# Machine Learning 2 – MAP569

Stéphane Gaïffas

ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Today (and next lecture) is about **optimization for machine learning**. We will learn about the main pillars:

- Proximal gradient descent and acceleration
- Coordinate descent, coordinate gradient descent
- Quasi-newton
- Stochastic gradient descent and beyond

We have seen a lot of problems of the form

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + g(w)$$

with $f$ a goodness-of-fit function

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle w, x_i \rangle)$$

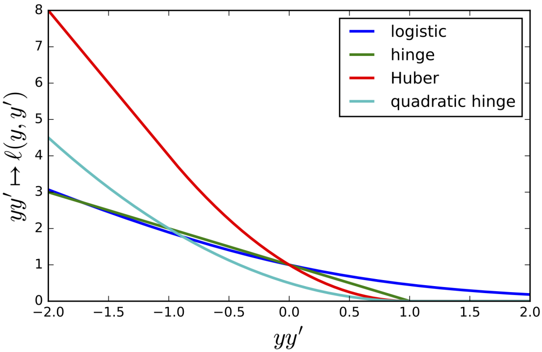
where $\ell$ is some loss and

$$g(w) = \frac{1}{C} \operatorname{pen}(w)$$

where $\operatorname{pen}(\cdot)$ is some penalization function, examples being $\operatorname{pen}(w) = \frac{1}{2}\|w\|_2^2$ (ridge) and $\operatorname{pen}(w) = \|w\|_1$ (Lasso)

## Example of losses for classification

- Logistic loss, $\ell(y, y') = \log(1 + e^{-yy'})$
- Hinge loss, $\ell(y, y') = (1 - yy')_+$
- Quadratic hinge loss, $\ell(y, y') = \frac{1}{2}(1 - yy')_+^2$
- Huber loss $\ell(y, y') = -4yy'\mathbf{1}_{yy'<-1} + (1 - yy')_+^2\mathbf{1}_{yy'\geq -1}$



Minimization of

$$F(w) = f(w) + g(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \langle x_i, w\rangle) + \frac{1}{C}\,\mathrm{pen}(w)$$

First, note that the gradient and Hessian matrix writes

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell'(y_i, \langle x_i, w\rangle) x_i$$

$$\nabla^2 f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell''(y_i, \langle x_i, w\rangle) x_i x_i^\top$$

with

$$\ell'(y, y') = \frac{\partial \ell'(y, y')}{\partial y'} \quad \text{and} \quad \ell''(y, y') = \frac{\partial^2 \ell'(y, y')}{\partial y'^2}$$

And note that $f$ is convex iff

$$y' \mapsto \ell(y_i, y')$$

is for any $i = 1, \ldots, n$.

**Definition.** We say that $f$ is $L$-**smooth** if it is continuously differentiable and if

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

If $f$ is twice differentiable, this is equivalent to assuming

$$\lambda_{\max}(\nabla^2 f(w)) \leq L \quad \text{for any } w \in \mathbb{R}^d$$

(largest eigenvalue of the Hessian matrix of $f$ is smaller than $L$)

For the least-squares loss

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n}(\langle x_i, w\rangle - y_i)x_i, \quad \nabla^2 f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$$

so that

$$L = \frac{1}{n}\lambda_{\max}\Big(\sum_{i=1}^{n} x_i x_i^\top\Big)$$

For the logit loss

$$\nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} y_i(\sigma(y_i\langle x_i, w\rangle) - 1)x_i$$

and

$$\nabla^2 f(w) = \frac{1}{n} \sum_{i=1}^{n} \sigma(y_i\langle x_i, w\rangle)(1 - \sigma(y_i\langle x_i, w\rangle))x_i x_i^\top$$

so that

$$L = \frac{1}{4n}\lambda_{\max}\Big(\sum_{i=1}^{n} x_i x_i^\top\Big)$$

**Gradient descent**. Now how to find

$$w^* \in \operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) \quad ?$$

A **key** point: the descent lemma. If $f$ is $L$-smooth, then

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

for any $w, w' \in \mathbb{R}^d$

**Proof.** Use the fact that

$$\begin{aligned} f(w') &= f(w) + \int_0^1 \langle \nabla f(w + t(w' - w)), w' - w \rangle dt \\ &= f(w) + \langle \nabla f(w), w' - w \rangle \\ &\quad + \int_0^1 \langle \nabla f(w + t(w' - w)) - \nabla f(w), w' - w \rangle dt \end{aligned}$$

So that

$$
\begin{aligned}
|f(w') - f(w) - \langle \nabla f(w), w' - w\rangle| \\
&\leq \int_0^1 |\langle \nabla f(w + t(w' - w)) - \nabla f(w), w' - w\rangle dt| \\
&\leq \int_0^1 \|\nabla f(w + t(w' - w)) - \nabla f(w)\| \|w' - w\| dt \\
&\leq \int_0^1 Lt \|w' - w\|^2 dt = \frac{L}{2}\|w' - w\|^2
\end{aligned}
$$

which proves the descent lemma. $\square$

It leads, around a point $w^k$ (where $k$ is an iteration counter) to

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

for any $w \in \mathbb{R}^d$

Remark that

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ f(w^k) + \langle \nabla f(w^k), w - w^k \rangle + \frac{L}{2} \|w - w^k\|_2^2 \right\}$$
$$= \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\| w - \left( w^k - \frac{1}{L} \nabla f(w^k) \right) \right\|_2^2$$

Hence, it is natural to choose

$$w^{k+1} = w^k - \frac{1}{L} \nabla f(w^k)$$

This is the basic **gradient descent** algorithm

But... where $g$ is gone?

**Proximal Gradient descent.** Let's put back $g$:

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

and again

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

**Proximal operator.** For any $g : \mathbb{R}^d \to \mathbb{R}$ convex, and any $w \in \mathbb{R}^d$, we define

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

We proved already that if $g(w) = \lambda\|w\|_1$ then

$$\mathrm{prox}_g(w) = S_\lambda(w) = \mathrm{sign}(w) \odot (|w| - \lambda)_+$$

(soft-thresholding) and that if $g(w) = \frac{\lambda}{2}\|w\|_2^2$ then

$$\mathrm{prox}_g(w) = \frac{1}{1+\lambda} w$$

(shrinkage)

**Proximal gradient descent** (GD)

- **Input**: starting point $w^0$, Lipschitz constant $L > 0$ for $\nabla f$
- For $k = 1, 2, \ldots$ until *convergence* do

$$w^k \leftarrow \mathrm{prox}_{g/L}\left(w^{k-1} - \frac{1}{L}\nabla f(w^{k-1})\right)$$

- **Return** last $w^k$

For Lasso

$$w^* \in \operatorname*{argmin}_{w \in \mathbb{R}^d} \left\{ \frac{1}{2n}\|y - Xw\|_2^2 + \lambda\|w\|_1 \right\},$$

the iteration is

$$w^k \leftarrow S_{\lambda/L}\left(w^{k-1} - \frac{1}{Ln}X^\top(Xw^{k-1} - y)\right),$$

where $S_\lambda$ is the soft-thresholding operator

## A theoretical guarantee

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

**Theorem.** If the sequence $\{w^k\}$ is generated by the proximal gradient descent algorithm, then if $f$ is $L$-smooth then

$$F(w^k) - F(w^*) \le \frac{L\|w^0 - w^*\|_2^2}{2k}$$

## Comments

- Convergence rate is $O(1/k)$
- $\varepsilon$-accuracy (namely $F(w^k) - F(w^*) \le \varepsilon$) achieved after $O(L/\varepsilon)$ iterations
- Is it possible to improve the $O(1/k)$ rate? It's very slow!
- Improving this rate **a lot** requires an extra assumption: **strong convexity**

$f$ is $\mu$-**strongly convex** if

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

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

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

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

$$\lambda_{\min}(\nabla^2 f(w)) \geq \mu$$

for any $w \in \mathbb{R}^d$ (smallest eigenvalue of $\nabla^2 f(w)$)

When $f$ is $L$-smooth, $\mu$-strongly convex and twice differentiable, then

$$\mu \leq \lambda_{\min}(\nabla^2 f(w)) \leq \lambda_{\max}(\nabla^2 f(w)) \leq L$$

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

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

as the **condition number** of $f$.

**Theorem.** If the sequence $\{w^k\}$ is generated by the proximal gradient descent algorithm, and if $f$ is $L$-smooth and $\mu$-strongly convex, we have

$$F(w^k) - F(w^*) \leq \frac{L}{2} \exp\Big( - \frac{4k}{\kappa + 1} \Big) \|w^0 - w^*\|$$

where $\kappa = L/\mu$ is the condition number of $f$.

## Comments

- Convergence rate is $O(e^{-ck})$
- $\varepsilon$-accuracy achieved after $O(\kappa \log(1/\varepsilon))$ iterations

**Acceleration.** Can we improve the number of iterations $O(L/\varepsilon)$ ($L$-smooth) and $O(\frac{L}{\mu}\log(1/\varepsilon))$ ($L$-smooth and $\mu$ strongly-convex) ?

Yes: the idea is to combine $w^k$ and $w^{k-1}$ to find $w^{k+1}$

**Accelerated Proximal Gradient Descent** (AGD)

- **Input**: starting points $z^1 = w^0$, Lipschitz constant $L > 0$ for $\nabla f$, $t_1 = 1$
- For $k = 1, 2, \ldots$ until *converged* do

$$
\begin{aligned}
w^k &\leftarrow \mathrm{prox}_{g/L}(z^k - \frac{1}{L}\nabla f(z^k)) \\
t_{k+1} &\leftarrow \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\
z^{k+1} &\leftarrow w^k + \frac{t_k - 1}{t_{k+1}}(w^k - w^{k-1})
\end{aligned}
$$

- **Return** last $w^k$

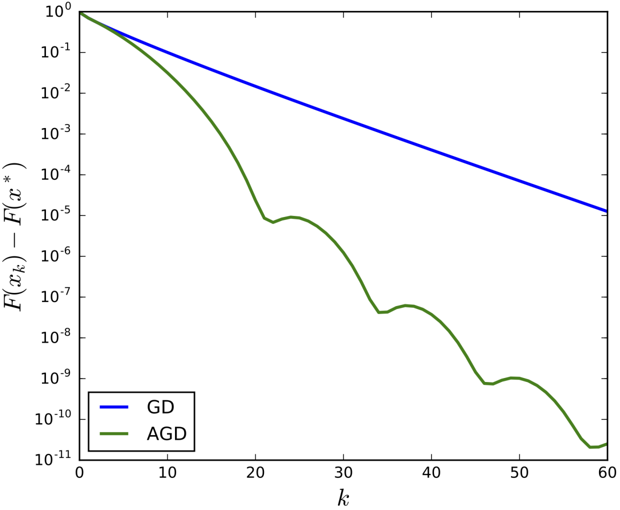
**Theorem.** Accelerated proximal gradient descent needs

$$O(L/\sqrt{\varepsilon}) \text{ iterations to achieve } \varepsilon\text{-precision}$$

in the $L$-smooth case and

$$O\left(\sqrt{\frac{L}{\mu}}\log(1/\varepsilon)\right) \text{ iterations to achieve } \varepsilon\text{-precision}$$

in the $L$-smooth and $\mu$-strongly convex case




**Remark**. AGD is **not** a descent algorithm, while GD is

Another approach: **coordinate descent**

- Received a lot of attention in machine learning and statistics the last 10 years
- It is state-of-the-art on several machine learning problems, when possible
- This is what is used in many `R` packages and for `scikit-learn` Lasso / Elastic-net and `LinearSVC`

**Idea.** Minimize one coordinate at a time (keeping all others fixed)

Given $f : \mathbb{R}^d \to \mathbb{R}$ convex and smooth if we have

$$f(w + ze_j) \geq f(w) \quad \text{for all} \quad z \in \mathbb{R} \text{ and } j = 1, \ldots, d$$

(where $e_j = j$-th canonical vector of $\mathbb{R}^d$) then we have

$$f(w) = \min_{w' \in \mathbb{R}^d} f(w')$$

**Proof.** $f(w + ze_j) \geq f(w)$ for all $z \in \mathbb{R}$ implies that

$$\frac{\partial f}{\partial w^j}(w) = 0$$

which entails $\nabla f(w) = 0$, so that $w$ is a minimum for $f$ convex and smooth

**Exact coordinate descent** (CD)

- For $t = 1, \ldots,$
- Choose $j \in \{1, \ldots, d\}$
- Compute

$$
\begin{aligned}
w_j^{t+1} &= \operatorname*{argmin}_{z \in \mathbb{R}} f(w_1^t, \ldots, w_{j-1}^t, z, w_{j+1}^t, \ldots, w_d^t) \\
w_{j'}^{t+1} &= w_{j'}^t \quad \text{for } j' \neq j
\end{aligned}
$$

**Remarks**

- Cycling through the coordinates is arbitrary: uniform sampling, pick a permutation and cycle over it every each $d$ iterations
- Only 1D optimization problems to solve, but a lot of them

**Example.** Least-squares linear regression

- Let $f(w) = \frac{1}{2n}\|\boldsymbol{X}w - y\|_2^2$
- $\boldsymbol{X}$ features matrix with columns $X^1, \ldots, X^d$
- Minimization over $w_j$ with all other coordinates fixed:

$$0 = \nabla_{w_j} f(w) = \langle X^j, Xw - y\rangle = \langle X^j, X^j w_j + \boldsymbol{X}^{-j} w_{-j} - y\rangle$$

  where $X^{-j}$ is $\boldsymbol{X}$ with $j$-th columns removed and $w_{-j}$ is $w$ with $j$-th coordinate removed
- Namely

$$w_j = \frac{\langle X^j, y - \boldsymbol{X}^{-j} w_{-j}\rangle}{\|X^j\|_2^2}$$

- Repeat these updates cycling through the coordinates $j = 1, \ldots, d$

- Namely pick $j \in \{1, \ldots, d\}$ at iteration $t$ and do

$$w_j^{t+1} \leftarrow \frac{\langle X^j, y - \boldsymbol{X}^{-j} w_{-j}^t \rangle}{\|X^j\|_2^2}$$
$$w_{j'}^{t+1} \leftarrow w_{j'}^t \quad \text{for } j' \neq j$$

- Written like this, one update complexity is $n \times d$ (matrix-vector product $\boldsymbol{X}^{-j} w_{-j}$ and inner product with $X_j$)
- Update of all coordinates is $O(nd^2)$ ? While GD is $O(nd)$ at each iteration...
- No! There is a trick. Defining the current **residual** $r^t \leftarrow y - \boldsymbol{X} w^t$ we can write an update as

$$w_j^{t+1} \leftarrow w_j^t + \frac{\langle X^j, r^t \rangle}{\|X_2^j\|^2} \quad \text{and} \quad r^{t+1} \leftarrow r^t + (w_j^{t+1} - w_j^t) X^j$$

- This is $2n$, which makes the full coordinates update $O(nd)$, like an iteration of GD

**Theorem (Warga (1963))**
If $f$ is continuously differentiable and strictly convex, then exact coordinate descent converges to a minimum.

**Remarks.**

- A 1D optimization problem to solve at each iteration: cheap for least-squares, but **can be expensive for other problems**
- Let's solve it approximately, since we have many iterations left
- Replace exact minimization w.r.t. one coordinate by a single gradient step in the 1D problem

**Coordinate gradient descent** (CGD)

- For $t = 1, \ldots,$
- Choose $j \in \{1, \ldots, d\}$
- Compute

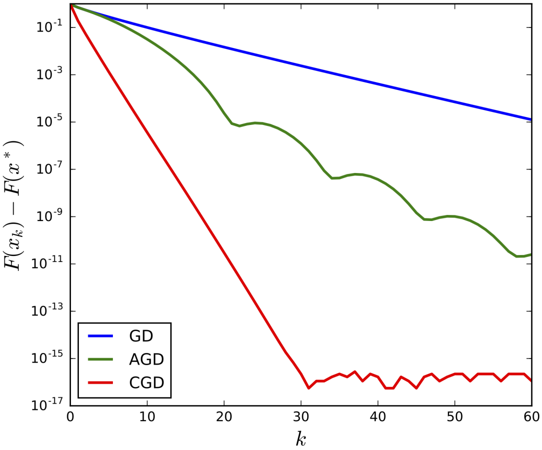
$$w_j^{t+1} = w_j^t - \eta_j \nabla_{w_j} f(w^t)$$
$$w_{j'}^{t+1} = w_{j'}^t \quad \text{for } j' \neq j$$

where

- $\eta_j =$ the step-size for coordinate $j$, can be taken as $\eta_j = 1/L_j$ where $L_j$ is the Lipchitz constant of

$$f^j(z) = f(w + ze_j) = f(w_1, \ldots, w_{j-1}, z, w_{j+1}, \ldots, w_d)$$

- Cool. Let's try it...




Wow! Coordinate gradient descent is much faster than GD and AGD! But why ?

**Theorem (Nesterov (2012)).** Assume that $f$ is convex and smooth and that each $f^j$ is $L_j$-smooth.
Consider a sequence $\{w^t\}$ given by CGD with $\eta_j = 1/L_j$ and coordinates $j_1, j_2, \ldots$ chosen at random: i.i.d and uniform distribution in $\{1, \ldots, d\}$. Then

$$\mathbb{E}f(w^{t+1}) - f(w^*) \leq \frac{n}{n+t}\Big(\big(1-\frac{1}{n}\big)(f(w^0)-f(w^*)) + \frac{1}{2}\|w^0 - w^*\|_L^2\Big)$$

with $\|w\|_L^2 = \sum_{j=1}^d L_j w_j^2$.

**Remark.** Bound in expectation, since coordinates are taken at random. For cycling cordinates $j = (t \mod d) + 1$ the bound is much worse.

## Comparison with gradient descent

- GD achieves $\varepsilon$-precision with

$$\frac{L\|w^0 - w^*\|_2^2}{2\varepsilon}$$

iterations. A single iteration for GD is $O(nd)$

- CGD achieves $\varepsilon$-precision with

$$\frac{d}{\varepsilon}\Big(\big(1 - \frac{1}{n}\big)\big(f(w^0) - f(w^*)\big) + \frac{1}{2}\|w^0 - w^*\|_L^2\Big)$$

iterations. A single iteration for CGD is $O(n)$

- Note that $f(w^0) - f(w^*) \leq \frac{L}{2}\|w^0 - w^*\|_2^2$ but typically $f(w^0) - f(w^*) \ll \frac{L}{2}\|w^0 - w^*\|_2^2$

- So, this is actually

$$\frac{L\|w^0 - w^*\|_2^2}{\varepsilon} \text{ against } \frac{1}{\varepsilon}\|w^0 - w^*\|_L^2$$

- Namely $L$ against the $L_j$
- For least-squares we have $L = \lambda_{\max}(\boldsymbol{X}^\top \boldsymbol{X})$ and $L_j = \|X^j\|_2^2$
- We always have

$$L_j = \|X^j\|_2^2 = \|\boldsymbol{X} e_j\|_2^2 \le \max_{u:\|u\|_2=1} \|\boldsymbol{X} u\|_2^2 = \lambda_{\max}(\boldsymbol{X}^\top \boldsymbol{X}) = L$$

- And actually it often happens that $L_j \ll L$. For instance, if features are normalized then $L_j = 1$, while $L \approx d$ meaning $L_j = O(L/d)$
- This explains roughly why CGD is much faster than GD for ML problems

- What about non-smooth penalization using CGD ?
- What if I want to use an L1 penalization $g(w) = \lambda \|w\|_1$ ?
- We only talk about the minimization of $f(w)$ convex and **smooth** using CGD
- What if we want to minimize $f(w) + g(w)$ for $g$ a penalization function, like we did with GD and AGD

**Proximal coordinate gradient descent** allows to minimize $f(w) + g(w)$ for a **separable** function $g$, namely a function of the form

$$g(w) = \sum_{j=1}^{d} g_j(w^j)$$

with each $g_j$ convex (eventually not smooth) and such that $\mathrm{prox}_{g_j}$ is easy to compute. For Lasso, take $g^j(w^j) = \lambda |w^j|$ for the Lasso (we saw 3 weeks ago that $\mathrm{prox}_{g_j}$ is easy to compute)

**Proximal coordinate gradient descent** (PCGD)

- For $t = 1, \dots,$
- Choose $j \in \{1, \dots, d\}$
- Compute

$$
\begin{aligned}
w_j^{t+1} &\leftarrow \mathrm{prox}_{\eta_j g_j}(w_j^t - \eta_j \nabla_{w_j} f(w^t)) \\
w_{j'}^{t+1} &= w_{j'}^t \quad \text{for } j' \neq j
\end{aligned}
$$

where we recall that

- $\eta_j =$ the step-size for coordinate $j$, can be taken as $\eta_j = 1/L_j$
- And where $\mathrm{prox}_{\eta_j g_j}$ is

$$
\mathrm{prox}_{\eta_j g_j}(w_j) = \operatorname*{argmin}_{z \in \mathbb{R}} \frac{1}{2}(z - w_j)^2 + \eta_j g_j(z)
$$

The same Theorem holds as for (CGD) (under the same assumptions, for random draws of coordinates)

**Applications to machine learning**. Minimization of

$$\min_{w \in \mathbb{R}^d} f(w) + \sum_{j=1}^{d} g_j(w^j)$$

- Regression elastic-net: $f(w) = \frac{1}{2n}\|\boldsymbol{X} w - y\|_2^2$ and $g_j(w) = \lambda(\tau|w_j| + (1-\tau)w_j^2)$
- Logistic regression $\ell_1$: $f(w) = \log(1 + \exp(-y \odot \boldsymbol{X} w))$ and $g_j(w) = \lambda|w_j|$
- Box-constrained regression $f(w) = \frac{1}{2n}\|\boldsymbol{X} w - y\|_2^2$ such that $\|w\|_\infty \leq r$
- Non-linear least-squares $f(w) = \frac{1}{2n}\|\boldsymbol{X} w - y\|_2^2$ such that $w_j \geq 0$
- This is what is used in `scikit-learn` for `LinearSVC` when `dual=True` (even if constraint is not separable)

Next week

- Some grand-mother recipes for supervised learning
- Quasi-newton
- Stochastic gradient descent and beyond
- **You** will implement all the algorithms and compare them !
- This means that **you need to bring your laptop next week to the PC**

# Thank you!