CPE LYON

Année universitaire
2022 - 2023

IMI – Optimisation et problèmes inverses – S8

TP2 – Optimisation non-lisse, méthodes proximales

Serge Mazauric
Éric Van Reeth

Objectifs.

- Compréhension des notions d'**optimisation non-lisse** : sous-différentielle, enveloppe de Moreau, opérateur proximal
- Compréhension des **modèles LASSO/TV**
- Implémentation de l'algorithme de sous-gradient
- Implémentation de l'**algorithme du point proximal**
- Implémentation de l'**algorithme du gradient proximal**
- Implémentation de l'**algorithme Forward-Backward**
- Compréhension des limites des modèles et des méthodes

Déroulement. Ce TP se déroule sur **2 séances** et est à effectuer par binôme sous Matlab/Octave. Vous trouverez sur CPe-campus une archive contenant l'ensemble des fichiers nécessaires à la réalisation de ce TP.

Ce TP fait partie intégrante de la construction de votre cours sur les algorithmes proximaux, et les modèles d'optimisation non-lisse. Il doit vous permettre à la fois de comprendre le fonctionnement de ces méthodes, d'illustrer votre cours, mais aussi de mettre en évidence l'importance de certaines hypothèses, l'apport de ces méthodes par rapport à des méthodes de résolution classiques, et les limites de ces méthodes. Il est dès lors indispensable que vous preniez le temps d'étudier l'influence des paramètres et, le cas échéant, de vos choix d'implémentation.

Bien qu'il soit relativement guidé, n'hésitez donc pas à sortir des sentier battus, à vous poser vos propres questions et à prendre du recul sur les pistes de réflexion proposées dans ce TP.

Configuration. Ce TP nécessite les librairies suivantes :

- Matlab : *Image Processing Toolbox*
- Octave : *Image toolbox* `pkg install -forge image`
  puis, en début de script : `pkg load image`

Évaluation. L'évaluation de TP portera sur la compréhension des notions, qui sera évaluée en séance.

Nous rappelons que toute tentative de copie entrainera une sanction de l'ensemble des binômes concernés.

Serge Mazauric
Éric Van Reeth

Contexte

On se concentrera dans ce module sur des exemples en débruitage et déconvolution 1D (signal) et 2D (image). Pour autant, nous rappelons que l'optimisation est un domaine transversal, et permet, en fonction du choix de la fonction de coût, de traiter de nombreuses applications.

En particulier, on souhaite effectuer la restauration (débruiter et supprimer le flou) de l'image ci-dessous :

Pour cela, nous avons considéré au TP1 un modèle classique en traitement d'image (et en machine learning, de manière plus générale), connu sous le nom de modèle de Tikhonov. Il est composé d'une attache à la donnée et d'une régularisation différentiable :

$$\hat{x} = \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \ \|Hx - z\|_2^2 + \lambda \|\Gamma x\|_2^2 \tag{1}$$

où Γ modélise un opérateur de dérivation (identité, gradient, Laplacien...).
L'ajout d'un terme de régularisation est une technique incontournable en machine learning pour éviter les problèmes de sur-apprentissage, autrement dit, pour empêcher l'estimation de coller"trop parfaitement" aux observations.

Le modèle de Tikhonov possède l'avantage d'être entièrement différentiable : il est donc possible de calculer une solution analytique, ou de le résoudre avec un algorithme de descente de gradient. Cependant, les résultats obtenus sont peu satisfaisants : le bruit est atténué, au prix de contours flous.

On se propose alors, pour améliorer la qualité de reconstruction, d'utiliser une régularisation $R(x)$ non lisse, autrement dit, non différentiable. Le problème d'optimisation qui nous intéresse dans cette seconde partie de module est le suivant :

$$\hat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \ \|Hx - z\|_2^2 + \lambda \|\Gamma x\|_1 \tag{2}$$

où Γ modélise un opérateur au choix (identité, gradient, Laplacien...).
Pour $\Gamma = \mathrm{Id}$, ce problème est connu sous le nom de **LASSO**.
Pour $\Gamma = D$ ou L, ce problème est connu sous le nom de modèle **TV**.

Pistes de réflexion (pouvant être menées plus tard dans le TP) :

- avantages et inconvénients de la norme ℓ_1 par rapport à la norme ℓ_2^2
- quel choix semble pertinent pour l'opérateur Γ ?

Serge Mazauric
Éric Van Reeth

1 TD : notions d'optimisation non-lisse

L'étude du terme d'attache aux données ayant été largement menée au TP précédent, nous nous concentrerons essentiellement dans ce TP sur le terme de régularisation (non-lisse).
Nous nous proposons donc de découvrir les notions d'optimisation non-lisse sur le problème 1D suivant :

$$\hat{x} \in \underset{x \in \mathbb{R}}{\operatorname{argmin}} \ |x| \tag{3}$$

1.1 Sous-différentielle et algorithme de sous-gradient

Bien que non différentiable, le problème d'optimisation (3) peut être résolu avec des méthodes similaires aux méthodes d'optimisation dans le cas différentiable, et notamment la descente de gradient. Cela nécessite toutefois de généraliser la notion de différentielle.
Après avoir revu cette notion, on s'intéresse donc à la résolution du problème (3) avec l'algorithme de sous-gradient, pour lequel la direction de descente n'est non plus l'opposé du gradient, mais l'opposé de la **sous-différentielle**.

Pistes de réflexion/travail demandé :

- résultat attendu ?
- solution analytique du problème (3) ?
- implémentation de la méthode de descente de sous-gradient
- avantages / inconvénients / limites ?

1.2 Enveloppe de Moreau

Une manière de s'affranchir des limitations de l'algorithme de sous-gradient pour les fonctions de coût f non-lisses est d'utiliser l'**enveloppe de Moreau** de f, notée $\mathcal{M}_f$. De part sa définition, elle permet de "lisser" la fonction f, tout en conservant certaines propriétés remarquables, notamment lorsque f est convexe.
Cette section vise à s'approprier la notion fondamentale d'enveloppe de Moreau d'une fonction, et de comprendre les propriétés de cette approximation particulière de notre fonction de coût. Pour cela, quelques pistes de réflexions sont proposées, autour des calculs théorique et numérique de l'enveloppe pour des fonctions usuelles.

Pistes de réflexion/travail demandé :

- calcul analytique 1D de l'enveloppe de Moreau de : $|ax+b|$
- implémentation numérique de l'enveloppe de Moreau calculée et visualisation
- interprétation ?
- propriétés de l'enveloppe de Moreau $\mathcal{M}_f$ par rapport à f ?
- influence du paramètre γ

Avant de poursuivre avec la section suivante, il sera nécessaire de faire valider les propriétés de l'enveloppe de Moreau que vous aurez exhibées.

1.3 Opérateur proximal

L'opérateur proximal est étroitement lié à l'enveloppe de Moreau. En effet, **le point proximal**, noté $\operatorname{prox}_f(x)$, est défini comme le point (unique) qui réalise le minimum définissant $\mathcal{M}_f$. Au vu

Serge Mazauric
Éric Van Reeth

des propriétés de l'enveloppe de Moreau, et notamment celle concernant la minimisation, que vous avez dû exhiber à la section précédente, il est donc naturel de s'intéresser à cet opérateur.
Tout comme pour l'enveloppe de Moreau, en pratique le calcul de l'opérateur proximal peut être long et coûteux (résolution numérique d'un problème d'optimisation). Pour cette raison, on cherche en général une **forme explicite** (*closed form expression* en anglais) de l'opérateur. Outre la réduction du temps de calcul, elle est nécessaire à la convergence de la plupart des algorithmes utilisant l'opérateur proximal.

Pistes de réflexion :

- calcul 1D du prox de fonctions usuelles : $(\cdot - z)^2$, $\lambda|\cdot|$
- visualisation
- interprétation ?
- propriétés (translation, rescaling, point fixe, etc.)
- influence du paramètre γ

2 Algorithmes proximaux

Les algorithmes proximaux forment une classe d'algorithmes permettant de résoudre des problèmes d'optimisation convexe de la forme $\hat{x} \in \underset{x\in\mathbb{R}^N}{\operatorname{argmin}} f(x)$, via l'utilisation de l'opérateur proximal de la fonction de coût f (convexe). Ils peuvent être vus comme une généralisation des méthodes de descente classiques aux fonctions de coût non-lisses. Ils s'appliquent dans des cas très généraux, sont rapides, et peuvent s'appliquer à des problèmes de grande dimension.

2.1 Algorithme du point proximal

L'algorithme proximal le plus simple adapté au problème d'optimisation $\hat{x} \in \underset{x\in\mathbb{R}^N}{\operatorname{argmin}} f(x)$, quand f est une fonction convexe, est l'**algorithme du point proximal**. Il repose sur les propriétés de $\mathcal{M}_f$ et le lien qui existe (admis) entre l'enveloppe de Moreau $\mathcal{M}_f$ et l'opérateur proximal prox_f :

$$\nabla \mathcal{M}_{\gamma f}(x) = \frac{1}{\gamma}(x - \operatorname{prox}_{\gamma f}(x)). \tag{4}$$

Il est constitué de l'itération suivante (souvent connue sous le nom d'**itération proximale**) :

$$x_{k+1} = \operatorname{prox}_{\gamma f}(x_k) := \underset{y\in\mathbb{R}^N}{\operatorname{argmin}}\ f(y) + \frac{1}{2\gamma}\|y - x_k\|_2^2 \tag{5}$$

Pistes de réflexion :

- liens entre l'algorithme proximal (5) et descente de gradient :
  - ▷ via la relation (4)
  - ▷ dans le cas où f est différentiable, directement via la définition du prox
- interprétation et choix de γ (convergence de l'algorithme)
- implémentation de l'algorithme du point proximal pour la résolution de $\operatorname{argmin} f(x)$ pour $f(x) = |x|$
- limites de l'algorithme

Serge Mazauric
Éric Van Reeth

2.2 Algorithme de gradient proximal (*forward-backward*)

L'algorithme du point proximal s'applique à la minimisation d'une fonction de coût relativement simple, qu'il est difficile de minimiser telle quelle, mais plus facile à minimiser lorsqu'on lui rajoute une régularisation quadratique.
En général, les fonctions de coût qui nous intéressent sont définies comme la somme de plusieurs fonctions (attache aux données + régularisation). Autrement dit, étant données $f, g : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ deux fonctions convexes et propres, le problème d'optimisation que l'on regarde souvent est de la forme suivante :

$$\hat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \, f(x) + g(x), \tag{6}$$

Ce type de fonction de coût nécessite d'être capable de calculer $\operatorname{prox}_{f+g}$, ce qui est souvent très compliqué. Cependant, si f est différentiable, on peut utiliser l'**algorithme de gradient proximal** défini comme suit, pour $\gamma > 0$:

$$x_{k+1} = \operatorname{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k)) \tag{7}$$

Cette itération est donc la composition d'une étape explicite (descente de gradient sur f) et d'une étape implicite (prox sur g), ce pourquoi cet algorithme est aussi connu sous le nom d'**algorithme forward-backward**.

Pour étudier le principe des algorithmes proximaux, on se propose dans un premier d'aborder le problème en dimension 1, afin de résoudre le problème de minimisation suivant :

$$\hat{x} = \underset{x \in \mathbb{R}}{\operatorname{argmin}} \; x^2 + \lambda |x| \tag{8}$$

Par la suite, l'algorithme sera généralisé en dimension N, afin de résoudre le problème suivant :

$$\hat{x} = \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \; \|x\|_2^2 + \lambda \|x\|_1 \tag{9}$$

Pistes de réflexion :

- interprétation et choix de γ (convergence de l'algorithme)
- implémentation de l'algorithme de gradient proximal pour la résolution de (8) et (9)
- limites de l'algorithme

3 Utilisation des algorithmes proximaux pour la résolution de problèmes d'optimisation non-lisses

Dans cette section, on se propose d'utiliser la méthode du gradient proximal pour résoudre différents modèles de restauration d'image.
On rappelle que tous ces modèles sont construits à partir du modèle direct suivant :

$$z = Hx + n,$$

avec :

✔	$z \in \mathbb{R}^N$	observation
✔	$H \in \mathbb{R}^{N \times N}$	dégradation linéaire, e.g. un flou
✔		densité de probabilité du bruit
🔍	$\hat{x} \in \mathbb{R}^N$	le plus proche possible de $\bar{x}$ (inconnu)

Création des données :

Serge Mazauric
Éric Van Reeth

- création d'une vérité terrain 1D $\bar{x}$ au choix (e.g. Heaviside, polynôme, etc.)
- création d'une observation z associée

3.1 Résolution du modèle du LASSO

Le modèle du LASSO correspond au choix de $\Gamma = Id$ dans le problème (2), autrement dit :

$$\hat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \ \|Hx - z\|_2^2 + \lambda \|x\|_1 \tag{10}$$

Pistes de réflexion/travail demandé :

- solution attendue ?
- solution analytique du problème (10) ?
- quel algorithme à mettre en œuvre ?
- solution numérique du problème (10) ?

3.2 Résolution du modèle TV (*Total Variation*)

Dans le cas où $\Gamma \neq Id$ dans le problème (2), le modèle est appelé **Variation Totale**, abrégé en TV :

$$\hat{x} \in \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \ \|Hx - z\|_2^2 + \lambda \|\Gamma x\|_1 \tag{11}$$

L'algorithme de gradient proximal, bien que permettant la minimisation de la somme de deux fonctions, n'est pas adapté lorsque la fonction non-différentiable contient un opérateur linéaire. Autrement dit, dans le cas (11), il n'est pas possible dans ce cas-là de calculer l'opérateur proximal de $\|\Gamma \cdot\|_1$.

On pose alors une variable auxiliaire $y = Dx$, et on réécrit le problème sous la forme d'un problème contraint

$$\begin{cases} (\hat{x}, \hat{y}) \in \underset{x,y}{\operatorname{argmin}} \ \|Hx - z\|_2^2 + \lambda \|y\|_1 \\ \text{subject to } y = Dx \end{cases} \tag{12}$$

En pratique, la résolution du problème (11) est donc réalisée en cherchant une solution au problème

$$(\hat{x}, \hat{y}) \in \underset{x,y}{\operatorname{argmin}} \ \underbrace{\|Hx - z\|_2^2 + \lambda \|y\|_1 + \frac{\mu}{2} \|y - Dx\|_2^2}_{f(x,y)}, \tag{13}$$

Cette méthode est appelée *Alternated Direction Method of Multipliers* (ADMM).

Pour minimiser numériquement une telle fonction de coût, on adopte très souvent une stratégie de minimisation alternée. Autrement dit, à chaque itération, on fixe la valeur de y et on minimise par rapport à x, puis inversement. Cela conduit aux deux sous-problèmes suivants :

$$x_{k+1} = \operatorname{argmin}_{x \in \mathbb{R}^N} f(x, y_k) \tag{Px}$$
$$y_{k+1} = \operatorname{argmin}_{y \in \mathbb{R}^{2N}} f(x_{k+1}, y) \tag{Py}$$

Serge Mazauric
Éric Van Reeth

L'algorithme qui en résulte est le suivant :

```
1: Initialiser x0, y0;
2: Choix des paramètres pour la résolution de (Px);
3: Choix des paramètres pour la résolution de (Py);
4: For k=1:niter
5:   Résoudre (Px);
6:   Résoudre (Py);
7: end
```

Une attention particulière sera portée au choix des paramètres λ et μ, qui s'avère critique dans cet algorithme.

Pistes de réflexion/travail demandé :

- type de solution attendue ?
- solution analytique du problème (10) ?
- quel algorithme à mettre en œuvre ?
- solution numérique du problème (10) ?
- influence des paramètres λ et μ ?
- comparaison avec les régularisations dérivables ?

Serge Mazauric
Éric Van Reeth

A Mesure de la qualité des résultats

Lorsque l'on essaie d'estimer un phénomène physique à partir d'une observation dégradée de ce dernier (z), il peut être intéressant de quantifier objectivement la qualité de la reconstruction $\hat{x}$ en sortie de traitement.
Cela permet notamment d'aider au paramétrage d'un modèle. On trace la courbe d'une métrique en fonction de différentes valeurs de paramètres, et on extrait le paramètre optimisant le score choisi.

Il existe différents scores, dont trois sont particulièrement utilisés en image :

- l'Erreur Quadratique Moyenne (MSE),
- le Rapport Signal à Bruit (SNR),
- la Structural SIMilarity (SSIM).

Ces scores nécessitent en général d'**avoir accès à une référence** (on choisira souvent la vérité terrain $\bar{x}$).

MSE : Mean Squared Error. La MSE repose sur une différence de reconstruction pixel à pixel, entre l'image de référence $\bar{x}$ et l'image reconstruite $\hat{x}$:

$$MSE(I) = \|\hat{x} - \bar{x}\|_2^2.$$

SNR : Signal to Noise Ratio. Le SNR permet de quantifier la dégradation de l'image (a priori porteuse d'information utile) par un bruit (non informatif).
Le SNR (en dB) est défini comme le rapport de la puissance du signal P_s sur celle du bruit P_B :

$$SNR(s) = 10\ \log_{10}\left(\frac{P_s}{P_B}\right)\ \ dB.$$

Dans le cadre d'un problème inverse, nous n'avons en général par accès à ces deux informations séparément. Aussi, on calcule le SNR du signal estimé $\hat{x}$ relativement à une référence $\bar{x}$:

$$SNR(s) = 10\ \log_{10}\left(\frac{\|\bar{x}\|_2^2}{\|\hat{x} - \bar{x}\|_2^2}\right)\ \ dB.$$

SSIM : Structural SIMilarity. Le SSIM permet quant à lui de mesurer la similarité de structure par rapport à une image de référence $\bar{x}$. Il essaie de mieux tenir compte de la perception de l'oeil humain, a priori plus sensible aux changements de structure que de couleur.

Ce score est calculé sur plusieurs patchs de l'image. Il repose sur trois métriques de comparaison : la luminance l, le contraste c et la structure s. Pour chaque patch p, le SSIM est défini comme le produit de ces trois valeurs :

$$SSIM(p_{\hat{x}}, p_{\bar{x}}) = l(p_{\hat{x}}, p_{\bar{x}}) \cdot c(p_{\hat{x}}, p_{\bar{x}}) \cdot s(p_{\hat{x}}, p_{\bar{x}}).$$

Serge Mazauric
Éric Van Reeth

B Index des fonctions

Normes.

Si `x` est un **vecteur** :

`norm(x)`	calcule la norme 2 d'un vecteur $\|x\|_2 = \sqrt{\sum_i x_i^2}$
`norm(x,2)`	calcule la norme 2 d'un vecteur $\|x\|_2 = \sqrt{\sum_i x_i^2}$

Si A est une **matrice** :

`norm(A)`	calcule la norme d'opérateur $\|\|\|A\|\|\| = \max(\mathrm{SVD}(A))$
`norm(A,2)`	calcule la norme d'opérateur $\|\|\|A\|\|\| = \max(\mathrm{SVD}(A))$
`norm(A,'fro')`	calcule la norme de Frobenius $\|A\|_2 = \sqrt{\sum_{i,j} A_{ij}^2}$

Remarque : dans le cas d'une matrice, `norm(A,'fro')` = `norm(A(:),2)`.

Matrices.

`D = matGamma(s,'gradient')`	Crée la matrice D de gradient (`s = size(x)`)
`L = matGamma(s,'laplacian')`	Crée la matrice L de Laplacien (`s = size(x)`)

Opérateurs (à privilégier en Octave).

`y = opD(x)`	Calcule l'opération matricielle Dx (x **non** vectorisé)
`y = opDt(x)`	Calcule l'opération matricielle $D^\top x$ (x **non** vectorisé)
`y = opL(x)`	Calcule l'opération matricielle Lx (x **non** vectorisé)
`y = opLt(x)`	Calcule l'opération matricielle $L^\top x$ (x **non** vectorisé)

On créera, à partir de ces fonctions, les fonctions anonymes suivantes en début de script :

```
D = @(x) opD(x);
Dt = @(x) opDt(x);

L = @(x) opL(x);
Lt = @(x) opLt(x);

H = @(x) opH(x,type,N);
Ht = @(x) opHt(x,type,N);
```

Calcul des normes d'opérateur.

À partir des fonctions anonymes créées à la section précédente :

`lip = lipschCst(D,Dt,1)`	Calcule la constante de Lipschitz de l'opérateur D en 1D
`lip = lipschCst(D,Dt,2)`	Calcule la constante de Lipschitz de l'opérateur D en 2D
`lip = lipschCst(L,Lt,1)`	Calcule la constante de Lipschitz de l'opérateur L en 1D
`lip = lipschCst(L,Lt,2)`	Calcule la constante de Lipschitz de l'opérateur L en 2D
`lip = lipschCst(H,Ht,1)`	Calcule la constante de Lipschitz de l'opérateur H en 1D
`lip = lipschCst(H,Ht,2)`	Calcule la constante de Lipschitz de l'opérateur H en 2D

Serge Mazauric
Éric Van Reeth