Master 2 Data Science, Univ. Paris Saclay

Optimization for Data Science

Stéphane Gaïffas, Alexandre Gramfort

Who I am?

- Stéphane Gaïffas
- Associate Professor
- Center for Applied Mathematics
- `http://www.cmap.polytechnique.fr/~gaiffas/`
- `stephane.gaiffas@polytechnique.edu`

Supervised learning

Setting

- Data $a_i \in \mathcal{A}$, $b_i \in \mathcal{B}$ for $i = 1, \ldots, n$
- a_i is the vector of **features** of i, b_i is the **label** of i
- Usually, assume (a_i, b_i) are i.i.d

Featuring step

- Features vectors a_i are obtained from **raw features**
- We assume $\mathcal{A} = \mathbb{R}^d$ (strings are hashed into an integer, pictures are decomposed into a basis, etc.)

Labels

- Labels b_i are scalar numbers. We assume $\mathcal{B} \in \mathbb{R}$
- $\mathcal{B} = \{-1, 1\}$, $\mathcal{B} = \{0, 1\}$ for binary classification
- $\mathcal{B} = \{1, \ldots, K\}$ for multiclass classification
- $\mathcal{B} = \mathbb{R}$ for regression

Aim

- Based on (a_i, b_i), learn a function that predicts b based on a new a (generalization property)

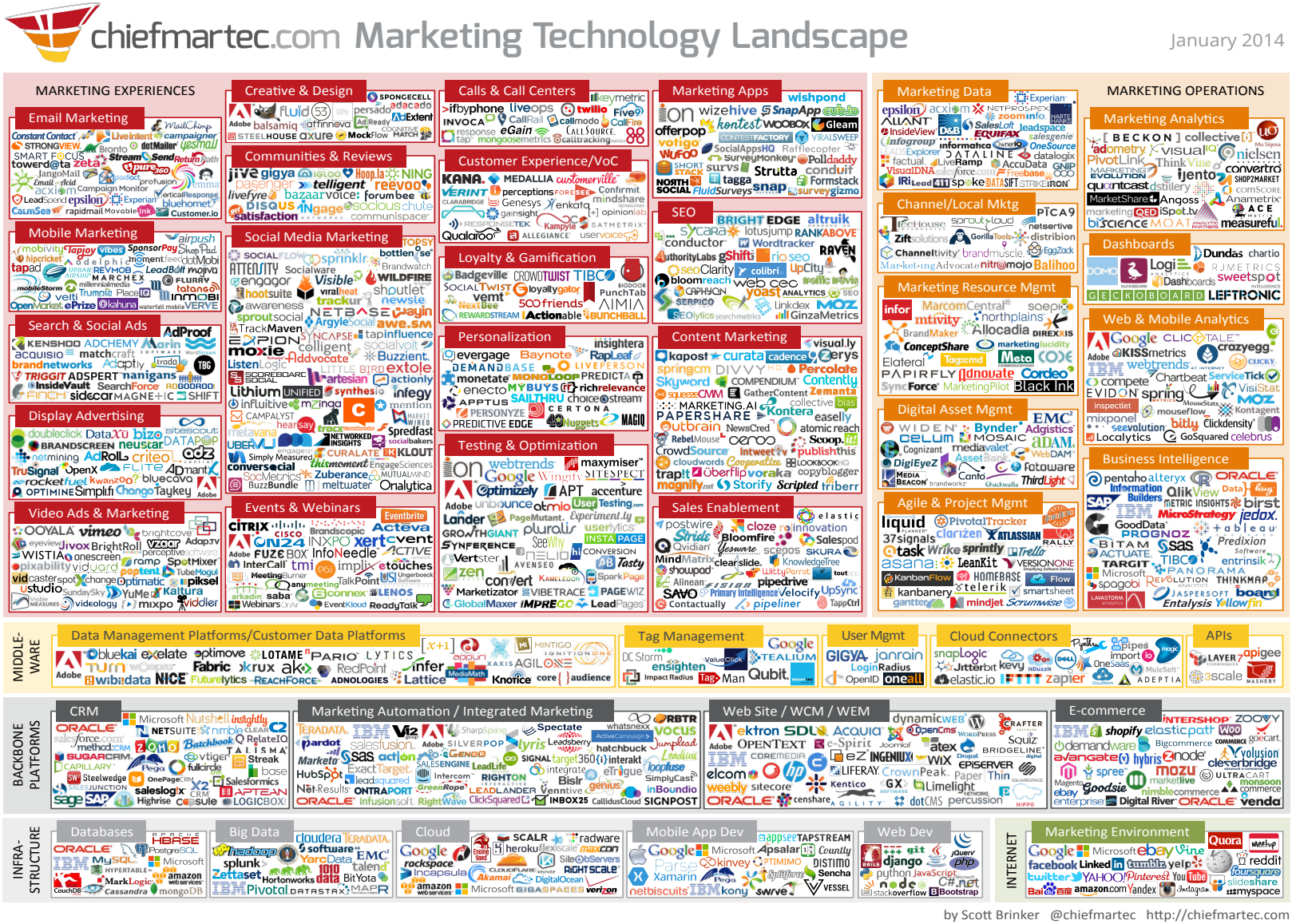
Scaling

- *High-dimension*: d is large, say $d \geq 10^4$
- *Big data*: n is large, say $n \geq 10^6$

Scenarios

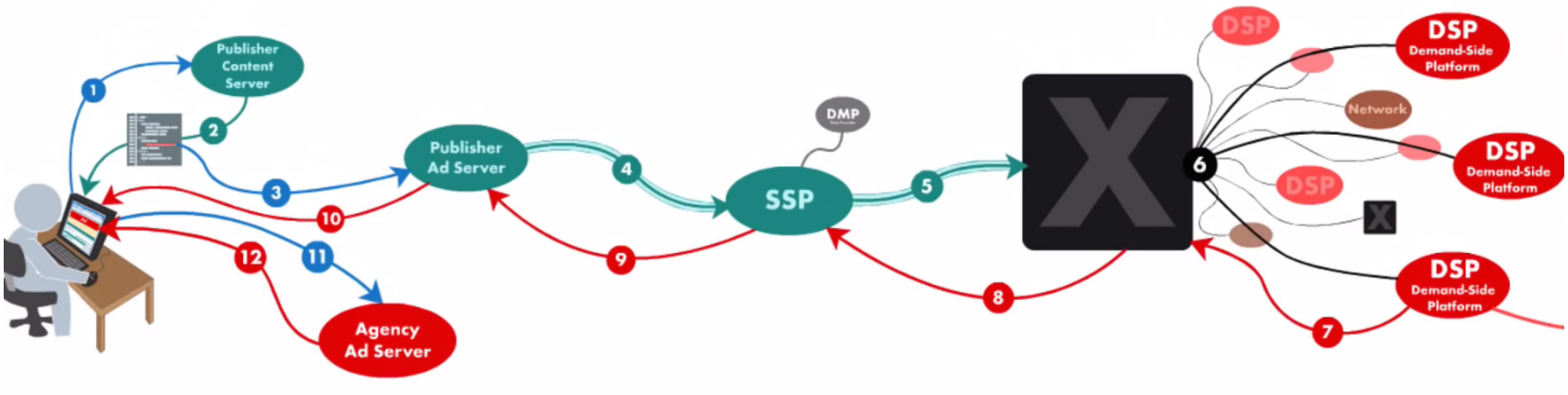
- d is large, n is small: computational biology
- d is small, n is large: marketing
- d is large, n is large: web-advertisement, ad display

Application: marketing



by Scott Brinker @chiefmartec http://chiefmartec.com

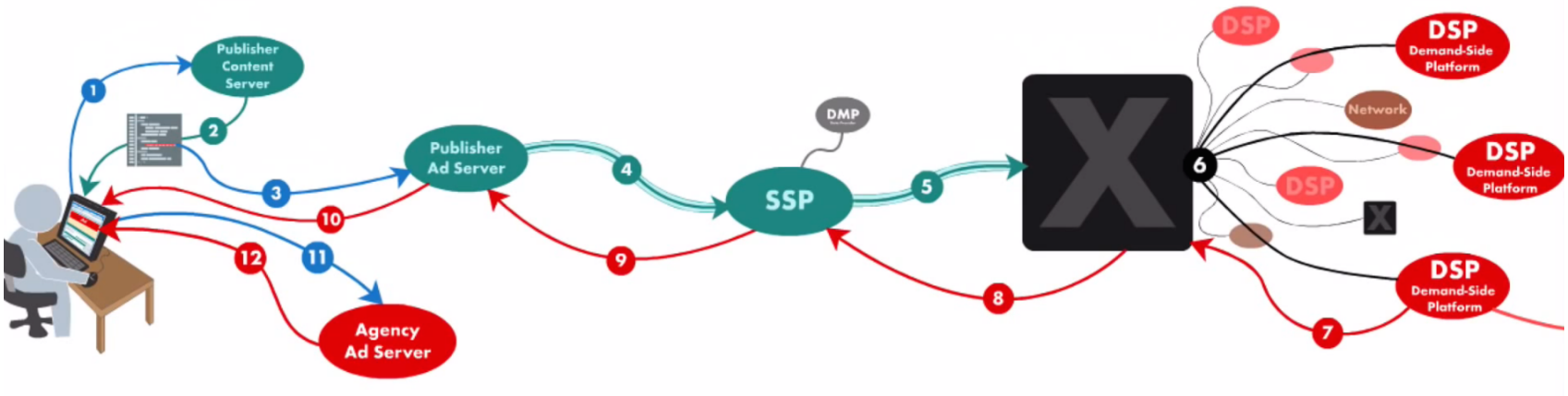
An example: Real Time Bidding



- A **customer** visits a webpage with his browser: a complex process of content selection and delivery begins.
- An **advertiser** might want to display an ad on the webpage where the user is going. The webpage belongs to a **publisher**.
- The publisher sells ad space to advertisers who want to reach customers

In some cases, an auction starts: **RTB** (Real Time Bidding)

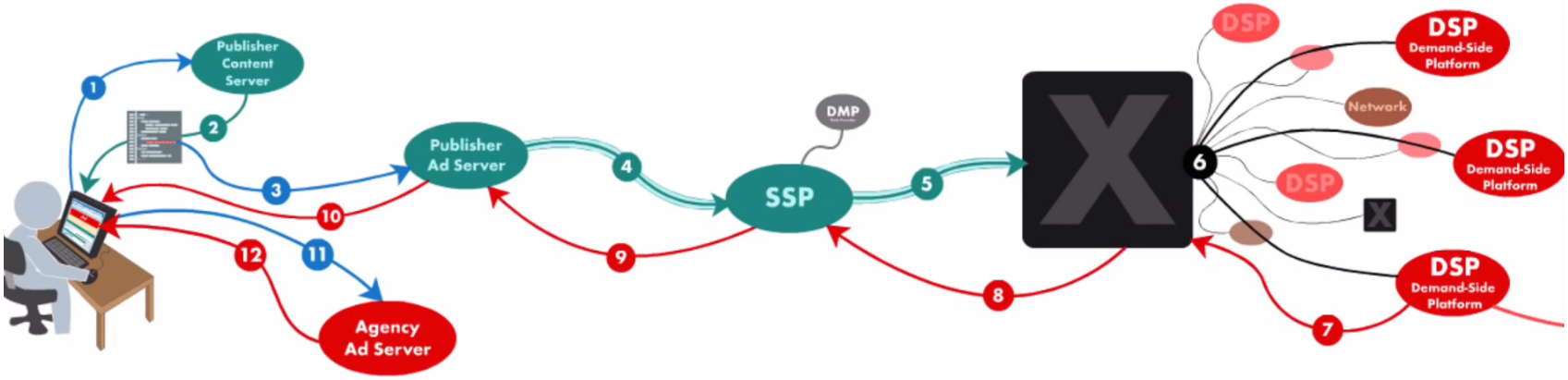
An example: Real Time Bidding



- Advertisers have **10ms** (!) to give a price: they need to assess quickly how willing they are to display the ad to this customer
- Machine learning is used here to predict the probability of click on the ad. Time constraint: few model parameters to answer quickly
- Feature selection / dimension reduction is crucial here

Full process takes < 100ms

An example: Real Time Bidding



Some figures:

- 10 million prediction of click probability per second
- answers within 10ms
- stores 20Terabytes of data daily

Aim

- Based on past data, you want to find users that will click on some ads

This problem can be formulated as a **binary classification problem**

What to do ?

Minimize with respect to $h : \mathbb{R}^d \to \mathbb{R}$

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(b_i, h(a_i))$$

where

- ℓ is a **loss** function: $\ell(b_i, h(a_i))$ small means b_i is close to $h(a_i)$
- $R_n(h)$ is called **goodness-of-fit** or **empirical risk**

Computation of h is called **training** or **estimation**

It requires the use of an **optimization algorithm**

Scaling problem

- n and d are large: training is too time-consuming for a complex function h

A simplification

- Choose a **linear** function h:

$$h(a) = \langle a, x \rangle = a^\top x = \sum_{j=1}^{d} a_j x_j,$$

for a parameter $x \in \mathbb{R}^d$ to be **trained**

Remark

- linear with respect to a_i, but **you** choose the features a_i
- usually not linear w.r.t the raw features: **feature engineering**

- **Least-squares** loss (linear regression): $\ell(b, b') = \frac{1}{2}(b - b')^2$ for $b \in \mathbb{R}$, namely

$$R_n(x) = \frac{1}{2n} \sum_{i=1}^{n} (b_i - \langle a_i, x \rangle)^2 = \frac{1}{2n} \|b - Ax\|_2^2$$

where $A = [a_1, \cdots, a_n]^\top \in \mathbb{R}^{n \times d}$ and $b = [b_1, \cdots, b_n] \in \mathbb{R}^d$

- **Logistic regression** loss (logit, log-linear regression): $\ell(b, b') = \log(1 + e^{-bb'})$ for $b \in \{-1, 1\}$, namely

$$R_n(x) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-b_i \langle a_i, x \rangle})$$

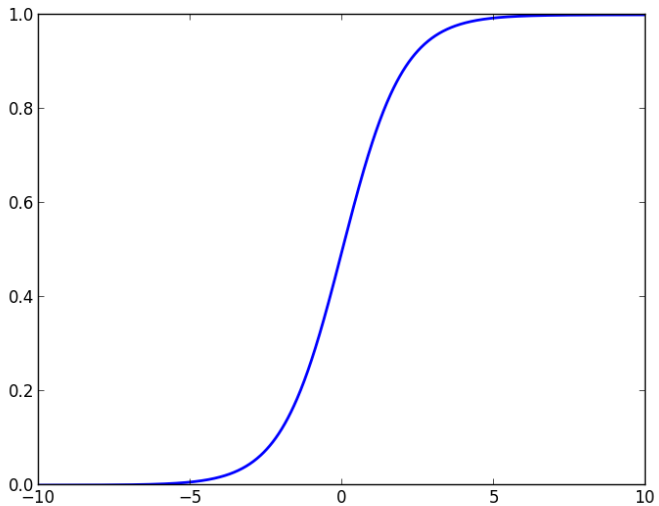
Supervised learning and regularization: loss functions

Binary classification: label $b \in \{0, 1\}$. Assume that

$$\mathbb{P}(b|a) = \text{Bernoulli}(\sigma_x(a))$$

with $\sigma_x(a) = \sigma(\langle x, a \rangle)$ where σ is the **sigmoid** function

$$\sigma(b) = \frac{1}{1 + e^{-b}}.$$



Binary classification: label $b \in \{0, 1\}$. Assume that

$$\mathbb{P}(b|a) = \text{Bernoulli}(\sigma_x(a))$$

with $\sigma_x(a) = \sigma(\langle x, a \rangle)$ where σ is the **sigmoid** function

$$\sigma(b) = \frac{1}{1 + e^{-b}}.$$

Hence for $b \in \{0, 1\}$:

$$\mathbb{P}(b|a) = \sigma_x(a)^b (1 - \sigma_x(a))^{1-b} = \sigma_x(a)^b \sigma_x(-a)^{1-b}$$

and the log-likelihood is given by (if we replace label 0 by -1 for convenience)

$$\sum_{i=1}^{n} \log \mathbb{P}[b_i|a_i] = -\sum_{i=1}^{n} \log(1 + e^{-b_i \langle a_i, x \rangle})$$

Goodness of fit $=-$ log-likelihood, so this leads to

$$R_n(x) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-b_i \langle a_i, x\rangle})$$

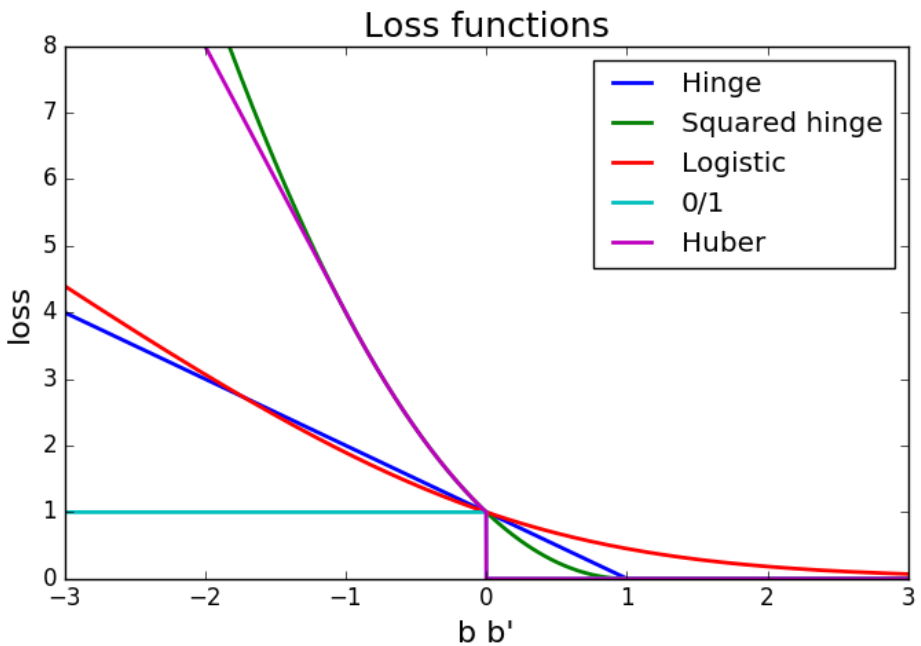
Remark: this parametrization using the sigmoid is equivalent to assuming that

$$\log\left(\frac{\mathbb{P}[b=1|a]}{\mathbb{P}[b=0|a]}\right) = \langle a, x\rangle$$

This leads to a **linear** separation between the 1s and -1s. Logistic regression is a **linear** classifier

Other classical loss functions for binary classication

- 0/1 loss, $\ell(b, b) = \mathbf{1}_{bb' \leq 0}$
- Hinge loss (SVM), $\ell(b, b) = (1 - bb')_+$
- Quadratic hinge loss (SVM), $\ell(b, b) = \frac{1}{2}(1 - bb')_+^2$
- Huber loss $\ell(b, b') = -4bb'\mathbf{1}_{bb' < -1} + (1 - bb')_+^2 \mathbf{1}_{bb' \geq -1}$



Penalization

Minimizing only

$$\hat{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(b_i, \langle a_i, x \rangle)$$

is generally a bad idea. Minimize instead

$$\hat{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(b_i, \langle a_i, x \rangle) + \lambda \operatorname{pen}(x) \right\}$$

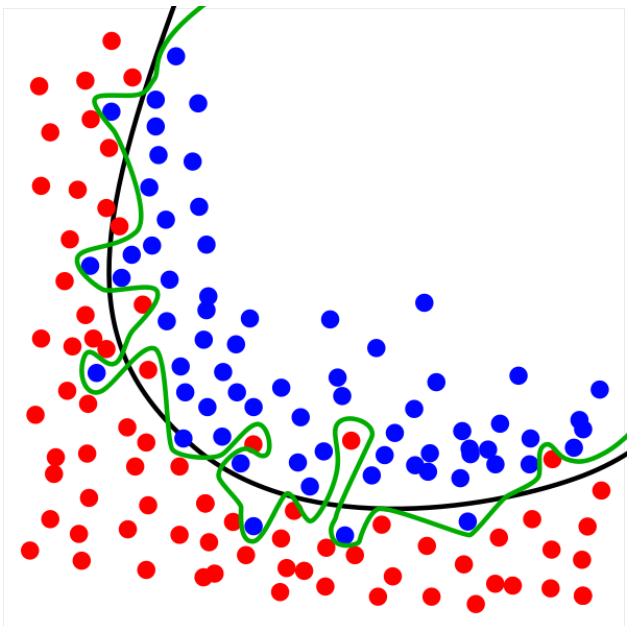
where

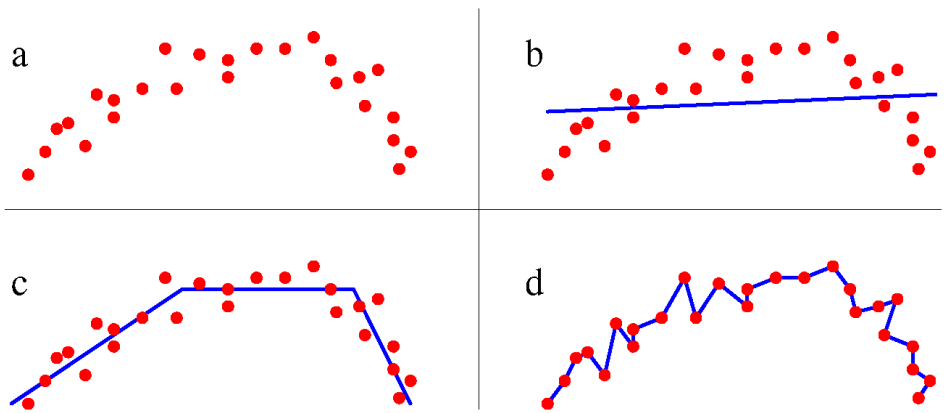
- pen is a **penalization** function, it forbids x to be "too complex"
- $\lambda > 0$ is a **tuning** or **smoothing** parameter, that **balances** goodness-of-fit and penalization

Supervised learning and regularization: penalization

$$\hat{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(b_i, \langle a_i, x \rangle) + \lambda \, \mathrm{pen}(x) \right\}$$

Penalization, for a well-chosen $\lambda > 0$, allows to avoid **overfitting**





What penalization?

- **Ridge**: $\text{pen}(x) = \frac{1}{2}\|x\|_2^2$
- This penalizes the energy of x

But, it would be nice if x contained **zeros**, and many of them: only **few** features explain the label. Simpler model, with a reduced dimension. How to do it?

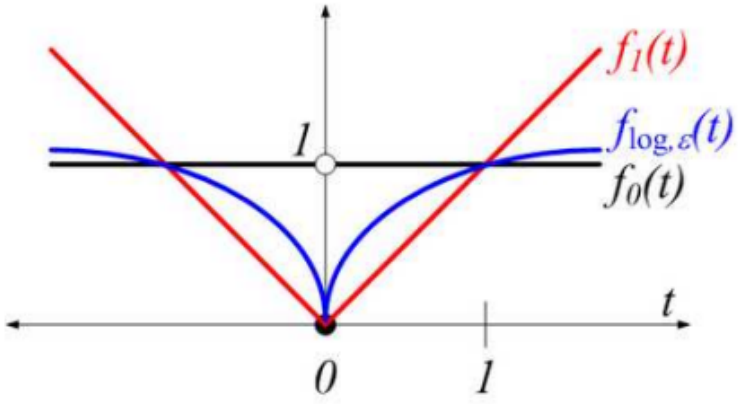
$$\hat{x} \in \underset{x \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(b_i, \langle a_i, x \rangle) + \lambda \|x\|_0 \right\},$$

where

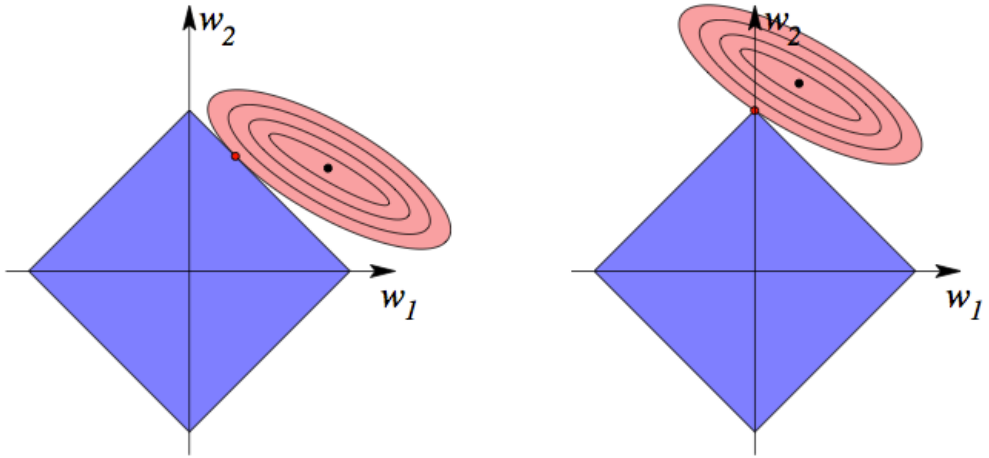
$$\|x\|_0 = \#\{j : x_j \neq 0\}.$$

To solve this, explore **all** possible supports of x. Too long!
[Link with model-selection, AIC and BIC]

Find a convex proxy of $\|\cdot\|_0$: the ℓ_1-**norm** $\|x\|_1 = \sum_{j=1}^{d} |x_j|$



Why it induces sparsity?



Why ℓ_2 (ridge) does not induce sparsity?

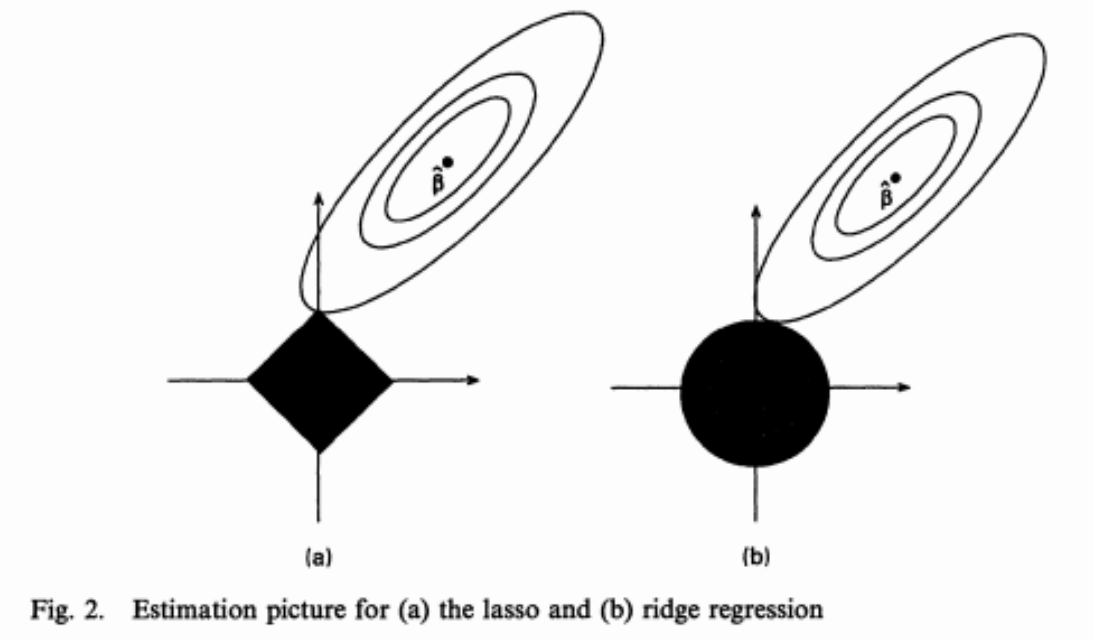


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

Consider the minimization problem

$$\min_{x \in \mathbb{R}} \frac{1}{2}(x-y)^2 + \lambda |x|$$

for $\lambda > 0$ and $y \in \mathbb{R}$

- Derivative at 0_+: $d_+ = \lambda - y$
- Derivative at 0_-: $d_- = -\lambda - y$

Let x_* be the solution

- $x_* = 0$ iff $d_+ \geq 0$ and $d_- \leq 0$, namely $|y| \leq \lambda$
- $x_* \geq 0$ iff $d_+ \leq 0$, namely $y \geq \lambda$ and $x_* = y - \lambda$
- $x_* \leq 0$ iff $d_- \geq 0$, namely $y \leq -\lambda$ and $x_* = y + \lambda$

Hence

$$x_* = \text{sign}(y)(|y| - \lambda)_+.$$

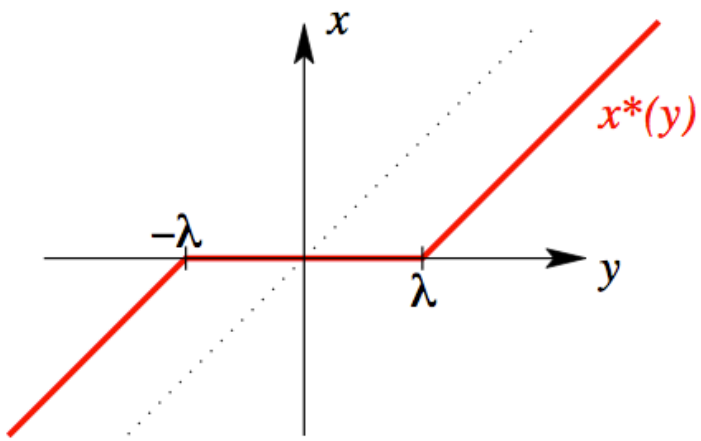
And as a consequence, we have

$$x_* = \underset{x \in \mathbb{R}^d}{\text{argmin}} \frac{1}{2}\|x - y\|_2^2 + \lambda \|x\|_1 = S_\lambda(y)$$

where

$$S_\lambda(y) = \text{sign}(y) \odot (|y| - \lambda)+$$

is the **soft-thresholding** operator



For the least squares loss

$$\hat{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ \frac{1}{2n} \|b - Ax\|_2^2 + \frac{\lambda}{2} \|x\|_2^2 \right\}$$

is called **ridge** regression, and

$$\hat{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ \frac{1}{2n} \|b - Ax\|_2^2 + \lambda \|x\|_1 \right\}$$

is called **Lasso** (least absolute shrinkage and selection operator).

Supervised learning and regularization: proximal operator

- We say that the soft-thresholding is the **proximal operator** of the ℓ_1-norm
- For any $g : \mathbb{R}^d \to \mathbb{R}$ convex, and any $y \in \mathbb{R}^d$, we define

$$\text{prox}_g(y) = \underset{x \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2}\|x - y\|_2^2 + g(x) \right\}$$

- Hence

$$\text{prox}_{\lambda\|\cdot\|_1}(y) = S_\lambda(y)$$

- $g(x) = c$ for a constant c, $\text{prox}_g = Id$
- If C convex set, and

$$g(x) = \delta_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

then

$$\text{prox}_g = \text{proj}_C = \text{projection onto } C.$$

- If $g(x) = \langle b, x \rangle + c$, then

$$\text{prox}_{\lambda g}(x) = x - \lambda b$$

- If $g(x) = \frac{1}{2} x^\top A x + \langle b, x \rangle + c$ with A symetric positive, then

$$\text{prox}_{\lambda g}(x) = (I + \lambda A)^{-1}(x - \lambda b)$$

- If $g(x) = \frac{1}{2}\|x\|_2^2$ then

$$\text{prox}_{\lambda g}(x) = \frac{1}{1+\lambda} x = \text{shrinkage operator}$$

- If $g(x) = -\log x$ then

$$\text{prox}_{\lambda g}(x) = \frac{x + \sqrt{x^2 + 4\lambda}}{2}$$

- If $g(x) = \|x\|_2$ then

$$\text{prox}_{\lambda g}(x) = \left(1 - \frac{\lambda}{\|x\|_2}\right)_+ x,$$

the block soft-thresholding operator

- If $g(x) = \|x\|_1 + \frac{\gamma}{2}\|x\|_2^2$ (elastic-net) where $\gamma > 0$, then

$$\text{prox}_{\lambda g}(x) = \frac{1}{1+\lambda\gamma}\,\text{prox}_{\lambda\|\cdot\|_1}(x)$$

  [Elastic-net and features correlation]

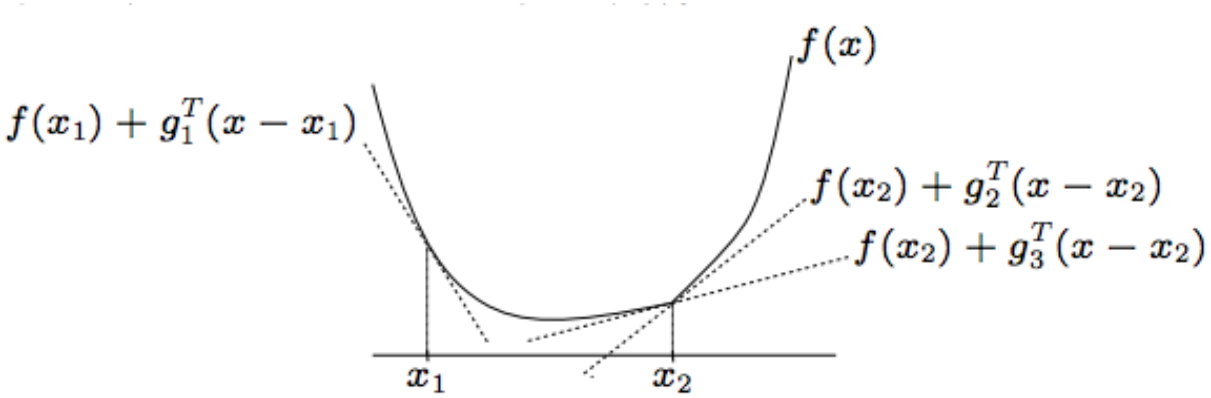
- If $g(x) = \sum_{g\in G}\|x_g\|_2$ where G partition of $\{1,\dots,d\}$,

$$(\text{prox}_{\lambda g}(x))_g = \left(1 - \frac{\lambda}{\|x_g\|_2}\right)_+ x_g,$$

  for $g \in G$. Block soft-thresholding, used for group-Lasso

The **subdifferential** of a convex f on $\mathbb{R}^d$ at x is the set

$$\partial f(x) = \left\{ g \in \mathbb{R}^d : f(y) \geq \langle g, y - x \rangle + f(x) \text{ for all } y \in \mathbb{R}^d \right\}$$



- Each element is called a **subgradient**
- **Optimality criterion**: for any convex function f on $\mathbb{R}^d$, a point $x \in \mathbb{R}^d$ is a global minizer of f iff $0 \in \partial f(x)$
- If f is differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$
- Example: $\partial |0| = [-1, 1]$

We will say that:

- f is L-**Lipschitz** if

$$|f(x) - f(y)| \leq L\|x - y\|_2 \quad \text{for any } x, y \in \mathbb{R}^d$$

- f is L-**smooth** if it is continuously differentiable and if

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \quad \text{for any } x, y \in \mathbb{R}^d.$$

If f is twice differentiable, this is equivalent to assuming

$$\nabla^2 f(x) \preceq L\boldsymbol{I}$$

- f is μ-strongly convex if

$$f(\cdot) - \frac{\mu}{2}\|\cdot\|_2^2$$

is convex. When f if differentiable, it is equivalent to

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|_2^2$$

for any $x, y \in \mathbb{R}^d$. When f is twice differentiable, this is equivalent to

$$\nabla^2 f(x) \succeq \mu \boldsymbol{I}.$$

When f is L-smooth, μ-strongly convex and twice differentiable, then

$$\mu \boldsymbol{I} \preceq \nabla^2 f(x) \preceq L \boldsymbol{I}$$

for any $x \in \mathbb{R}^d$. We define in this case

$$\kappa = \frac{L}{\mu} \geq 1$$

as the condition number of f.

We will see that such a function f can be minimized very efficiently (**linear** convergence rate)

An optimization problem

How to solve

$$\hat{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \Big\{ R_n(x) + \lambda \, \mathrm{pen}(x) \Big\} \quad ???$$

Put for short

$$f(x) = R_n(x) \;\text{ and }\; g(x) = \lambda \, \mathrm{pen}(x).$$

In what follows, we assume that:

- f is convex and L-smooth
- g is convex and continuous, but possibly non-smooth (for instance ℓ_1 penalization)
- g is prox-capable, namely it is computationaly not hard to compute its proximal operator

An optimization problem

Smoothness of f:

- For the least-squares: $\nabla R_n(x) = \frac{1}{n} A^\top (Ax - b)$, so $L = \|A^\top A\|_{\mathrm{op}}/n$ works
- For logistic regression, $L = \|A^\top A\|_{\mathrm{op}}/(4n)$ works

Prox-capability of g:

- we gave the explicit prox for many penalization functions above!

Optimality criterion: $x_* \in \mathrm{argmin}_{x \in \mathbb{R}^d} \{f(x) + g(x)\}$ iff

$$-\nabla f(x_*) \in \partial g(x_*)$$

namely

$$-\frac{1}{\lambda} \nabla R_n(x_*) \in \partial g(x_*)$$

An optimization problem

For the Lasso

$$\hat{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ \frac{1}{2n} \|b - Ax\|_2^2 + \lambda \|x\|_1 \right\}$$

this optimality criterion is

$$\begin{cases} \frac{1}{n} |\langle a^j, b - A\hat{x} \rangle| \leq \lambda & \text{if } \hat{x}_j = 0 \\ \frac{1}{n} \langle a^j, b - A\hat{x} \rangle = \lambda \operatorname{sign}(\hat{x}_j) & \text{if } \hat{x}_j \neq 0 \end{cases}$$

for any $j = 1, \ldots, d$, where a^j is the j-th conumn of A.

Ok. Now how do I minimize $f + g$?

- Key point: the **descent lemma**. If f is L-smooth, then for any $L' \geq L$:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L'}{2} \|x - y\|_2^2$$

  for any $x, y \in \mathbb{R}^d$

- Imagine we are at iteration k, the current point is x_k. I use the descent lemma:

$$f(x) \leq f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L'}{2} \|x - x_k\|_2^2.$$

Gradient descent

- Remark that

$$\operatorname*{argmin}_{x\in\mathbb{R}^d}\left\{f(x_k)+\langle\nabla f(x_k),x-x_k\rangle+\frac{L'}{2}\|x-x_k\|_2^2\right\}$$
$$=\operatorname*{argmin}_{x\in\mathbb{R}^d}\left\|x-\left(x_k-\frac{1}{L'}\nabla f(x_k)\right)\right\|_2^2$$

  Hence, choose

$$x_{k+1}=x_k-\frac{1}{L'}\nabla f(x_k)$$

  This is the basic **gradient descent** algorithm

- But where is gone g?

Let's put back g:

$$f(x) + g(x) \leq f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2}\|x - x_k\|_2^2 + g(x)$$

and again

$$\begin{aligned}
&\operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2}\|x - x_k\|_2^2 + g(x) \right\} \\
&\quad = \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ \frac{L}{2} \left\| x - \left(x_k - \frac{1}{L} \nabla f(x_k) \right) \right\|_2^2 + g(x) \right\} \\
&\quad = \operatorname{prox}_{g/L} \left(x_k - \frac{1}{L} \nabla f(x_k) \right)
\end{aligned}$$

Proximal gradient descent algorithm [also called ISTA]

- **Input**: starting point x_0, Lipschitz constant $L > 0$ for ∇f
- For $k = 1, 2, \ldots$ until *converged* do
  - $x_k = \text{prox}_{g/L}\left(x_{k-1} - \frac{1}{L}\nabla f(x_{k-1})\right)$
- **Return** last x_k

For Lasso, the iteration is

$$x_k = S_{\lambda/L}\left(x_{k-1} - \frac{1}{Ln}(A^\top A x_{k-1} - A^\top b)\right),$$

where S_λ is the soft-thresholding operator

A theoretical guarantee

- Put for short $F = f + g$,
- Take any $x_* \in \operatorname{argmin}_{x \in \mathbb{R}^d} F(x)$

Theorem (Beck Teboulle (2009))

If the sequence $\{x_k\}$ is generated by the proximal gradient descent algorithm, then

$$F(x_k) - F(x_*) \leq \frac{L\|x_0 - x_*\|_2^2}{2k}$$

- Convergence rate is $O(1/k)$
- Is it possible to improve the $O(1/k)$ rate?

Yes! Using **accelerated proximal gradient descent** (also called FISTA) [Beck Teboule (2009)]

- Idea: to find x_{k+1}, use an interpolation between x_k and x_{k-1}

Accelerated proximal gradient descent algorithm

- **Input**: starting points $z_1 = x_0$, Lipschitz constant $L > 0$ for ∇f, $t_1 = 1$
- For $k = 1, 2, \ldots$ until *converged* do
  - $x_k = \text{prox}_{g/L}(z_k - \frac{1}{L}\nabla f(z_k))$
  - $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
  - $z_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1})$
- **Return** last x_k

Theorem (Beck Teboulle (2009))

If the sequence $\{x_k\}$ is generated by the accelerated proximal gradient descent algorithm, then

$$F(x_k) - F(x_*) \leq \frac{2L\|x_0 - x_*\|_2^2}{(k+1)^2}$$

- Convergence rate is $O(1/k^2)$
- Is $O(1/k^2)$ the optimal rate?

Yes. Put $g = 0$

Theorem (Nesterov)

For any optimization procedure satisfying

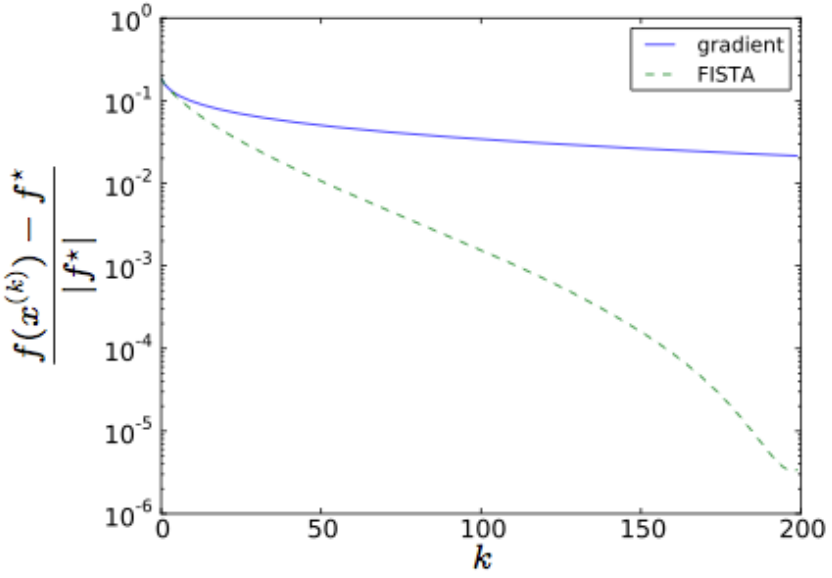
$$x_{k+1} \in x_1 + \operatorname{span}(\nabla f(x_1), \ldots, \nabla f(x_k)),$$

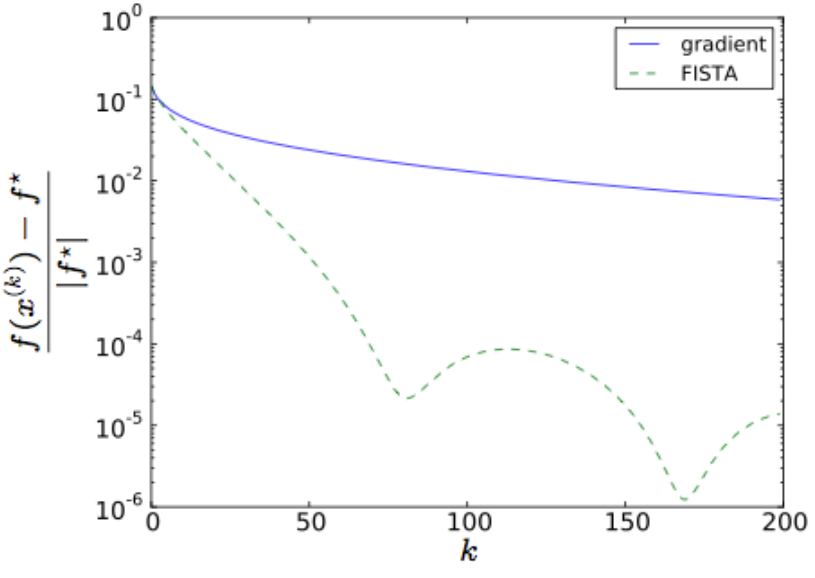
there is a function f on $\mathbb{R}^d$ convex and L-smooth such that

$$\min_{1 \le j \le k} f(x_j) - f(x_*) \ge \frac{3L}{32} \frac{\|x_1 - x^*\|_2^2}{(k+1)^2}$$

for any $1 \le k \le (d-1)/2$.

Comparison of ISTA and FISTA





Remark: FISTA is **not** a descent algorithm, while ISTA is