Large Scale Learning and Optimization

Aymeric DIEULEVEUT

CMAP, Polytechnique (X)

October 3 -12 , 2019

Autumn School on Machine Learning - Tbilisi

Slides on my web page: `www.cmap.polytechnique.fr/~aymeric.dieuleveut/`

Outline

Outline

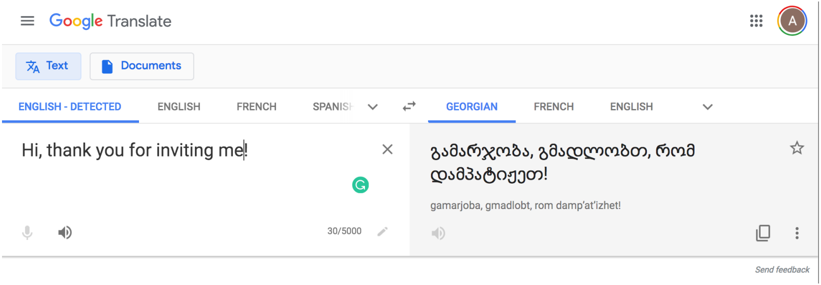
Large scale learning: multiple contexts and applications

What happened over the last 20 years?

1. **Increase in computational power**
2. **Data everywhere $\rightarrow$ learning from examples.**
2. **New algorithms, new models**

Large scale framework:
Data increase in number *n* and quality/dimension *d*.

New Applications: Translation



NLP tasks:

1. **Words representations, sentence representations, etc.**
2. **Automatic translation**
3. **Text generation, ...**

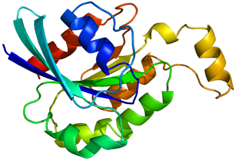
number n: billions of observations (wikipedia)
features dimension d: high dimensional representations of words

Advertisement

number n: billions of people

features dimension d: cookies, clicks

Bio-informatics



Bio-informatics

Input: DNA/RNA sequence,
Output: Drug responsiveness
number n: not always many patients
features dimension d: e.g., number of basis $\to 10^6$.

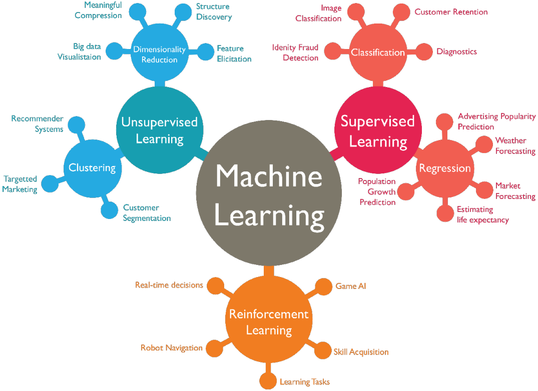
Image recognition

Image classification

Input: Images, Videos
Output: Digit , more complex category, action recognition...

number n: millions of images
features dimension d: millions of pixels, potentially thousand of frames in short video.

Large scale learning : Tons of applications, fewer algorithms & frameworks



- Sometimes combine supervised + unsupervised
- Many methods for each domain. For example for regression: Nearest neighbours, Linear regression, Kernel Regression, etc.
- Why is optimization about?

Image taken from https://wordstream-files-prod.s3.amazonaws.com/s3fs-public/machine-learning.png

Optimization is a key tool for large scale learning.

What is optimization about?

$$\min_{\theta \in \Theta} f(\theta)$$

With θ a parameter, and f a cost function.

Why?
We formulate our problem as a cost minimization problem.
A few examples:

- **Supervised machine learning**
- **Signal Processing**
- **Optimal transport**
- **GANS**

Optimization: some Examples 1/4

Example 1: Supervised Machine Learning

Consider an input/output pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, $(X, Y) \sim \rho$.

Goal: function $\theta : \mathcal{X} \to \mathbb{R}$, s.t. $\theta(X)$ good prediction for Y.

Here, as a linear function $\langle \theta, \Phi(X) \rangle$ of features $\Phi(X) \in \mathbb{R}^d$.

Consider a loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$

Define the Generalization risk :

$$\mathcal{R}(\theta) := \mathbb{E}_\rho \left[\ell(Y, \langle \theta, \Phi(X) \rangle) \right].$$

Empirical Risk minimization (I)

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

Empirical risk (or training error):

$$\hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle \theta, \Phi(x_i) \rangle).$$

Empirical risk minimization (ERM) : find $\hat{\theta}$ solution of

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, \langle \theta, \Phi(x_i) \rangle\big) \quad + \quad \mu \Omega(\theta).$$

convex data fitting term + regularizer

Empirical Risk minimization (II)

For example, least-squares regression:

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

and logistic regression:

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

Optimization: some Examples 2/4

Example 2: Signal processing

Observe a signal $\boldsymbol{Y} \in \mathbb{R}^{\boldsymbol{n}\times \boldsymbol{q}}$, try to recover the source $\boldsymbol{B} \in \mathbb{R}^{\boldsymbol{p}\times \boldsymbol{q}}$, knowing the "forward matrix" $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n}\times \boldsymbol{p}}$. (multi-task regression)

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{X}\beta - \boldsymbol{Y}\|_{\boldsymbol{F}}^2$$

Ω sparsity inducing regularization.

How to choose λ?

$\looparrowright$ non smooth optimization, optimization with sparsity inducing norms, etc.

Optimization: some Examples 3/4

Example 3: Optimal transport

$$\min_{\pi \in \Pi} \int c(x, y) \mathrm{d}\pi(x, y)$$

Π set of probability distributions $c(x, y)$ "distance" from x to y.

$+$ regularization

Kantorovic formulation of OT.

$\looparrowright$ alternating directions algorithms, ...

Optimization: some Examples 4/4

GANS

$$\min_G \max_D \{\mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z))]\}$$

- D discriminator: tries to discriminate between real and fake images
- G generator: tries to fool the discriminator.

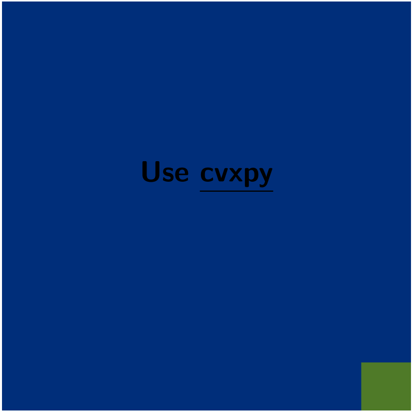
$\looparrowright$ minimax optimization, non convex optimization....

- Optimization is at the heart of most Learning methods.
- Is it difficult ?

Is it a (hard) problem?

for convex optimization, in 99 % of the cases, no.

In the words of Steven Boyd:

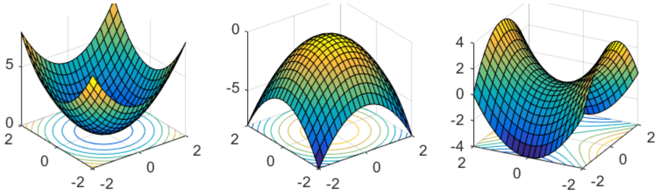


⇈

Interesting (or hard) problems

What makes it hard: 1. Convexity

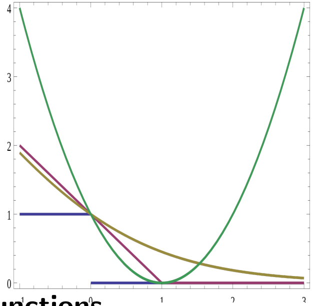
Why?



Typical non-convex problems:

Empirical risk minimization with 0-1 loss.

$$\hat{\mathcal{R}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{y_i \neq \mathrm{sign}\langle\theta,\Phi(x_i)\rangle}.$$



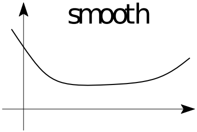
Neural networks: parametric non-convex functions.

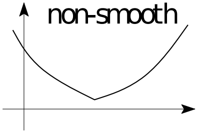
What makes it hard: 2. Regularity of the function

a. Smoothness

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

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





For all $\theta \in \mathbb{R}^d$:

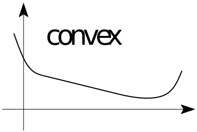
$$f(\theta) \leq f(\theta') + \langle f(\theta'), \theta - \theta' \rangle + \frac{L}{2} \left\| \theta - \theta' \right\|^2$$

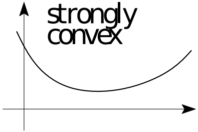
What makes it hard: 2. Regularity of the function

b. Strong Convexity

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

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





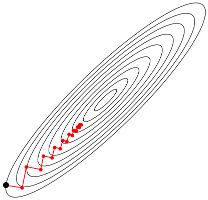
For all $\theta \in \mathbb{R}^d$:

$$f(\theta) \geq f(\theta') + \langle f(\theta'), \theta - \theta' \rangle + \frac{\mu}{2} \left\| \theta - \theta' \right\|^2$$

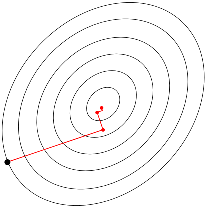
What makes it hard: 2. Regularity of the function

Why?

Rates typically depend on the condition number $\kappa = \frac{L}{\mu}$:



Large κ
harder to optimize



Small κ
easier to optimize

Smoothness and strong convexity in ML

We consider an a.s. convex loss in θ. Thus $\hat{\mathcal{R}}$ and $\mathcal{R}$ are convex.

Hessian of $\hat{\mathcal{R}} \approx$ covariance matrix $\frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^\top$

If ℓ is smooth, and $\mathbb{E}[\|\Phi(X)\|^2] \leq r^2$, $\mathcal{R}$ is smooth.

If ℓ is μ-strongly convex, and data has an invertible covariance matrix (low correlation/dimension), $\mathcal{R}$ is strongly convex.

Importance of regularization: provides strong convexity, and avoids overfitting.

Note: when considering dual formulation of the problem:

- L-smoothness $\leftrightarrow 1/L$-strong convexity.
- μ-strong convexity $\leftrightarrow 1/\mu$-smoothness

What makes it hard: 3. Set Θ, complexity of f

a. **Set Θ:** (if Θ is a convex set.)

- May be described implicitly (via equations):
  $\Theta = \{\theta \in \mathbb{R}^d \text{ s.t. } \|\theta\|_2 \leq R \text{ and } \langle \theta, 1 \rangle = r\}$.
  ↬ Use **dual formulation** of the problem.
- Projection might be difficult or impossible.
- Even when $\Theta = \mathbb{R}^d$, d might be very large (typically millions)
  ↬ use only first order methods

b. **Structure of f.** If $f = \hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle \theta, \Phi(x_i) \rangle)$, computing a gradient has a cost proportional to n.

Optimization

Take home

- We express problems as minimizing a function over a set
- Many convex problems are solved
- Difficulties come from non-convexity, lack of regularity, complexity of the set Θ, complexity of computing gradients

Our focus in this course:

- Supervised Machine Learning.
- Stochastic algorithms.

Goals:

- present algorithms (convex, large dimension, high number of observations)
- show how rates depend on smoothness and strong convexity
- show how we can use the structure
- not forgetting the initial problem: Generalization properties

Roadmap

Goals:

1. **Rates**
2. **Proof techniques**

Some slides from this section are adapted from Francis Bach's lecture in Orsay.

Classical rates for deterministic methods

- **Assumption: f convex on $\mathbb{R}^d$**
- **Classical generic algorithms**
  - **Gradient descent and accelerated gradient descent**
  - **Newton method**
  - **Subgradient method (and ellipsoid algorithm)**
- **Key additional properties of f**
  - **Lipschitz continuity, smoothness or strong convexity**
- **Key references: Nesterov (2004), Bubeck (2015)**

Several criteria for characterizing convergence

- **Objective function values**

$$f(\theta) - \inf_{\eta \in \mathbb{R}^d} f(\eta)$$

  - **Usually weaker condition**
- **Iterates**

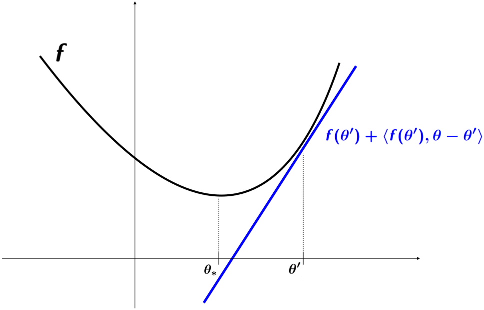
$$\inf_{\eta \in \arg\min f} \left\| \theta - \eta \right\|^2$$

  - **Typically used for strongly-convex problems**

- **NB 1: relationships between the two types in several situations**
- **NB 2: similarity with prediction vs. estimation in statistics**

Toolbox

We use a lot a few very useful inequalities.
Convex: the function is above the tangent line:

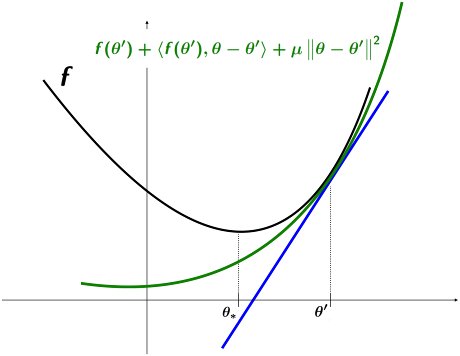


$$f(\theta) \geq f(\theta') + \langle f(\theta'), \theta - \theta' \rangle \tag{1}$$

Toolbox

We use a lot a few very useful inequalities.

Strongly-convex: function above the tangent line + μ* quadratic.



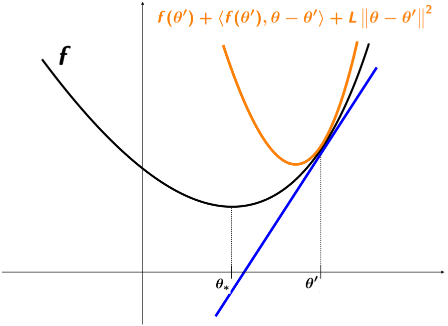
$$f(\theta) \geq f(\theta') + \langle f(\theta'), \theta - \theta' \rangle + \frac{\mu}{2} \|\theta - \theta'\|^2 \qquad (2)$$

$$\langle f'(\theta') - f'(\theta), \theta' - \theta \rangle \geq \mu \|\theta - \theta'\|^2 \qquad (3)$$

Toolbox

We use a lot a few very useful inequalities.

Smooth-convex: function below the tangent line + L* quadratic.



$$f(\theta) \leq f(\theta') + \langle f(\theta'), \theta - \theta' \rangle + \frac{L}{2} \left\| \theta - \theta' \right\|^2 \qquad (4)$$

Co-coercivity:

$$\left\| f'(\theta) - f'(\theta') \right\|^2 \leq L \langle f(\theta') - f(\theta'), \theta - \theta' \rangle \qquad (5)$$

Toolbox

3 Starting Points:

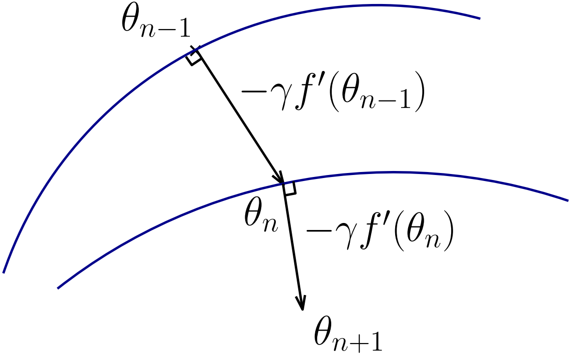
1. **Expand $\|\theta_{t+1} - \theta_*\|^2$ "Lyapunov approach"**
2. **Expand $f(\theta_{t+1}) - f(\theta_t)$ (if smooth!)**
3. **Expand $\theta_{t+1} - \theta_t$**

(smooth) Gradient Descent

- **Assumptions**
  - **f convex with L-Lipschitz-continuous gradient (e.g., L-smooth)**

- **Algorithm:**

$$\theta_t = \theta_{t-1} - \frac{1}{L} g'(\theta_{t-1})$$



(smooth) Gradient Descent - strong convexity

- **Assumptions**
  - f convex with L-Lipschitz-continuous gradient (e.g., L-smooth)
  - f μ-strongly convex
- **Algorithm:**

$$\theta_t = \theta_{t-1} - \frac{1}{L} f'(\theta_{t-1})$$

- **Bound:**

$$f(\theta_t) - f(\theta_*) \leqslant (1 - \mu/L)^t \big[f(\theta_0) - f(\theta_*) \big]$$

- Three-line proof. Challenge 1 ! (start from $(\|\theta_t - \theta_*\|^2)$
- Line search, steepest descent or constant step-size

Proof

(smooth) Gradient Descent - slow rate

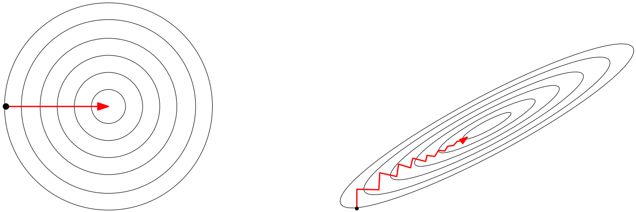
- **Assumptions**
  - f convex with L-Lipschitz-continuous gradient (e.g., L-smooth)
  - Minimum attained at θ_*
- **Algorithm:**

$$\theta_t = \theta_{t-1} - \frac{1}{L} f'(\theta_{t-1})$$

- **Bound:**

$$f(\theta_t) - f(\theta_*) \leqslant \frac{2L\|\theta_0 - \theta_*\|^2}{t+4}$$

- **Five-lines proof**
- **Adaptivity of gradient descent to problem difficulty**



Gradient descent - Proof for quadratic functions

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

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

- 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: $f(\theta_t) - f(\theta_*) \leqslant (1 - \mu/L)^{2t}\big[f(\theta_0) - f(\theta_*)\big]$

Gradient descent - Proof for quadratic functions

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

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

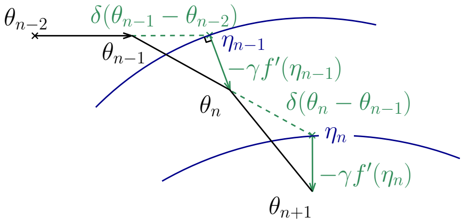
- Convexity $\mu = 0$: eigenvalues of $(I - \frac{1}{L}H)^t$ in $[0, 1]$
  - No convergence of iterates: $\|\theta_t - \theta_*\|^2 \leqslant \|\theta_0 - \theta_*\|^2$
  - Function values:

$$\begin{aligned}f(\theta_t) - f(\theta_*) &\leqslant \max_{v \in [0,L]} v(1 - v/L)^{2t}\|\theta_0 - \theta_*\|^2 \\ &\leqslant \frac{L}{t}\|\theta_0 - \theta_*\|^2\end{aligned}$$

Accelerated gradient methods (Nesterov, 1983)

- Assumptions f convex and smooth L
- Algorithm:

$$\theta_t = \eta_{t-1} - \frac{1}{L} f'(\eta_{t-1})$$
$$\eta_t = \theta_t + \frac{t-1}{t+2}(\theta_t - \theta_{t-1})$$



- Bound:

$$f(\theta_t) - f(\theta_*) \leqslant \frac{2L\|\theta_0 - \theta_*\|^2}{(t+1)^2}$$

- Ten-line proof (see, e.g., Schmidt et al., 2011)
- Not improvable
- Extension to strongly-convex functions

Accelerated gradient methods - strong convexity

- **Assumptions**
  - f convex with L-Lipschitz-cont. gradient , min. attained at θ_*
  - f μ-strongly convex
- **Algorithm:**

$$\begin{aligned}\theta_t &= \eta_{t-1} - \frac{1}{L} f'(\eta_{t-1}) \\ \eta_t &= \theta_t + \frac{1-\sqrt{\mu/L}}{1+\sqrt{\mu/L}}(\theta_t - \theta_{t-1})\end{aligned}$$

- **Bound:** $f(\theta_t) - f(\theta_*) \leqslant L\|\theta_0 - \theta_*\|^2 (1 - \sqrt{\mu/L})^t$
  - Ten-line proof (see, e.g., Schmidt et al., 2011)
  - Not improvable
  - Relationship with conjugate gradient for quadratic functions

Proof in the quadratic setting: compute the largest eigenvalue of a non-symmetric matrix. Challenge 2! Simple an insightful computation!

Other methods: Projected gradient descent

- Problems of the form: $\boxed{\min_{\theta \in \mathcal{K}} f(\theta)}$
  - $\theta_{t+1} = \arg\min_{\theta \in \mathcal{K}} f(\theta_t) + (\theta - \theta_t)^\top \nabla f(\theta_t) + \frac{L}{2}\|\theta - \theta_t\|_2^2$
  - $\theta_{t+1} = \arg\min_{\theta \in \mathcal{K}} \frac{1}{2}\left\|\theta - \left(\theta_t - \frac{1}{L}\nabla f(\theta_t)\right)\right\|_2^2$
  - Projected gradient descent
- Similar convergence rates than smooth optimization
  - Acceleration methods (Nesterov, 2007; Beck and Teboulle, 2009)

Other methods: Newton method

- Given θ_{t-1}, minimize second-order Taylor expansion

$$\begin{aligned}\tilde{f}(\theta) =& f(\theta_{t-1}) + f'(\theta_{t-1})^\top(\theta - \theta_{t-1}) \\ &+ \frac{1}{2}(\theta - \theta_{t-1})^\top f''(\theta_{t-1})^\top(\theta - \theta_{t-1})\end{aligned}$$

- Expensive Iteration: $\theta_t = \theta_{t-1} - f''(\theta_{t-1})^{-1} f'(\theta_{t-1})$
  - Running-time complexity: $O(d^3)$ in general
- Quadratic convergence: If $\|\theta_{t-1} - \theta_*\|$ small enough, for some constant C, we have

$$(C\|\theta_t - \theta_*\|) = (C\|\theta_{t-1} - \theta_*\|)^2$$

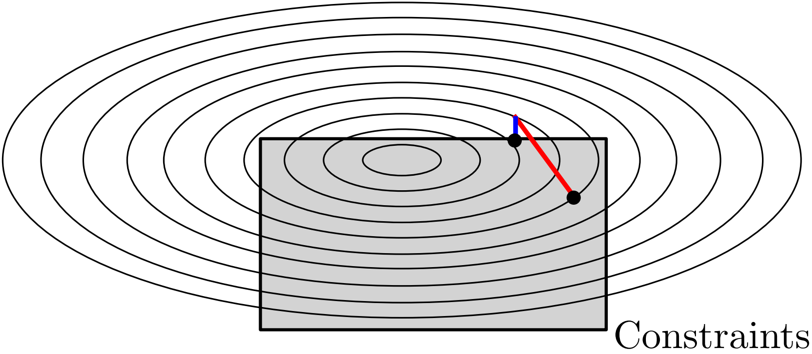
  - See Boyd and Vandenberghe (2003)

Summary: minimizing smooth convex functions

- Assumption: f convex
- Gradient descent: $\theta_t = \theta_{t-1} - \gamma_t f'(\theta_{t-1})$
  - $O(1/t)$ convergence rate for smooth convex functions
  - $O(e^{-t\mu/L})$ convergence rate for strongly smooth convex functions
  - Optimal rates $O(1/t^2)$ and $O(e^{-t\sqrt{\mu/L}})$ with FOI.
- Newton method: $\theta_t = \theta_{t-1} - f''(\theta_{t-1})^{-1} f'(\theta_{t-1})$
  - $O\big(e^{-\rho 2^t}\big)$ convergence rate
- From smooth to non-smooth
  - Subgradient method

Subgradient method/"descent" (Shor et al., 1985)

- **Assumptions**
  - f convex and B-Lipschitz-continuous on $\{\|\theta\|_2 \leqslant D\}$
- **Algorithm:** $\theta_t = \Pi_D\left(\theta_{t-1} - \frac{2D}{B\sqrt{t}} f'(\theta_{t-1})\right)$
  - Π_D : orthogonal projection onto $\{\|\theta\|_2 \leqslant D\}$



Subgradient method/"descent" (Shor et al., 1985)

- **Assumptions**
  - f convex and B-Lipschitz-continuous on $\{\|\theta\|_2 \leqslant D\}$
- **Algorithm:** $\theta_t = \Pi_D\left(\theta_{t-1} - \frac{2D}{B\sqrt{t}} f'(\theta_{t-1})\right)$
  - Π_D : orthogonal projection onto $\{\|\theta\|_2 \leqslant D\}$
- **Bound:**

$$f\left(\frac{1}{t}\sum_{k=0}^{t-1}\theta_k\right) - f(\theta_*) \leqslant \frac{2DB}{\sqrt{t}}$$

- **Three-line proof**
- **Best possible convergence rate after $O(d)$ iterations (Bubeck, 2015)**

Need for decaying steps

Example of $|x|$

Subgradient method/"descent" - proof - I

- **Iteration:** $\theta_t = \Pi_D(\theta_{t-1} - \gamma_t f'(\theta_{t-1}))$ with $\gamma_t = \frac{2D}{B\sqrt{t}}$
- **Assumption:** $\|f'(\theta)\|_2 \leqslant B$ and $\|\theta\|_2 \leqslant D$

$$\begin{aligned}
\|\theta_t - \theta_*\|_2^2 &\leqslant \|\theta_{t-1} - \theta_* - \gamma_t f'(\theta_{t-1})\|_2^2 \quad \text{by contractivity of projections} \\
&= \|\theta_{t-1} - \theta_*\|_2^2 + \gamma_t^2 \|f'(\theta_{t-1})\|_2^2 - 2\gamma_t(\theta_{t-1} - \theta_*)^\top g'(\theta_{t-1}) \\
&\leqslant \|\theta_{t-1} - \theta_*\|_2^2 + B^2\gamma_t^2 - 2\gamma_t(\theta_{t-1} - \theta_*)^\top f'(\theta_{t-1}) \\
&\qquad \text{because } \|f'(\theta_{t-1})\|_2 \leqslant B \\
&\leqslant \|\theta_{t-1} - \theta_*\|_2^2 + B^2\gamma_t^2 - 2\gamma_t\big[f(\theta_{t-1}) - f(\theta_*)\big] \\
&\qquad \text{(property of subgradients)}
\end{aligned}$$

- **leading to**

$$f(\theta_{t-1}) - f(\theta_*) \leqslant \frac{B^2\gamma_t}{2} + \frac{1}{2\gamma_t}\big[\|\theta_{t-1} - \theta_*\|_2^2 - \|\theta_t - \theta_*\|_2^2\big]$$

Subgradient method/"descent" - proof - II

- Starting from

$$f(\theta_{t-1}) - f(\theta_*) \leqslant \frac{B^2\gamma_t}{2} + \frac{1}{2\gamma_t}\left[\|\theta_{t-1} - \theta_*\|_2^2 - \|\theta_t - \theta_*\|_2^2\right]$$

- Constant step-size $\gamma_t = \gamma$

$$\sum_{u=1}^{t}\left[f(\theta_{u-1}) - f(\theta_*)\right] \leqslant \sum_{u=1}^{t}\frac{B^2\gamma}{2} + \sum_{u=1}^{t}\frac{1}{2\gamma}\left[\|\theta_{u-1} - \theta_*\|_2^2 - \|\theta_u - \theta_*\|_2^2\right]$$

$$\leqslant t\frac{B^2\gamma}{2} + \frac{1}{2\gamma}\|\theta_0 - \theta_*\|_2^2 \leqslant t\frac{B^2\gamma}{2} + \frac{2}{\gamma}D^2$$

- Optimized step-size $\gamma_t = \frac{2D}{B\sqrt{t}}$ depends on "horizon" t
  - Leads to bound of $2DB\sqrt{t}$
  - Slightly more complex proof for online setting (decreasing steps)

- Using convexity:

$$f\left(\frac{1}{t}\sum_{k=0}^{t-1}\theta_k\right) - f(\theta_*) \leqslant \frac{1}{t}\sum_{k=0}^{t-1} f(\theta_k) - f(\theta_*) \leqslant \frac{2DB}{\sqrt{t}}$$

Subgradient method/"descent" - proof - III

- Starting from
$$f(\theta_{t-1}) - f(\theta_*) \leqslant \frac{B^2\gamma_t}{2} + \frac{1}{2\gamma_t}\big[\|\theta_{t-1}-\theta_*\|_2^2 - \|\theta_t-\theta_*\|_2^2\big]$$
- Decreasing step-size

$$\begin{aligned}
\sum_{u=1}^{t}\big[f(\theta_{u-1}) - f(\theta_*)\big] &\leqslant \sum_{u=1}^{t}\frac{B^2\gamma_u}{2} + \sum_{u=1}^{t}\frac{1}{2\gamma_u}\big[\|\theta_{u-1}-\theta_*\|_2^2 - \|\theta_u-\theta_*\|_2^2\big] \\
&= \sum_{u=1}^{t}\frac{B^2\gamma_u}{2} + \sum_{u=1}^{t-1}\|\theta_u-\theta_*\|_2^2\Big(\frac{1}{2\gamma_{u+1}} - \frac{1}{2\gamma_u}\Big) + \frac{\|\theta_0-\theta_*\|_2^2}{2\gamma_1} \\
&\leqslant \sum_{u=1}^{t}\frac{B^2\gamma_u}{2} + \sum_{u=1}^{t-1}4D^2\Big(\frac{1}{2\gamma_{u+1}} - \frac{1}{2\gamma_u}\Big) + \frac{4D^2}{2\gamma_1} \\
&= \sum_{u=1}^{t}\frac{B^2\gamma_u}{2} + \frac{4D^2}{2\gamma_t} \leqslant 3DB\sqrt{t} \text{ with } \gamma_t = \frac{2D}{B\sqrt{t}}
\end{aligned}$$

- Using convexity: $f\big(\frac{1}{t}\sum_{k=0}^{t-1}\theta_k\big) - f(\theta_*) \leqslant \frac{3DB}{\sqrt{t}}$

Subgradient descent - strong convexity

- Assumptions
  - f convex and B-Lipschitz-continuous on $\{\|\theta\|_2 \leqslant D\}$
  - f μ-strongly convex
- Algorithm: $\theta_t = \Pi_D\left(\theta_{t-1} - \frac{2}{\mu(t+1)} f'(\theta_{t-1})\right)$
- Bound:

$$f\left(\frac{2}{t(t+1)}\sum_{k=1}^{t} k\theta_{k-1}\right) - f(\theta_*) \leqslant \frac{2B^2}{\mu(t+1)}$$

- Three-line proof
- Best possible convergence rate after $O(d)$ iterations (Bubeck, 2015)

Subgradient method - strong convexity - proof - I

- **Iteration:** $\theta_t = \Pi_D(\theta_{t-1} - \gamma_t f'(\theta_{t-1}))$ with $\gamma_t = \frac{2}{\mu(t+1)}$
- **Assumption:** $\|f'(\theta)\|_2 \leqslant B$ and $\|\theta\|_2 \leqslant D$ and μ-strong convexity of f

$$\|\theta_t - \theta_*\|_2^2 \leqslant \|\theta_{t-1} - \theta_* - \gamma_t f'(\theta_{t-1})\|_2^2$$

by contractivity of projections

$$\leqslant \|\theta_{t-1} - \theta_*\|_2^2 + B^2\gamma_t^2 - 2\gamma_t(\theta_{t-1} - \theta_*)^\top f'(\theta_{t-1})$$

because $\|f'(\theta_{t-1})\|_2 \leqslant B$

$$\leqslant \|\theta_{t-1} - \theta_*\|_2^2 + B^2\gamma_t^2 - 2\gamma_t\big[f(\theta_{t-1}) - f(\theta_*) + \frac{\mu}{2}\|\theta_{t-1} - \theta_*\|_2^2\big]$$

(property of subgradients and strong convexity)

↬ leading to

$$f(\theta_{t-1}) - f(\theta_*) \leqslant \frac{B^2\gamma_t}{2} + \frac{1}{2}\Big[\frac{1}{\gamma_t} - \mu\Big]\|\theta_{t-1} - \theta_*\|_2^2 - \frac{1}{2\gamma_t}\|\theta_t - \theta_*\|_2^2$$

$$\leqslant \frac{B^2}{\mu(t+1)} + \frac{\mu}{2}\Big[\frac{t-1}{2}\Big]\|\theta_{t-1} - \theta_*\|_2^2 - \frac{\mu(t+1)}{4}\|\theta_t - \theta_*\|_2^2$$

Subgradient method - strong convexity- proof - II

$$f(\theta_{t-1}) - f(\theta_*) \leqslant \frac{B^2\gamma_t}{2} + \frac{1}{2}\Big[\frac{1}{\gamma_t} - \mu\Big]\|\theta_{t-1} - \theta_*\|_2^2 - \frac{1}{2\gamma_t}\|\theta_t - \theta_*\|_2^2$$
$$\leqslant \frac{B^2}{\mu(t+1)} + \frac{\mu}{2}\Big[\frac{t-1}{2}\Big]\|\theta_{t-1} - \theta_*\|_2^2 - \frac{\mu(t+1)}{4}\|\theta_t - \theta_*\|_2^2$$

$$\sum_{u=1}^{t} u\big[f(\theta_{u-1}) - f(\theta_*)\big] \leqslant \sum_{t=1}^{u} \frac{B^2 u}{\mu(u+1)} + \frac{1}{4}\sum_{u=1}^{t}\big[u(u-1)\|\theta_{u-1} - \theta_*\|_2^2 - u(u+1)\|\theta_u - \theta_*\|_2^2\big]$$
$$\leqslant \frac{B^2 t}{\mu} + \frac{1}{4}\big[0 - t(t+1)\|\theta_t - \theta_*\|_2^2\big] \leqslant \frac{B^2 t}{\mu}$$

- Using convexity: $f\left(\frac{2}{t(t+1)}\sum_{u=1}^{t} u\theta_{u-1}\right) - f(\theta_*) \leqslant \frac{2B^2}{t+1}$
- NB: with step-size $\gamma_n = 1/(n\mu)$, extra logarithmic factor

Summary: minimizing convex functions

Gradient descent: $\theta_t = \theta_{t-1} - \gamma_t \, f'(\theta_{t-1})$

Convergence rate (= speed of convergence)

$O(1/\sqrt{t})$ for non-smooth convex functions

$O(1/t)$ for smooth convex functions

$O(e^{-t\mu/L})$ for strongly smooth convex functions

Summary of rates of convergence

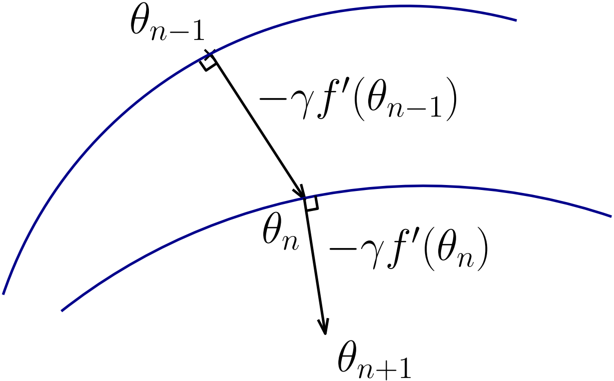
- **Problem parameters**
  - D diameter of the domain
  - B Lipschitz-constant
  - L smoothness constant
  - μ strong convexity constant

	convex	strongly convex
nonsmooth	deterministic: $BD/\sqrt{t}$	deterministic: $B^2/(t\mu)$
smooth	deterministic: LD^2/t^2	deterministic: $\exp(-t\sqrt{\mu/L})$
quadratic	deterministic: LD^2/t^2	deterministic: $\exp(-t\sqrt{\mu/L})$

Summary of the first session

1. **Optimizing a cost function is at the heart of Large scale learning**
2. **Difficulty comes from the fact that both the number of examples n and the number of dimensions d are very large.**

First method: Gradient descent: $\theta_t = \theta_{t-1} - \gamma_t f'(\theta_{t-1})$.



Summary of the first session

First method: Gradient descent: $\theta_t = \theta_{t-1} - \gamma_t f'(\theta_{t-1})$.

Convergence rate (= speed of convergence)

$O(1/t)$ for smooth convex functions

$O(e^{-t\mu/L})$ for strongly smooth convex functions

Optimal rates $O(1/t^2)$ and $O(e^{-t\sqrt{\mu/L}})$ with acceleration (optimal - not seen).

Spirit - Goals

Goals:

1. Understand what SGD is.
2. Comparison to GD (cost, convergence speed)
3. Important variants.

Approach:

1. convergence speed helps to choose between algorithms
2. influence of parameters $\rightarrow$ choice of paramaters (e.g., step size)
3. proofs help to understand assumptions

Roadmap

Back to Supervised Machine Learning framework

Example 1: Supervised Machine Learning

Consider an input/output pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, $(X, Y) \sim \rho$.

Goal: function $\theta : \mathcal{X} \to \mathbb{R}$, s.t. $\theta(X)$ good prediction for Y.

Here, as a linear function $\langle \theta, \Phi(X) \rangle$ of features $\Phi(X) \in \mathbb{R}^d$.

Consider a loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$

Define the Generalization risk :

$$\mathcal{R}(\theta) := \mathbb{E}_\rho \left[\ell(Y, \langle \theta, \Phi(X) \rangle) \right].$$

Empirical Risk minimization (I)

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

Empirical risk (or training error):

$$\hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle \theta, \Phi(x_i) \rangle).$$

Empirical risk minimization (ERM) : find $\hat{\theta}$ solution of

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, \langle \theta, \Phi(x_i) \rangle\big) \quad + \quad \mu \Omega(\theta).$$

convex data fitting term + regularizer

Empirical Risk minimization (II)

For example, least-squares regression:

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

and logistic regression:

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

Empirical Risk Minimization (ERM) setting.

$$\min_{\theta \in \mathbb{R}^d} \left\{ \hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle \theta, \Phi(x_i) \rangle) \right\}.$$

Two fundamental questions: (a) computing (b) analyzing $\hat{\theta}$.

"Large scale" framework: number of examples n and the number of explanatory variables d are both large.

1. **High dimension $d \implies$ First order algorithms**

Gradient Descent (GD) :

$$\boxed{\theta_t = \theta_{t-1} - \gamma_t \hat{\mathcal{R}}'(\theta_{t-1})}$$

Problem: computing the gradient costs $O(dn)$ per iteration.

Gradient descent for ERM

- **Assumptions** ($\mathcal{R}$ is the expected risk, $\hat{\mathcal{R}}$ the empirical risk)
  - $\hat{\mathcal{R}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \Phi(x_i)^\top \theta)$
  - ℓ smooth.
- **Cost:** At each step, compute

$$\hat{\mathcal{R}}'(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell'(y_i, \Phi(x_i)^\top \theta)\Phi(x_i).$$

cost $= nd$ each step

- **Convergence**: after t iterations of subgradient method

$$\hat{\mathcal{R}}(\theta_t) - \min_{\eta \in \Theta} \hat{\mathcal{R}}(\eta) \leqslant \frac{L}{t}$$

- **Summary:** for $t = \sqrt{n}$ iterations, convergence $L/\sqrt{n}$, with total running-time complexity of $O(n^{3/2}d)$

Empirical Risk Minimization (ERM) setting.

$$\min_{\theta \in \mathbb{R}^d} \left\{ \hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle \theta, \Phi(x_i) \rangle) \right\}.$$

Two fundamental questions: (a) computing (b) analyzing $\hat{\theta}$.

"Large scale" framework: number of examples n and the number of explanatory variables d are both large.

1. High dimension d $\Longrightarrow$ First order algorithms

Gradient Descent (GD) :

$$\boxed{\theta_t = \theta_{t-1} - \gamma_t \hat{\mathcal{R}}'(\theta_{t-1})}$$

Problem: computing the gradient costs $O(dn)$ per iteration.

2. Large n $\Longrightarrow$ Stochastic algorithms

Stochastic Gradient Descent (SGD)

Idea of SGD

What is our main problem? computing

$$\hat{\mathcal{R}}'(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell'(y_i, \Phi(x_i)^\top\theta)\Phi(x_i) =: \frac{1}{n}\sum_{i=1}^{n} f_i'(\theta)$$

costs *nd* per iteration

Solution?
Use instead for the gradient just one element of the sum!!

$$f_i'(\theta) \qquad \big(= \ell'(y_i, \Phi(x_i)^\top\theta)\Phi(x_i)\big)$$

with $i \in \mathcal{U}\{1, \dots, n\}$.

One observation at each step $\rightarrow$ complexity d per iteration.

SGD for ERM: $f = \hat{\mathcal{R}}$

Loss for a single pair of observations, for any $j \leq n$:

$$f_j(\theta) := \ell(y_j, \langle \theta, \Phi(x_j) \rangle).$$

For the empirical risk $\hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{t=1}^{n} \ell(y_t, \langle \theta, \Phi(x_t) \rangle)$.

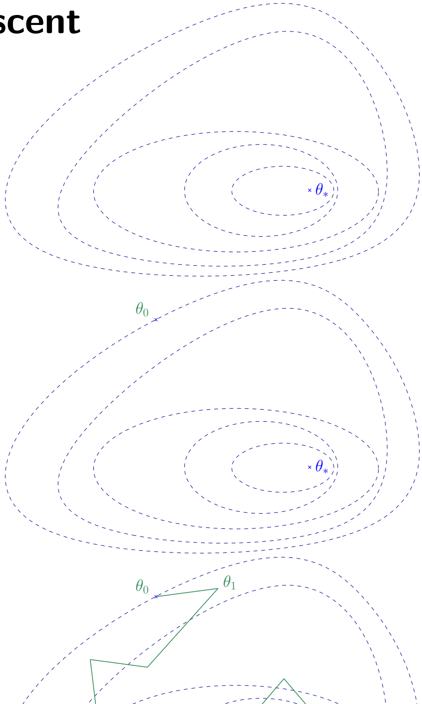
- At each step $t \in \mathbb{N}^*$, sample $I_t \sim \mathcal{U}\{1, \dots n\}$:

$$f'_{I_t}(\theta_{t-1}) = \ell'(y_{I_t}, \langle \theta_{t-1}, \Phi(x_{I_t}) \rangle)$$

$$\mathbb{E}[f'_{I_t}(\theta_{t-1})] = \frac{1}{n} \sum_{t=1}^{n} f'_i(\theta_{t-1}) = \hat{\mathcal{R}}'(\theta_{t-1}).$$

More generally, let's define SGD for a general function f.

Stochastic Gradient descent



- **Goal:**

$$\min_{\theta \in \mathbb{R}^d} f(\theta)$$

  given unbiased gradient estimates f'_n

- $\theta_* := \mathrm{argmin}_{\mathbb{R}^d} f(\theta)$.

Why is randomness not a problem

Key insights from Bottou and Bousquet (2008)

1. In machine learning, no need to optimize below statistical error
2. In machine learning, cost functions are averages
3. Testing errors are more important than training errors

Take home

SGD is :

1. Necessary in the Large Scale setting (complexity)
2. Well suited to Learning problems !

Convergence ?

Analysis: behaviour of $(\theta_n)_{n\geq 0}$

$$\boxed{\theta_t = \theta_{t-1} - \gamma_t\, f_t'(\theta_{t-1})}$$

Importance of the learning rate $(\gamma_t)_{t\geq 0}$.

For smooth and strongly convex problem, $\theta_t \to \theta_*$ a.s. if

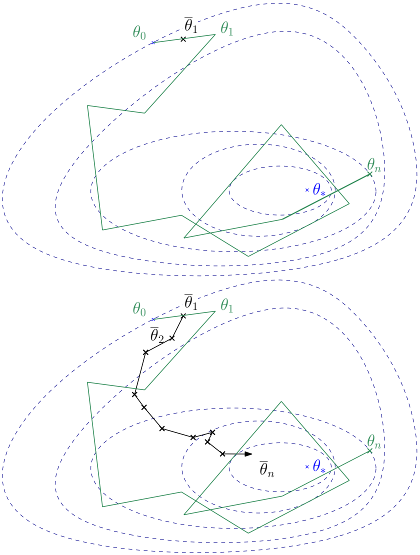
$$\sum_{t=1}^{\infty} \gamma_t = \infty \qquad \sum_{t=1}^{\infty} \gamma_t^2 < \infty.$$

Converges as

$$\frac{L}{\mu^2 t}$$

- Limit (variance) scales as $1/\mu^2$
- Very sensitive to ill-conditioned problems.
- μ generally unknown...

Polyak Ruppert averaging



Introduced by Polyak and Juditsky (1992) and Ruppert (1988):

$$\bar{\theta}_t = \frac{1}{t+1}\sum_{i=0}^{t}\theta_i.$$

- **off line averaging reduces the noise effect.**
- **on line computing:** $\bar{\theta}_{t+1} = \frac{1}{t+1}\theta_{t+1} + \frac{t}{t+1}\bar{\theta}_t$.

Convex stochastic approximation: convergence

Known global minimax rates for non-smooth problems

- **Strongly convex: $O((\mu t)^{-1})$**
  Attained by averaged stochastic gradient descent with $\gamma_t \propto (\mu t)^{-1}$
- **Non-strongly convex: $O(t^{-1/2})$**
  Attained by averaged stochastic gradient descent with $\gamma_t \propto t^{-1/2}$

For smooth problems, use larger steps

- **Strongly convex: $O(\mu t)^{-1}$**
  for $\gamma_t \propto t^{-1/2}$: adapts to strong convexity.

Convergence rate for $f(\tilde{\theta}_t) - f(\theta_*)$, smooth f.

	$\min \hat{\mathcal{R}}$	
	SGD	**GD**
Convex	$O\left(\frac{1}{\sqrt{t}}\right)$	$O\left(\frac{1}{t}\right)$
Stgly-Cvx	$O\left(\frac{1}{\mu t}\right)$	$O(e^{-\mu t})$

Convergence rate for $f(\tilde{\theta}_t) - f(\theta_*)$, smooth f.

$\min \hat{\mathcal{R}}$

	SGD	GD
Convex	$O\left(\frac{1}{\sqrt{t}}\right)$	$O\left(\frac{1}{t}\right)$
Stgly-Cvx	$O\left(\frac{1}{\mu t}\right)$	$O(e^{-\mu t})$

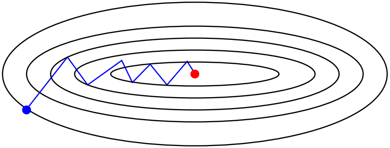
$\ominus$ Gradient descent update costs n times as much as SGD update.

Which one to choose?
Can we get best of both worlds?

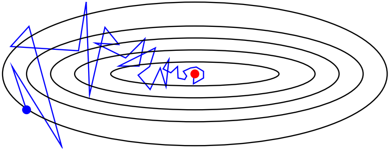
Stochastic vs. deterministic methods

- **Batch** gradient descent:

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



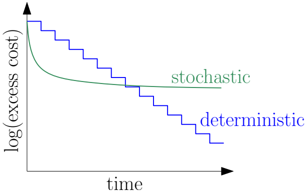
- **Stochastic** gradient descent: $\theta_t = \theta_{t-1} - \gamma_t f'_{i(t)}(\theta_{t-1})$



Comparison of convergence : SGD vs GD

Which one to choose?

1. **Depends on the precision we want.**



Example: non strongly convex case.

2. **If our goal is to get a convergence of $1/\sqrt{n}$, then**
   - Complexity of GD: $n^{3/2}d$
   - Complexity of SGD: nd.
3. **If our goal is to get a convergence of $1/n^2$, then**
   - Complexity of GD: $n^3 d$ (n^2 iterations)
   - Complexity of SGD: $n^4 d$ (n^4 iterations).

Why one is the most likely in Learning ? (Details later...)

Take home

1. **SGD is a great algorithm**
2. **Exactly suited for Large Scale Learning**
   2.1 **Low complexity per iteration**
   2.2 $\looparrowright$ **rapid convergence to a correct precision**

Question 2: Can we get best of both worlds?

Methods for finite sum minimization

- GD: at step t, use $\frac{1}{n}\sum_{i=0}^{n} f_i'(\theta_t)$
- SGD: at step t, sample $i_t \sim \mathcal{U}[1;n]$, use $f_{i_t}'(\theta_t)$
- SAG: at step t,
  - keep a "full gradient" $\frac{1}{n}\sum_{i=0}^{n} f_i'(\theta_{t_i})$, with $\theta_{t_i} \in \{\theta_1, \ldots \theta_t\}$
  - sample $i_t \sim \mathcal{U}[1;n]$, use

$$\frac{1}{n}\left(\sum_{i=0}^{n} f_i'(\theta_{t_i}) - f_{i_t}'(\theta_{t_{i_t}}) + f_{i_t}'(\theta_t)\right),$$

In other words:

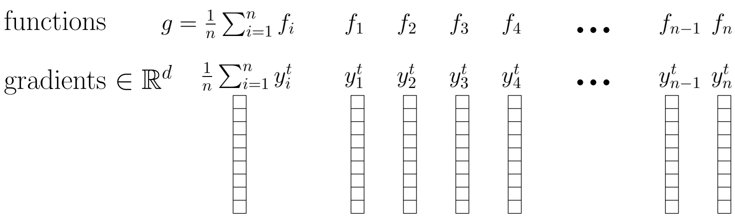
- Keep in memory past gradients of all functions f_i, $i = 1, \ldots, n$
- Random selection $i(t) \in \{1, \ldots, n\}$ with replacement
- Iteration: $\theta_t = \theta_{t-1} - \frac{\gamma_t}{n}\sum_{i=1}^{n} y_i^t$ with

$$y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$$

SAG

- Keep in memory past 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

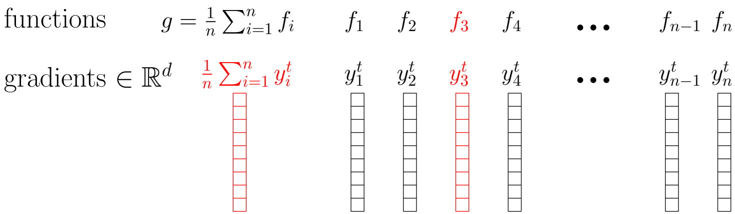
$$y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$$



SAG

- Keep in memory past 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

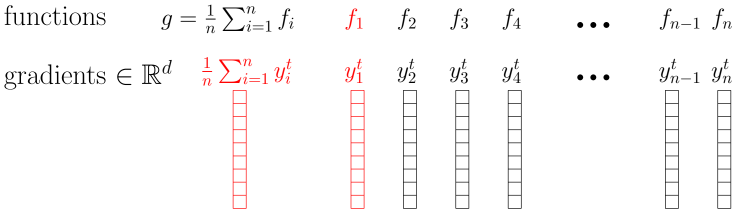
$$y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$$



SAG

- Keep in memory past 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

$$y_i^t = \begin{cases} f_i'(\theta_{t-1}) & \text{if } i = i(t) \\ y_i^{t-1} & \text{otherwise} \end{cases}$$



↬ ⊕ update costs the same as SGD

↬ ⊖ needs to store all gradients $f_i'(\theta_{t_i})$ at "points in the past"

Variance reduced methods

Some references:

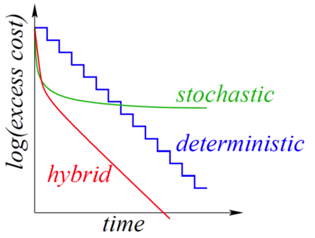
- **SAG Schmidt et al. (2013), SAGA Defazio et al. (2014a)**
- **SVRG Johnson and Zhang (2013) (reduces memory cost but 2 epochs...)**
- **FINITO Defazio et al. (2014b)**
- **S2GD Konečnỳ and Richtárik (2013)...**

And many others... See for example Niao He's lecture notes for a nice overview.

Convergence rate for $f(\tilde{\theta}_t) - f(\theta_*)$, smooth objective f.

$\min \hat{\mathcal{R}}$

	SGD	GD	SAG
Convex	$O\left(\frac{1}{\sqrt{t}}\right)$	$O\left(\frac{1}{t}\right)$	
Stgly-Cvx	$O\left(\frac{1}{\mu t}\right)$	$O(e^{-\mu t})$	$O\left(1-(\mu \wedge \frac{1}{n})\right)^t$



GD, SGD, SAG (Fig. from Schmidt et al. (2013))

Summary

Take home

1. **Variance reduced algorithms can have both:**
   - **low iteration cost**
   - **fast asymptotic convergence**

How precisely do I need to converge?

Generalization gap: the overfitting problem ?

My true goal is to control $\mathcal{R}$:

$$\mathcal{R}(\theta) := \mathbb{E}_{\rho}\left[\ell(Y, \langle \theta, \Phi(X) \rangle)\right].$$

Optimization: after t iterations of one method

$$\hat{\mathcal{R}}(\hat{\theta}) - \hat{\mathcal{R}}(\theta_*) \leqslant \frac{C}{t^?}$$

Statistics: with probability greater than $1 - \delta$

$$\sup_{\theta \in \Theta} |\hat{\mathcal{R}}(\theta) - \mathcal{R}(\theta)| \leqslant \frac{GRD}{\sqrt{n}}\left[2 + \sqrt{2 \log \frac{2}{\delta}}\,\right]$$

SGD for the generalization risk: $f = \mathcal{R}$

SGD: key assumption $\mathbb{E}[f_n'(\theta_{n-1})|\mathcal{F}_{n-1}] = f'(\theta_{n-1})$.

For the risk

$$\mathcal{R}(\theta) = \mathbb{E}_\rho\left[\ell(Y, \langle\theta, \Phi(X)\rangle)\right]$$

- At step $0 < k \leq n$, use a new point independent of θ_{k-1}:

$$f_k'(\theta_{k-1}) = \ell'(y_k, \langle\theta_{k-1}, \Phi(x_k)\rangle)$$

- For $0 \leq k \leq n$, $\mathcal{F}_k = \sigma((x_i, y_i)_{1 \leq i \leq k})$.

$$\begin{aligned}\mathbb{E}[f_k'(\theta_{k-1})|\mathcal{F}_{k-1}] &= \mathbb{E}_\rho[\ell'(y_k, \langle\theta_{k-1}, \Phi(x_k)\rangle)|\mathcal{F}_{k-1}] \\ &= \mathbb{E}_\rho\left[\ell'(Y, \langle\theta_{k-1}, \Phi(X)\rangle)\right] = \mathcal{R}'(\theta_{k-1})\end{aligned}$$

- Single pass through the data, Running-time $= O(nd)$,
- "Automatic" regularization.

SGD for the generalization risk: $f = \mathcal{R}$

	ERM minimization	Gen. risk minimization
	several passes : $0 \leq k$	One pass $0 \leq k \leq n$
x_i, y_i is	$\mathcal{F}_t$-measurable for any t	$\mathcal{F}_t$-measurable for $t \geq i$.

Convergence rate for $f(\tilde{\theta}_k) - f(\theta_*)$, smooth objective f.

	$\min \hat{\mathcal{R}}$			$\min \mathcal{R}$
	SGD	GD	SAG	SGD
Convex	$O\left(\frac{1}{\sqrt{k}}\right)$	$O\left(\frac{1}{k}\right)$		$O\left(\frac{1}{\sqrt{k}}\right)$
Stgly-Cvx	$O\left(\frac{1}{\mu k}\right)$	$O(e^{-\mu k})$	$O\left(1-(\mu \wedge \frac{1}{n})\right)^k$	$O\left(\frac{1}{\mu k}\right)$

Convergence rate for $f(\tilde{\theta}_k) - f(\theta_*)$, smooth objective f.

	$\min \hat{\mathcal{R}}$			$\min \mathcal{R}$
	SGD	GD	SAG	SGD
Convex	$O\left(\frac{1}{\sqrt{k}}\right)$	$O\left(\frac{1}{k}\right)$		$O\left(\frac{1}{\sqrt{n}}\right)$
Stgly-Cvx	$O\left(\frac{1}{\mu k}\right)$	$O(e^{-\mu k})$	$O\left(1 - (\mu \wedge \frac{1}{n})\right)^k$	$O\left(\frac{1}{\mu n}\right)$
	$0 \leq k$			$0 \leq k \leq n$

Gradient is unknown

Take home

- In the context of large scale learning, we have to use SGD
- It is a stochastic algorithm
- Typically, steps sizes have to decay to 0
- For smooth problems, larger steps are allowed and adapts to strong convexity.

Moreover: "one epoch = one pass over my observations"

Take home

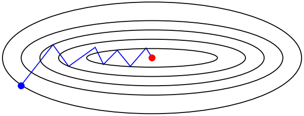
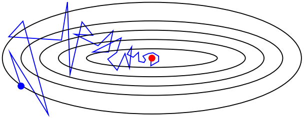
- It is possible to use variance reduced algorithms to have a faster convergence rate after many epochs.
- During the first epoch, we optimize the (unknown!) generalization error!!
  - powerful remark
  - e.g., streaming setting.

Next Goals

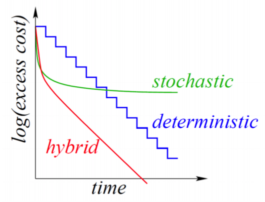
1. **Even larger steps ?**
2. **Mini-batch algorithms.**
3. **Adaptive algorithms.**

Summary of the first two days

1. **Large Scale Learning framework**
2. **Optimization**
   - **First order methods: speed of convergence of GD**
   - **SGD vs GD: SGD is fast & low precision**

   - **Variance reduced SGD**
   - **Generalization with SGD: we can optimize an unknown function!**



Convergence rate $f(\tilde{\theta}_k) - f(\theta_*)$, smooth objective f.

	$\min \hat{\mathcal{R}}$			$\min \mathcal{R}$
	SGD	GD	SAG	SGD
Convex	$O\left(\frac{1}{\sqrt{k}}\right)$	$O\left(\frac{1}{k}\right)$		$O\left(\frac{1}{\sqrt{n}}\right)$
Stgly-Cvx	$O\left(\frac{1}{\mu k}\right)$	$O(e^{-\mu k})$	$O\left(1 - (\mu \wedge \frac{1}{n})\right)^k$	$O\left(\frac{1}{\mu n}\right)$
	$0 \leq k$			$0 \leq k \leq n$

Today

1. Mini-batch algorithms
2. Adaptive algorithms
3. (Markov chain point of view)

Outline

See the very good post:

`http://ruder.io/optimizing-gradient-descent/`

Minibatch SGD for ERM: $f = \hat{\mathcal{R}}$

Loss for a single pair of observations, for any $j \leq n$:

$$f_j(\theta) := \ell(y_j, \langle \theta, \Phi(x_j) \rangle).$$

Empirical risk $\hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{t=1}^{n} \ell(y_t, \langle \theta, \Phi(x_t) \rangle)$.

SGD:

- At each step $t \in \mathbb{N}^*$, sample $I_t \sim \mathcal{U}\{1, \dots n\}$:

$$\theta_t = \theta_{t-1} - \gamma_t f'_{I_t}(\theta_{t-1})$$

Mini-batch SGD: choose $m \leq n$

- At each step $t \in \mathbb{N}^*$, sample $(I_{1,t}, \dots, I_{m,t}) \sim \mathcal{U}\{1, \dots n\}^{\otimes m}$:

$$\theta_t = \theta_{t-1} - \gamma_t \frac{1}{m} \sum_{i=1}^{m} f'_{I_{i,t}}(\theta_{t-1})$$

Minibatch SGD : behavior

1. Gradient is still stochastic (if $m < n$)
2. Level of noise in the gradient is reduced: formally

$$\mathrm{var}\left(\frac{1}{m}\sum_{i=1}^{m} f'_{I_{i,t}}(\theta_{t-1})\right) = \frac{1}{m}\,\mathrm{var}\left(f'_{I_t}(\theta_{t-1})\right)$$

3. Cost/time per iteration?
   - cost/complexity: $O(md)$ per iteration
   - In practice, distribution of the computation over many cores can reduce the time par iteration to less than $O(md)$.
4. Convergence ?

We denote $\sigma^2 = \mathrm{var}\left(f'_{I_t}(\theta_{t-1})\right)$.

Convergence of SGD for smooth smooth f

SGD:

1. **What matters? For smooth functions - the Variance of stochastic gradient. Bound $\simeq$:**

$$f(\bar{\theta}_t) - f(\theta_*) \leq \frac{\|\theta_0 - \theta_*\|^2}{\gamma_t t} + \gamma_t \operatorname{var}\left(f'_{I_t}(\theta_{t-1})\right).$$

2. **"Optimal" step size: $\gamma_t = \sqrt{\frac{\|\theta_0 - \theta_*\|^2}{\sigma^2 t}}$: gives a rate**

$$f(\bar{\theta}_t) - f(\theta_*) \leq 2\sqrt{\frac{\sigma^2 \|\theta_0 - \theta_*\|^2}{t}}.$$

Step size has always to be $\leq \frac{2}{L}$ otherwise SGD diverges.

Convergence of mini-batch SGD for smooth f

Mini-batch SGD:

- to keep same total complexity: $t \leftarrow t/m$
- Reduced variance : $\sigma^2 \leftarrow \sigma^2/m$

1. For smooth functions - the Variance of stochastic gradient. Bound $\simeq$:

$$f(\bar{\theta}_{t/m}) - f(\theta_*) \leq \frac{\|\theta_0 - \theta_*\|^2}{\gamma_t t/m} + \frac{\gamma_t \sigma^2}{m}.$$

2. "Optimal" step size: $\gamma_t = \sqrt{\frac{\|\theta_0 - \theta_*\|^2}{\sigma^2/m \quad t/m}} = m\sqrt{\frac{\|\theta_0 - \theta_*\|^2}{\sigma^2 t}}$: gives a rate

$$f(\bar{\theta}_t) - f(\theta_*) \leq 2\sqrt{\frac{\sigma^2\|\theta_0 - \theta_*\|^2}{t}}.$$

Step size has always to be $\leq \frac{2}{L}$ otherwise SGD diverges.

Convergence of mini-batch SGD for smooth f

	SGD	$m-$Mini-batch SGD	
Steps $\mathbb{C} = O(td)$	t	$\frac{t}{m}$	
Gradient Variance	σ^2	$\frac{\sigma^2}{m}$	
Optimal step	$\frac{c_{\theta_0,\sigma^2}}{\sqrt{t}}$	$m\frac{c_{\theta_0,\sigma^2}}{\sqrt{t}}$	$\wedge 2L^{-1}$!
Global rate		$\sqrt{\frac{\sigma^2\|\theta_0-\theta_*\|^2}{t}}$	

1. Same Global convergence rate
2. If mini-batch size starts being too large, saturation because of the upper bound on the step size
3. Reasonable (n-minibatch = GD !)
4. In practice, used a lot because time $<$ complexity.

Convergence of SGD for smooth non-smooth f

SGD:

1. What matters? For non-smooth functions - the upper bound B^2 on stochastic gradient. Bound $\simeq$:

$$f(\bar{\theta}_t) - f(\theta_*) \leq \frac{\|\theta_0 - \theta_*\|^2}{\gamma_t t} + \gamma_t \sup \mathbb{E}\|f'_{I_t}(\theta_{t-1})\|^2.$$

Mini-batch SGD:

$$\sup \mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m} f'_{I_{i,t}}(\theta_{t-1})\right\|^2 \lesssim \sup \mathbb{E}\|f'_{I_t}(\theta_{t-1})\|^2$$

1. Same bound for same number of iterations
2. Higher cost par iteration

Using mini-batch is a bad idea.

Convergence of minibatch SAG

In variance reduced method:

1. **The variance is already reduced by the method itself**
2. **No need to use mini-batch**

Take home

Mini-batch gradient descent:

1. **Simple algorithm derived for SGD using a small "batch" of examples**
2. **Reduces the variance of the random gradients**
3. **Helps when**
   - **1. Function is smooth, &**
   - **2. m not too large (Saturation) &**
   - **3. Time $<$ Complexity**
4. **Does not help much for non smooth function or Variance reduced methods.**

Remark: all these insights come from theory and proofs.

Take home

Read papers or ask people with theoretical knowledge :)

Outline

See the very good post:

`http://ruder.io/optimizing-gradient-descent/`

Challenge number 1: Acceleration

1. Earlier we saw that we could accelerate GD getting a better rate
2. Similar process for SGD.
   - Might cause instability or divergence
   - Not fully understood theoretically
   - Used a lot in practice

Momentum algorithmI I

Aim: related to Nesterov Acceleration but older (1964)

Particularly useful for stochastic gradient descent.

https://distill.pub/2017/momentum/

Momentum algorithm II

Polyak's momentum algorithm - Heavy ball method

1. starting point $\theta^{(0)}$,
2. learning rate $\gamma_t > 0$,
3. momentum $\beta \in [0, 1]$ (default $\beta = 0.9$).

Iterate

$$\theta_{t+1} = \theta_t - \gamma_t \nabla f(\theta_t) \quad + \beta(\theta_t - \theta_{t-1})$$

Return last $\theta^{(t+1)}$.

Image 2: SGD without momentum

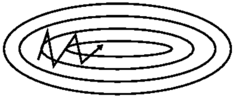


Image 3: SGD with momentum

Challenge number 2: Adaptation

1. **Same learning rate for all coordinates. Could we use a different learning rate for all coordinates ?**
   i.e., for $1 \leq j \leq d$:

$$(\theta_t)_j = (\theta_{t-1})_j - \gamma_{t,j}(f'_{I_t}(\theta_{t-1}))_j$$

Intuition: Gradient descent

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

$$\theta_t - \theta_* \;=\; (I - \gamma H)(\theta_{t-1} - \theta_*) = (I - \gamma H)^t(\theta_0 - \theta_*)$$

- If $H = \mathrm{Diag}(\alpha_1, \ldots, \alpha_d)$, $\alpha_1 = L$, $\alpha_d = \mu$
- For coordinate j, we have:

$$(\theta_t)_j = \left(1 - \gamma\alpha_j\right)^t(\theta_0 - \theta_*)_j$$

- $\looparrowright$ step size cannot be larger than $2/\alpha_1 = 2/L$ otherwise first coefficient $|\,(1 - \gamma\alpha_1)\,| > 1$ and this coordinate diverges.
- $\looparrowright$ Rate is dictated by the smallest coordinate: rate $(1 - \alpha_d/\alpha_1)^t = (1 - \mu/L)^t$

Using different γ per coordinate would be great.

Notations

$$(\theta_t)_j = (\theta_{t-1})_j - \gamma_{t,k}(f'_{I_t}(\theta_{t-1}))_j$$

1. $g_t = f'_{I_t}(\theta_{t-1})$ stochastic gradient at time t

$$(\theta_t)_j = (\theta_{t-1})_j - \gamma_{t,j}(g_t)_j$$

2. Avoiding double subscript:

$$(\theta^t)_j = (\theta^{t-1})_j - \gamma_j^t(g^t)_j$$

$$\theta_j^t = \theta_j^{t-1} - \gamma_j^t g_j^t$$

ADAGRAD

$$\theta_j^t = \theta_j^{t-1} - \gamma_j^t g_j^t$$

Special choice for step-sizes:

$$\theta_j^t = \theta_j^{t-1} - \frac{\gamma}{\sqrt{C_{t,j} + \varepsilon}} g_j^t$$

ADAptive GRADient algorithm

1. starting point θ^0,
2. learning rate $\gamma > 0$, (default value of 0.01)
3. momentum β, constant ε.

For $t = 1, 2, \ldots$ until convergence do for $1 \leq j \leq d$

$$\theta_j^{t+1} \leftarrow \theta_j^t - \frac{\gamma}{\sqrt{\sum_{\tau=1}^t (g_j^\tau)^2 + \varepsilon}} g_j^t$$

Return last θ^t

ADAGRAD

Update equation for ADAGRAD $\theta_j^{t+1} \leftarrow \theta_j^t - \frac{\gamma}{\sqrt{\sum_{\tau=1}^{t}(g_j^\tau)^2+\varepsilon}} g_j^t$

Pros:

- Different dynamic rates on each coordinate
- Dynamic rates grow as the inverse of the gradient magnitude:
  1. Large/small gradients have small/large learning rates
  2. The dynamic over each dimension tends to be of the same order
  3. Interesting for NN in which gradient at different layers can be of different order of magnitude.
- Accumulation of gradients in the denominator act as a decreasing learning rate.

Cons:

- Very sensitive to initial condition: large initial gradients lead to small learning rates.
- Can be fought by increasing the learning rate thus making the algorithm sensitive to the choice of the learning rate.

Improving upon AdaGrad: AdaDelta

Idea : restricts the window of accumulated past gradients to some fixed size.

1. starting point θ^0, constant ε,
2. new params : decay rate $\rho > 0$

Update:

$$\theta_j^{t+1} = \theta_j^t - \frac{\gamma_j^t}{\sqrt{C_{j,t} + \varepsilon}} g_j^t$$

Before: $C_{j,t} = \sum_{\tau=1}^{t} (g_j^\tau)^2$
Now: $C_{j,t} = \rho C_j^{t-1} + (1-\rho)(g_j^t)^2$

Adadelta

Interpretation:

- **Less sensitivity to initial parameters than Adagrad.**
- γ_j^t**is chosen to by size the previous step in memory and enforce larger steps along directions in which large steps were made.**
- **The denominator keeps the size of the previous gradients in memory and acts as a decreasing learning rate. Weights are lower than in Adagrad due to the decay rate ρ.**

RMSprop

Unpublished methode, from the online course of Geoff Hinton

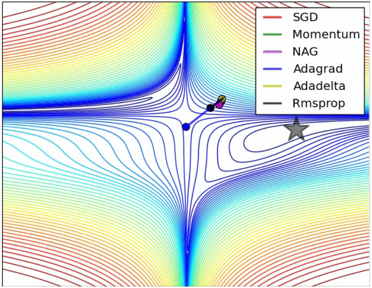
http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

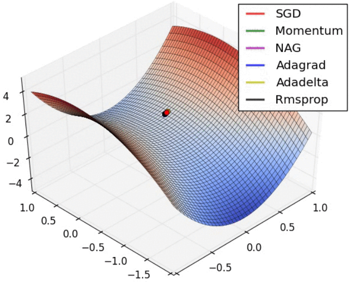
1. **starting point θ^0, constant ε,**
2. **decay rate $\rho > 0$**
3. **"new" step size γ (default = 0.001)**

Update:

$$\theta_j^{t+1} = \theta_j^t - \frac{\gamma}{\sqrt{C_{j,t} + \varepsilon}} g_j^t$$

Animation of Stochastic Gradient algorithms





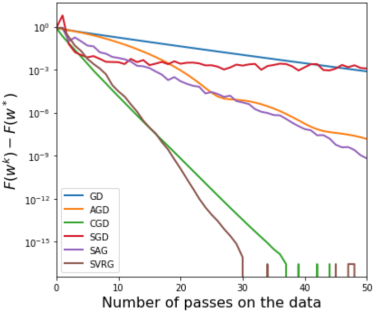
Credits to Alec Radford for the animations.

Wednesday

Goal: Code:

1. **gradient descent (GD)**
2. **accelerated gradient descent (AGD)**
3. **coordinate gradient descent (CD)**
4. **stochastic gradient descent (SGD)**
5. **stochastic variance reduced gradient descent (SAG)**
6. **Adagrad**

for the linear regression and logistic regression models, with the ridge penalization.



Wednesday

1. **Who knows python ?**
2. **Who's using anaconda?**

Least Mean Squares: rate independent of μ

Least-squares: $\mathcal{R}(\theta) = \frac{1}{2}\mathbb{E}\big[(Y - \langle \Phi(X), \theta\rangle)^2\big]$

Analysis for averaging and constant step-size $\gamma = 1/(4R^2)$ (?)

- Assume $\|\Phi(x_n)\| \leqslant r$ and $|y_n - \langle \Phi(x_n), \theta_*\rangle| \leqslant \sigma$
- No assumption regarding lowest eigenvalues of the Hessian

$$\mathbb{E}\mathcal{R}(\bar{\theta}_n) - \mathcal{R}(\theta_*) \leqslant \frac{4\sigma^2 d}{n} + \frac{\|\theta_0 - \theta_*\|^2}{\gamma n}$$

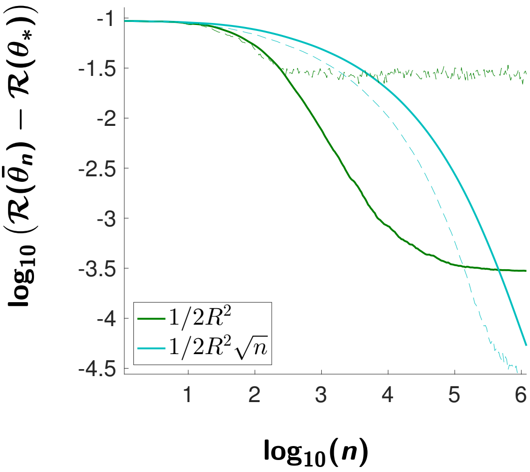
- Matches statistical lower bound (Tsybakov, 2003).
- Optimal rate with “large” step sizes

Take home

- SGD can be used to minimize the true risk directly
- Stochastic algorithm to minimize unknown function
- No regularization needed, only one pass
- For Least Squares, with constant step, optimal rate .

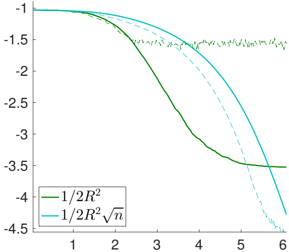
Beyond least squares. Logistic regression

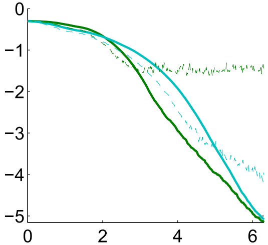
$$\min_{\theta \in \mathbb{D}^d} \quad \mathbb{E} \log \Big(1 + \exp(-Y \langle \theta, \Phi(X) \rangle) \Big).$$



Logistic regression. **Final iterate (dashed), and averaged recursion (plain).**

Motivation 2/ 2. Difference between quadratic and logistic loss





Logistic Regression

$$\mathbb{E}\mathcal{R}(\bar{\theta}_n) - \mathcal{R}(\theta_*) = O(\gamma^2)$$

with $\gamma = 1/(4R^2)$

Least-Squares Regression

$$\mathbb{E}\mathcal{R}(\bar{\theta}_n) - \mathcal{R}(\theta_*) = O\left(\frac{1}{n}\right)$$

with $\gamma = 1/(4R^2)$

SGD: an homogeneous Markov chain

Consider a $L-$smooth and $\mu-$strongly convex function $\mathcal{R}$.

SGD with a step-size $\gamma > 0$ is an homogeneous Markov chain:

$$\theta_{k+1}^{\gamma} = \theta_{k}^{\gamma} - \gamma\big[\mathcal{R}'(\theta_{k}^{\gamma}) + \varepsilon_{k+1}(\theta_{k}^{\gamma})\big] \ ,$$

- satisfies Markov property
- is homogeneous, for γ constant, $(\varepsilon_k)_{k\in\mathbb{N}}$ i.i.d.

Also assume:

- $\mathcal{R}'_k = \mathcal{R}' + \varepsilon_{k+1}$ is almost surely L-co-coercive.
- Bounded moments

$$\mathbb{E}[\|\varepsilon_k(\theta_*)\|^4] < \infty.$$

Stochastic gradient descent as a Markov Chain: Analysis framework†

- **Existence of a limit distribution π_γ, and linear convergence to this distribution:**

$$\theta_k^\gamma \xrightarrow{d} \pi_\gamma.$$

- **Convergence of second order moments of the chain,**

$$\bar{\theta}_k^\gamma \xrightarrow[k\to\infty]{L^2} \bar{\theta}_\gamma := \mathbb{E}_{\pi_\gamma}[\theta].$$

- **Behavior under the limit distribution ($\gamma \to 0$): $\bar{\theta}_\gamma = \theta_* + ?$.**

↬ **Provable convergence improvement with extrapolation tricks.**

†Dieuleveut, Durmus, Bach [2017], published in AOS 19

Existence of a limit distribution $\gamma \to 0$

Goal: $(\theta_k^\gamma)_{k \geq 0} \xrightarrow{d} \pi_\gamma$.

Theorem

For any $\gamma < L^{-1}$, the chain $(\theta_k^\gamma)_{k \geq 0}$ admits a unique stationary distribution π_γ. In addition for all $\theta_0 \in \mathbb{R}^d$, $k \in \mathbb{N}$:

$$W_2^2(\theta_k^\gamma, \pi_\gamma) \leq (1 - 2\mu\gamma(1 - \gamma L))^k \int_{\mathbb{R}^d} \|\theta_0 - \vartheta\|^2 \, \mathrm{d}\pi_\gamma(\vartheta) \; .$$

Wasserstein metric: distance between probability measures.

Behavior under limit distribution.

Ergodic theorem: $\bar{\theta}_k \to \mathbb{E}_{\pi_\gamma}[\theta] =: \bar{\theta}_\gamma$. Where is $\bar{\theta}_\gamma$?

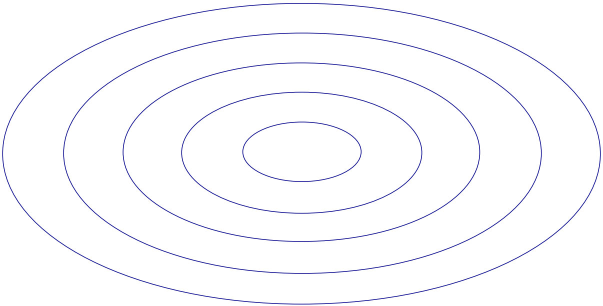
If $\theta_0 \sim \pi_\gamma$, then $\theta_1 \sim \pi_\gamma$.

$$\theta_1^\gamma = \theta_0^\gamma - \gamma\big[\mathcal{R}'(\theta_0^\gamma) + \varepsilon_1(\theta_0^\gamma)\big] \ .$$

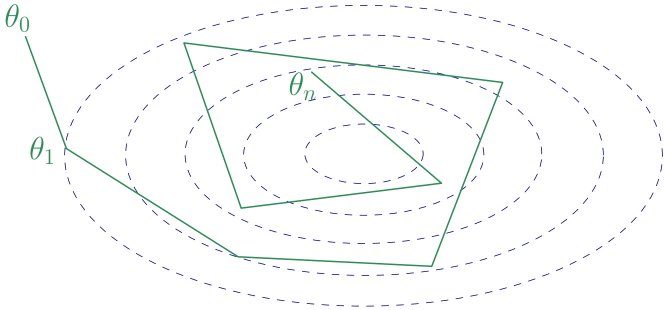
$$\mathbb{E}_{\pi_\gamma}\left[\mathcal{R}'(\theta)\right] = 0$$

In the quadratic case (linear gradients) $\Sigma\mathbb{E}_{\pi_\gamma}[\theta - \theta_*] = 0$: $\bar{\theta}_\gamma = \theta_*$!

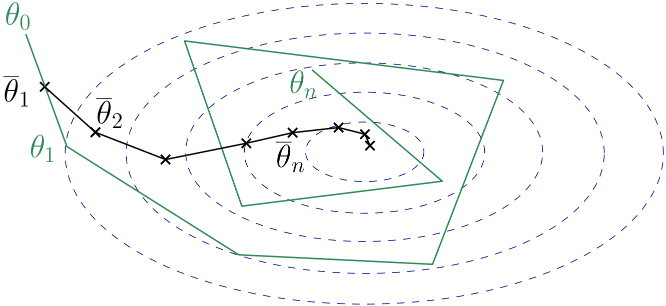
Constant learning rate SGD: convergence in the quadratic case



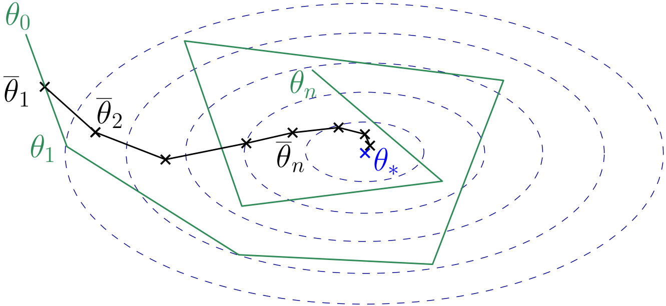
Constant learning rate SGD: convergence in the quadratic case



Constant learning rate SGD: convergence in the quadratic case



Constant learning rate SGD: convergence in the quadratic case



Behavior under limit distribution.

Ergodic theorem: $\bar{\theta}_n \to \mathbb{E}_{\pi_\gamma}[\theta] =: \bar{\theta}_\gamma$. Where is $\bar{\theta}_\gamma$?

If $\theta_0 \sim \pi_\gamma$, then $\theta_1 \sim \pi_\gamma$.

$$\theta_1^\gamma = \theta_0^\gamma - \gamma\big[\mathcal{R}'(\theta_0^\gamma) + \varepsilon_1(\theta_0^\gamma)\big] .$$

$$\mathbb{E}_{\pi_\gamma}\left[\mathcal{R}'(\theta)\right] = 0$$

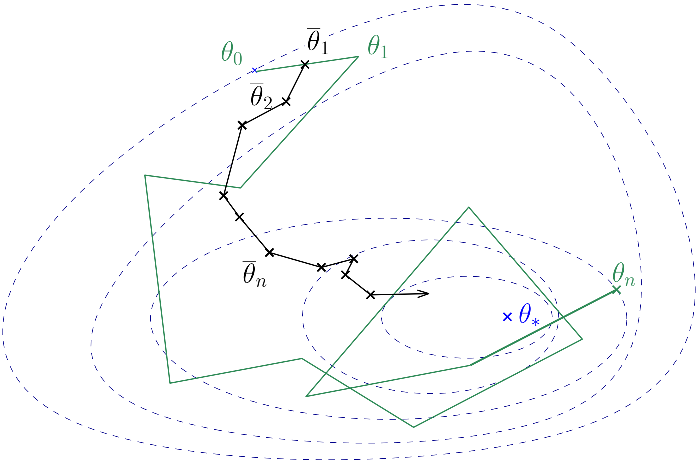
In the quadratic case (linear gradients) $\Sigma \mathbb{E}_{\pi_\gamma}[\theta - \theta_*] = 0$: $\bar{\theta}_\gamma = \theta_*$!

In the general case, Taylor expansion of $\mathcal{R}$, and same reasoning on higher moments of the chain leads to

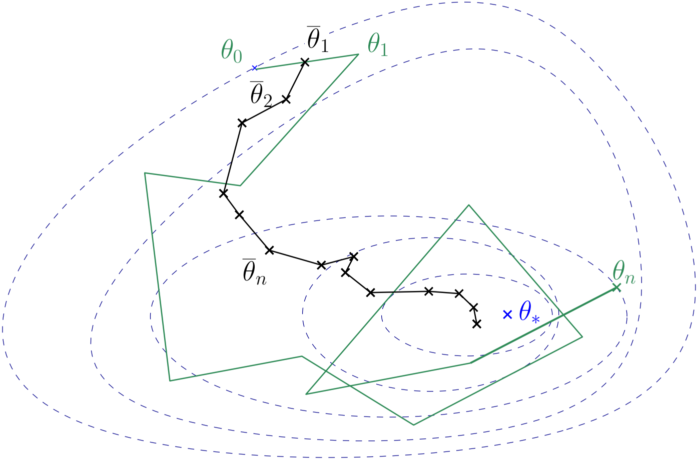
$$\bar{\theta}_\gamma - \theta_* \simeq \gamma \mathcal{R}''(\theta_*)^{-1}\mathcal{R}'''(\theta_*)\Big(\big[\mathcal{R}''(\theta_*) \otimes I + I \otimes \mathcal{R}''(\theta_*)\big]^{-1}\mathbb{E}_\varepsilon[\varepsilon(\theta_*)^{\otimes 2}]\Big)$$

Overall, $\bar{\theta}_\gamma - \theta_* = \gamma\Delta + O(\gamma^2)$.

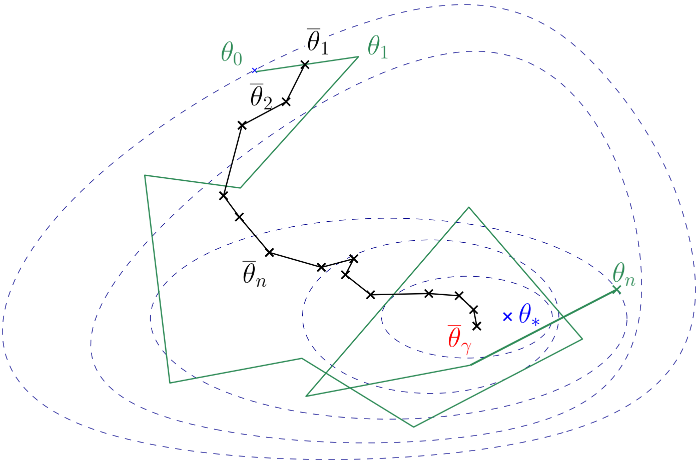
Constant learning rate SGD: convergence in the non-quadratic case



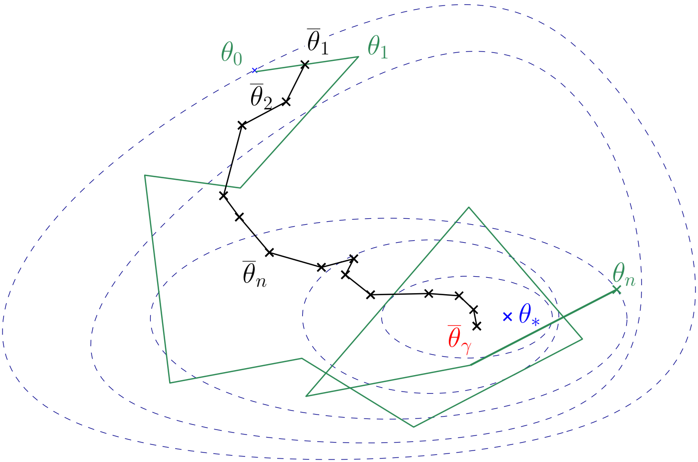
Constant learning rate SGD: convergence in the non-quadratic case



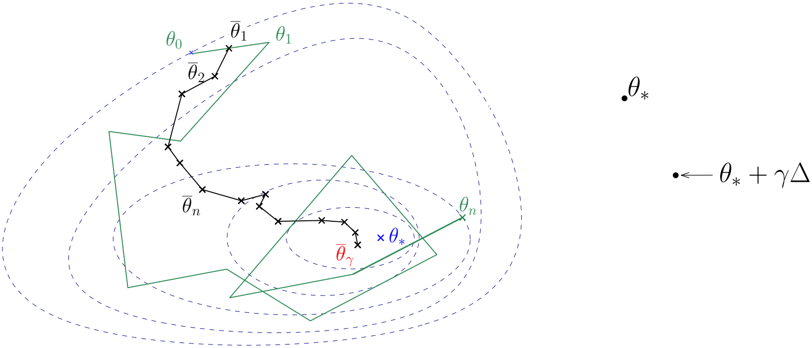
Constant learning rate SGD: convergence in the non-quadratic case



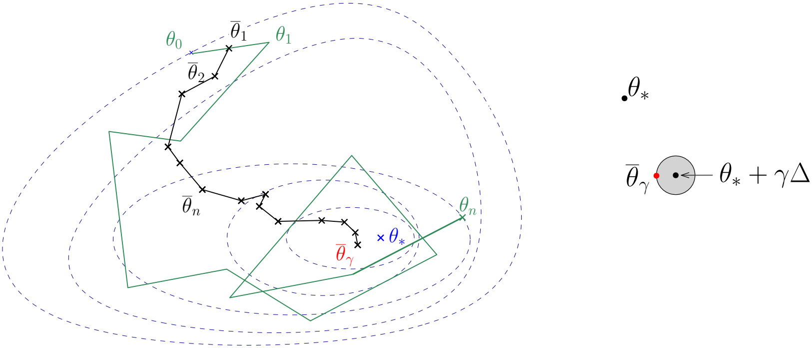
Constant learning rate SGD: convergence in the non-quadratic case



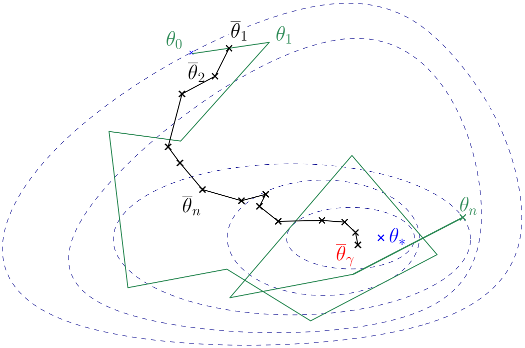
Richardson extrapolation

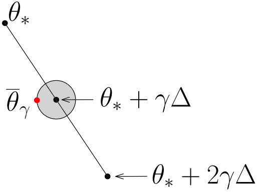


Richardson extrapolation

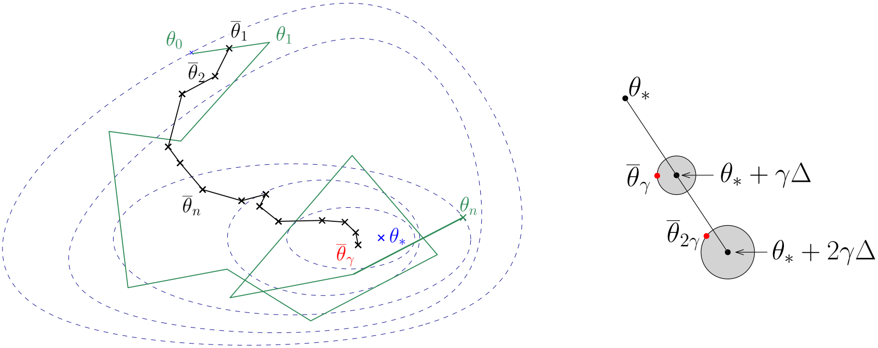


Richardson extrapolation

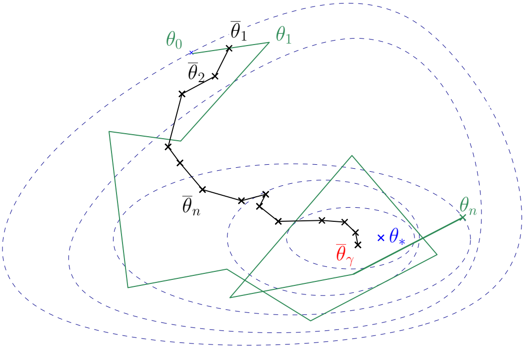


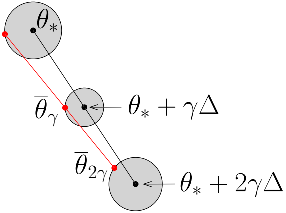


Richardson extrapolation

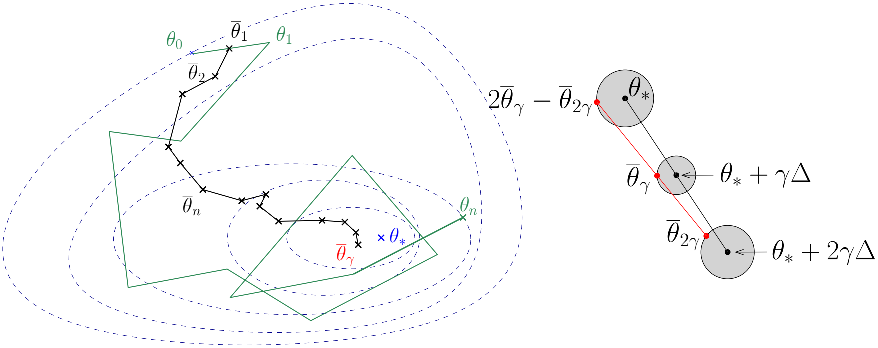


Richardson extrapolation



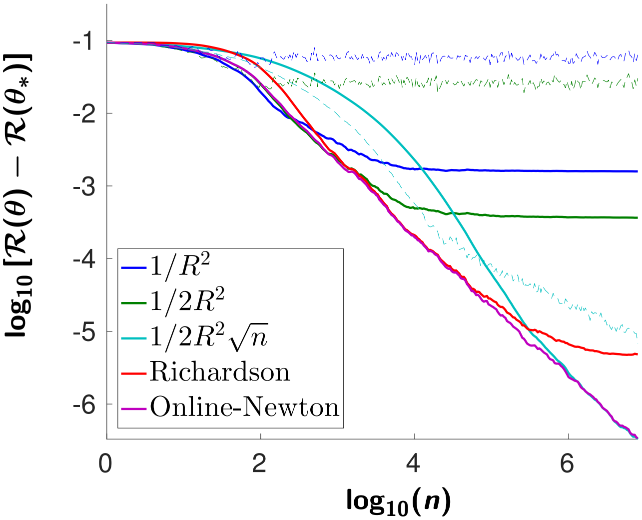


Richardson extrapolation



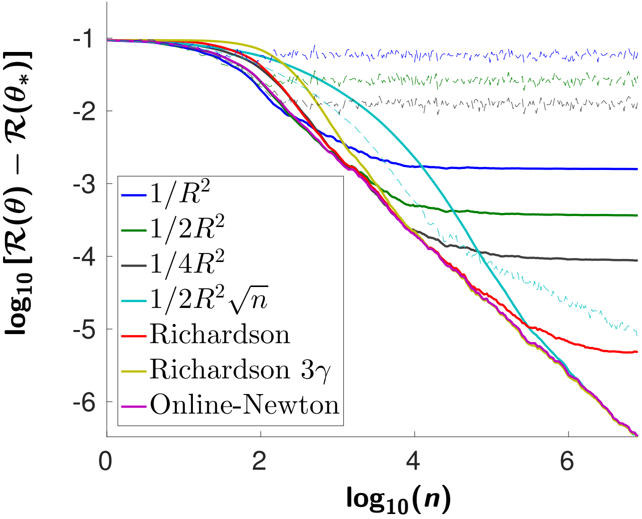
Recovering convergence closer to θ_* by Richardson extrapolation $2\bar{\theta}_n^\gamma - \bar{\theta}_n^{2\gamma}$

Experiments: smaller dimension



Synthetic data, logistic regression, $n = 8.10^6$

Experiments: Double Richardson



Synthetic data, logistic regression, $n = 8.10^6$

"Richardson 3γ": estimator built using Richardson on 3 different sequences: $\tilde{\theta}_n^3 = \frac{8}{3}\bar{\theta}_n^{\gamma} - 2\bar{\theta}_n^{2\gamma} + \frac{1}{3}\bar{\theta}_n^{4\gamma}$

Conclusion MC

Take home

- Asymptotic sometimes matter less than first iterations: consider large step size.
- Constant step size SGD is a homogeneous Markov chain.
- Difference between LS and general smooth loss is intuitive.

For smooth strongly convex loss:

- Convergence in terms of Wasserstein distance.
- Decomposition as three sources of error: variance, initial conditions, and “drift”
- Detailed analysis of the position of the limit point: the direction does not depend on γ at first order $\implies$ Extrapolation tricks can help.

Further references

Many stochastic algorithms not covered in this talk (coordinate descent, online Newton, composite optimization, non convex learning) ...

- **Good introduction: Francis's lecture notes at Orsay**
- **Book:**
  Convex Optimization: Algorithms and Complexity,
  Sébastien Bubeck

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences, 2(1):183–202.

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

Boyd, S. and Vandenberghe, L. (2003). Convex Optimization. Cambridge University Press.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. Foundations and Trends® in Machine Learning, 8(3-4):231–357.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014a). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In Advances in Neural Information Processing Systems, pages 1646–1654.

Defazio, A., Domke, J., and Caetano, T. (2014b). Finito: A faster, permutable incremental gradient method for big data problems. In Proceedings of the 31st international conference on machine learning (ICML-14), pages 1125–1133.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In Advances in neural information processing systems, pages 315–323.

Konečnỳ, J. and Richtárik, P. (2013). Semi-stochastic gradient descent methods. arXiv preprint arXiv:1312.1666.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate o (1/k2). In Soviet Mathematics Doklady, volume 27, pages 372–376.

Nesterov, Y. (2004). Introductory lectures on convex optimization: A basic course. Springer.

Nesterov, Y. (2007). Gradient methods for minimizing composite objective function. Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, Tech. Rep, 76.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. SIAM J. Control Optim., 30(4):838–855.

Robbins, H. and Monro, S. (1951). A stochastic approxiation method. The Annals of mathematical Statistics, 22(3):400–407.

Ruppert, D. (1988). Efficient estimations from a slowly convergent Robbins-Monro process. Technical report, Cornell University Operations Research and Industrial Engineering.

Schmidt, M., Le Roux, N., and Bach, F. (2011). Convergence rates for inexact proximal-gradient method. In Adv. NIPS.

Schmidt, M., Le Roux, N., and Bach, F. (2013). Minimizing finite sums with the stochastic average gradient. Mathematical Programming, 162(1-2):83–112.

Shor, N. Z., Kiwiel, K. C., and Ruszcay?ski, A. (1985). Minimization methods for non-differentiable functions. Springer-Verlag New York, Inc.

Tsybakov, A. B. (2003). Optimal rates of aggregation. In Proceedings of the Annual Conference on Computational Learning Theory.