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

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**: find $\hat{\theta}$ solution of

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

data fitting term $+$ regularizer

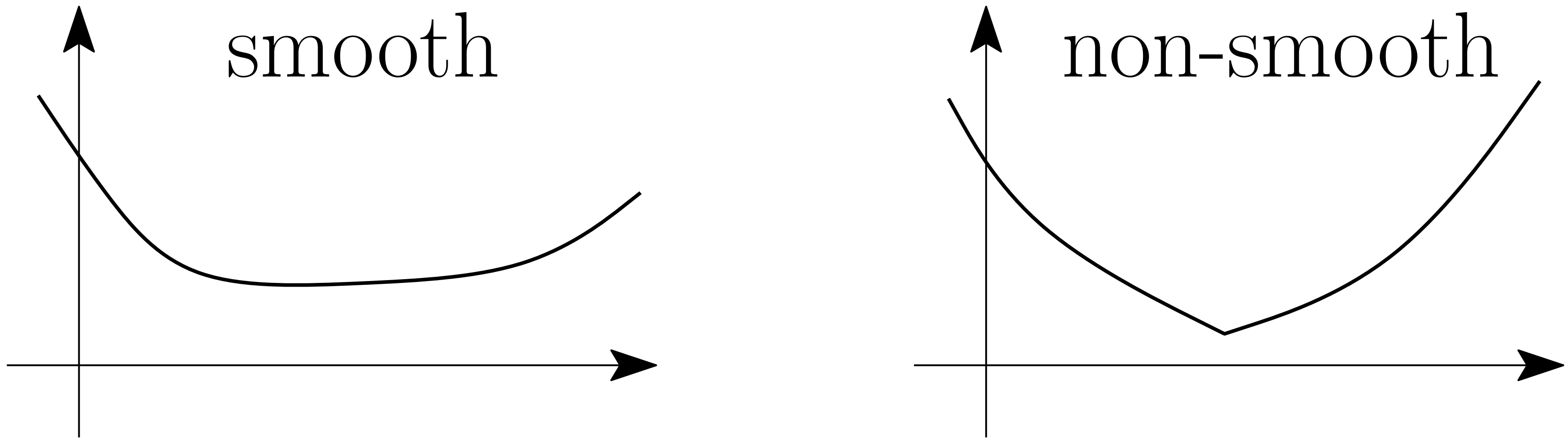
- Optimization: optimization of regularized risk training cost

- 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
  - *(see board)*

Board

- Function $g(\theta) = \dfrac{1}{n}\sum_{i=1}^{n} \ell(y_i, \theta^\top \Phi(x_i))$

Board

- Function $g(\theta) = \dfrac{1}{n} \sum_{i=1}^{n} \ell(y_i, \theta^\top \Phi(x_i))$

- Gradient $g'(\theta) = \dfrac{1}{n} \sum_{i=1}^{n} \ell'(y_i, \theta^\top \Phi(x_i)) \Phi(x_i)$

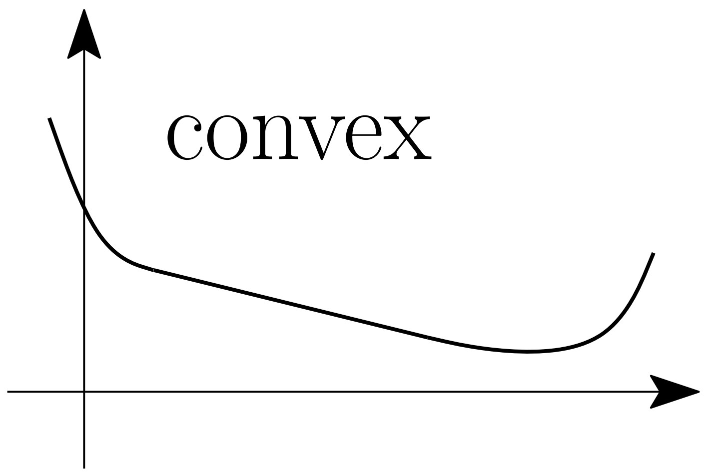
Board

- Function $g(\theta) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \ell(y_i, \theta^\top \Phi(x_i))$
- Gradient $g'(\theta) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \ell'(y_i, \theta^\top \Phi(x_i))\Phi(x_i)$
- Hessian $g''(\theta) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \ell''(y_i, \theta^\top \Phi(x_i))\Phi(x_i)\Phi(x_i)^\top$
  - Smooth loss $\Rightarrow \ell''(y_i, \theta^\top \Phi(x_i))$ bounded

Smoothness and (strong) convexity

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

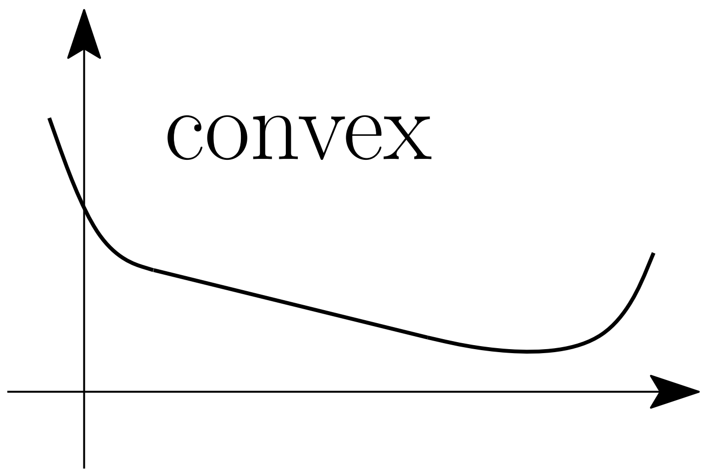
$$\forall \theta \in \mathbb{R}^d, \text{ eigenvalues}\big[g''(\theta)\big] \geqslant 0$$

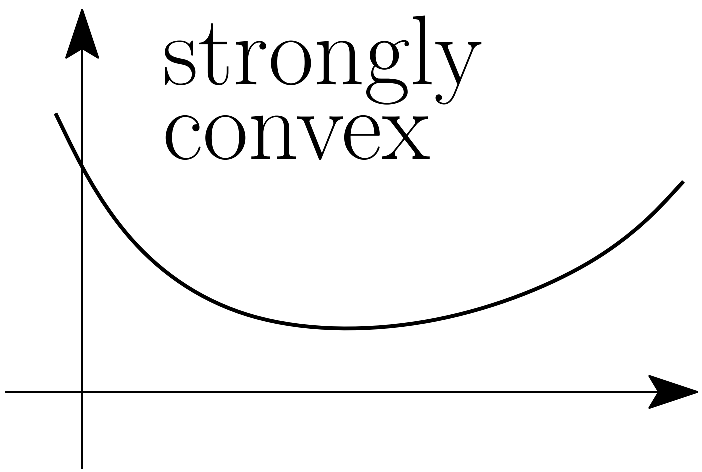


Smoothness and (strong) convexity

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

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



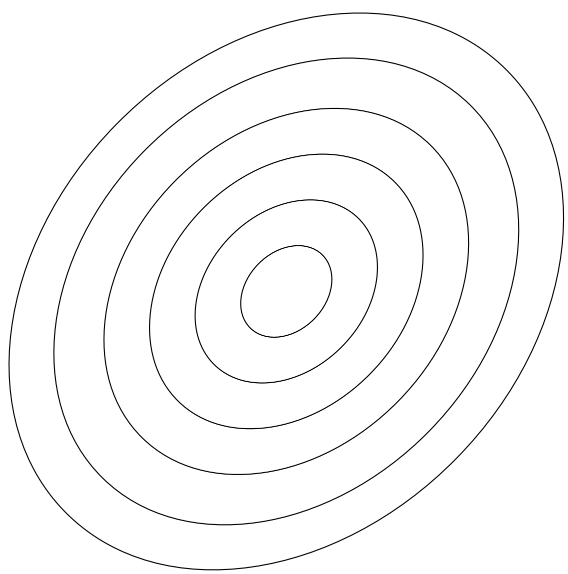


Smoothness and (strong) convexity

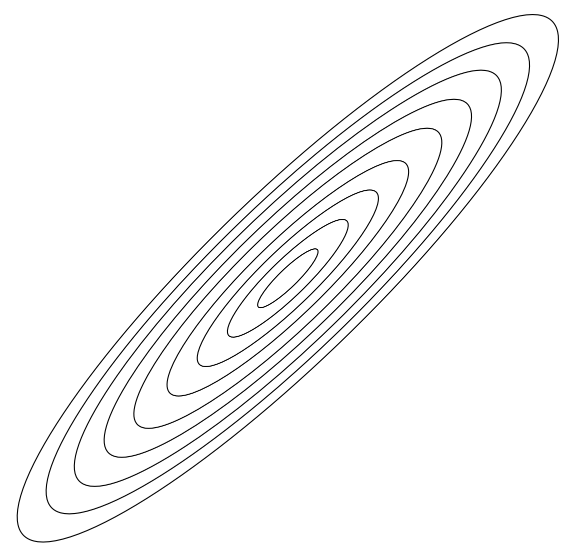
- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if

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

  – 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 $\frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^\top \Rightarrow n \geqslant d$ *(board)*
  - Even when $\mu > 0$, μ may be arbitrarily small!

Board

- Function $g(\theta) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \ell(y_i, \theta^\top \Phi(x_i))$

- Gradient $g'(\theta) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \ell'(y_i, \theta^\top \Phi(x_i))\Phi(x_i)$

- Hessian $g''(\theta) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \ell''(y_i, \theta^\top \Phi(x_i))\Phi(x_i)\Phi(x_i)^\top$
  - Smooth loss $\Rightarrow \ell''(y_i, \theta^\top \Phi(x_i))$ bounded

- Square loss $\Rightarrow \ell''(y_i, \theta^\top \Phi(x_i)) = 1$
  - Hessian proportional to $\frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^\top$

Smoothness and (strong) convexity

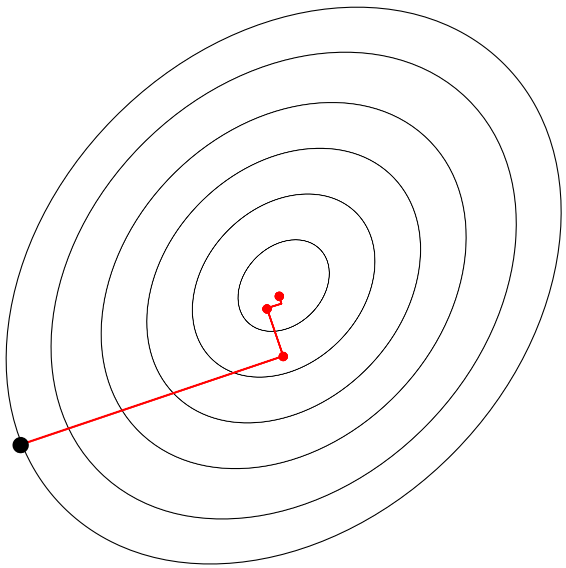
- A twice differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if and only if
$$\forall \theta \in \mathbb{R}^d, \ \text{eigenvalues}\big[g''(\theta)\big] \geqslant \mu$$

- **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 $\frac{1}{n}\sum_{i=1}^n \Phi(x_i)\Phi(x_i)^\top \Rightarrow n \geqslant d$ *(board)*
  - Even when $\mu > 0$, μ may be arbitrarily small!

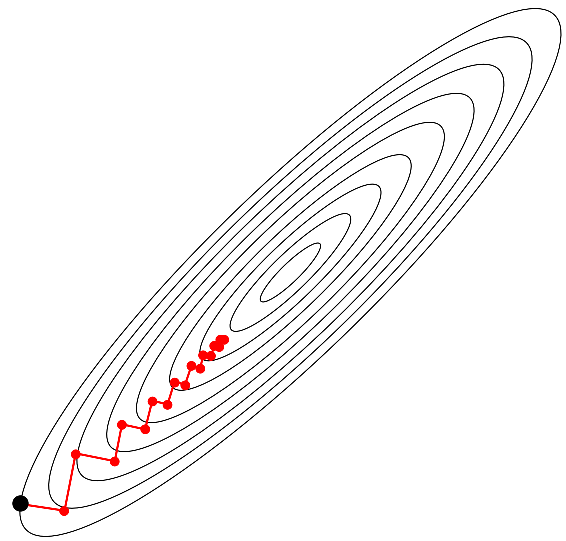
- **Adding regularization by $\frac{\mu}{2}\|\theta\|^2$**
  - creates additional bias unless μ is small, but reduces variance
  - Typically $L/\sqrt{n} \geqslant \mu \geqslant L/n$

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 $\kappa = L/\mu$)

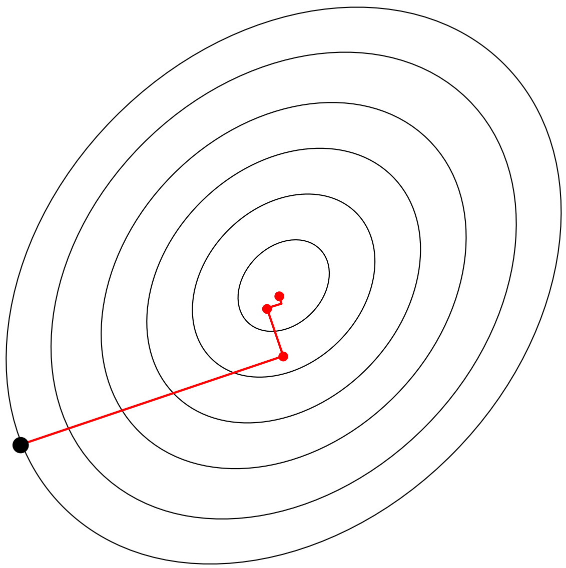


(large $\kappa = L/\mu$)

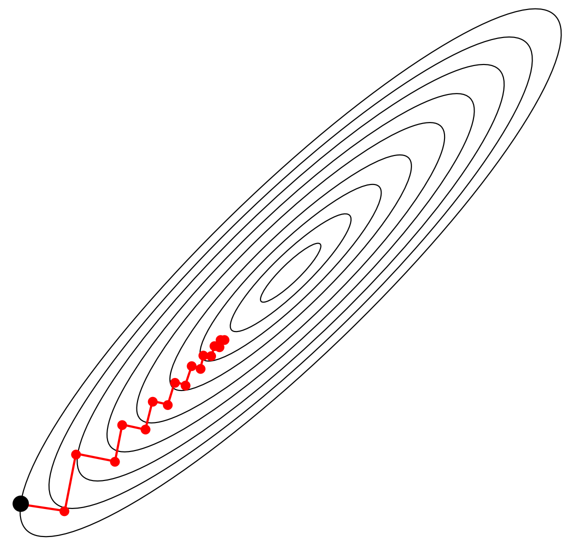
Iterative methods for minimizing smooth functions

- **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)*

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



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



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

Gradient descent - Proof for quadratic functions

- Quadratic convex function: $g(\theta) = \frac{1}{2}\theta^\top H\theta - c^\top \theta$
  - μ and L are the smallest and largest eigenvalues of H
  - Global optimum $\theta_* = H^{-1}c$ (or $H^\dagger c$) such that $H\theta_* = c$

Gradient descent - Proof for quadratic functions

- Quadratic convex function: $g(\theta) = \frac{1}{2}\theta^\top H\theta - c^\top\theta$
  - μ and L are the smallest and largest eigenvalues of H
  - Global optimum $\theta_* = H^{-1}c$ (or $H^\dagger c$) such that $H\theta_* = c$

- Gradient descent with $\gamma = 1/L$:

$$\begin{aligned} \theta_t &= \theta_{t-1} - \frac{1}{L}(H\theta_{t-1} - c) = \theta_{t-1} - \frac{1}{L}(H\theta_{t-1} - H\theta_*) \\ \theta_t - \theta_* &= (I - \frac{1}{L}H)(\theta_{t-1} - \theta_*) = (I - \frac{1}{L}H)^t(\theta_0 - \theta_*) \end{aligned}$$

Gradient descent - Proof for quadratic functions

- Quadratic convex function: $g(\theta) = \frac{1}{2}\theta^\top H\theta - c^\top\theta$
  - μ and L are the smallest and largest eigenvalues of H
  - Global optimum $\theta_* = H^{-1}c$ (or $H^\dagger c$) such that $H\theta_* = c$

- Gradient descent with $\gamma = 1/L$:

$$\begin{aligned}\theta_t &= \theta_{t-1} - \frac{1}{L}(H\theta_{t-1} - c) = \theta_{t-1} - \frac{1}{L}(H\theta_{t-1} - H\theta_*) \\ \theta_t - \theta_* &= (I - \frac{1}{L}H)(\theta_{t-1} - \theta_*) = (I - \frac{1}{L}H)^t(\theta_0 - \theta_*)\end{aligned}$$

- **Strong convexity** $\mu > 0$: eigenvalues of $(I - \frac{1}{L}H)^t$ in $[0, (1 - \frac{\mu}{L})^t]$
  - Convergence of iterates: $\|\theta_t - \theta_*\|^2 \leqslant (1 - \mu/L)^{2t}\|\theta_0 - \theta_*\|^2$
  - Function values: $g(\theta_t) - g(\theta_*) \leqslant (1 - \mu/L)^{2t}\big[g(\theta_0) - g(\theta_*)\big]$

Gradient descent - Proof for quadratic functions

- Quadratic convex function: $g(\theta) = \frac{1}{2}\theta^\top H\theta - c^\top\theta$
  - μ and L are the smallest and largest eigenvalues of H
  - Global optimum $\theta_* = H^{-1}c$ (or $H^\dagger c$) such that $H\theta_* = c$

- Gradient descent with $\gamma = 1/L$:

$$\begin{aligned}
\theta_t &= \theta_{t-1} - \frac{1}{L}(H\theta_{t-1} - c) = \theta_{t-1} - \frac{1}{L}(H\theta_{t-1} - H\theta_*) \\
\theta_t - \theta_* &= (I - \frac{1}{L}H)(\theta_{t-1} - \theta_*) = (I - \frac{1}{L}H)^t(\theta_0 - \theta_*)
\end{aligned}$$

- **Convexity** $\mu = 0$: eigenvalues of $(I - \frac{1}{L}H)^t$ in $[0, 1]$
  - No convergence of iterates: $\|\theta_t - \theta_*\|^2 \leqslant \|\theta_0 - \theta_*\|^2$
  - Function values: $g(\theta_t) - g(\theta_*) \leqslant \max_{v\in[0,L]} v(1 - v/L)^{2t}\|\theta_0 - \theta_*\|^2$

    $g(\theta_t) - g(\theta_*) \leqslant \frac{L}{t}\|\theta_0 - \theta_*\|^2$ *(board)*

Board

- No convergence of iterates: $\|\theta_t - \theta_*\|^2 \leqslant \|\theta_0 - \theta_*\|^2$

- $g(\theta_t) - g(\theta_*) = \frac{1}{2}(\theta_t - \theta_*)^\top H(\theta_t - \theta_*)$, which is equal to

$$\frac{1}{2}(\theta_0 - \theta_*)^\top H(I - \gamma H)^{2t}\theta_0 - \theta_*)$$

- Function values: $g(\theta_t) - g(\theta_*) \leqslant \max_{v\in[0,L]} v(1 - v/L)^{2t}\|\theta_0 - \theta_*\|^2$

Board

- No convergence of iterates: $\|\theta_t - \theta_*\|^2 \leqslant \|\theta_0 - \theta_*\|^2$
- $g(\theta_t) - g(\theta_*) = \frac{1}{2}(\theta_t - \theta_*)^\top H(\theta_t - \theta_*)$, which is equal to

$$\frac{1}{2}(\theta_0 - \theta_*)^\top H(I - \gamma H)^{2t}\theta_0 - \theta_*)$$

- Function values: $g(\theta_t) - g(\theta_*) \leqslant \max_{v \in [0,L]} v(1 - v/L)^{2t}\|\theta_0 - \theta_*\|^2$

$$\begin{aligned}
v(1-v/L)^{2t} &\leqslant v\exp(-v/L)^{2t} = v\exp(-2tv/L) \\
&\leqslant (2tv/L)\exp(-2tv/L) \times \frac{L}{2t} \\
&\leqslant \max_{\alpha \geqslant 0} \alpha \exp(-\alpha) \times \frac{L}{2t} = O(\frac{L}{2t})
\end{aligned}$$

Iterative methods for minimizing smooth functions

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

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

Iterative methods for minimizing smooth functions

- **Assumption**: g convex and L-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 *(see board)*

Board

- Second-order Taylor expansion

$$g(\theta) \approx g(\theta_{t-1}) + g'(\theta_{t-1})^\top(\theta - \theta_{t-1}) + \frac{1}{2}(\theta - \theta_{t-1})^\top g''(\theta_{t-1})(\theta - \theta_{t-1})$$

  – Minimization by zeroing gradient:

$$g'(\theta_{t-1}) + g''(\theta_{t-1})(\theta - \theta_{t-1}) = 0$$

  – Iteration: $\theta_t = \theta_{t-1} - g''(\theta_{t-1})^{-1} g'(\theta_{t-1})$

- Local quadratic convergence: $\|\theta_t - \theta_*\| = O(\|\theta_{t-1} - \theta_*\|^2)$

Iterative methods for minimizing smooth functions

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

- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_t \, g'(\theta_{t-1})$
  - $O(1/t)$ 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 L-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 L-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 L-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

Outline

1. **Introduction/motivation: Supervised machine learning**
   - Machine learning $\approx$ optimization of finite sums
   - Batch optimization methods

2. **Fast stochastic gradient methods for convex problems**
   - Variance reduction: for *training* error
   - Constant step-sizes: for *testing* error

3. **Beyond convex problems**
   - Generic algorithms with generic "guarantees"
   - Global convergence for over-parameterized neural networks

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**: find $\hat{\theta}$ solution of

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

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

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: $O(d \cdot \kappa/\varepsilon)$

Impact of averaging (Bach and Moulines, 2011)

- Stochastic gradient descent with learning rate $\gamma_t = Ct^{-\alpha}$

- **Strongly convex smooth objective functions**
  - Non-asymptotic analysis with explicit constants
  - Forgetting of initial conditions
  - Robustness to the choice of C

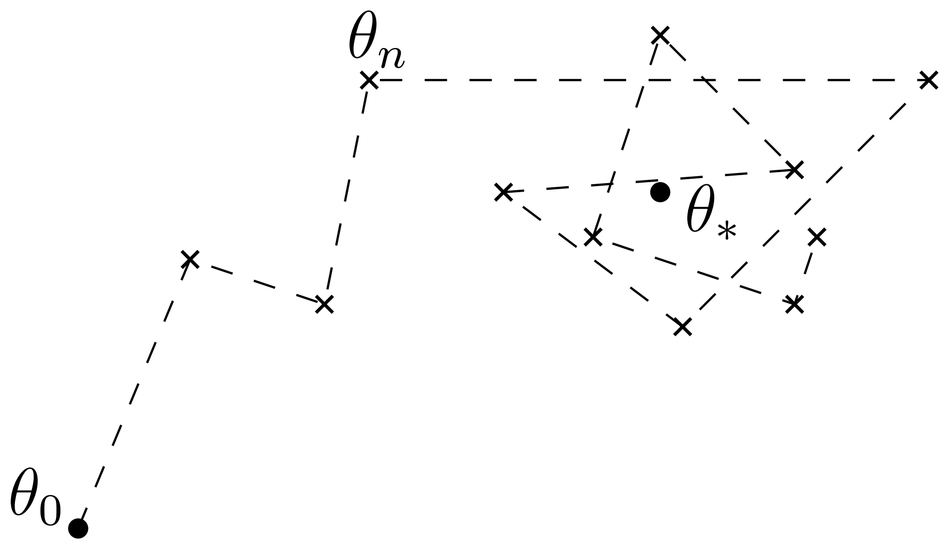
Impact of averaging (Bach and Moulines, 2011)

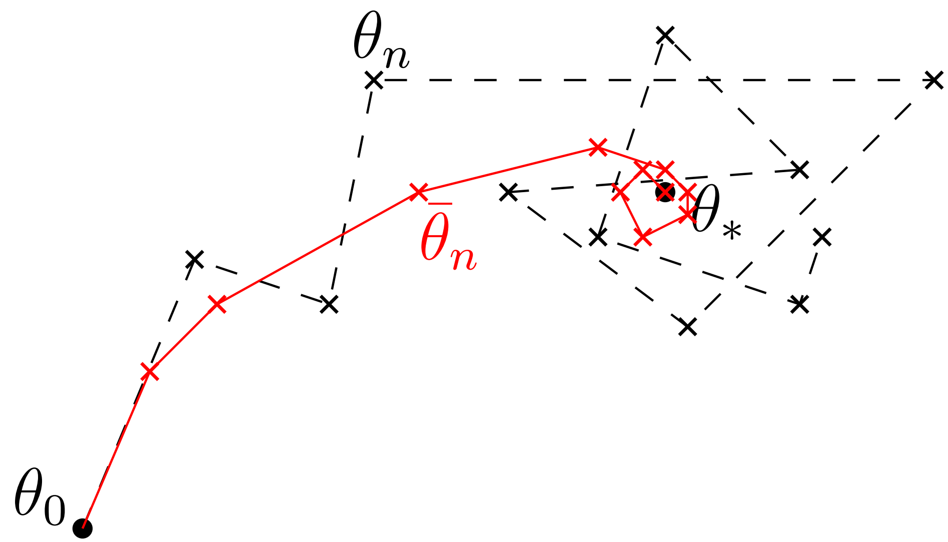
- Stochastic gradient descent with learning rate $\gamma_t = Ct^{-\alpha}$
- **Strongly convex smooth objective functions**
  - Non-asymptotic analysis with explicit constants
  - Forgetting of initial conditions
  - Robustness to the choice of C
- **Convergence rates** for $\mathbb{E}\|\theta_t - \theta_*\|^2$ and $\mathbb{E}\|\bar{\theta}_t - \theta_*\|^2$
  - no averaging: $O\Big(\frac{\sigma^2\gamma_t}{\mu}\Big) + O(e^{-\mu t \gamma_t})\|\theta_0 - \theta_*\|^2$
  - averaging: $\frac{\operatorname{tr} H(\theta_*)^{-1}}{t} + O\Big(\frac{\|\theta_0 - \theta_*\|^2}{\mu^2 t^2}\Big)$

    (see board)

Board

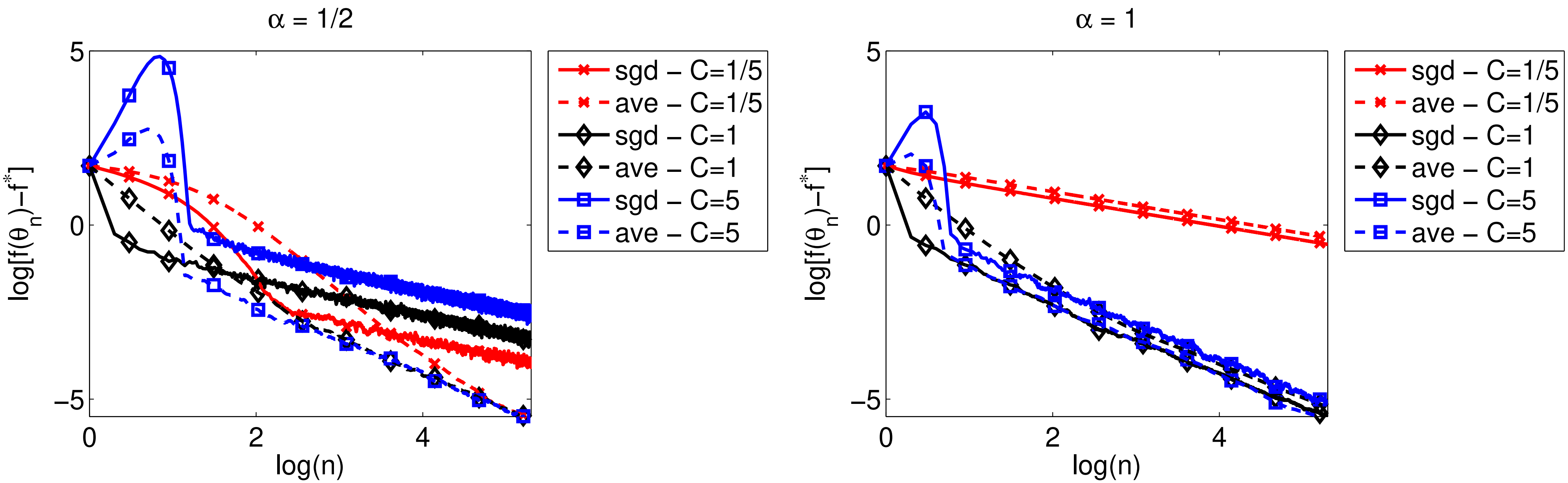
- Leaving initial point θ_0 to reach θ_*
- Impact of averaging





Robustness to wrong constants for $\gamma_t = Ct^{-\alpha}$

- $f(\theta) = \frac{1}{2}|\theta|^2$ with i.i.d. Gaussian noise ($d = 1$)
- Left: $\alpha = 1/2$
- Right: $\alpha = 1$



- See also `http://leon.bottou.org/projects/sgd`

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

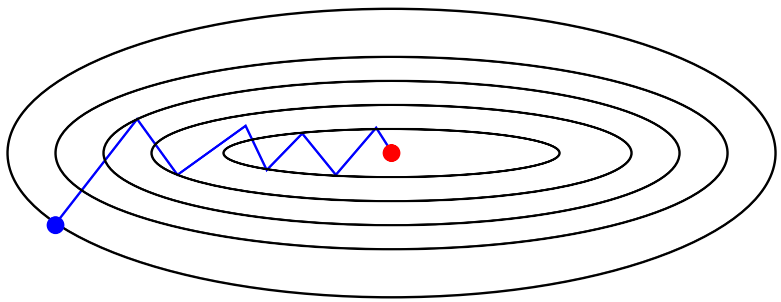
- Minimizing $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i,\theta)\big) + \lambda\Omega(\theta)$

- Batch gradient descent: $\theta_t = \theta_{t-1} - \gamma_t g'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma_t}{n}\sum_{i=1}^{n} f_i'(\theta_{t-1})$
  - Linear (e.g., exponential) convergence rate in $O(e^{-t/\kappa})$
  - 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: $\theta_t = \theta_{t-1} - \gamma_t g'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma_t}{n}\sum_{i=1}^{n} f_i'(\theta_{t-1})$



Stochastic vs. deterministic methods

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

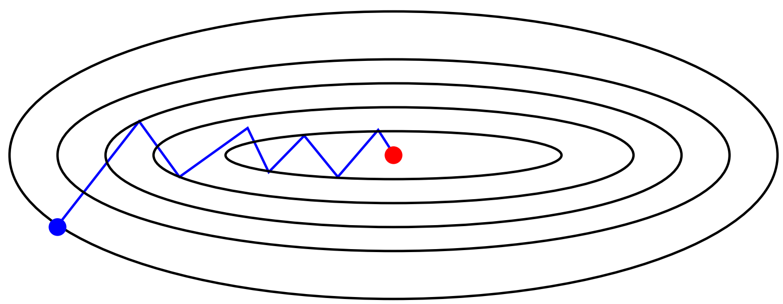
- Batch gradient descent: $\theta_t = \theta_{t-1} - \gamma_t g'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma_t}{n}\sum_{i=1}^{n} f_i'(\theta_{t-1})$
  - Linear (e.g., exponential) convergence rate in $O(e^{-t/\kappa})$
  - Iteration complexity is linear in n

- Stochastic gradient descent: $\theta_t = \theta_{t-1} - \gamma_t f'_{i(t)}(\theta_{t-1})$
  - Sampling with replacement: $i(t)$ random element of $\{1,\dots,n\}$
  - Convergence rate in $O(\kappa/t)$
  - Iteration complexity is independent of n

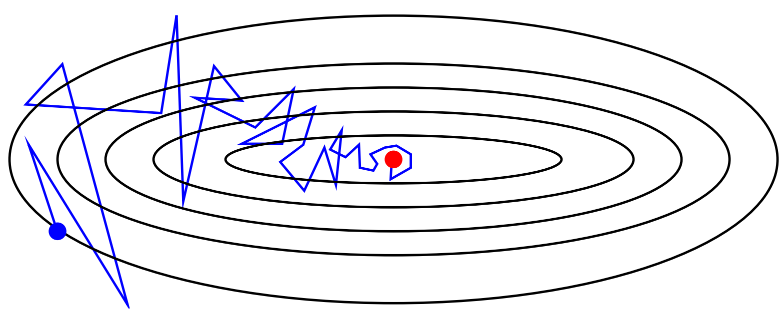
Stochastic vs. deterministic methods

- 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: $\theta_t = \theta_{t-1} - \gamma_t g'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma_t}{n}\sum_{i=1}^{n} f_i'(\theta_{t-1})$

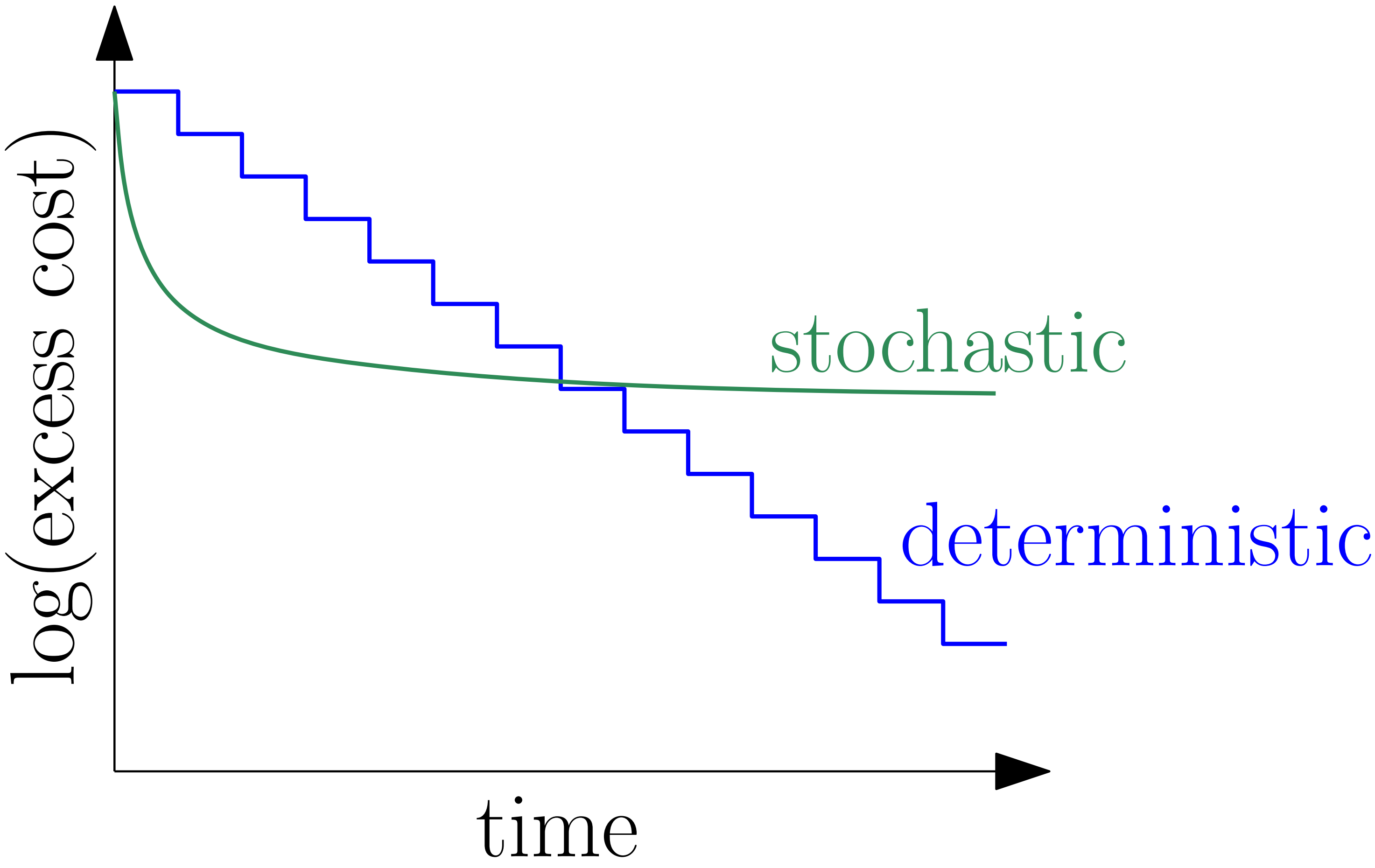


- Stochastic gradient descent: $\theta_t = \theta_{t-1} - \gamma_t f'_{i(t)}(\theta_{t-1})$



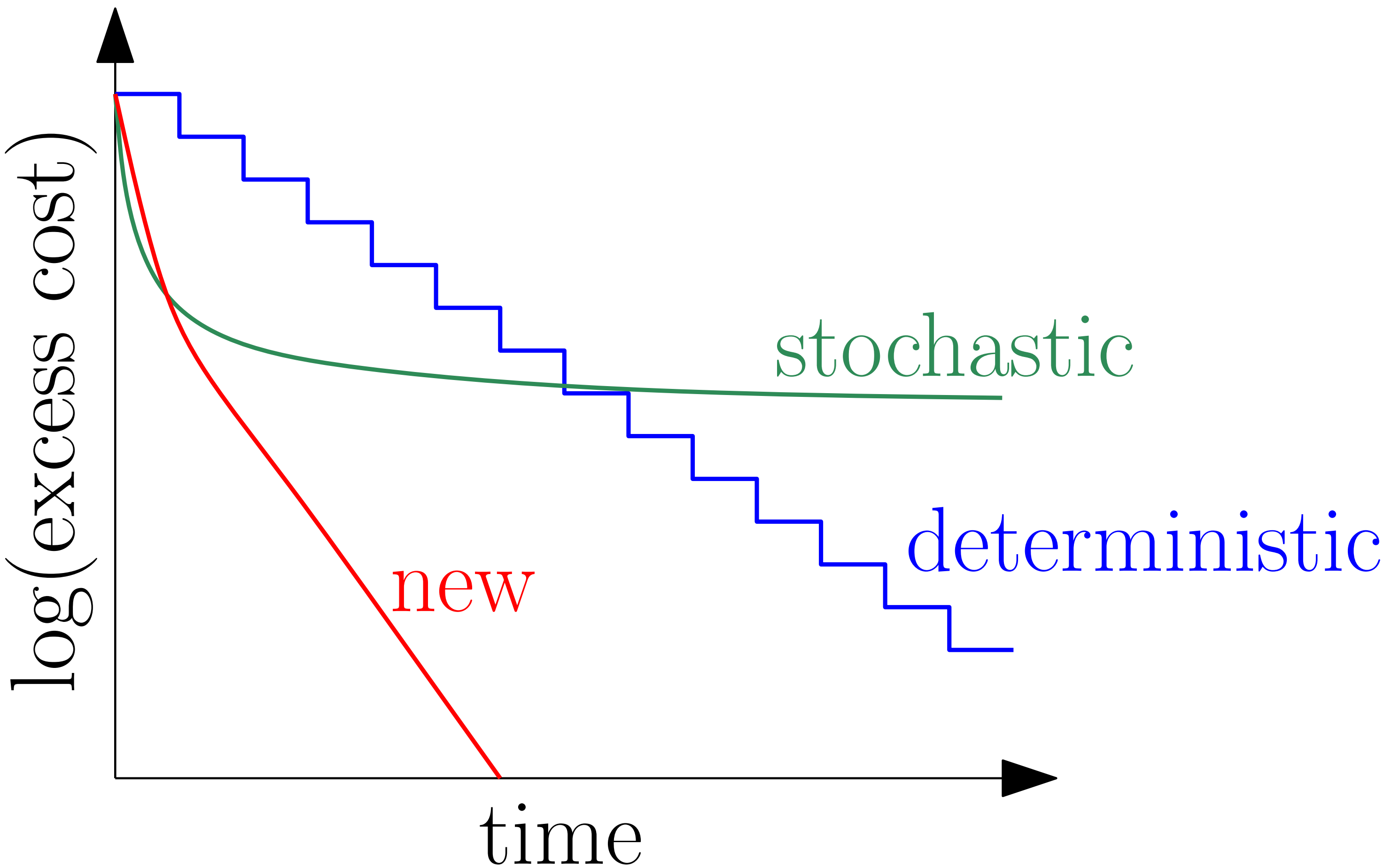
Stochastic vs. deterministic methods

- **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



Accelerating gradient methods - Related work

- **Generic acceleration** (Nesterov, 1983, 2004)

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

Accelerating gradient methods - Related work

- **Generic acceleration** (Nesterov, 1983, 2004)

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

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

    $g(\theta_t) - g(\theta_*) \leqslant O(1/t^2)$

    $g(\theta_t) - g(\theta_*) \leqslant O(e^{-t\sqrt{\mu/L}}) = O(e^{-t/\sqrt{\kappa}})$ if μ-strongly convex
  - Optimal rates after $t = O(d)$ iterations (Nesterov, 2004)

Accelerating gradient methods - Related work

- **Generic acceleration** (Nesterov, 1983, 2004)

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

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

    $g(\theta_t) - g(\theta_*) \leqslant O(1/t^2)$

    $g(\theta_t) - g(\theta_*) \leqslant O(e^{-t\sqrt{\mu/L}}) = O(e^{-t/\sqrt{\kappa}})$ if μ-strongly convex
  - Optimal rates after $t = O(d)$ iterations (Nesterov, 2004)
  - Still $O(nd)$ iteration cost: complexity $= O(nd \cdot \sqrt{\kappa} \log \frac{1}{\varepsilon})$

Accelerating gradient methods - Related work

- **Constant step-size stochastic gradient**
  - Solodov (1998); Nedic and Bertsekas (2000)
  - Linear convergence, but only up to a fixed tolerance

Accelerating gradient methods - Related work

- **Constant step-size stochastic gradient**
  - Solodov (1998); Nedic and Bertsekas (2000)
  - Linear convergence, but only up to a fixed tolerance
- **Stochastic methods in the dual (SDCA)**
  - Shalev-Shwartz and Zhang (2013)
  - Similar linear rate but limited choice for the f_i's
  - Extensions without duality: see Shalev-Shwartz (2016)

Accelerating gradient methods - Related work

- **Constant step-size stochastic gradient**
  - Solodov (1998); Nedic and Bertsekas (2000)
  - Linear convergence, but only up to a fixed tolerance
- **Stochastic methods in the dual (SDCA)**
  - Shalev-Shwartz and Zhang (2013)
  - Similar linear rate but limited choice for the f_i's
  - Extensions without duality: see Shalev-Shwartz (2016)
- **Stochastic version of accelerated batch gradient methods**
  - Tseng (1998); Ghadimi and Lan (2010); Xiao (2010)
  - Can improve constants, but still have sublinear $O(1/t)$ rate

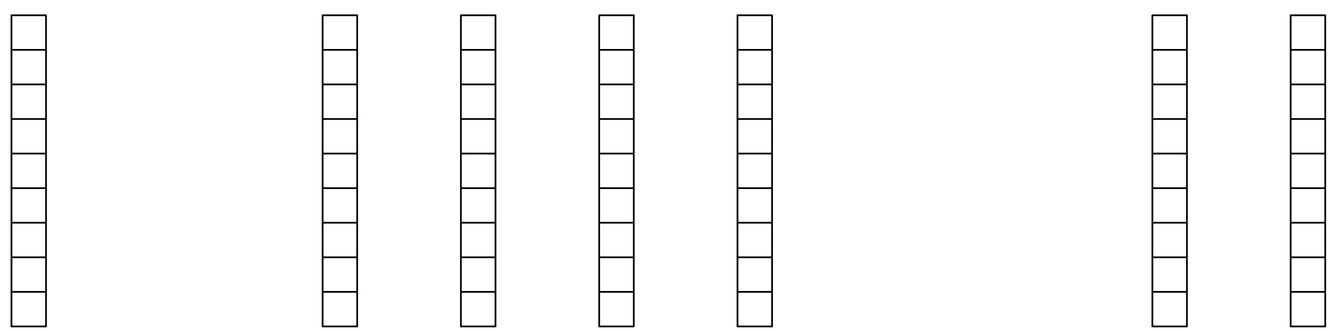
Stochastic average gradient (Le Roux, Schmidt, and Bach, 2012)

- **Stochastic average gradient** (SAG) iteration
  - Keep in memory the gradients of all functions f_i, $i = 1, \dots, n$
  - Random selection $i(t) \in \{1, \dots, n\}$ with replacement
  - Iteration: $\theta_t = \theta_{t-1} - \frac{\gamma_t}{n} \sum_{i=1}^{n} y_i^t$ with $y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$

Stochastic average gradient (Le Roux, Schmidt, and Bach, 2012)

- **Stochastic average gradient** (SAG) iteration
  - Keep in memory the gradients of all functions f_i, $i = 1, \dots, n$
  - Random selection $i(t) \in \{1, \dots, n\}$ with replacement
  - Iteration: $\theta_t = \theta_{t-1} - \frac{\gamma_t}{n} \sum_{i=1}^{n} y_i^t$ with $y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$

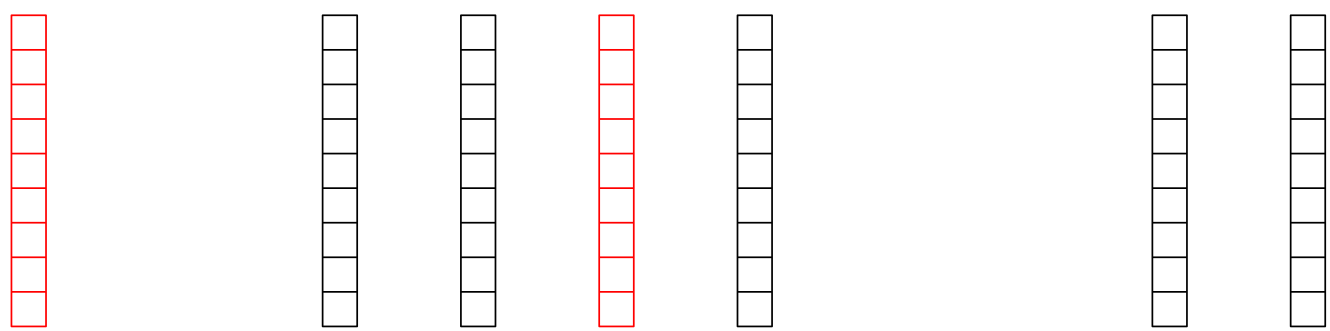
functions	$g = \frac{1}{n}\sum_{i=1}^{n} f_i$	f_1	f_2	f_3	f_4	$\cdots$	f_{n-1}	f_n
gradients $\in \mathbb{R}^d$	$\frac{1}{n}\sum_{i=1}^{n} y_i^t$	y_1^t	y_2^t	y_3^t	y_4^t	$\cdots$	y_{n-1}^t	y_n^t



Stochastic average gradient
(Le Roux, Schmidt, and Bach, 2012)

- **Stochastic average gradient** (SAG) iteration
  - Keep in memory the gradients of all functions f_i, $i = 1, \dots, n$
  - Random selection $i(t) \in \{1, \dots, n\}$ with replacement
  - Iteration: $\theta_t = \theta_{t-1} - \dfrac{\gamma_t}{n} \displaystyle\sum_{i=1}^{n} y_i^t$ with $y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$

functions	$g = \frac{1}{n}\sum_{i=1}^n f_i$	f_1	f_2	f_3	f_4	$\cdots$	f_{n-1}	f_n
gradients $\in \mathbb{R}^d$	$\frac{1}{n}\sum_{i=1}^n y_i^t$	y_1^t	y_2^t	y_3^t	y_4^t	$\cdots$	y_{n-1}^t	y_n^t

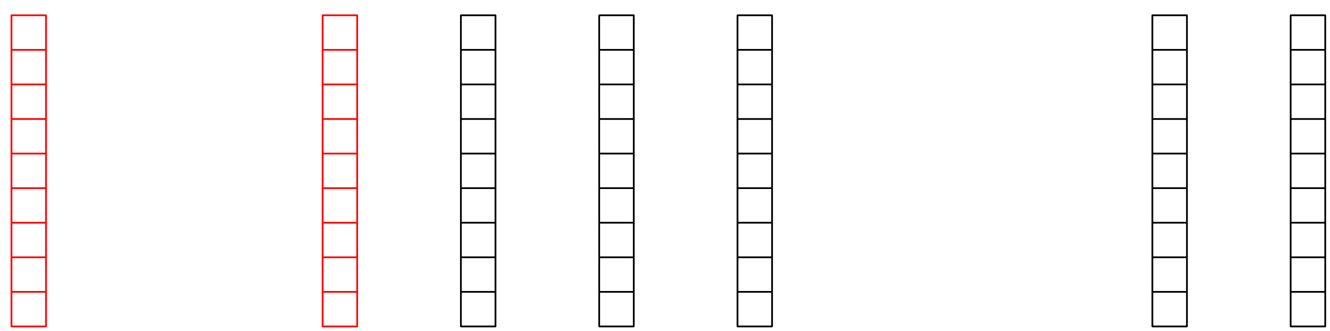


Stochastic average gradient (Le Roux, Schmidt, and Bach, 2012)

- **Stochastic average gradient** (SAG) iteration
  - Keep in memory the gradients of all functions f_i, $i = 1, \dots, n$
  - Random selection $i(t) \in \{1, \dots, n\}$ with replacement
  - Iteration: $\theta_t = \theta_{t-1} - \frac{\gamma_t}{n} \sum_{i=1}^{n} y_i^t$ with $y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$

functions $\quad g = \frac{1}{n} \sum_{i=1}^{n} f_i \quad f_1 \quad f_2 \quad f_3 \quad f_4 \quad \cdots \quad f_{n-1} \quad f_n$

gradients $\in \mathbb{R}^d \quad \frac{1}{n} \sum_{i=1}^{n} y_i^t \quad y_1^t \quad y_2^t \quad y_3^t \quad y_4^t \quad \cdots \quad y_{n-1}^t \quad y_n^t$



Stochastic average gradient (Le Roux, Schmidt, and Bach, 2012)

- **Stochastic average gradient** (SAG) iteration
  - Keep in memory the gradients of all functions f_i, $i = 1, \dots, n$
  - Random selection $i(t) \in \{1, \dots, n\}$ with replacement
  - Iteration: $\theta_t = \theta_{t-1} - \frac{\gamma_t}{n} \sum_{i=1}^{n} y_i^t$ with $y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$

- Stochastic version of incremental average gradient (Blatt et al., 2008)

Stochastic average gradient (Le Roux, Schmidt, and Bach, 2012)

- **Stochastic average gradient** (SAG) iteration
  - Keep in memory the gradients of all functions f_i, $i = 1, \dots, n$
  - Random selection $i(t) \in \{1, \dots, n\}$ with replacement
  - Iteration: $\theta_t = \theta_{t-1} - \frac{\gamma_t}{n} \sum_{i=1}^{n} y_i^t$ with $y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$
- Stochastic version of incremental average gradient (Blatt et al., 2008)
- **Extra memory requirement**: n gradients in $\mathbb{R}^d$ in general
- **Linear supervised machine learning**: only n real numbers
  - If $f_i(\theta) = \ell(y_i, \Phi(x_i)^\top \theta)$, then $f_i'(\theta) = \ell'(y_i, \Phi(x_i)^\top \theta)\, \Phi(x_i)$

Running-time comparisons (strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth and g μ-strongly convex

Stochastic gradient descent	$d\times$	$\frac{L}{\mu}$	$\times \; \frac{1}{\varepsilon}$
Gradient descent	$d\times$	$n\frac{L}{\mu}$	$\times \log \frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times$	$n\sqrt{\frac{L}{\mu}}$	$\times \log \frac{1}{\varepsilon}$
SAG	$d\times$	$(n + \frac{L}{\mu})$	$\times \log \frac{1}{\varepsilon}$

NB: slightly different (smaller) notion of condition number for batch methods

Running-time comparisons (strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth and g μ-strongly convex

Stochastic gradient descent	$d\times$	$\frac{L}{\mu}$	$\times$	$\frac{1}{\varepsilon}$
Gradient descent	$d\times$	$n\frac{L}{\mu}$	$\times \log$	$\frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times$	$n\sqrt{\frac{L}{\mu}}$	$\times \log$	$\frac{1}{\varepsilon}$
SAG	$d\times$	$(n+\frac{L}{\mu})$	$\times \log$	$\frac{1}{\varepsilon}$

- **Beating two lower bounds** (Nemirovski and Yudin, 1983; Nesterov, 2004): with additional assumptions

(1) stochastic gradient: exponential rate for finite sums
(2) full gradient: better exponential rate using the sum structure

Running-time comparisons (non-strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth
  - Ill conditioned problems: g may not be strongly-convex ($\mu = 0$)

Stochastic gradient descent	$d\times$	$1/\varepsilon^2$
Gradient descent	$d\times$	n/ε
Accelerated gradient descent	$d\times$	$n/\sqrt{\varepsilon}$
SAG	$d\times$	$\sqrt{n}/\varepsilon$

- Adaptivity to potentially hidden strong convexity

- No need to know the local/global strong-convexity constant

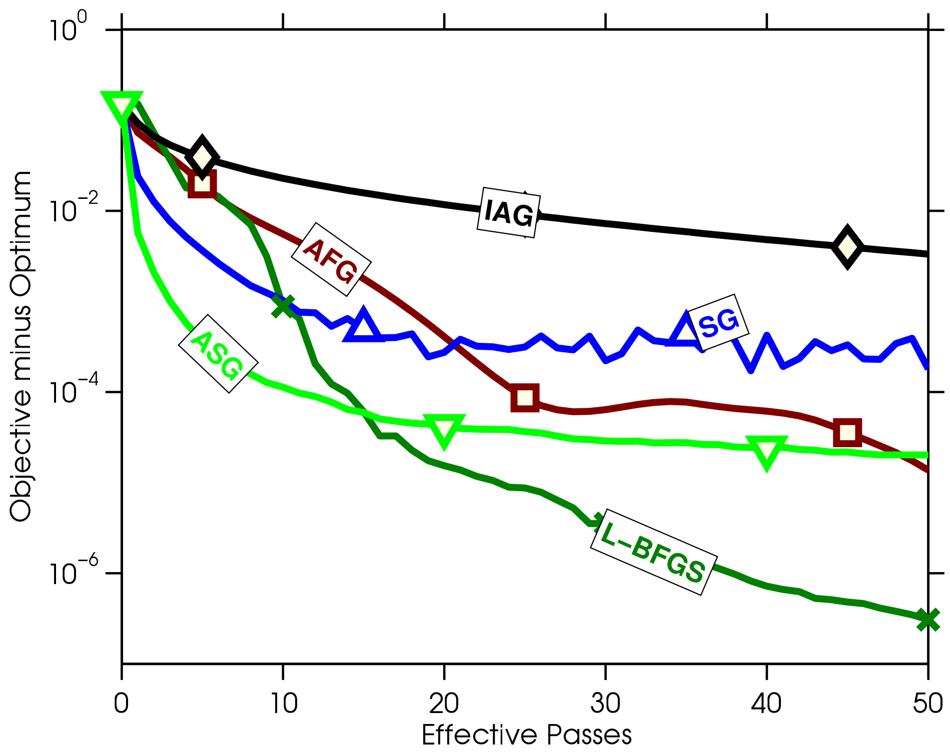
Stochastic average gradient
Implementation details and extensions

- **Sparsity in the features**
  - Just-in-time updates $\Rightarrow$ replace $O(d)$ by number of non zeros
  - See also Leblond, Pedregosa, and Lacoste-Julien (2016)
- **Mini-batches**
  - Reduces the memory requirement $+$ block access to data
- **Line-search**
  - Avoids knowing L in advance
- **Non-uniform sampling**
  - Favors functions with large variations
- See `www.cs.ubc.ca/~schmidtm/Software/SAG.html`

Experimental results (logistic regression)

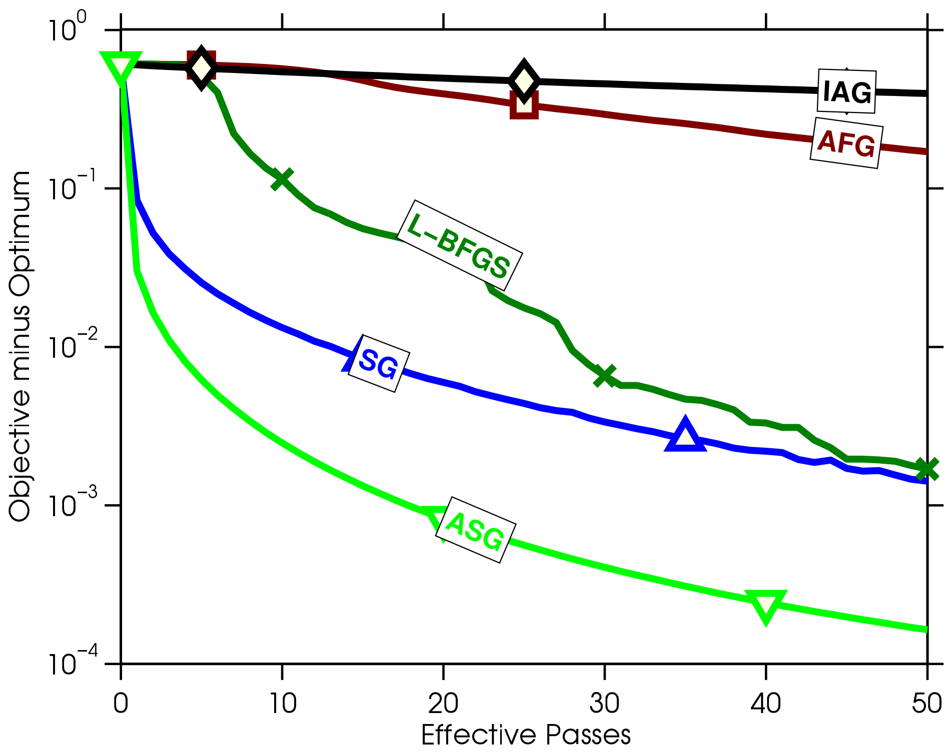
`quantum` dataset

$(n = 50\ 000,\ d = 78)$



`rcv1` dataset

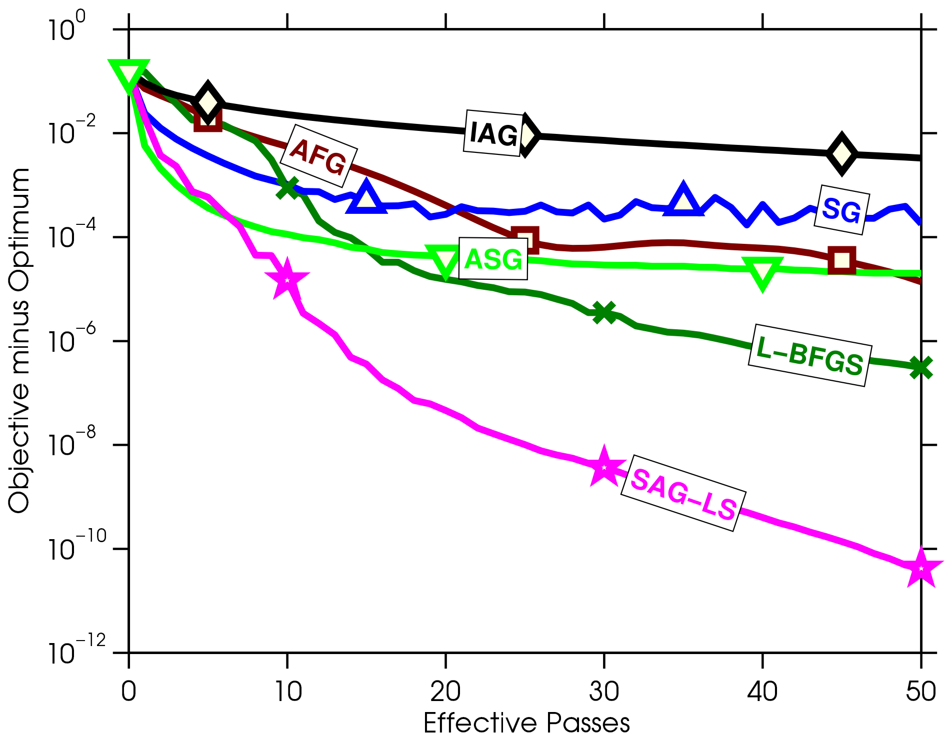
$(n = 697\ 641,\ d = 47\ 236)$

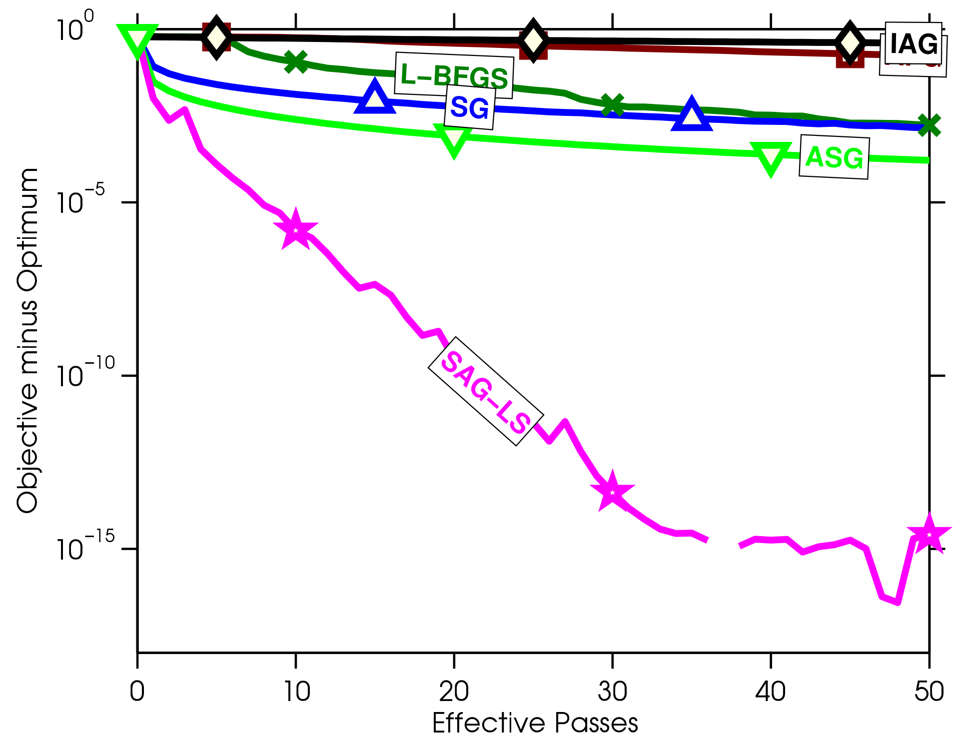


Experimental results (logistic regression)

`quantum` dataset
$(n = 50\ 000,\ d = 78)$

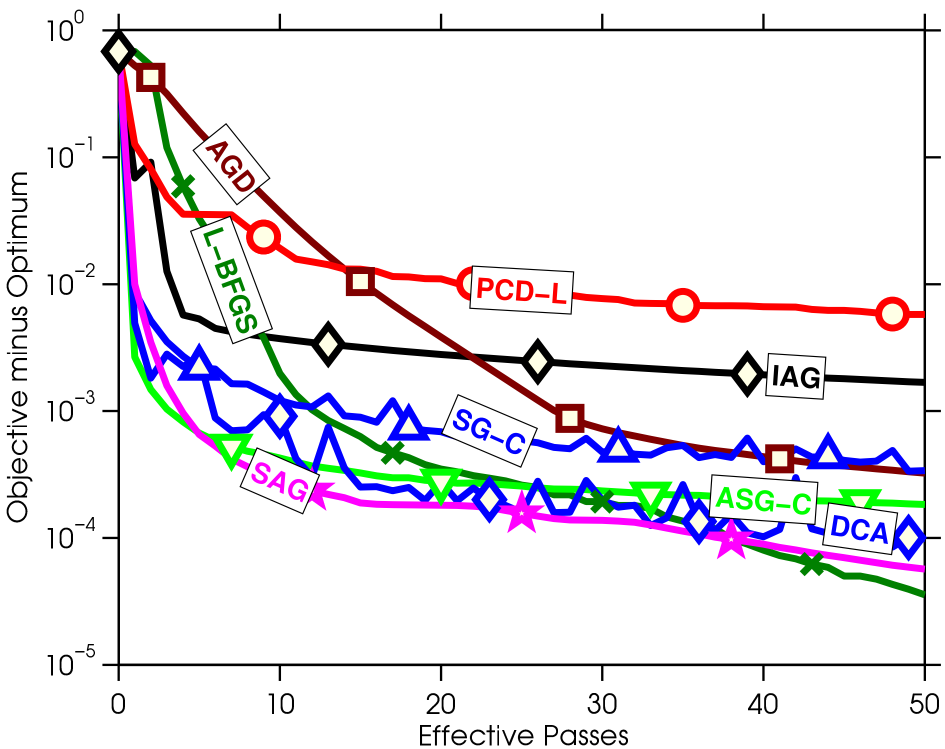
`rcv1` dataset
$(n = 697\ 641,\ d = 47\ 236)$



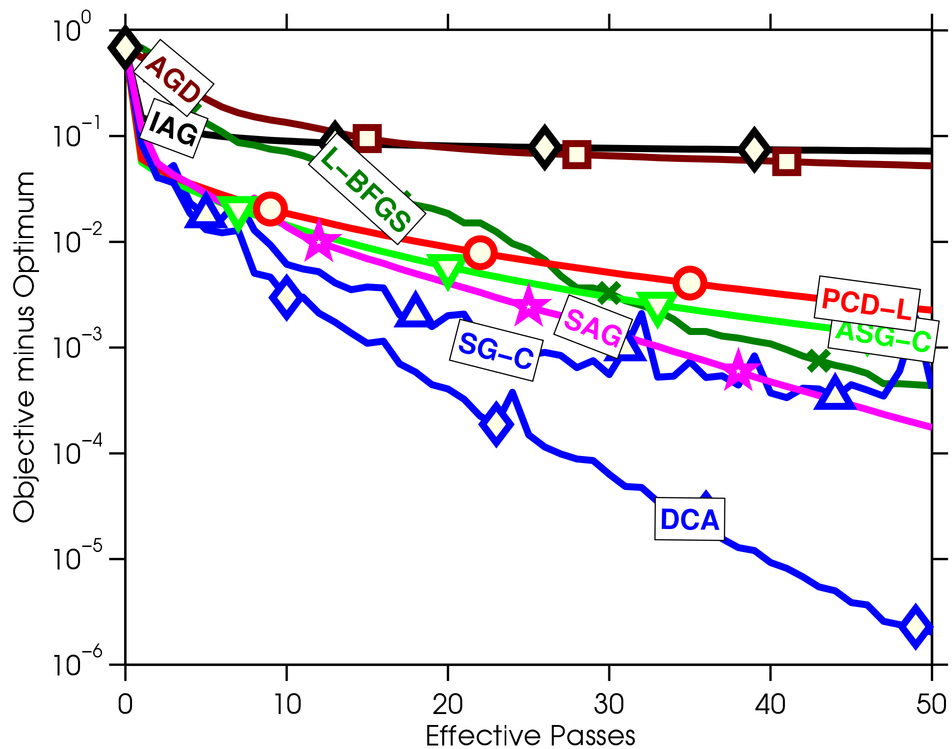


Before non-uniform sampling

`protein` dataset
$(n = 145\ 751,\ d\ = 74)$

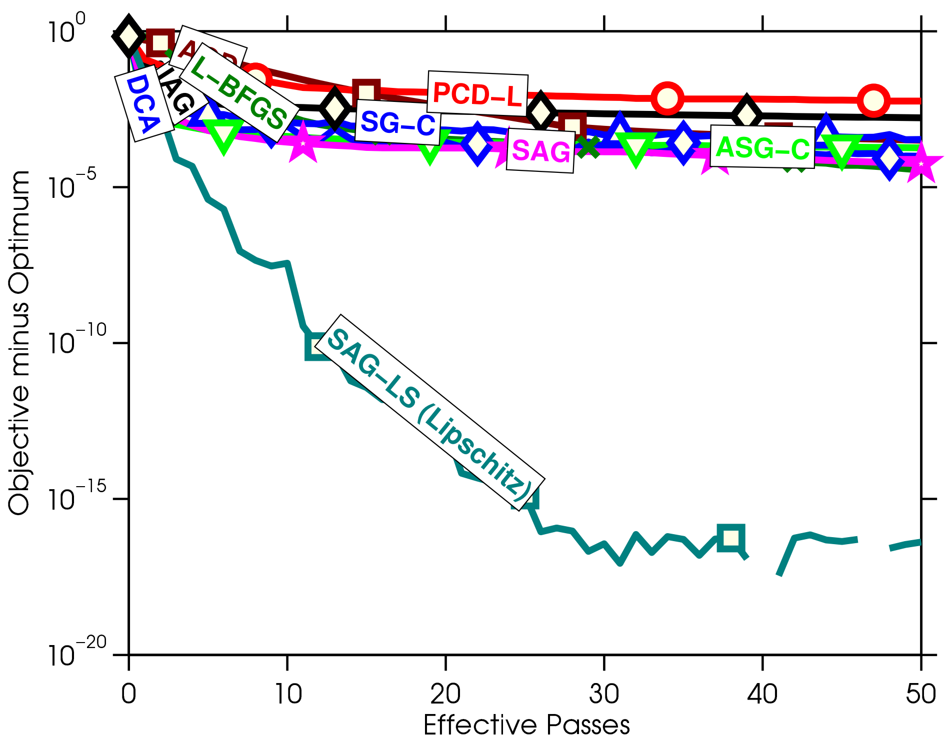


`sido` dataset
$(n = 12\ 678,\ d = 4\ 932)$

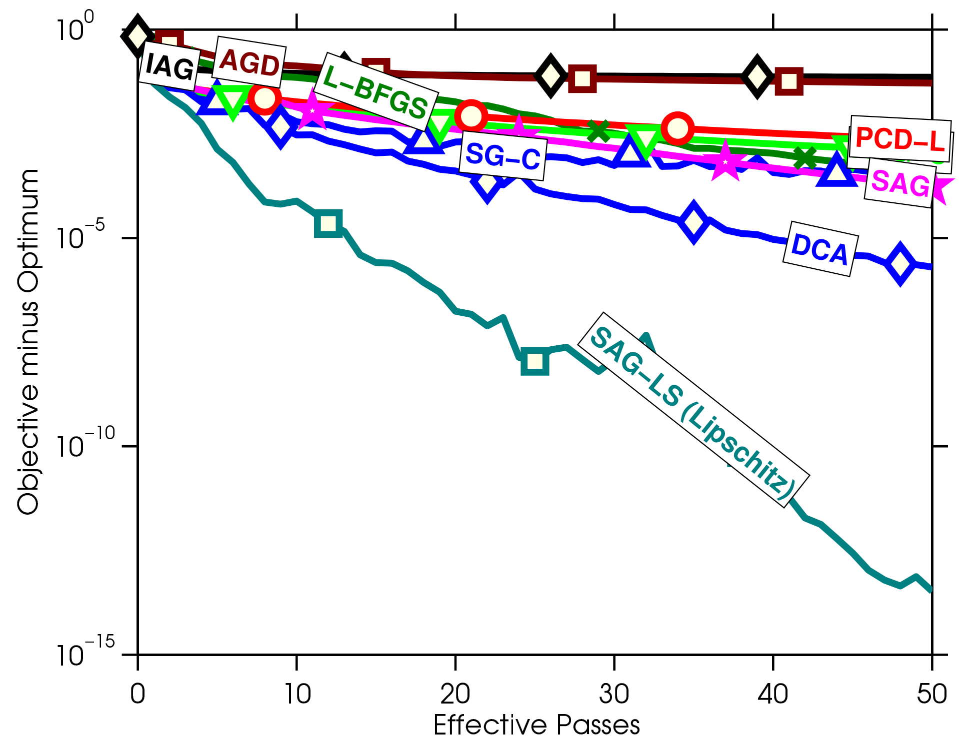


After non-uniform sampling

`protein` dataset
$(n = 145\ 751,\ d = 74)$



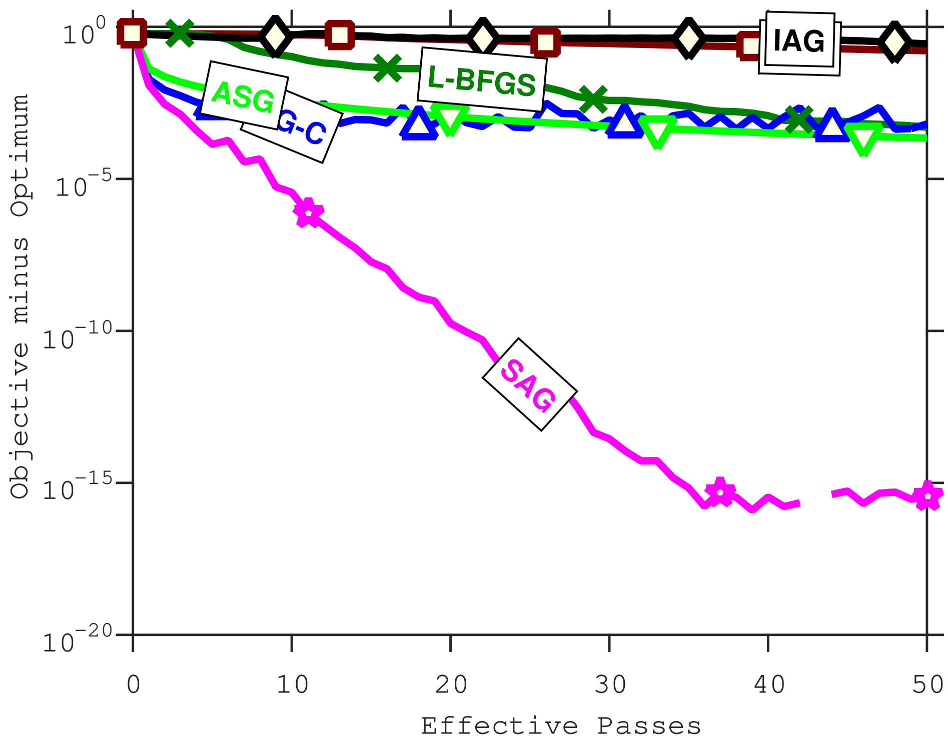
`sido` dataset
$(n = 12\ 678,\ d = 4\ 932)$



From training to testing errors

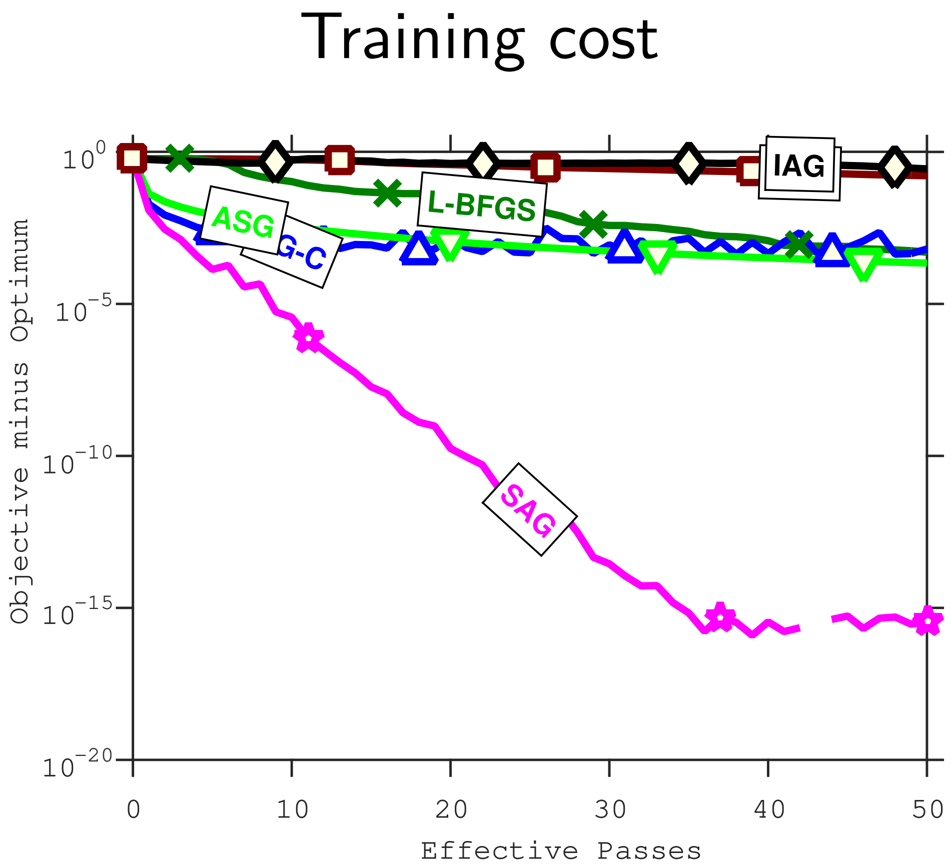
- rcv1 dataset $(n = 697\ 641,\ d = 47\ 236)$
  - NB: IAG, SG-C, ASG with optimal step-sizes in hindsight

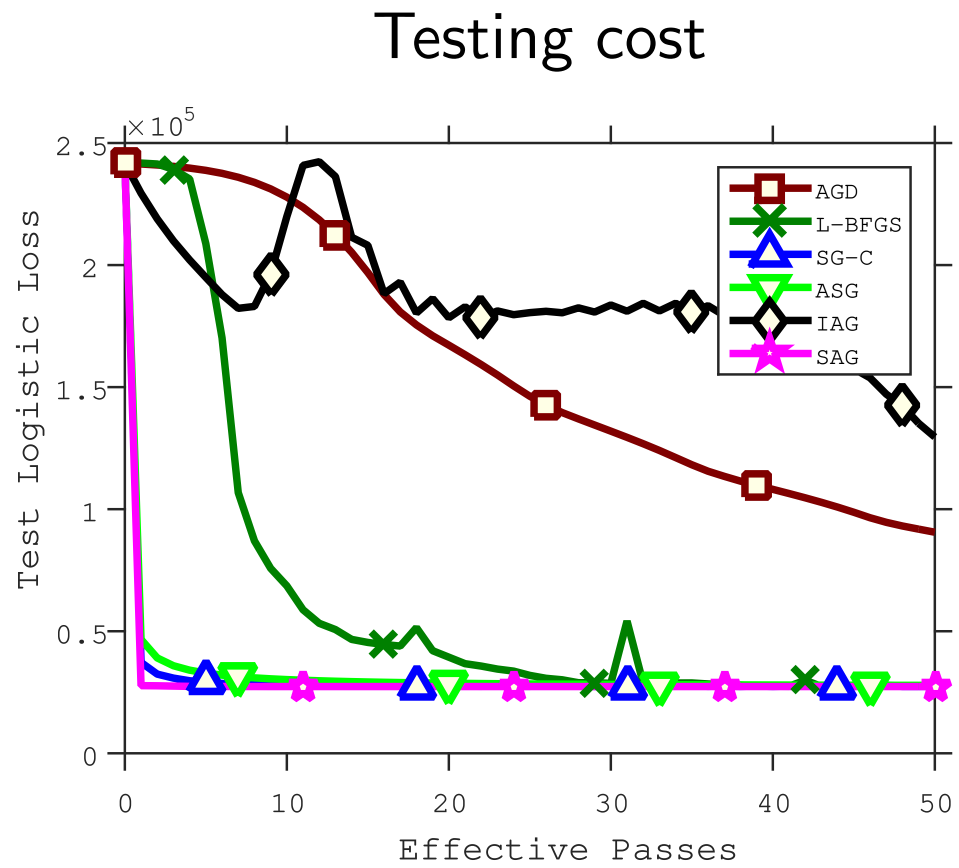
Training cost



From training to testing errors

- rcv1 dataset ($n = 697\ 641$, $d = 47\ 236$)
  - NB: IAG, SG-C, ASG with optimal step-sizes in hindsight





Linearly convergent stochastic gradient algorithms

- **Many related algorithms**
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SDCA (Shalev-Shwartz and Zhang, 2013)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - MISO (Mairal, 2015)
  - Finito (Defazio et al., 2014b)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014a)
  - ...

- **Similar rates of convergence and iterations**

Linearly convergent stochastic gradient algorithms

- **Many related algorithms**
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SDCA (Shalev-Shwartz and Zhang, 2013)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - MISO (Mairal, 2015)
  - Finito (Defazio et al., 2014b)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014a)
  - ...

- **Similar rates of convergence and iterations**

- **Different interpretations and proofs / proof lengths**
  - Lazy gradient evaluations
  - Variance reduction

Acceleration

- **Similar guarantees for finite sums**: SAG, SDCA, SVRG (Xiao and Zhang, 2014), SAGA, MISO (Mairal, 2015)

Gradient descent	$d\times$	$n\frac{L}{\mu}$	$\times \log\frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times$	$n\sqrt{\frac{L}{\mu}}$	$\times \log\frac{1}{\varepsilon}$
SAG(A), SVRG, SDCA, MISO	$d\times$	$(n+\frac{L}{\mu})$	$\times \log\frac{1}{\varepsilon}$
Accelerated versions	$d\times$	$(n+\sqrt{n\frac{L}{\mu}})$	$\times \log\frac{1}{\varepsilon}$

- **Acceleration for special algorithms** (e.g., Shalev-Shwartz and Zhang, 2014; Nitanda, 2014; Lan, 2015; Defazio, 2016)

- **Catalyst** (Lin, Mairal, and Harchaoui, 2015a)
  - Widely applicable generic acceleration scheme

SGD minimizes the testing cost!

- **Goal**: minimize $f(\theta) = \mathbb{E}_{p(x,y)} \ell(y, h(x, \theta))$
  - Given n independent samples (x_i, y_i), $i = 1, \dots, n$ from $p(x, y)$
  - Given a single pass of stochastic gradient descent
  - Bounds on the excess testing cost $\mathbb{E} f(\bar{\theta}_n) - \inf_{\theta \in \mathbb{R}^d} f(\theta)$

SGD minimizes the testing cost!

- **Goal**: minimize $f(\theta) = \mathbb{E}_{p(x,y)}\ell(y, h(x, \theta))$
  - Given n independent samples (x_i, y_i), $i = 1, \ldots, n$ from $p(x, y)$
  - Given a single pass of stochastic gradient descent
  - Bounds on the excess testing cost $\mathbb{E}f(\bar{\theta}_n) - \inf_{\theta\in\mathbb{R}^d} f(\theta)$

- **Optimal convergence rates**: $O(1/\sqrt{n})$ and $O(1/(n\mu))$
  - Optimal for non-smooth losses (Nemirovski and Yudin, 1983)
  - Attained by averaged SGD with decaying step-sizes

SGD minimizes the testing cost!

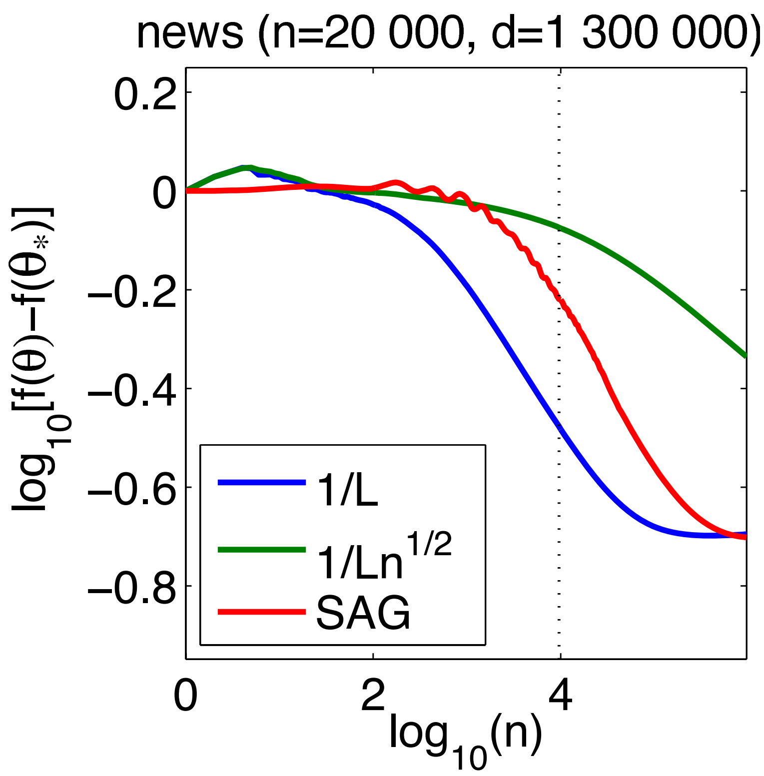
- **Goal**: minimize $f(\theta) = \mathbb{E}_{p(x,y)}\ell(y, h(x, \theta))$
  - Given n independent samples (x_i, y_i), $i = 1, \dots, n$ from $p(x, y)$
  - Given a single pass of stochastic gradient descent
  - Bounds on the excess testing cost $\mathbb{E}f(\bar{\theta}_n) - \inf_{\theta \in \mathbb{R}^d} f(\theta)$

- **Optimal convergence rates**: $O(1/\sqrt{n})$ and $O(1/(n\mu))$
  - Optimal for non-smooth losses (Nemirovski and Yudin, 1983)
  - Attained by averaged SGD with decaying step-sizes

- **Constant-step-size SGD**
  - Linear convergence up to the noise level for strongly-convex problems (Solodov, 1998; Nedic and Bertsekas, 2000)
  - Full convergence and robustness to ill-conditioning?

Robust averaged stochastic gradient (Bach and Moulines, 2013)

- **Constant-step-size SGD is convergent for least-squares**
  - Convergence rate in $O(1/n)$ without any dependence on μ
  - Simple choice of step-size (equal to $1/L$) *(see board)*

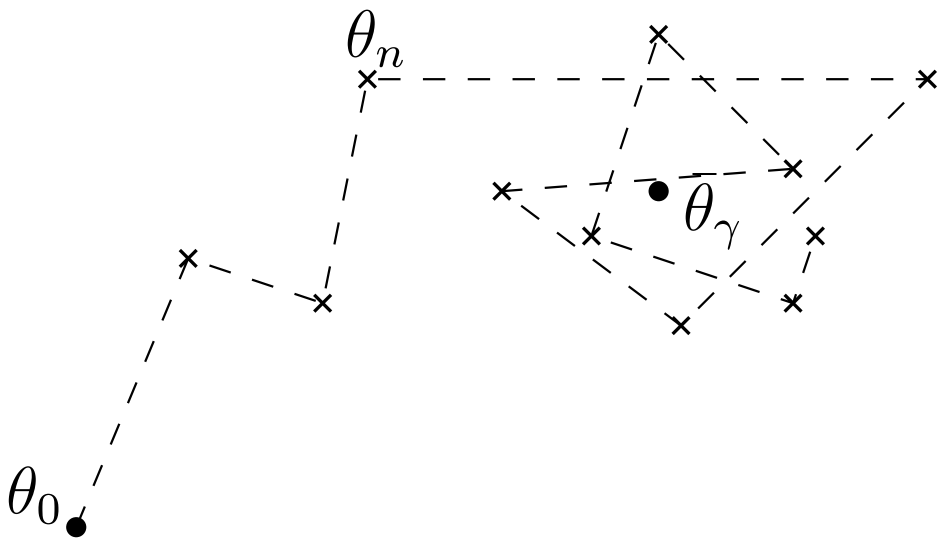


Markov chain interpretation of constant step sizes

- LMS recursion for $f_n(\theta) = \frac{1}{2}\big(y_n - \langle \Phi(x_n), \theta \rangle\big)^2$

$$\theta_n = \theta_{n-1} - \gamma\big(\langle \Phi(x_n), \theta_{n-1} \rangle - y_n\big)\Phi(x_n)$$

- The sequence $(\theta_n)_n$ is a homogeneous Markov chain
  - convergence to a stationary distribution π_γ
  - with expectation $\bar{\theta}_\gamma \stackrel{\text{def}}{=} \int \theta \pi_\gamma(\mathrm{d}\theta)$



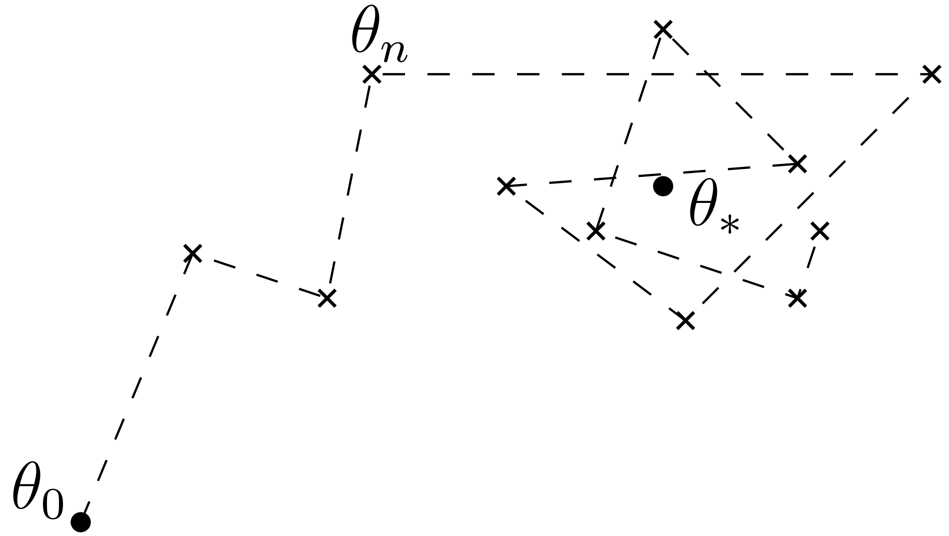
Markov chain interpretation of constant step sizes

- LMS recursion for $f_n(\theta) = \frac{1}{2}\big(y_n - \langle \Phi(x_n), \theta \rangle\big)^2$

$$\theta_n = \theta_{n-1} - \gamma\big(\langle \Phi(x_n), \theta_{n-1} \rangle - y_n\big)\Phi(x_n)$$

- The sequence $(\theta_n)_n$ is a homogeneous Markov chain
  - convergence to a stationary distribution π_γ
  - with expectation $\bar{\theta}_\gamma \stackrel{\text{def}}{=} \int \theta \pi_\gamma(\mathrm{d}\theta)$

- **For least-squares, $\bar{\theta}_\gamma = \theta_*$**



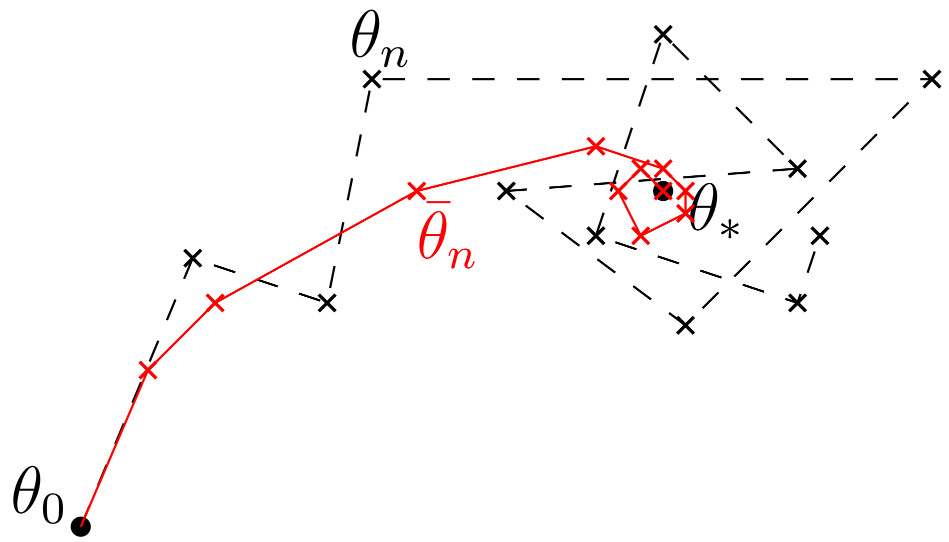
Markov chain interpretation of constant step sizes

- LMS recursion for $f_n(\theta) = \frac{1}{2}\big(y_n - \langle \Phi(x_n), \theta \rangle\big)^2$

$$\theta_n = \theta_{n-1} - \gamma\big(\langle \Phi(x_n), \theta_{n-1} \rangle - y_n\big)\Phi(x_n)$$

- The sequence $(\theta_n)_n$ is a homogeneous Markov chain
  - convergence to a stationary distribution π_γ
  - with expectation $\bar{\theta}_\gamma \stackrel{\mathrm{def}}{=} \int \theta \pi_\gamma(\mathrm{d}\theta)$

- **For least-squares,** $\bar{\theta}_\gamma = \theta_*$



Markov chain interpretation of constant step sizes

- LMS recursion for $f_n(\theta) = \frac{1}{2}\big(y_n - \langle \Phi(x_n), \theta \rangle\big)^2$

$$\theta_n = \theta_{n-1} - \gamma\big(\langle \Phi(x_n), \theta_{n-1} \rangle - y_n\big)\Phi(x_n)$$

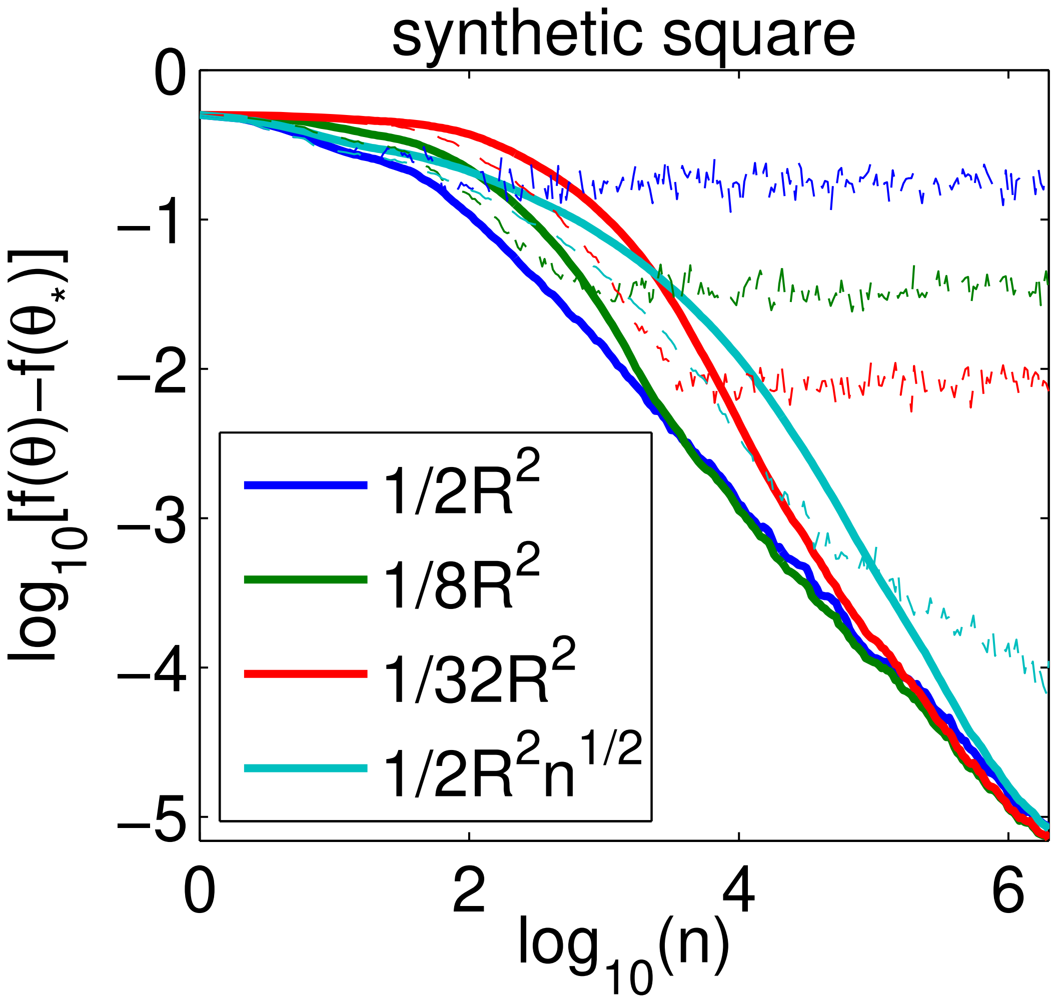
- The sequence $(\theta_n)_n$ is a homogeneous Markov chain
  - convergence to a stationary distribution π_γ
  - with expectation $\bar{\theta}_\gamma \stackrel{\text{def}}{=} \int \theta \pi_\gamma(\mathrm{d}\theta)$

- **For least-squares, $\bar{\theta}_\gamma = \theta_*$**
  - θ_n does not converge to θ_* but oscillates around it
  - oscillations of order $\sqrt{\gamma}$

- **Ergodic theorem:**
  - Averaged iterates converge to $\bar{\theta}_\gamma = \theta_*$ at rate $O(1/n)$

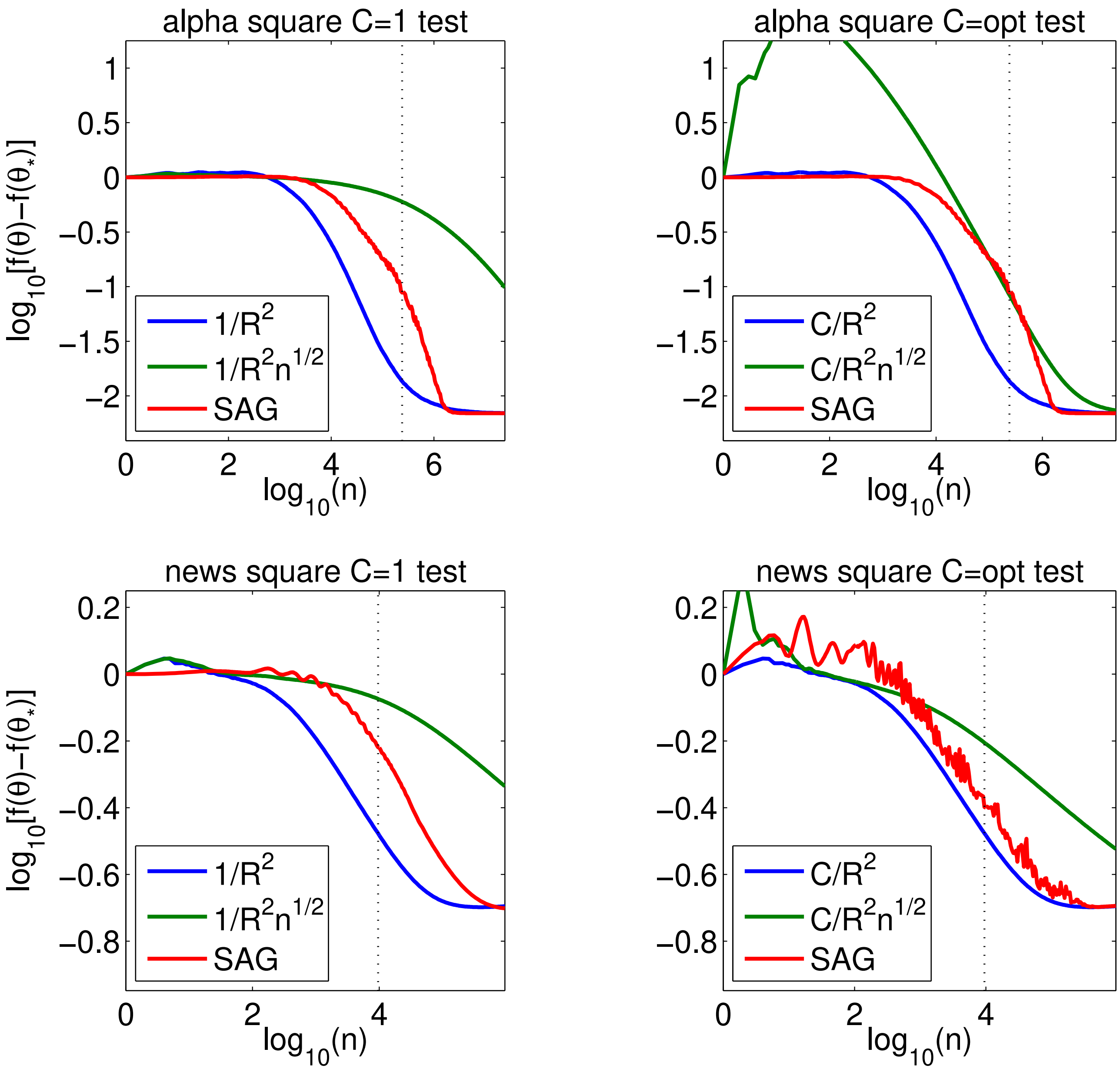
Simulations - synthetic examples

- Gaussian distributions - $p = 20$



Simulations - benchmarks

- *alpha* ($p = 500$, $n = 500\ 000$), *news* ($p = 1\ 300\ 000$, $n = 20\ 000$)



Robust averaged stochastic gradient (Bach and Moulines, 2013)

- **Constant-step-size SGD is convergent for least-squares**
  - Convergence rate in $O(1/n)$ without any dependence on μ
  - Simple choice of step-size (equal to $1/L$)

- **Constant-step-size SGD can be made convergent**
  - Online Newton correction with same complexity as SGD
  - Replace $\theta_n = \theta_{n-1} - \gamma f_n'(\theta_{n-1})$
    by $\theta_n = \theta_{n-1} - \gamma\big[f_n'(\bar\theta_{n-1}) + f''(\bar\theta_{n-1})(\theta_{n-1} - \bar\theta_{n-1})\big]$
  - Simple choice of step-size and convergence rate in $O(1/n)$

Robust averaged stochastic gradient (Bach and Moulines, 2013)

- **Constant-step-size SGD is convergent for least-squares**
  - Convergence rate in $O(1/n)$ without any dependence on μ
  - Simple choice of step-size (equal to $1/L$)

- **Constant-step-size SGD can be made convergent**
  - Online Newton correction with same complexity as SGD
  - Replace $\theta_n = \theta_{n-1} - \gamma f_n'(\theta_{n-1})$
    by $\theta_n = \theta_{n-1} - \gamma\big[f_n'(\bar\theta_{n-1}) + f''(\bar\theta_{n-1})(\theta_{n-1} - \bar\theta_{n-1})\big]$
  - Simple choice of step-size and convergence rate in $O(1/n)$

- **Multiple passes still work better in practice**
  - See Pillaud-Vivien, Rudi, and Bach (2018)

Perspectives

- **Linearly-convergent stochastic gradient methods**
  - Provable and precise rates
  - Improves on two known lower-bounds (by using structure)
  - Several extensions / interpretations / accelerations

Perspectives

- **Linearly-convergent stochastic gradient methods**
  - Provable and precise rates
  - Improves on two known lower-bounds (by using structure)
  - Several extensions / interpretations / accelerations

- **Extensions and future work**
  - Lower bounds for finite sums (Lan, 2015)
  - Sampling without replacement (Gurbuzbalaban et al., 2015)

Perspectives

- **Linearly-convergent stochastic gradient methods**
  - Provable and precise rates
  - Improves on two known lower-bounds (by using structure)
  - Several extensions / interpretations / accelerations

- **Extensions and future work**
  - Lower bounds for finite sums (Lan, 2015)
  - Sampling without replacement (Gurbuzbalaban et al., 2015)
  - Bounds on testing errors for incremental methods

Perspectives

- **Linearly-convergent stochastic gradient methods**
  - Provable and precise rates
  - Improves on two known lower-bounds (by using structure)
  - Several extensions / interpretations / accelerations

- **Extensions and future work**
  - Lower bounds for finite sums (Lan, 2015)
  - Sampling without replacement (Gurbuzbalaban et al., 2015)
  - Bounds on testing errors for incremental methods
  - Parallelization (Leblond, Pedregosa, and Lacoste-Julien, 2016; Hendrikx, Bach, and Massoulié, 2019)
  - Non-convex problems (Reddi et al., 2016)
  - Other forms of acceleration (Scieur, d'Aspremont, and Bach, 2016)

Outline

1. **Introduction/motivation: Supervised machine learning**

   – Machine learning $\approx$ optimization of finite sums
   – Batch optimization methods

2. **Fast stochastic gradient methods for convex problems**

   – Variance reduction: for *training* error
   – Constant step-sizes: for *testing* error

2. **Beyond convex problems**

   – Generic algorithms with generic "guarantees"
   – Global convergence for over-parameterized neural networks

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

- **Actual goal**: minimize test error $\mathbb{E}_{p(x,y)} \ell(y, h(x, \theta))$

Convex optimization problems

- **Convexity in machine learning**
  - Convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$

Convex optimization problems

- **Convexity in machine learning**
  - Convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$

- **(approximately) matching theory and practice**
  - Fruitful discussions between theoreticians and practitioners
  - Quantitative theoretical analysis suggests practical improvements

Convex optimization problems

- **Convexity in machine learning**
  - Convex loss and linear predictions $h(x, \theta) = \theta^\top \Phi(x)$
- **(approximately) matching theory and practice**
  - Fruitful discussions between theoreticians and practitioners
  - Quantitative theoretical analysis suggests practical improvements
- **Golden years of convexity in machine learning** (1995 to 201*)
  - Support vector machines and kernel methods
  - Inference in graphical models
  - Sparsity / low-rank models (statistics + optimization)
  - Convex relaxation of unsupervised learning problems
  - Optimal transport
  - Stochastic methods for large-scale learning and online learning

Convex optimization problems

- **Convexity in machine learning**
  - Convex loss and linear predictions $h(x,\theta) = \theta^\top \Phi(x)$
- **(approximately) matching theory and practice**
  - Fruitful discussions between theoreticians and practitioners
  - Quantitative theoretical analysis suggests practical improvements
- **Golden years of convexity in machine learning** (1995 to 201*)
  - Support vector machines and kernel methods
  - Inference in graphical models
  - Sparsity / low-rank models (statistics + optimization)
  - Convex relaxation of unsupervised learning problems
  - Optimal transport
  - Stochastic methods for large-scale learning and online learning

Exponentially convergent SGD for smooth finite sums

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

Exponentially convergent SGD for smooth finite sums

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

- **Non-accelerated algorithms** (with similar properties)
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SDCA (Shalev-Shwartz and Zhang, 2013)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - MISO (Mairal, 2015), Finito (Defazio et al., 2014b)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014a), etc...

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

Exponentially convergent SGD for smooth finite sums

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

- **Non-accelerated algorithms** (with similar properties)
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SDCA (Shalev-Shwartz and Zhang, 2013)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - MISO (Mairal, 2015), Finito (Defazio et al., 2014b)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014a), etc...

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

Exponentially convergent SGD for smooth finite sums

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

- **Non-accelerated algorithms** (with similar properties)
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SDCA (Shalev-Shwartz and Zhang, 2013)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - MISO (Mairal, 2015), Finito (Defazio et al., 2014b)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014a), etc...

- **Accelerated algorithms**
  - Shalev-Shwartz and Zhang (2014); Nitanda (2014)
  - Lin et al. (2015b); Defazio (2016), etc...
  - Catalyst (Lin, Mairal, and Harchaoui, 2015a)

Exponentially convergent SGD for finite sums

- **Running-time to reach precision** ε (with κ = condition number)

Stochastic gradient descent	$d\times$	κ	$\times$	$\frac{1}{\varepsilon}$
Gradient descent	$d\times$	$n\kappa$	$\times$	$\log \frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times$	$n\sqrt{\kappa}$	$\times$	$\log \frac{1}{\varepsilon}$

Exponentially convergent SGD for finite sums

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

Stochastic gradient descent	$d \times \quad \kappa \quad \times \quad \frac{1}{\varepsilon}$
Gradient descent	$d \times \quad n\kappa \quad \times \log \frac{1}{\varepsilon}$
Accelerated gradient descent	$d \times \quad n\sqrt{\kappa} \quad \times \log \frac{1}{\varepsilon}$
SAG(A), SVRG, SDCA, MISO	$d \times \quad (n + \kappa) \quad \times \log \frac{1}{\varepsilon}$

Exponentially convergent SGD for finite sums

- **Running-time to reach precision** ε (with κ = condition number)

Stochastic gradient descent	$d\times \quad \kappa \quad \times \quad \frac{1}{\varepsilon}$
Gradient descent	$d\times \quad n\kappa \quad \times \log\frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times \quad n\sqrt{\kappa} \quad \times \log\frac{1}{\varepsilon}$
SAG(A), SVRG, SDCA, MISO	$d\times \quad (n+\kappa) \quad \times \log\frac{1}{\varepsilon}$
Accelerated versions	$d\times \quad (n+\sqrt{n\kappa}) \quad \times \log\frac{1}{\varepsilon}$

NB: slightly different (smaller) notion of condition number for batch methods

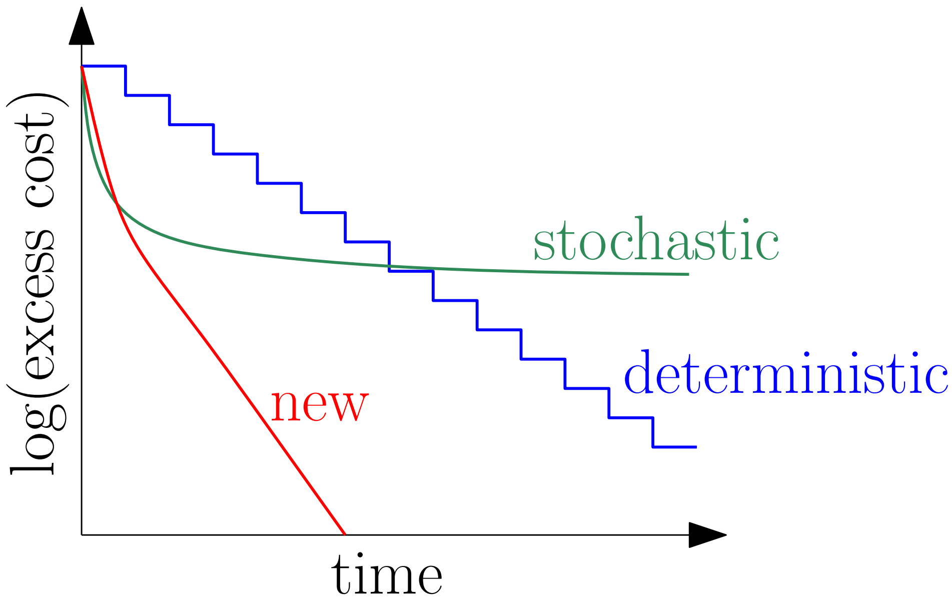
Exponentially convergent SGD for finite sums

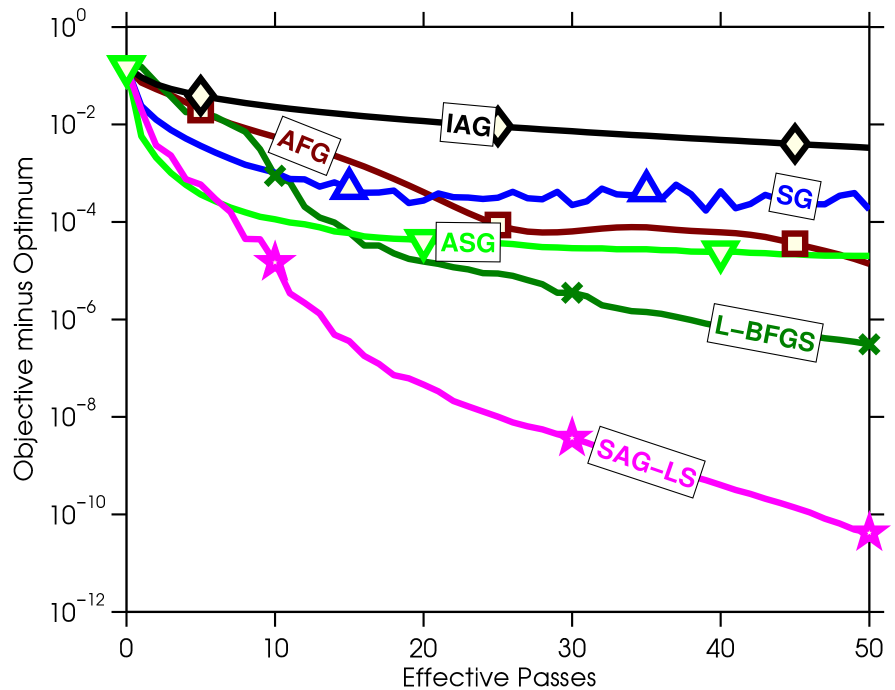
- **Running-time to reach precision** ε (with κ = condition number)

Stochastic gradient descent	$d\times \quad \kappa \quad \times \quad \frac{1}{\varepsilon}$
Gradient descent	$d\times \quad n\kappa \quad \times \log \frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times \quad n\sqrt{\kappa} \quad \times \log \frac{1}{\varepsilon}$
SAG(A), SVRG, SDCA, MISO	$d\times \quad (n+\kappa) \quad \times \log \frac{1}{\varepsilon}$
Accelerated versions	$d\times \quad (n+\sqrt{n\kappa}) \quad \times \log \frac{1}{\varepsilon}$

- **Matching lower bounds** (Woodworth and Srebro, 2016; Lan, 2015)

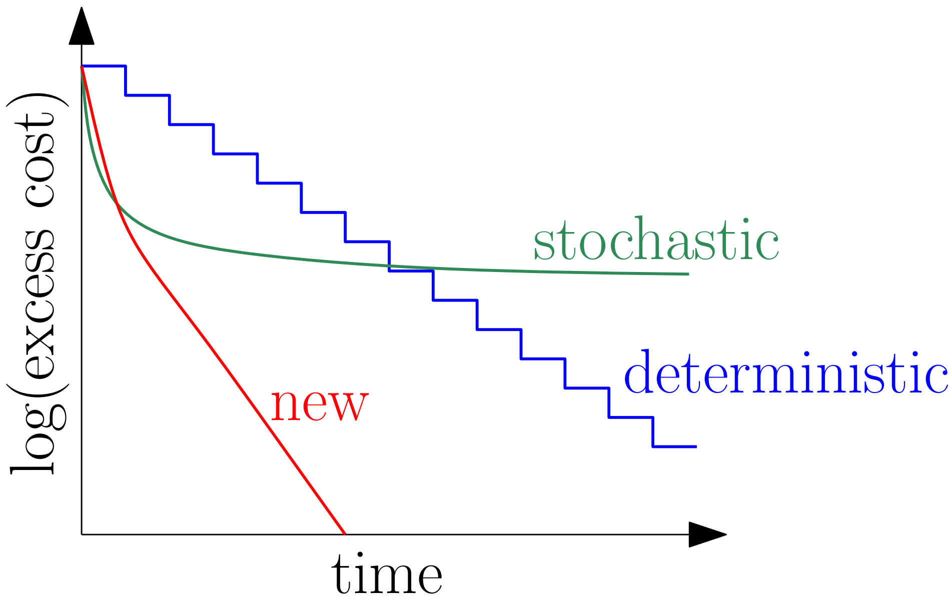
Exponentially convergent SGD for finite sums
From theory to practice and vice-versa

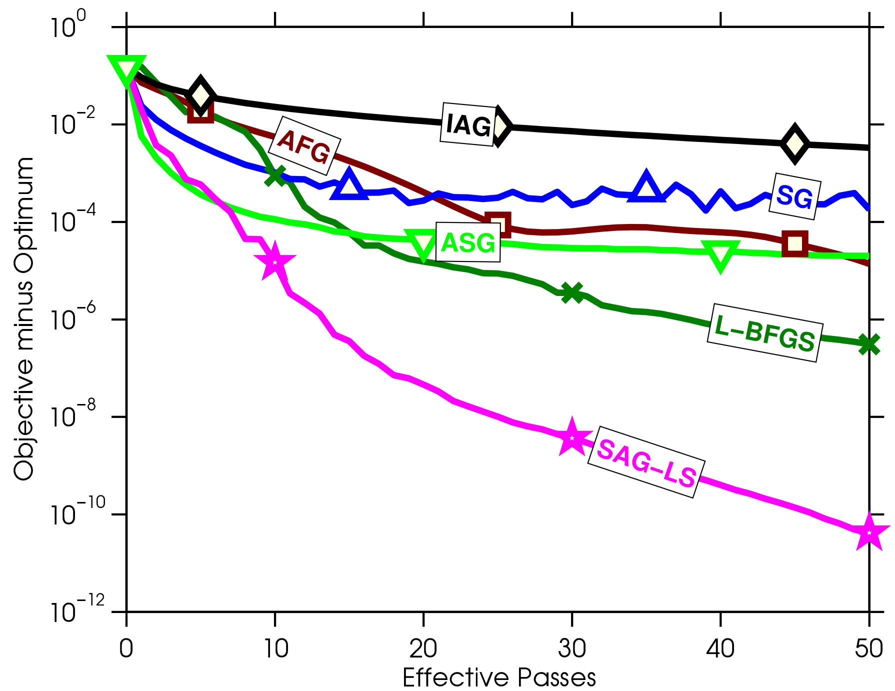




- **Empirical performance "matches" theoretical guarantees**

Exponentially convergent SGD for finite sums
From theory to practice and vice-versa





- **Empirical performance "matches" theoretical guarantees**
- **Theoretical analysis suggests practical improvements**
  - Non-uniform sampling, acceleration
  - Matching upper and lower bounds

Convex optimization for machine learning
From theory to practice and vice-versa

- Empirical performance "matches" theoretical guarantees

- Theoretical analysis suggests practical improvements

Convex optimization for machine learning
From theory to practice and vice-versa

- **Empirical performance "matches" theoretical guarantees**
- **Theoretical analysis suggests practical improvements**
- **Many other well-understood areas**
  - Single pass SGD and generalization errors
  - From least-squares to convex losses
  - High-dimensional inference
  - Non-parametric regression
  - Randomized linear algebra
  - Bandit problems
  - etc...

Convex optimization for machine learning
From theory to practice and vice-versa

- **Empirical performance “matches” theoretical guarantees**

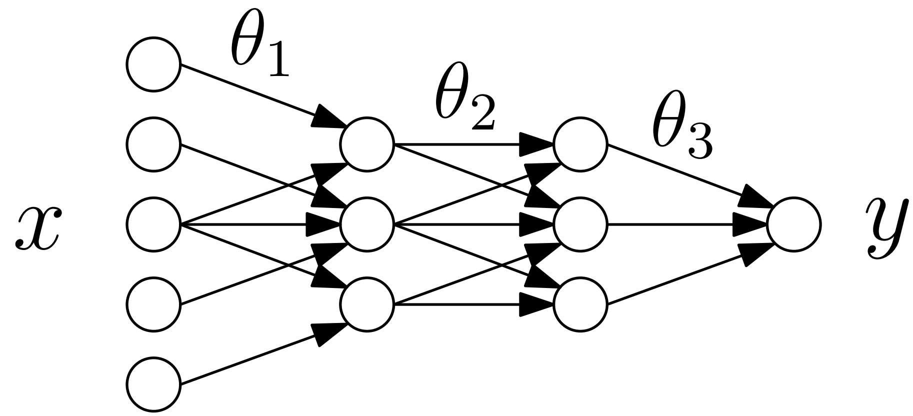
- **Theoretical analysis suggests practical improvements**

- **Many other well-understood areas**
  - Single pass SGD and generalization errors
  - From least-squares to convex losses
  - High-dimensional inference
  - Non-parametric regression
  - Randomized linear algebra
  - Bandit problems
  - etc...

- **What about deep learning?**

Theoretical analysis of deep learning

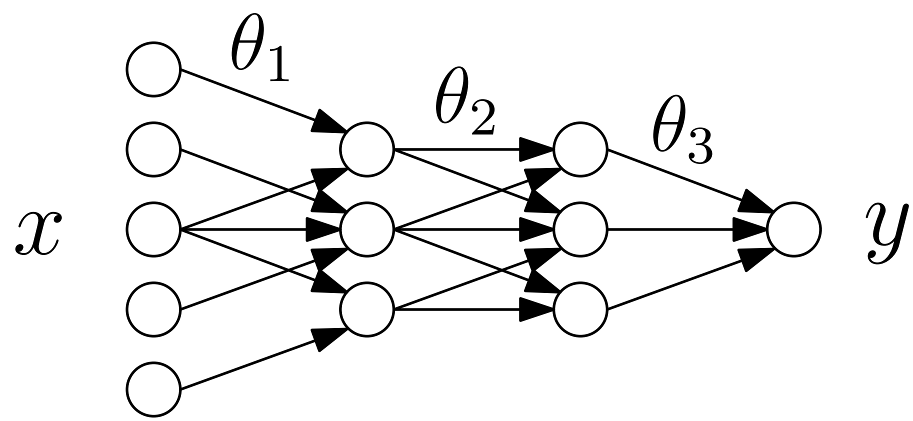
- **Multi-layer neural network** $h(x,\theta) = \theta_r^\top \sigma(\theta_{r-1}^\top \sigma(\cdots \theta_2^\top \sigma(\theta_1^\top x))$



  - NB: already a simplification

Theoretical analysis of deep learning

- **Multi-layer neural network** $h(x,\theta) = \theta_r^\top \sigma(\theta_{r-1}^\top \sigma(\cdots \theta_2^\top \sigma(\theta_1^\top x))$



  – NB: already a simplification

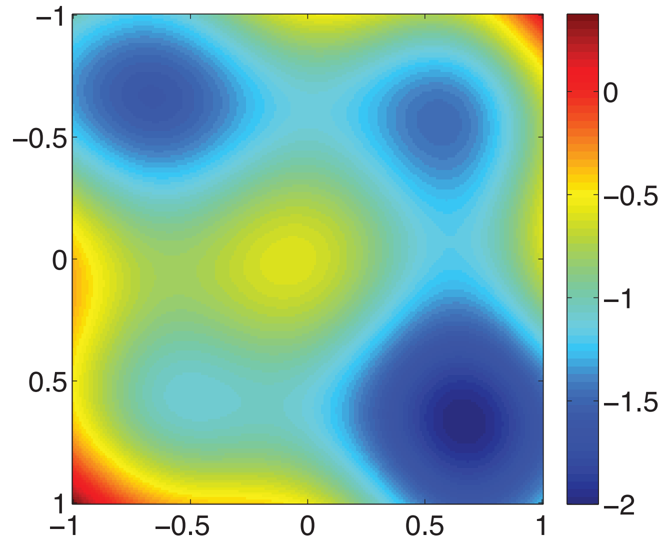
- **Main difficulties**

  1. Non-convex optimization problems

  2. Generalization guarantees in the overparameterized regime

Optimization for multi-layer neural networks

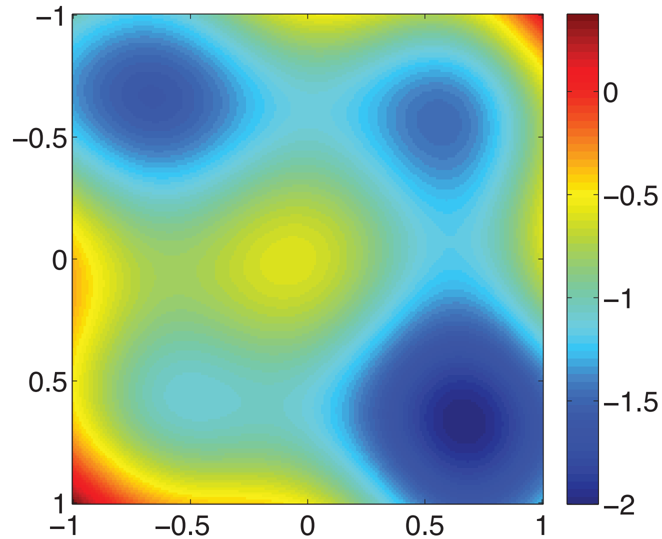
- **What can go wrong with non-convex optimization problems?**
  - Local minima
  - Stationary points
  - Plateaux
  - Bad initialization
  - etc...



Optimization for multi-layer neural networks

- **What can go wrong with non-convex optimization problems?**

  - Local minima
  - Stationary points
  - Plateaux
  - Bad initialization
  - etc...



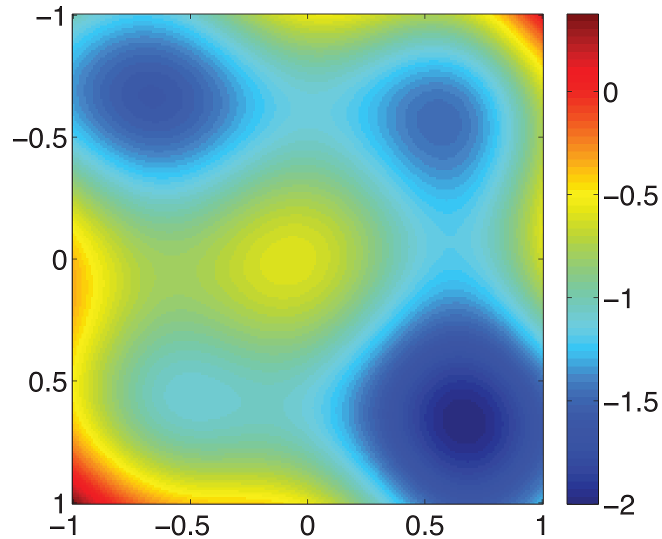
- **Generic local theoretical guarantees**

  - Convergence to stationary points or local minima
  - See, e.g., Lee et al. (2016); Jin et al. (2017)

Optimization for multi-layer neural networks

- **What can go wrong with non-convex optimization problems?**

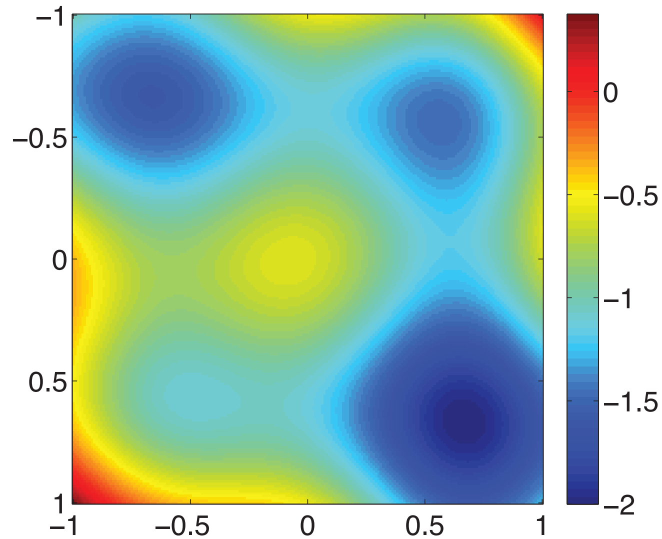
  - Local minima
  - Stationary points
  - Plateaux
  - Bad initialization
  - etc...



- **General global performance guarantees impossible to obtain**

Optimization for multi-layer neural networks

- **What can go wrong with non-convex optimization problems?**
  - Local minima
  - Stationary points
  - Plateaux
  - Bad initialization
  - etc...



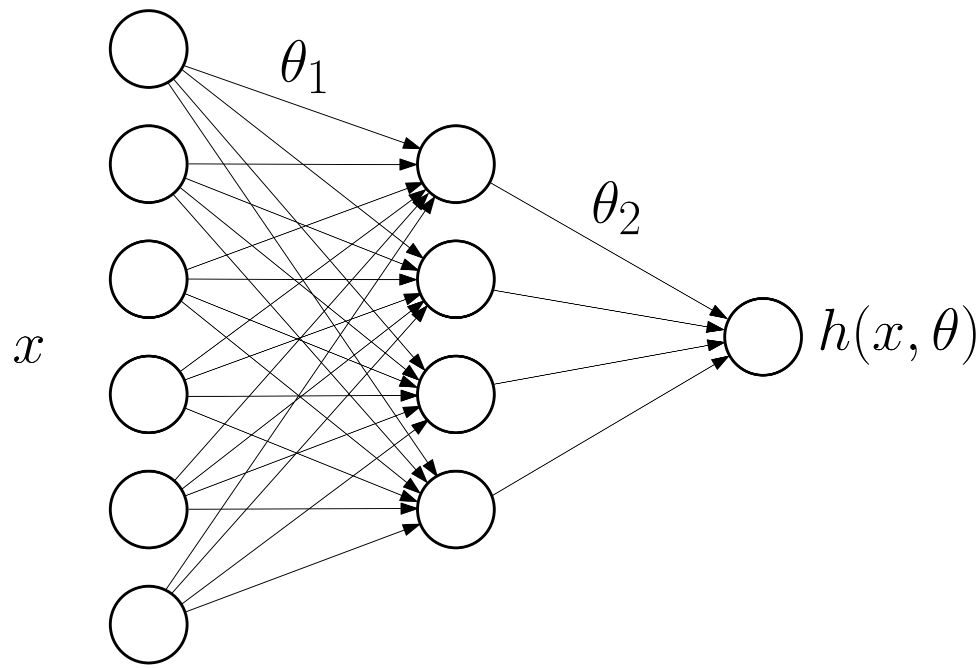
- **General global performance guarantees impossible to obtain**

- **Special case of (deep) neural networks**
  - Most local minima are equivalent (Choromanska et al., 2015)
  - No spurrious local minima (Soltanolkotabi et al., 2018)

Gradient descent for a single hidden layer

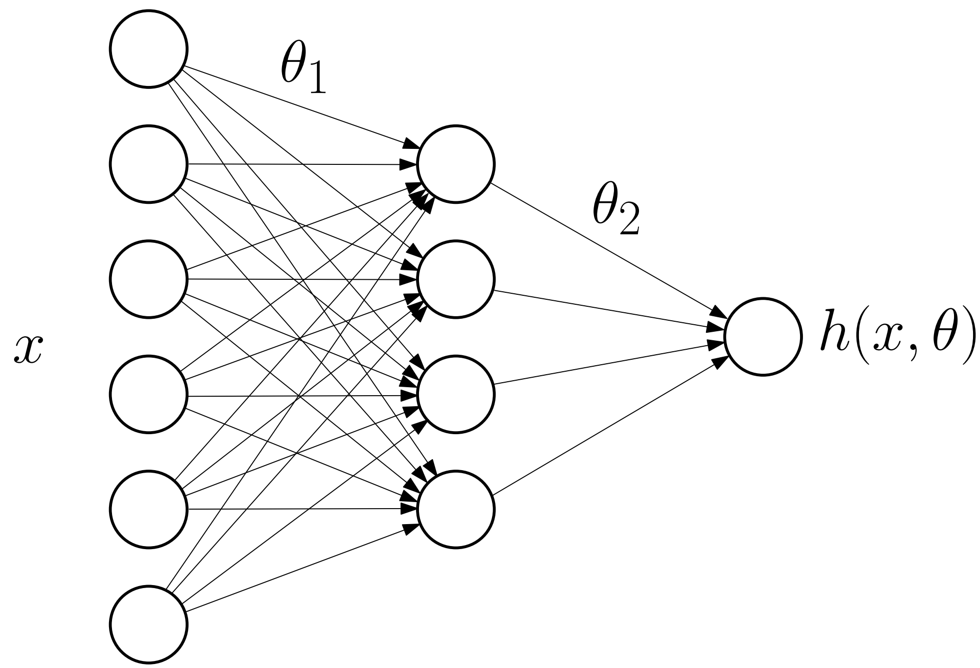
- **Predictor**: $h(x) = \frac{1}{m}\theta_2^\top \sigma(\theta_1^\top x) = \frac{1}{m}\sum_{j=1}^m \theta_2(j)\cdot\sigma\big[\theta_1(\cdot,j)^\top x\big]$

- **Goal**: minimize $R(h) = \mathbb{E}_{p(x,y)}\ell(y,h(x))$, with R convex



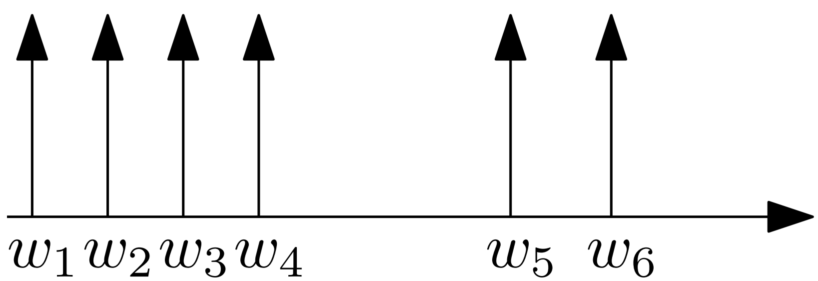
Gradient descent for a single hidden layer

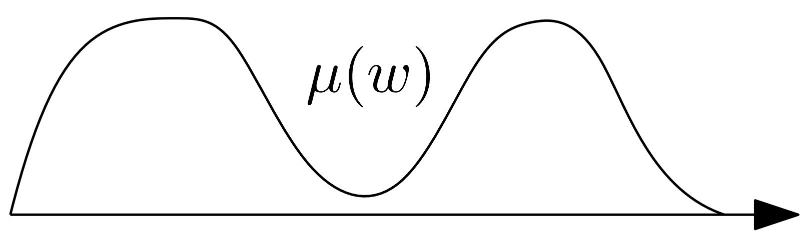
- **Predictor**: $h(x) = \frac{1}{m}\theta_2^\top \sigma(\theta_1^\top x) = \frac{1}{m}\sum_{j=1}^m \theta_2(j)\cdot\sigma\big[\theta_1(\cdot,j)^\top x\big]$
  - Family: $h = \dfrac{1}{m}\displaystyle\sum_{j=1}^m \Psi(w_j)$ with $\Psi(w_j)(x) = \theta_2(j)\cdot\sigma\big[\theta_1(\cdot,j)^\top x\big]$
- **Goal**: minimize $R(h) = \mathbb{E}_{p(x,y)}\ell(y,h(x))$, with R convex



Gradient descent for a single hidden layer

- **Predictor**: $h(x) = \frac{1}{m}\theta_2^\top \sigma(\theta_1^\top x) = \frac{1}{m}\sum_{j=1}^m \theta_2(j) \cdot \sigma\big[\theta_1(\cdot, j)^\top x\big]$
  - Family: $h = \dfrac{1}{m}\displaystyle\sum_{j=1}^m \Psi(w_j)$ with $\Psi(w_j)(x) = \theta_2(j) \cdot \sigma\big[\theta_1(\cdot, j)^\top x\big]$
- **Goal**: minimize $R(h) = \mathbb{E}_{p(x,y)}\ell(y, h(x))$, with R convex
- **Main insight**
  - $h = \dfrac{1}{m}\displaystyle\sum_{j=1}^m \Psi(w_j) = \int_{\mathcal{W}} \Psi(w) d\mu(w)$ with $d\mu(w) = \dfrac{1}{m}\displaystyle\sum_{j=1}^m \delta_{w_j}$





Gradient descent for a single hidden layer

- **Predictor**: $h(x) = \frac{1}{m}\theta_2^\top \sigma(\theta_1^\top x) = \frac{1}{m}\sum_{j=1}^m \theta_2(j)\cdot\sigma\big[\theta_1(\cdot,j)^\top x\big]$
  - Family: $h = \dfrac{1}{m}\displaystyle\sum_{j=1}^m \Psi(w_j)$ with $\Psi(w_j)(x) = \theta_2(j)\cdot\sigma\big[\theta_1(\cdot,j)^\top x\big]$
- **Goal**: minimize $R(h) = \mathbb{E}_{p(x,y)}\ell(y,h(x))$, with R convex
- **Main insight**
  - $h = \dfrac{1}{m}\displaystyle\sum_{j=1}^m \Psi(w_j) = \int_{\mathcal{W}} \Psi(w)d\mu(w)$ with $d\mu(w) = \dfrac{1}{m}\displaystyle\sum_{j=1}^m \delta_{w_j}$
  - Overparameterized models with m large $\approx$ measure μ with densities
  - Barron (1993); Kurkova and Sanguineti (2001); Bengio et al. (2006); Rosset et al. (2007); Bach (2017)

Optimization on measures

- **Minimize with respect to measure** μ: $R\Big(\int_{\mathcal{W}} \Psi(w) d\mu(w)\Big)$
  - Convex optimization problem on measures
  - Frank-Wolfe techniques for incremental learning
  - Non-tractable (Bach, 2017), not what is used in practice

Optimization on measures

- **Minimize with respect to measure** μ: $R\Big(\int_{\mathcal{W}} \Psi(w) d\mu(w)\Big)$
  - Convex optimization problem on measures
  - Frank-Wolfe techniques for incremental learning
  - Non-tractable (Bach, 2017), not what is used in practice

- **Represent** μ **by a finite set of "particles"** $\mu = \frac{1}{m}\sum_{j=1}^{m} \delta_{w_j}$
  - Backpropagation $=$ gradient descent on $(w_1, \dots, w_m)$

- **Three questions**:
  - Algorithm limit when number of particles m gets large
  - Global convergence to a global minimizer
  - Prediction performance (see Chizat and Bach, 2020)

Many particle limit and global convergence (Chizat and Bach, 2018a)

- **General framework**: minimize $F(\mu) = R\Big(\int_{\mathcal{W}} \Psi(w) d\mu(w) \Big)$
  - Algorithm: minimizing $F_m(w_1, \dots, w_m) = R\Big(\frac{1}{m} \sum_{j=1}^{m} \Psi(w_j) \Big)$

Many particle limit and global convergence (Chizat and Bach, 2018a)

- **General framework**: minimize $F(\mu) = R\Big(\int_{\mathcal{W}} \Psi(w) d\mu(w)\Big)$
  - Algorithm: minimizing $F_m(w_1, \dots, w_m) = R\Big(\frac{1}{m}\sum_{j=1}^{m} \Psi(w_j)\Big)$
  - Gradient flow $\dot{W} = -m\nabla F_m(W)$, with $W = (w_1, \dots, w_m)$
  - Idealization of (stochastic) gradient descent

Many particle limit and global convergence (Chizat and Bach, 2018a)

- **General framework**: minimize $F(\mu) = R\Big(\int_{\mathcal{W}} \Psi(w) d\mu(w) \Big)$
  - Algorithm: minimizing $F_m(w_1, \dots, w_m) = R\Big(\frac{1}{m} \sum_{j=1}^{m} \Psi(w_j) \Big)$
  - Gradient flow $\dot{W} = -m \nabla F_m(W)$, with $W = (w_1, \dots, w_m)$
  - Idealization of (stochastic) gradient descent
    1. Single pass SGD on the unobserved expected risk
    2. Multiple pass SGD or full GD on the empirical risk

Many particle limit and global convergence (Chizat and Bach, 2018a)

- **General framework**: minimize $F(\mu) = R\Big(\int_{\mathcal{W}} \Psi(w) d\mu(w)\Big)$
  - Algorithm: minimizing $F_m(w_1, \dots, w_m) = R\Big(\frac{1}{m}\sum_{j=1}^{m} \Psi(w_j)\Big)$
  - Gradient flow $\dot{W} = -m\nabla F_m(W)$, with $W = (w_1, \dots, w_m)$
  - Idealization of (stochastic) gradient descent

- **Limit when m tends to infinity**
  - Wasserstein gradient flow (Nitanda and Suzuki, 2017; Chizat and Bach, 2018a; Mei, Montanari, and Nguyen, 2018; Sirignano and Spiliopoulos, 2018; Rotskoff and Vanden-Eijnden, 2018)

- NB: for more details on gradient flows, see Ambrosio et al. (2008)

Many particle limit and global convergence (Chizat and Bach, 2018a)

- **(informal) theorem**: when the number of particles tends to infinity, the gradient flow converges to the global optimum

Many particle limit and global convergence (Chizat and Bach, 2018a)

- **(informal) theorem**: when the number of particles tends to infinity, the gradient flow converges to the global optimum
  - See precise definitions and statement in paper
  - Two key ingredients: homogeneity and initialization

Many particle limit and global convergence (Chizat and Bach, 2018a)

- **(informal) theorem**: when the number of particles tends to infinity, the gradient flow converges to the global optimum
  - See precise definitions and statement in paper
  - Two key ingredients: homogeneity and initialization

- **Homogeneity** (see, e.g., Haeffele and Vidal, 2017; Bach et al., 2008)
  - Full or partial, e.g., $\Psi(w_j)(x) = m\theta_2(j) \cdot \sigma\big[\theta_1(\cdot, j)^\top x\big]$
  - Applies to rectified linear units (but also to sigmoid activations)

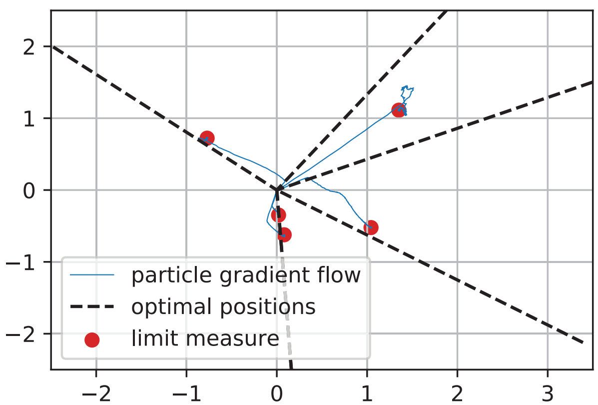
- **Sufficiently spread initial measure**
  - Needs to cover the entire sphere of directions

Many particle limit and global convergence (Chizat and Bach, 2018a)

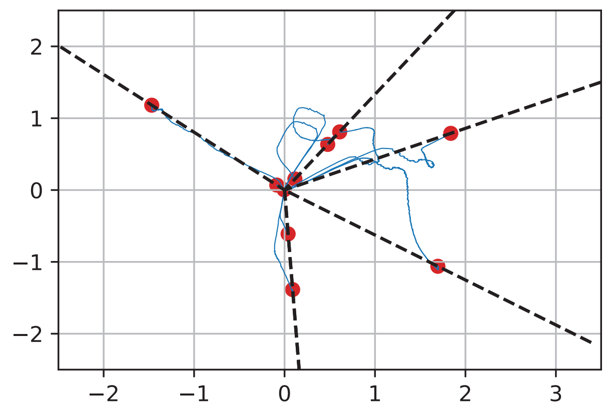
- **(informal) theorem**: when the number of particles tends to infinity, the gradient flow converges to the global optimum
  - See precise definitions and statement in paper
  - Two key ingredients: homogeneity and initialization
- **Homogeneity** (see, e.g., Haeffele and Vidal, 2017; Bach et al., 2008)
  - Full or partial, e.g., $\Psi(w_j)(x) = m\theta_2(j) \cdot \sigma\big[\theta_1(\cdot, j)^\top x\big]$
  - Applies to rectified linear units (but also to sigmoid activations)
- **Sufficiently spread initial measure**
  - Needs to cover the entire sphere of directions
- **Only qualititative!**

Simple simulations with neural networks

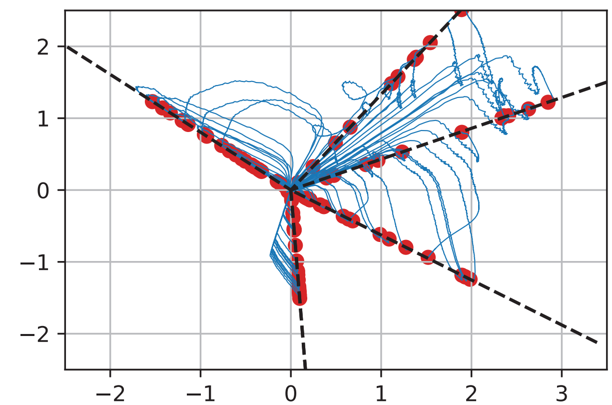
- ReLU units with $d = 2$ (optimal predictor has 5 neurons)



5 neurons



10 neurons



100 neurons

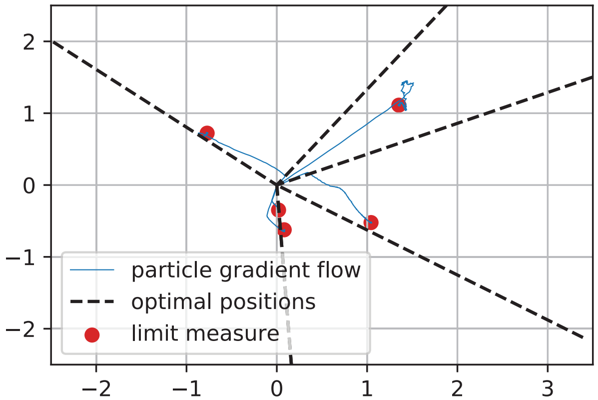
$$h(x) = \frac{1}{m}\sum_{j=1}^{m} \Psi(w_j)(x) = \frac{1}{m}\sum_{j=1}^{m} \theta_2(j)\big(\theta_1(\cdot,j)^\top x\big)_+$$

(plotting $|\theta_2(j)|\theta_1(\cdot,j)$ for each hidden neuron j)

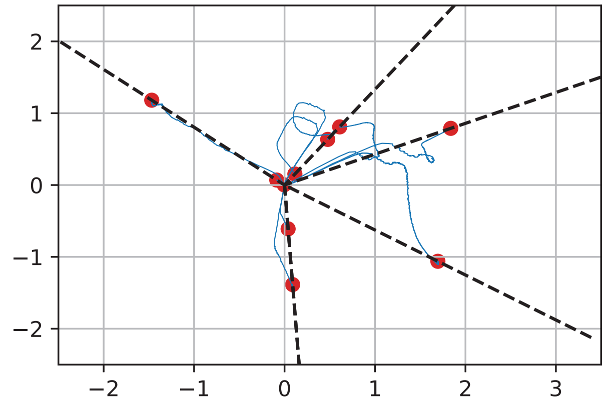
NB : also applies to spike deconvolution

Simple simulations with neural networks

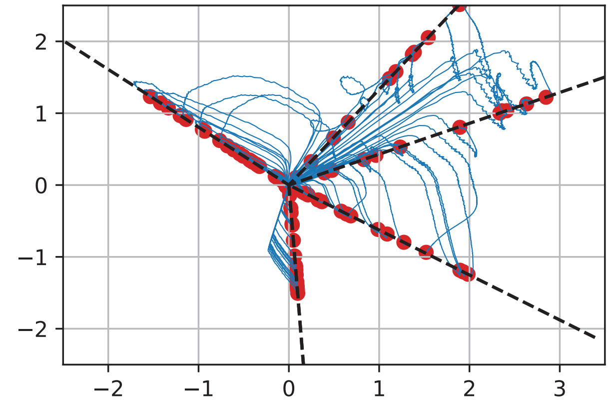
- ReLU units with $d = 2$ (optimal predictor has 5 neurons)



5 neurons



10 neurons



100 neurons

video!

NB : also applies to spike deconvolution

From qualitative to quantitative results ?

- **Adding noise** (Mei, Montanari, and Nguyen, 2018)
  - On top of SGD “à la Langevin” $\Rightarrow$ convergence to a diffusion
  - Quantitative analysis of the needed number of neurons
  - Recent improvement (Mei, Misiakiewicz, and Montanari, 2019)
  - Only applies to a single hidden layer

From qualitative to quantitative results ?

- **Adding noise** (Mei, Montanari, and Nguyen, 2018)
  - On top of SGD "à la Langevin" $\Rightarrow$ convergence to a diffusion
  - Quantitative analysis of the needed number of neurons
  - Recent improvement (Mei, Misiakiewicz, and Montanari, 2019)
  - Only applies to a single hidden layer

- **Recent bursty activity on ArXiv**
  - `https://arxiv.org/abs/1810.02054`
  - `https://arxiv.org/abs/1811.03804`
  - `https://arxiv.org/abs/1811.03962`
  - `https://arxiv.org/abs/1811.04918`
  - etc.

From qualitative to quantitative results ?

- **Adding noise** (Mei, Montanari, and Nguyen, 2018)
  - On top of SGD "à la Langevin" $\Rightarrow$ convergence to a diffusion
  - Quantitative analysis of the needed number of neurons
  - Recent improvement (Mei, Misiakiewicz, and Montanari, 2019)
  - Only applies to a single hidden layer

- **Recent bursty activity on ArXiv**
  - Global quantitative linear convergence of gradient descent (Du et al., 2018)
  - Extends to deep architectures and skip connections (Du et al., 2019; Allen-Zhu et al., 2018)

From qualitative to quantitative results ?

- **Adding noise** (Mei, Montanari, and Nguyen, 2018)
  - On top of SGD "à la Langevin" $\Rightarrow$ convergence to a diffusion
  - Quantitative analysis of the needed number of neurons
  - Recent improvement (Mei, Misiakiewicz, and Montanari, 2019)
  - Only applies to a single hidden layer

- **Recent bursty activity on ArXiv**
  - Global quantitative linear convergence of gradient descent (Du et al., 2018)
  - Extends to deep architectures and skip connections (Du et al., 2019; Allen-Zhu et al., 2018)

- **Any link?**

From qualitative to quantitative results ?

- **Mean-field limit**: $h(W) = \frac{1}{m}\sum_{i=1}^{m} \Psi(w_i)$
  - With w_i initialized randomly (with variance independent of m)
  - Dynamics equivalent to Wasserstein gradient flow
  - Convergence to global minimum of $R(\int \Psi d\mu)$

From qualitative to quantitative results ?

- **Mean-field limit**: $h(W) = \frac{1}{m} \sum_{i=1}^{m} \Psi(w_i)$
  - With w_i initialized randomly (with variance independent of m)
  - Dynamics equivalent to Wasserstein gradient flow
  - Convergence to global minimum of $R(\int \Psi d\mu)$

- **Recent bursty activity on ArXiv**
  - Corresponds to initializing with weights which are $\sqrt{m}$ times larger
  - Where does it converge to?

From qualitative to quantitative results ?

- **Mean-field limit**: $h(W) = \frac{1}{m}\sum_{i=1}^{m} \Psi(w_i)$
  - With w_i initialized randomly (with variance independent of m)
  - Dynamics equivalent to Wasserstein gradient flow
  - Convergence to global minimum of $R(\int \Psi d\mu)$
- **Recent bursty activity on ArXiv**
  - Corresponds to initializing with weights which are $\sqrt{m}$ times larger
  - Where does it converge to?
- **Equivalence to "lazy" training** (Chizat and Bach, 2018b)
  - Convergence to a positive-definite kernel method
  - Neurons move infinitesimally
  - Extension of results from Jacot et al. (2018)

Lazy training (Chizat and Bach, 2018b)

- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^{m} \Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)

Lazy training (Chizat and Bach, 2018b)

- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^{m} \Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)

- **Consequence**: around $W(0)$, $G_\alpha(W)$ has
  - Gradient "proportional" to $\nabla R(\alpha h(W(0)))/\alpha$
  - Hessian "proportional" to $\nabla^2 R(\alpha h(W(0)))$

Lazy training (Chizat and Bach, 2018b)

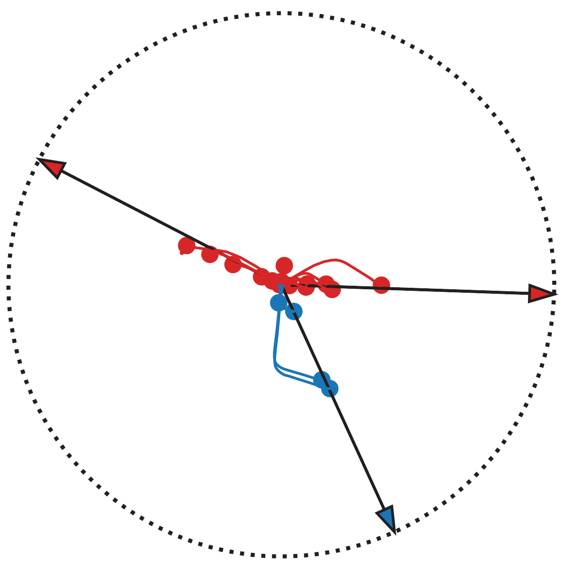
- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^{m}\Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)

- **Proposition** (informal, see paper for precise statement)
  - Assume differential of h at $W(0)$ is surjective
  - Gradient flow $\dot{W} = -\nabla G_\alpha(W)$ is such that

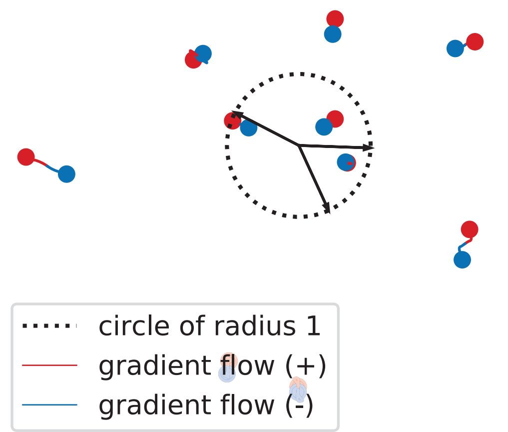
$$\|W(t) - W(0)\| = O(1/\alpha) \text{ and } \alpha h(W(t)) \to \arg\min_h R(h) \text{ “linearly”}$$

Lazy training (Chizat and Bach, 2018b)

- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^{m} \Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)

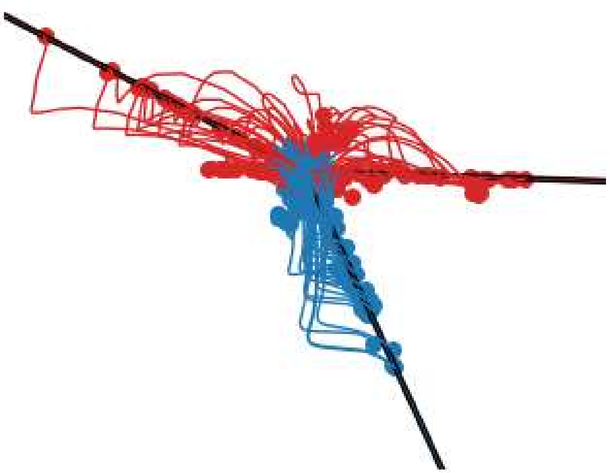


Lazy
$\Rightarrow$

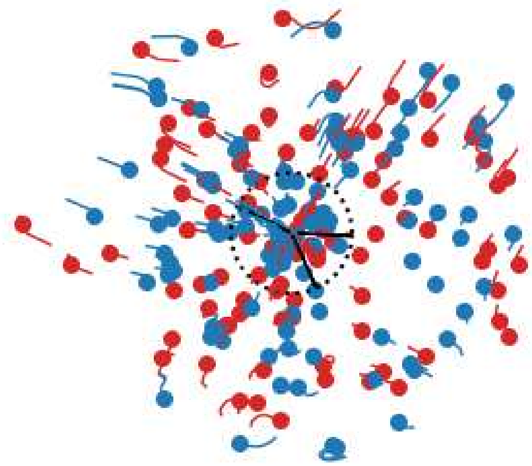


Lazy training (Chizat and Bach, 2018b)

- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^m \Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)

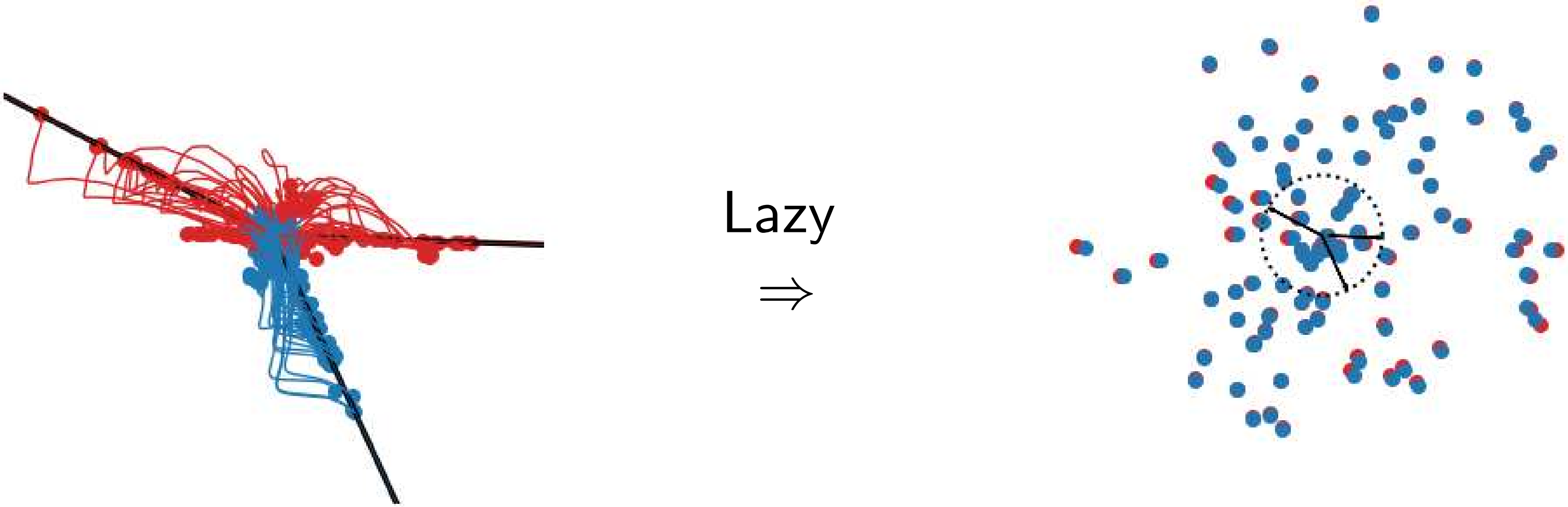


Lazy
$\Rightarrow$



Lazy training (Chizat and Bach, 2018b)

- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^m \Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)



Lazy training (Chizat and Bach, 2018b)

- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^{m}\Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)

- **Proposition** (informal, see paper for precise statement)
  - Assume differential of h at $W(0)$ is surjective
  - Gradient flow $\dot{W} = -\nabla G_\alpha(W)$ is such that

    $\|W(t)-W(0)\| = O(1/\alpha)$ and $\alpha h(W(t)) \to \arg\min_h R(h)$ "linearly"

Lazy training (Chizat and Bach, 2018b)

- **Generic criterion** $G(W) = R(h(W))$ **to minimize w.r.t.** W
  - Example: R loss, $h(W) = \frac{1}{m}\sum_{i=1}^{m} \Psi(w_i)$ prediction function
  - Introduce (large) scale factor $\alpha > 0$ and $G_\alpha(W) = R(\alpha h(W))/\alpha^2$
  - Initialize $W(0)$ such that $\alpha h(W(0))$ is bounded
    (using, e.g., $\mathbb{E}\Psi(w_i) = 0$)

- **Proposition** (informal, see paper for precise statement)
  - Assume differential of h at $W(0)$ is surjective
  - Gradient flow $\dot{W} = -\nabla G_\alpha(W)$ is such that

    $\|W(t) - W(0)\| = O(1/\alpha)$ and $\alpha h(W(t)) \to \arg\min_h R(h)$ "linearly"

$\Rightarrow$ Equivalent to a linear model
$h(W) \approx h(W(0)) + (W - W(0))^\top \nabla h(W(0))$

From lazy training to neural tangent kernel

- **Neural tangent kernel** (Jacot et al., 2018; Lee et al., 2019)
  - Linear model: $h(x, W) \approx h(x, W(0)) + (W - W(0))^\top \nabla h(x, W(0))$
  - Corresponding kernel $k(x, x') = \nabla h(x, W(0))^\top \nabla h(x', W(0))$
  - Direct effect of scaling of the model (Chizat and Bach, 2018b)
  - Applies to all differentiable models, including deep models

From lazy training to neural tangent kernel

- **Neural tangent kernel** (Jacot et al., 2018; Lee et al., 2019)
  - Linear model: $h(x, W) \approx h(x, W(0)) + (W - W(0))^\top \nabla h(x, W(0))$
  - Corresponding kernel $k(x, x') = \nabla h(x, W(0))^\top \nabla h(x', W(0))$
  - Direct effect of scaling of the model (Chizat and Bach, 2018b)
  - Applies to all differentiable models, including deep models

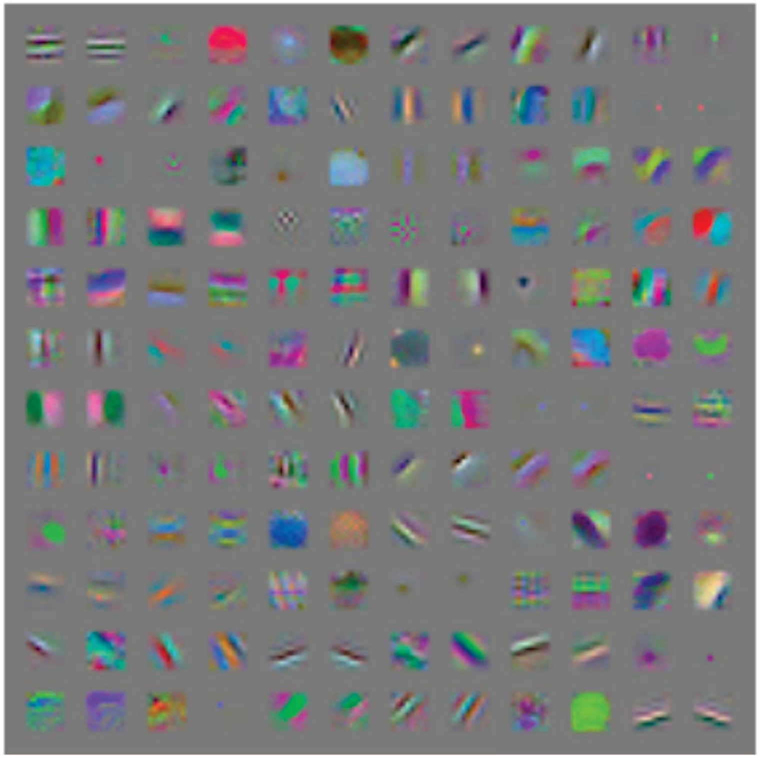
- **Two questions**:
  - Does this really "demystify" generalization in deep networks?
    (are state-of-the-art neural networks in the lazy regime?)
  - Can kernel methods beat neural networks?
    (is the neural tangent kernel useful in practice?)

Are state-of-the-art neural networks in the lazy regime?

- **Lazy regime**: Neurons move infinitesimally

Are state-of-the-art neural networks in the lazy regime?

- **Lazy regime**: Neurons move infinitesimally
- **Evidence 1**: the first layer of trained CNNs looks like Gabor filters



From Goodfellow, Bengio, and Courville (2016)

Are state-of-the-art neural networks in the lazy regime?

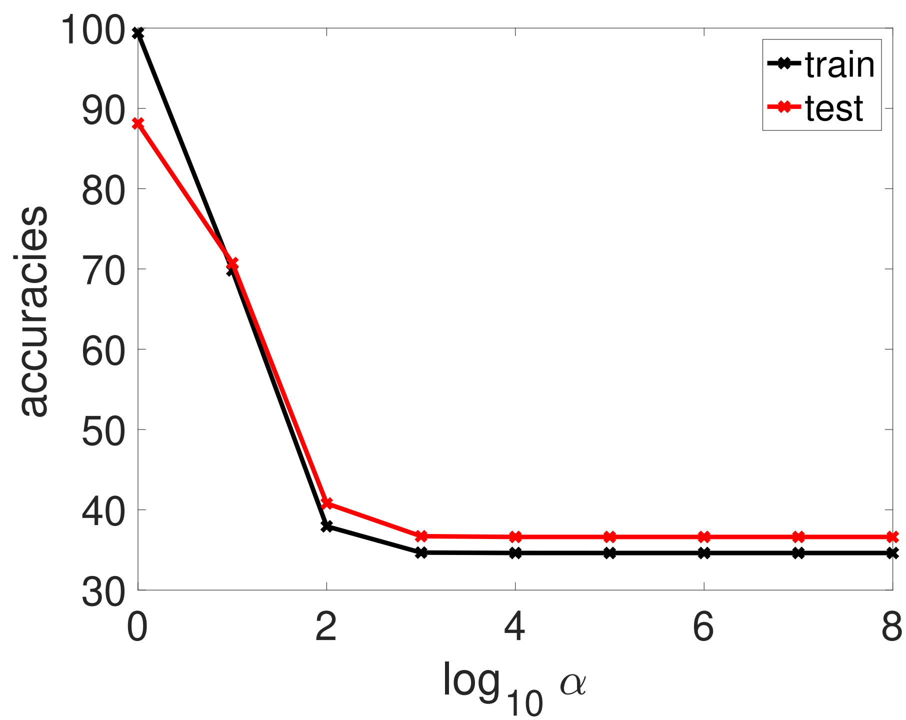
- **Lazy regime**: Neurons move infinitesimally
- **Evidence 1**: the first layer of trained CNNs looks like Gabor filters
- **Evidence 2**, by Zhang et al. (2019)
  - Neurons from first layers do move
  - Neurons of other layers do not need to move

Are state-of-the-art neural networks in the lazy regime?

- **Lazy regime**: Neurons move infinitesimally
- **Evidence 1**: the first layer of trained CNNs looks like Gabor filters
- **Evidence 2**, by Zhang et al. (2019)
  - Neurons from first layers do move
  - Neurons of other layers do not need to move
- **Evidence 3**: Take a state-of-the-art CNN and make it lazier
  - Chizat, Oyallon, and Bach (2019)

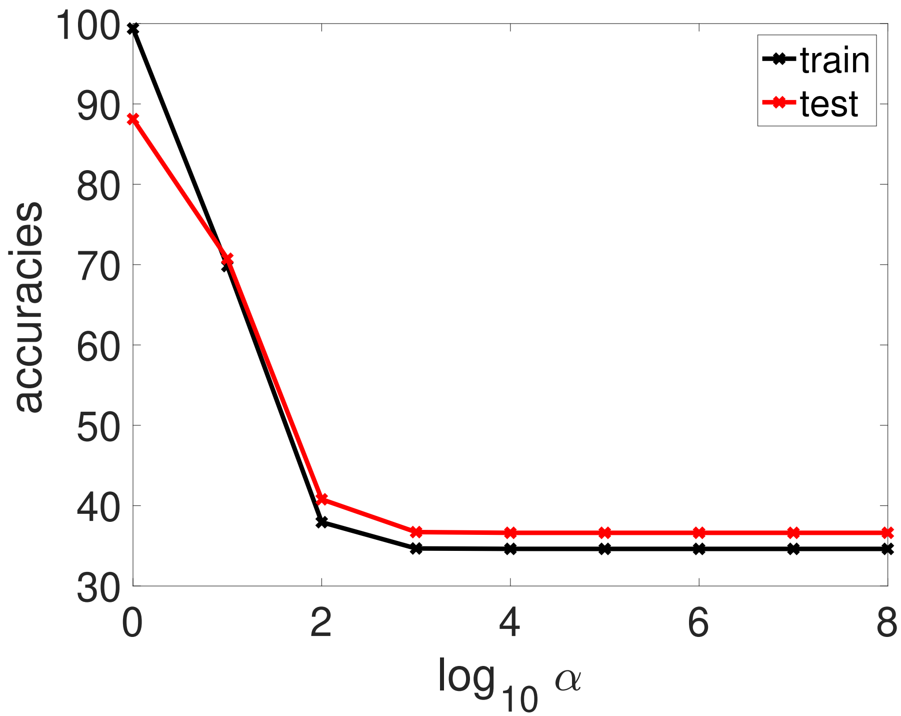
Lazy training seen in practice?

- **Convolutional neural network**
  - "VGG-11": 10 millions parameters
  - "CIFAR10" images: 60 000 32×32 color images and 10 classes
  - (almost) state-of-the-art accuracies



Lazy training seen in practice?

- **Convolutional neural network**
  - "VGG-11": 10 millions parameters
  - "CIFAR10" images: 60 000 32×32 color images and 10 classes
  - (almost) state-of-the-art accuracies



- **Understanding the mix of lazy and non-lazy regimes?**

Is the neural tangent kernel useful in practice?

- **Fully connected networks**
  - Gradient with respect to output weights: classical random features (Rahimi and Recht, 2007)
  - Gradient with respect to input weights: extra random features
  - Non-parametric estimation but no better than usual kernels (Ghorbani et al., 2019; Bietti and Mairal, 2019)

Is the neural tangent kernel useful in practice?

- **Fully connected networks**
  - Gradient with respect to output weights: classical random features (Rahimi and Recht, 2007)
  - Gradient with respect to input weights: extra random features
  - Non-parametric estimation but no better than usual kernels (Ghorbani et al., 2019; Bietti and Mairal, 2019)

- **Convolutional neural networks**
  - Theoretical and computational properties (Arora et al., 2019)
  - Good stability properties (Bietti and Mairal, 2019)
  - Can achieve state-of-the-art performance with additional tricks (Mairal, 2016; Novak et al., 2018) on the CIFAR10 dataset (Li et al., 2019)

Is the neural tangent kernel useful in practice?

- **Fully connected networks**
  - Gradient with respect to output weights: classical random features (Rahimi and Recht, 2007)
  - Gradient with respect to input weights: extra random features
  - Non-parametric estimation but no better than usual kernels (Ghorbani et al., 2019; Bietti and Mairal, 2019)
- **Convolutional neural networks**
  - Theoretical and computational properties (Arora et al., 2019)
  - Good stability properties (Bietti and Mairal, 2019)
  - Can achieve state-of-the-art performance with additional tricks (Mairal, 2016; Novak et al., 2018) on the CIFAR10 dataset (Li et al., 2019)
- **Going further without explicit representation learning?**

Healthy interactions between theory, applications, and hype?

Healthy interactions between theory, applications, and hype?

- **Empirical successes of deep learning cannot be ignored**

Healthy interactions between theory, applications, and hype?

- **Empirical successes of deep learning cannot be ignored**
- **Scientific standards should not be lowered**
  - Critics and limits of theoretical and empirical results
  - Rigor beyond mathematical guarantees

Conclusions
Optimization for machine learning

- **Well understood**
  - Convex case with a single machine
  - Matching lower and upper bounds for variants of SGD
  - Non-convex case: SGD for local risk minimization

Conclusions
Optimization for machine learning

- **Well understood**
  - Convex case with a single machine
  - Matching lower and upper bounds for variants of SGD
  - Non-convex case: SGD for local risk minimization

- **Not well understood**: many open problems
  - Step-size schedules and acceleration
  - Dealing with non-convexity
    (global minima vs. local minima and stationary points)
  - Distributed learning: multiple cores, GPUs, and cloud (see, e.g., Hendrikx, Bach, and Massoulié, 2019, and references therein)

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

F. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Adv. NIPS*, 2011.

F. Bach and E. Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In *Adv. NIPS*, 2013.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012a.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Structured sparsity through convex optimization, 2012b.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(1):629–681, 2017.

Francis Bach, Julien Mairal, and Jean Ponce. Convex sparse matrix factorizations. Technical Report 0812.1869, arXiv, 2008.

A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

Y. Bengio, N. Le Roux, P. Vincent, O. Delalleau, and P. Marcotte. Convex neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*, 2019.

D. Blatt, A. O. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2008.

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

Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in neural information processing systems*, pages 3036–3046, 2018a.

Lenaic Chizat and Francis Bach. A note on lazy training in supervised differentiable programming. Technical Report To appear, ArXiv, 2018b.

Lénaïc Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. *arXiv preprint arXiv:2002.04486*, 2020.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. 2019.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014a.

A. Defazio, J. Domke, and T. S. Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *Proc. ICML*, 2014b.

Aaron Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems*, pages 676–684, 2016.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1675–1685, 2019.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization. *Optimization Online*, July, 2010.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *arXiv preprint arXiv:1904.12191*, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

M. Gurbuzbalaban, A. Ozdaglar, and P. Parrilo. On the convergence rate of incremental aggregated gradient algorithms. Technical Report 1506.02081, arXiv, 2015.

Benjamin D. Haeffele and René Vidal. Global optimality in neural network training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7331–7339, 2017.

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

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8580–8589, 2018.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

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

V. Kurkova and M. Sanguineti. Bounds on rates of variable-basis and neural-network approximation. *IEEE Transactions on Information Theory*, 47(6):2659–2665, Sep 2001.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate {Frank-Wolfe} optimization for structural {SVMs}. In *Proceedings of The 30th International Conference on Machine Learning*, pages 53–61, 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.

R. Leblond, F. Pedregosa, and S. Lacoste-Julien. Asaga: Asynchronous parallel Saga. Technical Report 1606.04809, arXiv, 2016.

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey

Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on Learning Theory*, pages 1246–1257, 2016.

Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015a.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015b.

J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

Julien Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in neural information processing systems*, pages 1399–1407, 2016.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks. Technical Report 1804.06561, arXiv, 2018.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *arXiv preprint arXiv:1902.06015*, 2019.

A. Nedic and D. Bertsekas. Convergence rate of incremental subgradient algorithms. *Stochastic Optimization: Algorithms and Applications*, pages 263–304, 2000.

A. S. Nemirovski and D. B. Yudin. *Problem complexity and method efficiency in optimization.* Wiley & Sons, 1983.

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.

A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Atsushi Nitanda and Taiji Suzuki. Stochastic particle gradient descent for infinite ensembles. *arXiv preprint arXiv:1712.05438*, 2017.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. 2018.

Loucas Pillaud-Vivien, Alessandro Rudi, and Francis Bach. Statistical optimality of stochastic gradient descent on hard learning problems through multiple passes. Technical Report 1805.10074, arXiv, 2018.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20:1177–1184, 2007.

S. J. Reddi, A. Hefny, S. Sra, B. Póczós, and A. Smola. Stochastic variance reduction for nonconvex optimization. Technical Report 1603.06160, arXiv, 2016.

H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statistics*, 22:400–407, 1951.

S. Rosset, G. Swirszcz, N. Srebro, and J. Zhu. ℓ_1-regularization in infinite dimensional feature spaces. In *Proceedings of the Conference on Learning Theory (COLT)*, 2007.

Grant M. Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2001.

Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Advances In Neural Information Processing Systems*, pages 712–720, 2016.

S. Shalev-Shwartz. Sdca without duality, regularization, and individual convexity. Technical Report 1602.01582, arXiv, 2016.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *Proc. ICML*, 2014.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks. *arXiv preprint arXiv:1805.01053*, 2018.

M. V. Solodov. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, 11(1):23–35, 1998.

Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 2018.

P. Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

I. Tsochantaridis, Thomas Joachims, T., Y. Altun, and Y. Singer. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

Blake E. Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in neural information processing systems*, pages 3639–3647, 2016.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 9:2543–2596, 2010.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *arXiv preprint arXiv:1902.01996*, 2019.

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.