Valentin De Bortoli debortoli@cmla.ens-cachan.fr
Alain Durmus alain.durmus@cmla.ens-cachan.fr
TP 2 : optimisation numérique Première année Master Hadamard

Minimisation par optimisation proximale

En préambule de tout module python, on suppose que les paquets suivants ont été chargés

```
import numpy as np
import matplotlib.pyplot as plt
```

1 Implémentation de ISTA

L'algorithme ISTA [1] est un algorithme a pour objectif à minimiser une fonction $F : \mathbb{R}^d \to \mathbb{R}$ s'écrivant sous la forme $F = f + \lambda g$ où $\lambda > 0$ est une constante,

1. f est un fonction convexe, C^1, gradient Lipschitz *i.e.*pour tout $x, y \in \mathbb{R}^d$,
$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| \ .$$
2. g est une fonction continue convexe dont on peut calculer l'opérateur proximal prox_g^γ défini pour tout $\gamma > 0$ et $x \in \mathbb{R}^d$ par
$$\mathrm{prox}_g^\gamma = \arg\min_{y \in \mathbb{R}} \left\{ g(y) + (2\gamma)^{-1} \|x - y\|^2 \right\} \ .$$

L'algorithme ISTA consiste alors à construire la suite $(x_k)_{k \in \mathbb{N}}$ partant de $x_0 \in \mathbb{R}^d$ à partir de la récursion suivante :

$$x_{k+1} = \mathrm{prox}_{\lambda g}^\gamma \left\{ x_k - \gamma \nabla f(x_k) \right\} \ .$$

1. Montrer que pour tout $\gamma, \lambda > 0$, on a
$$\mathrm{prox}_{\lambda g}^\gamma = \mathrm{prox}_g^{\lambda\gamma} \ .$$
2. Implémenter alors l'algorithme ISTA. On pourra par exemple considérer un fonction sous la forme

```
ista(dim,prox_op_g, grad_f, fun_total, lambda_l, n_it=100):
Variables d'entree :
dim : dimension du probleme
prox_op_g : operateur proximal de g qui prend en entree x et gamma
grad_f : fonction qui a partir d'un point x retourne le gradient de f
       et la constante de Lipschitz du gradient de f
fun_total : fonction F = f+lambda*g
lambda_l : parametre lambda dans F
n_it : nombre d'iterations
Variables de sortie : x, fun_iterate
x : itere final de ISTA
fun_iterate : suite (f(x_k))
```

2 Acquisition comprimée

Dans de nombreux problèmes, notamment en traitement du signal, seulement des observations partielles, regroupées dans un vecteur $y \in \mathbb{R}^p$, d'un signal $x \in \mathbb{R}^d$ sont disponibles. Cela s'explique par une volonté de compresser le signal par exemple ou encore parce que les instruments de mesures du signal sont dans certains cas peu précis. Mathématiquement, cette perte de données se modélise par une relation $y = Ax$, où $A \in \mathbb{R}^{p,d}$, et $p << d$. Comme A n'est pas inversible, le plus souvent le problème inverse $y = Ax$ a de nombreuses solutions mais qui ne sont pas satisfaisantes. En effet, dans de nombreux cas, le signal d'origine x est le plus souvent parcimonieux, *i.e.* un grand nombre de ses composantes sont nulles. Ainsi, pour reconstruire ce type de signal, il a été proposé de considérer le minimum de la fonction

$$F(x) = f(x) + \lambda g(x) \ , \qquad f(x) = \|y - Ax\|^2 \ , \ g(x) = \sum_{i=1}^{d} |x_i| \ ,$$

où $\lambda > 0$ est un coefficient qui contrôle le niveau de parcimonie de x. On peut observer que F peut s'écrire comme $f + g$ où f est une fonction convexe gradient Lipschitz and g est un fonction convexe continue mais non différentiable. On se propose dans cette partie d'appliquer l'algorithme ISTA à la minimisation de F avec une application au traitement d'image. Ici y sera une mesure d'une image et x les coefficients en ondelettes de la même image. Ainsi la matrice A est simplement l'opérateur linéaire qui à une représentation en ondelettes associe un signal.

Avant de commencer le TP, installer les librairies suivantes de python. Taper dans un terminal :

```
pip install -U scikit-learn scikit-image
```

Vous aurez aussi besoin du fichier `tp2_tools.py` à télécharger sur la page suivante : https://vdeborto.github.io/project/optimization/

1. Créer un nouveau module python pour ce problème qui commencera par

```
import numpy as np
import matplotlib.pyplot as plt
from tp2_tools import *
import warnings
warnings.filterwarnings('ignore')
from ista import ista
```

2. Générer les données A et y et fixer pour le moment λ

```
lambda_l1 = 1
y, A = noisy_observations()
```

3. Calculer le gradient de f pour cet exemple.
4. Dans cette question, on s'intéresse au calcul de l'opérateur proximal de g.
   (a) Soit h une fonction convexe sci propre sur $\mathbb{R}^d$ sous la forme $h(x) = \sum_{i=1}^{d} h_i(x_i)$, pour tout $x = (x_i, \dots, x_d) \in \mathbb{R}^d$, où pour tout $i \in \{1, \dots, d\}$, $h_i : \mathbb{R} \to (-\infty, +\infty]$, est convexe sci propre. Montrer que pour tout $x \in \mathbb{R}^d$ et $\gamma > 0$,

$$\text{prox}_h^\gamma(x) = (\text{prox}_{h_i}^\gamma(x_i))_{i \in \{1,\dots,d\}} \ .$$

(b) Considérons $\phi(t) = |t|$ pour tout $t \in \mathbb{R}$. Montrer que pour tout $u \in \mathbb{R}$ et $\gamma > 0$

$$t^\star = \arg\min_{t\in\mathbb{R}} \left\{ |t| + (1/2\gamma)\,|t-u|^2 \right\} \ ,$$

si et seulement si

$$t^\star \in u - \gamma \partial\phi(t^\star) \ .$$

(c) Déterminer la sous-différentielle de ϕ sur $\mathbb{R}$.

(d) En distinguant les cas $|u| < \gamma$ et $|u| \geq \lambda$, déterminer pour tout $u \in \mathbb{R}$ et $\gamma > 0$, $\mathrm{prox}_\phi^\gamma(u)$.

(e) En déduire pour tout $x \in \mathbb{R}^d$ et $\gamma > 0$, $\mathrm{prox}_g^\gamma(x)$.

5. Implémenter le gradient de f et l'opérateur proximal de g.
6. Appliquer la fonction ista que vous avez précédemment coder pour minimiser F et afficher l'image que vous obtenez à l'aide la fonction `plot_image`.
7. Vérifier numériquement que l'ordre de convergence de ISTA est de l'ordre $O(k^{-1})$ où k est le nombre d'itérations.
8. Changer la valeur du paramètre λ et afficher les images que vous obtenez. Discuter de vos résultats.

3 Minimisation de la fonction max

Dans cette section, on s'intéresse à la minimisation de la fonction F définie sur $\mathbb{R}^d$ par

$$F(x) = f(x) + \lambda g(x) \ , \qquad f(x) = \|y - Ax\|^2 \ , \ g(x) = \max_{i\in\{1,\dots,d\}} x_i \ ,$$

où $y \in \mathbb{R}^d$ et $A \in \mathbb{R}^{d\times d}$.

1. Créer un nouveau module python pour ce problème qui commencera par

```
import numpy as np
import matplotlib.pyplot as plt
from tp2_tools import *
import warnings
warnings.filterwarnings('ignore')
from ista import ista
```

2. Générer les données A et y et fixer pour le moment λ

```
lambda_max = 1
y, A = noisy_observations_inf()
```

3. Calculer le gradient de f pour cet exemple.
4. Dans cette question, on s'intéresse au calcul de l'opérateur proximal de g.

(a) Montrer que le calcul de $\mathrm{prox}_g^\gamma(x)$ pour $\gamma > 0$ et $x \in \mathbb{R}^d$ revient au problème de minimisation sous contrainte

$$\begin{array}{rl} \min_{t\in\mathbb{R}, y\in\mathbb{R}^d} & t + (2\gamma)^{-1}\,\|y-x\|^2 \\ \text{sous contraintes} & y_i \leq t \text{ pour tout } i \in \{1,\dots,d\} \ . \end{array} \tag{1}$$

(b) Donner le Lagrangien associé à ce problème.

(c) En déduire à partir des conditions KKT, une expression implicite pour la solution $(t^\star, y^\star)$ du problème (1).

5. Télécharger le module `prox_max.py` à partir de https://vdeborto.github.io/project/optimization/ et utiliser la fonction ista que vous avez implémenté pour minimiser F.
6. Illustrer graphiquement la convergence de l'algorithme.

4 Complétion de matrice de faible rang

Dans cette section, on s'intéresse au problème de complétion de matrice. Dans certains problèmes statistiques, on a accès à seulement certaines entrées d'une matrice $X \in \mathbb{R}^{d\times d}$, $Y = A \odot X$ où $A = (\mathbf{1}_{I\times J}((i,j)))_{i,j\in\{1,\dots,d\}}$, où $I, J \subset \{1,\dots,d\}$ et $\odot$ est la multiplication élément par élément (ou appelé par certains "matlab"). Dans ce problème, on voudrait alors retrouver les composantes manquantes de X. Pour cela, il est commun de chercher une matrice X de faible rang. Cela revient alors à chercher à minimiser la fonction F définie sur $\mathbb{R}^{d\times d}$ par

$$F(X) = f(X) + \lambda g(X)\ , \qquad f(X) = \|Y - A\odot X\|_2^2\ ,\ g(X) = \|X\|_\star = \sum_{i=1}^{d} |\sigma_i(X)|\ ,$$

où $\|\cdot\|_2$ est la norme de Frobenius et $(\sigma_i)_{i\in\{1,\dots,d\}}$ sont les valeurs singulières de X. On rappelle le théorème de décomposition en valeurs propres singulières :

Théorème 1. *Soit $A \in \mathbb{R}^{d\times m}$, $d \le m$. Il existe alors deux matrices orthogonales O_1, O_2 et $\sigma_1(A) \ge \cdots \ge \sigma_d(A) \ge 0$ tel que $A = O_1 \Sigma O_2$ où $\Sigma_A \in \mathbb{R}^{d\times m}$ est la matrice dont les entrées sont données par $\Sigma_{i,i} = \sigma_i(A)$ pour $i \in \{1,\dots,d\}$ et $\Sigma_{i,j} = 0$ sinon. Les réels $(\sigma_1(A),\dots,\sigma_d(A))$ sont appelés les valeurs singulières de A et sont les valeurs propres de AA^{T}.*

1. Montrer que g est la norme dual de la norme opérateur $|||\cdot|||$ sur $\mathbb{R}^{d\times d}$, *i.e.*

$$g(X) = \sup_{V\in\mathbb{R}^{d\times d},\,|||V|||\le 1} \langle X, V\rangle = \sup_{V\in\mathbb{R}^{d\times d},\,|||V|||\le 1} \mathrm{Tr}(X^{\mathrm{T}}V)\ .$$

   En déduire que g est convexe.

On cherche alors à appliquer ISTA pour minimiser F.

2. On pourra dans un premier temps modifier ou créer une nouvelle fonction `ista_mat` qui s'applique directement à des matrices.
3. Créer un nouveau module python pour ce problème qui commencera par

```
import numpy as np
import matplotlib.pyplot as plt
from tp2_tools import *
import warnings
warnings.filterwarnings('ignore')
from ista import ista_mat
```

4. Générer les données A et y et fixer pour le moment λ

```
lambda_nuclear = 1
Y, A = noisy_observations_nuclear()
```

5. Calculer le gradient de f pour cet exemple.
6. Dans cette question, on s'intéresse au calcul de l'opérateur proximal de g.
   (a) En utilisant que la norme de Frobenius est invariant par rotation, montrer que pour tout $X \in \mathbb{R}^{d\times d}$ le problème de minimisation

$$\min_{Y\in\mathbb{R}^{d\times d}} \|Y\|_\star + (2\gamma)^{-1}\|Y - X\|_2^2 \ , \tag{2}$$

   est équivalent à

$$\min_{D\in\mathsf{E}} \|D\|_\star + (2\gamma)^{-1}\|D - \Sigma_X\|_2^2 \ , \tag{3}$$

   où E est l'ensemble des matrices diagonales de $\mathbb{R}^{d\times d}$ et Σ_X est la matrice diagonale donnée par la décomposition en valeurs singulières de X. Pour cela on utilisera le résultat suivant [2].

   Théorème 2. *Soit $A, B \in \mathbb{R}^{d\times d}$ deux matrices de dimension d. Alors pour toutes matrices orthogonales O_1, O_2,*

$$\langle O_1 A O_2, B\rangle \leq \langle \Sigma_A, \Sigma_B\rangle \ ,$$

   où Σ_A et Σ_B sont les deux matrices diagonales données par la décomposition en valeurs singulières de respectivement A et B.

   Si D est une solution de (3), comment construire Y solution de (2) ?
   (b) En déduire que pour tout $\gamma > 0$ et $X \in \mathbb{R}^{d\times d}$

$$\mathrm{prox}_g^\gamma(X) = U\,\mathrm{prox}_{\ell_1}^\gamma(\Sigma_X)V \ ,$$

   où $X = U\Sigma_X V$ est une décomposition en valeurs singulières de X et $\mathrm{prox}_{\ell_1}^\gamma$ est l'operateur proximal de la norme ℓ_1, $Y \mapsto \sum_{i,j=1}^d |Y_{i,j}|$.
7. Implémenter l'opérateur proximal de g.
8. Tester l'algorithme ISTA et illustrer graphiquement sa convergence.

Références

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithms for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1) :183–202, 2009.

[2] von Neumann J. Some matrix inequalities and metrization of metric-space. *Rev. Tomsk Univ.*, 1 :286–300, 1937.