TP sur les problèmes inverses

Sandrine Anthoine, Valentin Emiya - Aix-Marseille Université

M2 IAAA - UE SOAP - 2024-2025

Ce TP a pour objectif de prendre en main un problème inverse élémentaire de déconvolution 1D, l'algorithme de référence ISTA et de mener une démarche expérimentale classique pour maîtriser la résolution de ce problème avec cet algorithme. Il s'agit donc d'acquérir une méthodologie scientifique solide sur un contexte nouveau, dans un cas simple.

Les données se présentent sous la forme d'un vecteur

$$\boldsymbol{y} = \boldsymbol{h} * \boldsymbol{x}^o + \boldsymbol{b} \tag{1}$$

où $\boldsymbol{y} \in \mathbb{R}^N$ est le signal observé, $\boldsymbol{h} \in \mathbb{R}^N$ est un filtre connu, $\boldsymbol{x}^o \in \mathbb{R}^N$ est le signal original inconnu et $\boldsymbol{b} \in \mathbb{R}^N$ est du bruit additif inconnu. On suppose par ailleurs que le signal $\boldsymbol{x}$ original est parcimonieux, c'est-à-dire composés de quelques pics et de nombreux zéros.

L'objectif est de retrouver au mieux $\boldsymbol{x}^o$ à partir de $\boldsymbol{y}$ et $\boldsymbol{h}$. On remarque que l'équation (1) peut se réécrire sous la forme

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^o + \boldsymbol{b} \tag{2}$$

où $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ est une matrice de la forme

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{h}[N/2] & \boldsymbol{h}[N/2-1] & \dots & \boldsymbol{h}[0] & \boldsymbol{h}[N-1] & \dots & \boldsymbol{h}[N/2+1] \\ \boldsymbol{h}[N/2+1] & \boldsymbol{h}[N/2] & \boldsymbol{h}[N/2-1] & \dots & \boldsymbol{h}[0] & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \boldsymbol{h}[N-1] \\ \boldsymbol{h}[N-1] & \boldsymbol{h}[N-2] & \ddots & \ddots & \ddots & \ddots & \boldsymbol{h}[0] \\ \boldsymbol{h}[0] & \boldsymbol{h}[N-1] & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \boldsymbol{h}[N/2-1] \\ \boldsymbol{h}[N/2-1] & \dots & \boldsymbol{h}[0] & \boldsymbol{h}[N-1] & \dots & \boldsymbol{h}[N/2+1] & \boldsymbol{h}[N/2] \end{bmatrix} \tag{3}$$

Le code fourni avec cet énoncé permet de générer un filtre gaussien, une matrice $\boldsymbol{A}$ de la forme ci-dessous, d'ajouter du bruit en contrôlant le rapport signal-à-bruit (SNR).

Exercice 1 (Génération de la matrice)**.** Utilisez le code fourni pour générer un filtre gaussien d'écart-type 2 de taille $N = 128$ via `get_gaussian_filter` et une matrice $\boldsymbol{A}$ de la forme ci-dessous via `filter2matrix`. Affichez la matrice obtenue avec `plt.imshow` et assurez-vous que le résultat correspond à la forme de l'éq. (3).

Exercice 2 (Génération du signal bruité)**.** Générez un signal $\boldsymbol{x}^o$ avec $K = 5$ valeurs non-nulles tirées au hasard, un vecteur $\boldsymbol{y}$ (éq. (2)) avec un SNR de 20dB (avec `add_noise` pour ajouter le bruit) et affichez sur une même figure le signal original $\boldsymbol{x}^o$ et le signal $\boldsymbol{y}$.

On souhaite maintenant trouver une estimation de $\boldsymbol{x}$ en résolvant le problème du Lasso :

$$\underset{\boldsymbol{x}}{\text{Minimiser}}\, f(\boldsymbol{x}) + \alpha \|\boldsymbol{x}\|_1 \text{ où } f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{4}$$

L'algorithme 1 détaille le pseudo-code d'ISTA qui permet de résoudre le problème. On rappelle la fonction de seuillage doux : $S_\tau(x) = \text{sign}(x)\max(|x| - \tau, 0)$.

Algorithme 1 ISTA

Entrée(s): ∇f, $\boldsymbol{x}_0$, $\eta > 0$, $\alpha > 0$, $n_{\text{iterations}}$

$\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
pour $i = 1$ à $n_{\text{iterations}}$ **faire**
  $\boldsymbol{z} \leftarrow \boldsymbol{x} - \eta \nabla f(\boldsymbol{x})$
  $\boldsymbol{x} \leftarrow S_{\alpha\eta}(\boldsymbol{z})$
fin pour
renvoyer $\boldsymbol{x}$

Exercice 3 (Codage de l'algorithme ISTA)**.** Codez et affichez la fonction de seuillage doux. Codez ensuite l'algorithme sous la forme d'une fonction

```
ista_l1(f, grad_f, x0, step_size, alpha, n_iterations, return_iterates)
```

qui prend en arguments le gradient `grad_f` de l'attache aux données f, un vecteur initial `x0`, un pas de gradient `step_size`, un paramètre de régularisation `alpha`, un nombre d'itérations `n_iterations`, un booléen `return_iterates`, et renvoie la solution estimée $\boldsymbol{x}$ ou la liste complète des itérées $[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{\text{n\_iterations}}]$ (selon la valeur de `return_iterates`). Le pas $\eta = \frac{1}{\|\boldsymbol{A}\|_2^2}$ garantit une bonne décroissance.

Appliquez votre algorithme aux données générées à l'exercice 2, en prenant une valeur arbitraire pour α, par exemple $\alpha = 1$. Tracez la valeur de la fonctionnelle $f(\boldsymbol{x}) + \alpha \|\boldsymbol{x}\|_1$ en fonction des itérations pour vérifier la convergence. Tracez le signal $\boldsymbol{x}$ obtenu en le superposant aux autres signaux sur la figure de l'exercice 2.

Exercice 4 (Variation de l'hyperparamètre α)**.** On note $\|\boldsymbol{x}\|_0$ le nombre de valeurs non-nulles dans un vecteur $\boldsymbol{x}$. Faites varier α en prenant plusieurs valeurs selon une échelle logarithmique par exemple entre 10^{-9} et 10^2, puis pour chaque solution $\boldsymbol{x}$ obtenue, représentez sur deux figures différentes $f(\boldsymbol{x})$ et $\|\boldsymbol{x}\|_0$ en fonction de α, pour chaque $\boldsymbol{x}$ renvoyé par l'algorithme. Commentez les figures. Pour quelle valeur de α a-t-on $\|\boldsymbol{x}\|_0 = \|\boldsymbol{x}^o\|_0 = 5$?

Exercice 5 (Solution avec régularisation optimale)**.** Fixez α à la valeur trouvée à la fin de l'exercice 4, tracez les mêmes figures qu'à l'exercice 3 avec cette valeur et analysez les résultats.

Exercice 6 (Amélioration du temps de calcul.)**.** Sans optimiser α, faîtes varier la taille du signal N en prenant des puissances de 2 successives, mesurez le temps de calcul de l'algorithme avec `time.process_time` et tracez ce temps en fonction de N.

On remarque que l'algorithme effectue des produits matrice-vecteur par $\boldsymbol{A}$ et $\boldsymbol{A}^T$, avec une complexité en $\mathcal{O}(N^2)$. On souhaite aller plus vite, en utilisant une convolution. Vous allez créer une nouvelle version de la solution (dupliquez votre code et choisissez des noms différents pour garder l'ancienne version). Dans le cas présent, le filtre est de taille N : le calcul rapide peut se faire en passant par la transformée de Fourier rapide. Dans un premier temps, remplacez le produit par $\boldsymbol{A}$ par un filtrage. Dans un second temps, on veut aussi un calcul rapide pour remplacer la multiplication par $\boldsymbol{A}^T$. Pour cela, il faut considérer $\boldsymbol{A}^T$ comme l'adjoint de $\boldsymbol{A}$: l'opérateur linéaire $\boldsymbol{B}$ est l'adjoint de $\boldsymbol{A}$ si pour tous vecteur $\boldsymbol{x}, \boldsymbol{u}$ on a $\langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{u} \rangle = \langle \boldsymbol{x}, \boldsymbol{B}\boldsymbol{u} \rangle$. Dans le cas d'une matrice, l'adjoint est la transposée. Dans le cas, d'une convolution il faut faire le calcul. Faîtes le calcul, en considérant à la fois l'égalité $\langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{u} \rangle = \langle \boldsymbol{x}, \boldsymbol{A}^T\boldsymbol{u} \rangle$ et le fait que $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{h} * \boldsymbol{x}$. Ensuite, codez le produit par $\boldsymbol{A}^T$ comme un filtrage, mesurez les temps de calculs et comparez-les à ceux de votre implémentation initiale.

Exercice 7 (Variante avec différentiation automatique)**.** Créez une variante de la première solution (qui utilise une matrice A) où vous remplacez votre implémentation de ∇f par un gradient calculé automatiquement par Jax. Vous devez retrouver les mêmes performances. Qu'en est-il du temps de calcul ? Que se passe-t-il si vous faîtes la même chose avec l'autre solution (qui utilise une convolution) ?

Exercice 8 (Variante avec compilateur JIT)**.** Poursuivez en utilisant le compilateur JIT de Jax : obtenez-vous des gains en temps de calcul ?

Exercice 9 (Réflexion)**.** Comme vu en cours, un filtre gaussien 2D sur une image génère du flou. Réfléchissez à comment coder une déconvolution 2D pour déflouter une image.