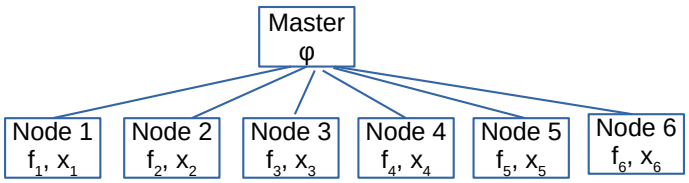
SD-TSIA 211
Optimization for Machine Learning
3 February 2023

Paper documents are allowed (lecture notes, exercises and books)
Electronic devices are forbidden
You may write in English or in French
2h-exam

Exercise 1 (Distributed optimization)**.**
We consider the following optimization problem

$$\min_{x\in\mathbb{R}} \sum_{i=1}^{n} f_i(x) \tag{1}$$

where each function f_i is convex and differentiable. We assume that the data to compute each function f_i is stored in a unique computer M_i. For privacy reasons, it is not possible to share this data. Instead, we are going to show how a master computer can manage communications between computers without using the private data.



1. Show that (1) is equivalent to

$$\min_{x_1\in\mathbb{R},\dots,x_n\in\mathbb{R}} \sum_{i=1}^{n} f_i(x_i) \tag{2}$$
$$\text{s.t.: } x_i = x_n, \quad \forall i \in \{1,\dots,n-1\}$$

   In particular, you should show how, knowing a minimizer of one optimization problem, one can construct a minimizer of the other optimization problem.

2. Write the Lagrangian of the optimization problem (2). How many Lagrange multipliers are there?

3. Write the dual problem of (2).

4. Show that the dual function can be written as

$$D(\varphi) = \sum_{i=1}^{n} \inf_{z_i\in\mathbb{R}} F_i(z_i, \varphi)$$

   where you should explicit the formula of the functions F_i.

5. Show that for all z, $(\varphi \mapsto F_i(z, \varphi))$ is linear with respect to φ.

6. Let $h_i : \varphi \mapsto \inf_z F_i(z, \varphi)$ and let $z_i^*(\varphi) \in \arg\min_z F_i(z, \varphi)$.
   Show that for all ψ, $F_i(z_i^*(\psi), \psi) \leq F_i(z_i^*(\varphi), \varphi) + \langle \nabla_\varphi F_i(z_i^*(\varphi), \varphi), \psi - \varphi \rangle$

7. Deduce from this an element of $g \in \partial(-D)(\varphi)$. We assume for the rest of the exercise that D is differentiable and that $\nabla D(\varphi) = -g$.

8. Write the gradient descent method on $-D$ (or said otherwise, gradient ascent on D).

9. What step size would you recommend among the following choices: fixed step size, decreasing step size or line search? Please give details and justify your choice.

10. Show that this algorithm can be implemented in such a way that Computer i communicates with the master computer only through the dual multipliers φ and their copy x_i of the primal variable.

Exercise 2 (Stochastic gradient descent for Huber loss regression)**.**
Let $\delta > 0$ and

$$\begin{aligned} h_\delta : \mathbb{R} &\to \mathbb{R} \\ z &\mapsto \begin{cases} |z| - \frac{\delta}{2} & \text{if } |z| \geq \delta \\ \frac{z^2}{2\delta} & \text{if } |z| < \delta \end{cases} \end{aligned}$$

We call this function the Huber loss.

1. Calculate the derivative of h_δ and show that h_δ is continuously differentiable.

2. Show that h_δ is convex.

We now consider the following optimization problem:

$$\min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} h_\delta(a_i^\top x - b_i) \tag{3}$$

where for all i, $a_i \in \mathbb{R}^d$ is a vector of features and $b_i \in \mathbb{R}$ is a label.

3. Denote $f_i(x) = h_\delta(a_i^\top x - b_i)$. Show that $\exists C \geq 0$ such that $\|\nabla f_i(x)\| \leq C$ for all x.

4. Write the stochastic gradient method for the resolution of (3). Your algorithm should use a single data point (a_i, b_i) per iteration and you should specify the step size (also known as learning rate) used in the algorithm.

5. Recall the convergence rate of stochastic gradient descent.

Exercise 3 (Coordinate descent for the Lasso)**.**
We consider the Lasso problem

$$\min_{x\in\mathbb{R}^d} \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1$$

where A is a $n \times d$ matrix, b is a vector of size n and $\lambda > 0$.

1. Show that the function $f : x \mapsto \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1$ is convex.
2. Calculate the gradient of $f_1 : x \mapsto \frac{1}{2}\|Ax-b\|_2^2$.
3. Give, without proof, the formula of the proximal operator of $f_2 : x \mapsto \lambda\|x\|_1$
4. Write the proximal gradient descent algorithm for the resolution of the Lasso problem.

We consider now an alternative algorithm called coordinate descent. The idea is to update one single coordinate per iteration, in order to get very cheap iterations.

Let $e_j = (0,\dots,0,1,0,\dots,0)$ where the 1 is in the j^{th} position. For $x \in \mathbb{R}^d$ and $j \in \{1,\dots,d\}$, we denote

$$\begin{aligned} g : \mathbb{R} &\to \mathbb{R} \\ t &\mapsto f(x+te_j) \end{aligned}$$

The coordinate descent algorithm is given by

$$\begin{aligned} &x^0 \in \mathbb{R}^d, r^0 = Ax^0 - b \\ &\forall k \geq 0 : \\ &\quad j \sim U(\{1,\dots,d\}) \\ &\quad t^{k+1} = \arg\min_t f(x_k + te_j) \\ &\quad x_{k+1} = x_k + t_{k+1}e_j \\ &\quad r^{k+1} = Ax^{k+1} - b \end{aligned}$$

In the algorithm, the vector $r^k = Ax^k - b$ is called the vector of residuals and its efficient update will be fundamental to get a fast algorithm.

5. Show that $g(t) = \frac{t^2}{2}\|Ae_j\|^2 + t\langle Ae_j, r^k\rangle + \frac{1}{2}\|r^k\|_2^2 + \lambda|t+x_j^k| + \lambda\sum_{i\neq j}|x_i^k|$. In particular, provided that r^k is pre-computed, computing $g(t)$ requires only $O(d)$ operations.
6. *[Bonus question]* Show that $\arg\min_t g(t)$ can be written in terms of the proximal operator of the absolute value.
7. Show that r^{k+1} can be computed as a function of r^k, t^{k+1} and the j^{th} column Ae_j of A. In particular, the costly matrix-vector product Ax^k only needs to be performed at initialization.

Ex 1.

1. Assume $\forall i \in \{1, \cdots, n-1\}$. $x_i = x_n = x^*$ is the solution of (2),

$$\sum_{i=1}^{n} f_i(x^*) \le \sum_{i=1}^{n} f_i(x_i)$$

which also indicates that $\sum_{i=1}^{n} f_i(x^*) \le \sum_{i=1}^{n} f_i(x)$

Therefore, (1) and (2) share the same solution under the condition of $x_i = x_n$.

Thus, (1) $\Leftrightarrow$ (2)

2. $L(x, \phi) = \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n-1} \phi_i (x_i - x_n)$

$\phi \in \mathbb{R}^{n-1}$

3. Dual: $\max_{\phi_i} \left(\min_{x_i} L(x, \phi) \right)$ $\sup_{\phi} \inf_{x} L(x, \phi) = \max_{\phi} D(\phi)$ with $D(\phi) = \inf_{x} L(x, \phi)$

4. $D(\phi) = \inf_{x_i} \left[\sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n-1} \phi_i (x_i - x_n) \right]$

$$= \inf_{x_i} \left[\sum_{i=1}^{n-1} \left(f_i(x_i) + \phi_i (x_i - x_n) \right) + f_n(x_n) \right] = \inf_{x_i} \left[\sum_{i=1}^{n-1} \left(f_i(x_i) + \phi_i x_i \right) - x_n \sum_{i=1}^{n-1} \phi_i + f_n(x_n) \right]$$

$$= \sum_{i=1}^{n-1} \inf_{x_i} \left(f_i(x_i) + \phi_i (x_i - x_n) \right) + \inf_{x_n} f_n(x_n)$$

$$F_i(z_i, \phi) = \begin{cases} f_i(z_i) + \phi_i (z_i - z_n) & i \neq n \\ f_n(z_n) & i = n \end{cases} \qquad F_i(z_i, \phi) = \begin{cases} f_i(z_i) + z_i \phi_i, & i \neq n \\ f_n(z_n) - z_n \sum_{j=1}^{n-1} \phi_j, & i = n \end{cases}$$

when $i = n$, $x_i - x_n = 0$ therefore,

$$F_i(z_i, \varphi) = f_i(z_i) + \varphi(z_i - z_n).$$

5. Let $z_i - z_n = A$, $f_i(z_i) = b$.

$F_i(z_i, \varphi) = b + A\varphi$ is affine wrt φ.

h_i is linear + concave.

6. $h_i(\varphi) = \inf_z F_i(z, \varphi)$

$h_i(\psi) \le h_i(\varphi) + \langle \nabla h_i(\varphi), \psi - \varphi \rangle$

$\Rightarrow F_i(z_i^*(\psi), \psi) \le F_i(z_i^*(\varphi), \varphi) + \langle \nabla F_i(z_i^*(\varphi), \varphi), \psi - \varphi \rangle$

$z_i^*(\varphi) \in \arg\min_z F_i(z, \varphi)$ indicates that $F_i(z_i^*(\varphi), \varphi) \le F_i(z_i^*(\psi), \varphi)$

$$F_i(z_i^*(\varphi), \varphi) \ge F_i(z_i^*(\psi), \psi) + \langle \nabla_\varphi F_i(z_i^*(\varphi), \varphi), \varphi - \psi \rangle$$

$$F_i(z_i^*(\psi), \varphi) + \langle \nabla_\varphi F_i(z_i^*(\psi), \varphi), \psi - \varphi \rangle \ge F_i(z_i^*(\psi), \psi)$$

7. $(z_i^*(\varphi), \varphi)$ is a saddle point.

$\partial D(\varphi) = \nabla_\varphi F_i(z_i^*(\varphi), \varphi)$.

$\partial(-D)(\varphi) = -\nabla_\varphi F_i(z_i^*(\varphi), \varphi) = -g$

by (6): $\sum_{i=1}^n h_i(\psi) \le \sum_{i=1}^n h_i(\varphi) + \langle -g, \psi - \varphi \rangle$

$-D(\psi) \ge -D(\varphi) + \langle g, \psi - \varphi \rangle$.

$\Leftrightarrow \sum_{i=1}^n \inf_{z_i} F_i(z_i, \psi) \le \sum_{i=1}^n \inf_{z_i} F_i(z_i, \varphi) + \langle -g, \psi - \varphi \rangle$

$\Leftrightarrow \sum_{i=1}^n h_i(\psi) \le \sum_{i=1}^n h_i(\varphi) + \langle -g, \psi - \varphi \rangle$

$\sum_{i=1}^n h_i(\psi) \le \sum_{i=1}^n h_i(\varphi) + \langle \sum_{i=1}^n \nabla h_i(\varphi), \psi - \varphi \rangle$

$g = -\sum_{i=1}^n \nabla h_i(\varphi) = \partial(-D)(\varphi)$

8. $\varphi_i^{(k+1)} = \varphi_i^{(k)} - \gamma_i^{(k)} g^{(k)}$.

9. If $F_i(z_i, \varphi)$ satisfy L-Lipschitz, then $\gamma_i = \frac{1}{L}$.

But it is recommand to choose γ_i as line search,

i.e. $\gamma_i = \arg\min_{\gamma \in \mathbb{R}^+} (f_i(z_i) - \gamma \nabla f_i(z_i))$.

10. In local computer, use $x_i = x_n$ to change the P to D.

In iterations, the calculation of g only uses φ.

To update z_i, uses g and z_i. These calculations can be

implemented locally.

Ex 2.

1. $$\partial h_\delta(z) = \begin{cases} \frac{z}{\delta} & |z| < \delta \\ \frac{z}{|z|} & |z| \geq \delta \wedge \delta \neq 0 \\ 0 & z = 0 \end{cases} \quad \text{sgn}(z)$$

which is continuous.

2. $$\partial^2 h_\delta(z) = \begin{cases} 1/\delta, & |z| < \delta \\ 0, & \text{otherwise} \end{cases}$$

$\partial^2 h_\delta(z) \geq 0 \Rightarrow h_\delta(z)$ is convex

3. $$\nabla f_i(x) = \partial h_\delta(a_i^T x - b_i) = \begin{cases} -a_i, & a_i^T x - b_i < -\delta \\ \frac{a_i^T x - b_i}{\delta} a_i, & |a_i^T x - b_i| < \delta \\ a_i, & a_i^T x - b_i > \delta \end{cases}$$

$$\|\nabla f_i(x)\| \leq \|a_i\| \Rightarrow C = \max \|a_i\|$$

4. Generate $I_{k+1} \sim U(\{1, \cdots, N\})$.

$$x_{k+1} = x_k - \gamma_k \nabla f_{I_{k+1}}(x_k)$$

$$\gamma_k = \frac{\gamma_0}{\sqrt{k}} \quad P_{25}$$

5. $$\mathbb{E}\left[f(\overline{x}_k^\gamma) - f(x^*)\right] \leq \frac{\mathbb{E}\left[\|x_0 - x^*\|^2\right] + C \sum_{i=1}^{k} \gamma_i^2}{2 \sum_{i=0}^{k} \gamma_i}$$

$$\overline{x}_k^\gamma = \frac{\sum_{i=0}^{k} \gamma_i x_i}{\sum_{i=0}^{k} \gamma_i}, \quad C = \max \|a_i\| \quad P_{24}$$

Ex 3.

1. $\|Ax-b\|^2$ is convex:

$$\partial \|Ax-b\|^2 = 2A^T(Ax-b).$$

$$\partial^2 \|Ax-b\|^2 = 2A^TA \rightarrow PSD.$$

$\lambda \|x\|$ is convex.

Therefore $f: x \mapsto \|Ax+b\|^2 + \lambda\|x\|$ is convex.

2. $\partial f_1(x) = A^T(Ax-b).$

3. $$\operatorname{prox}_{\gamma f_2}(x) = \operatorname*{argmin}_y \gamma f_2(y) + \frac{1}{2}\|x-y\|^2$$

$$= \operatorname*{argmin}_y \gamma\lambda\|y\| + \frac{1}{2}\|x-y\|^2$$

$$= \operatorname{sgn}(x)(|x|-\gamma\lambda)_+$$

4. $$x_{k+1} = \operatorname{prox}_{\gamma f_2}(x_k - \gamma \nabla f_1(x_k)) \qquad x_k - \gamma A^T(Ax_k-b)$$

$$= \operatorname{prox}_{\gamma f_2}(x_k - \gamma A^T(Ax_k-b))$$

$$= \operatorname{sgn}(x_k-\gamma A^T(Ax_k-b))\left(|x_k-\gamma A^T(Ax_k-b)| - \gamma\lambda\right)_+$$

5. $$g(t) = f(x+te_j) = \frac{1}{2}\|A(x+te_j)-b\|^2 + \lambda\|x+te_j\|$$

$$= \frac{1}{2}\|Ax+tAe_j-b\|^2 + \lambda\|x+te_j\|$$

$$= \frac{t^2}{2}\|Ae_j\|^2 + \frac{1}{2}\|Ax-b\|^2 + t\langle Ax-b, Ae_j\rangle + \lambda\|x+te_j\|$$

$$= \underbrace{\frac{t^2}{2}\|Aej\|^2}_{O(n)} + \underbrace{\frac{1}{2}\|r^k\|^2}_{O(1)} + \underbrace{t\langle Aej, r^k\rangle}_{O(n)} + \underbrace{\lambda|t+x_j^k|}_{O(1)} + \underbrace{\lambda\sum_{i\neq j}|x_i^k|}_{O(d)}$$

total cost: $O(2n+2+d) = O(2n+d) = O(n+d)$

6. $t^* = \arg\min_t g(t)$. assume $t + x_j^k = t'$

$$= x_j^k + \arg\min_{t'} \frac{1}{2}(t'-x_j^k)^2\|Aej\|^2 + (t'-x_j^k)\langle Aej, r^k\rangle + \lambda|t'| + \cdots$$

$$= x_j^k + \arg\min_{t'} \lambda|t'| + \frac{1}{2}\|(t'-x_j^k)Aej + r^k\|^2$$

7. $r^{k+1} = Ax^{k+1} - b = Ax^k + Aej\, t^{k+1} - b.$

$$= r^k + Aej\, t^{k+1}$$

$r^k = r^{k-1} + Aej\, t^k$

$r^2 = r^1 + Aej\, t^2$

$r^1 = r^0 + Aej\, t^1$

$r^0 = Ax^0 - b$. Ax^k only needs to be initialized such that r^0 can be calculated. Then by recurrance, r^{k+1} can be calculated.