Débruitage par minimisation de la variation totale

Hugo Lelièvre - Alexia Grenouillat

January 2023

1 Introduction

Il existe de nombreux algorithmes permettant de débruiter une image, qui peuvent être plus ou moins performants en fonction des caractéristiques de l'image et de celles du bruit. La minimisation de la variation totale est adaptée au cas des images avec une faible variation totale (norme ℓ_1 du gradient). Dans ce cas, on peut estimer l'image en minimisant une fonctionnelle. Attention, on parle ici du gradient de l'image, pas d'une fonctionnelle qui associe une image à un réel.

Le gradient d'une image est un champ de vecteur avec deux composantes, l'une verticale, l'autre horizontale. Ces champs se calculent de manière classique par différences finies.

Si on note $y = x^0 + b$ les observations où x^0 est l'image à estimer et b un bruit additif, cette fonctionnelle sera de la forme

$$f(x) = \frac{1}{2}\|x - y\|^2 + \lambda\|\nabla x\|_1 \tag{1}$$

Notre problème de minimisation est donc le suivant :

$$\textbf{Problème primal :}\ \min_x \frac{1}{2}\|x - y\|^2 + \lambda\|\nabla x\|_1 \tag{2}$$

La norme ℓ_1 du gradient n'est pas différentiable. On aimerait donc utiliser un algorithme Forward-Backward avec $f_1(x) = \frac{1}{2}\|x - y\|^2$ comme terme différentiable.

Cependant, on aurait alors besoin de calculer l'opérateur proximal de la fonction $f_2(x) = \lambda\|\nabla x\|_1$.

Or il n'existe pas de formule pour un tel opérateur. Ce qui implique aussi que réaliser un Forward-Backward avec le terme quadratique comme terme différentiable ne peut pas fonctionner.

Il existe cependant un moyen d'utiliser l'algorithme FB pour résoudre ce problème : on procède par dualisation.

La dualisation qui utilise la conjugué de Fenchel Rockafellar permet de montrer qu'on peut associer au problème précédent, le problème d'optimisation suivant

$$\textbf{Problème dual :}\ \min_p \frac{1}{2}\|div(p) + y\|^2 + \iota_{\mathcal{B}_{\infty,\lambda}}(p) \tag{3}$$

où :

- p est un champ de gradient ;
- $\mathcal{B}_{\infty,\lambda}$ est la boule ℓ_∞ de rayon λ ;
- div est l'opérateur de divergence.

La présence de la divergence est ici due au fait que $-div$ est l'opérateur adjoint du gradient ∇. La norme ∞ est ici présente car c'est la norme duale de la norme l_1.

Si p^* est la solution du problème précédent alors $x^* = y + div(p^*)$ est la solution du problème initial.

L'avantage de ce second problème est qu'il peut être résolu par l'algorithme FB dans la mesure où le terme non différentiable est une indicatrice d'un ensemble sur lequel on sait projeter :

- $\nabla\frac{1}{2}\|div(p) + y\|^2 = (-Div(p) - y)$;
- $prox(\iota_{\mathcal{B}_{\infty,\lambda}}(p)) = Proj_{B_{\infty,\lambda}}(p)$;
- $x_{n+1} = Proj_{B_{\infty,\lambda}}(x_n + h * (Div(x_n) + y)) = Tx_n$

2 Application aux images

Pour commencer, nous avons choisi une image de référence : Lenna en noir et blanc. Nous avons ajouté un bruit gaussien à cette image avant d'y appliquer nos algorithmes FB et FISTA.

On observe sur l'image reconstruite un effet singulier : le débruitage a aplati les couleurs. Les dégradés ont disparu et ont laissé place à des "escaliers" de couleurs, on a des paliers de couleurs uniformes. Cela n'est pas étonnant étant donné la nature du problème de minimisation de départ. Pour minimiser la norme ℓ_1 du gradient, on réduit fortement les dégradés et on favorise les couleurs uniformes.

Le paramètre λ permet de calibrer les aplats de couleurs en fonction du résultat attendu. Plus λ est grand et plus l'importance accordée à la norme ℓ_1 du gradient est grande, et donc plus l'image sera aplatie.

Nous avons décidé d'étendre nos algorithmes aux images couleurs en procédant à un débruitage canal par canal.

Pour les tests, nous avons cette fois choisi 3 images différentes :

- une image type cartoon de Dumbo ;
- la même image mais avec des couleurs beaucoup plus travaillées, des effets d'ombres et de lumière ;
- une photo d'un éléphant.

Pour un bruit relativement faible ($\sigma = 20$), le résultat est très bon pour les 2 premières images. Sur la troisième, bien qu'on reconnaisse parfaitement l'animal qui est dessus, l'éléphant subit un effet de lissage et on perd beaucoup de détails. Dans le cas de photographies comme celle-ci, il vaut mieux essayer de garder un λ relativement petit pour ne pas donner trop d'importance à la norme ℓ_1 du gradient. On conserva mieux les détails de la photo, quitte à garder un petit peu de bruit résiduel.

Pour des images de type cartoon, ce débruitage est tout à fait adapté et donne des résultats très satisfaisant. Cependant, quand le bruit est très grand ($\sigma = 100$), il est compliqué d'avoir un bon résultat.

Nous avons essayé 3 valeurs de λ différentes (1, 50 et 200) pour comparer les résultats. Pour $\lambda = 200$, l'importance accordé à la norme ℓ_1 du gradient est énorme. Il n'y a plus de bruit, mais on se retrouve avec des zones très uniformes, et les traits de contours deviennent presque inexistants. Le résultat donne un effet de flou peu agréable à l'oeil.

Pour $\lambda = 1$, l'image est très nette, mais toujours très bruitée, les couleurs ne sont pas uniformes. Le meilleur rendu semble être le compromis des 2 avec $\lambda = 50$. On a toujours du bruit, mais beaucoup moins que pour $\lambda = 1$, tout en ayant une image assez nette.

3 Application aux GIFS

Les GIFS étant des successions d'images, nous avons décidé d'étendre notre algorithme à ces derniers en traitant chaque frame une à une. Cette méthode s'applique donc à chaque frame indépendamment des autres, sans tenir compte de l'aspect temporel des images. Or, les frames successives sont souvent très ressemblantes, il serait donc possible d'améliorer cette méthode en gardant l'information des frames précédentes.

Pour tester l'adaptation de notre algorithme, nous avons décidé d'utiliser un GIF de Donald qui est de type cartoon. Il correspond donc parfaitement au type de GIF que nous pouvons débruiter efficacement.

Une fois le GIF bruité en rajoutant un bruit gaussien indépendant à chaque frame, on s'est rendu compte que l'affichage du fichier GIF lors d'une ouverture classique ne donnait pas le même résultat que lorsque l'on regardait les frames une à une sur python. Seule la première frame semble être bruitée, et les autres semblent n'avoir du bruit qu'aux endroits où il y a du mouvement. Il doit exister dans le code d'affichage des gifs un algorithme de compression qui tient compte de la temporalité des frames. Les frames affichées ne correspondent pas exactement à celles codées. Il est possible que pour accélérer le chargement et l'affichage des gifs, on garde le même squelette d'une image à l'autre, en ne modifiant que les zones où il y a du mouvement. Cela donne un effet de débruitage supplémentaire qui rend plus difficile à voir notre débruitage avec FB.

4 Annexe : Rappels sur la descente de gradient explicite, sur Forward-Backward et FISTA

4.1 Forward-Backward

Si $F = f + g$ est une fonction convexe composite, somme de deux fonctions convexes, f différentiable à gradient L-Lipschitz et g une fonction convexe dont on sait calculer l'opérateur proximal, l'algorithme Forward-Backward s'écrit

$x_{n+1} = prox_{hg}(x_n - h\nabla f(x_n)) = Tx_n \quad$ avec $T := prox_{hg} \circ (Id - h\nabla f)$

On peut montrer que la suite de terme général $F(x_n) - F(x^*)$ est décroissante et de plus que $F(x_n) - F(x^*) \leq \frac{2\|x_0 - x^*\|^2}{hn}$

Cette vitesse en $\frac{1}{n}$ est optimale au sens où il n'est pas possible de trouver des bornes qui décroissent en $\frac{1}{n^\delta}$ avec $\delta > 1$ pour toutes les fonctions convexes. Cela étant on peut montrer que si $h < \frac{1}{L}$ on a en fait $F(x_n) - F(x^*) = o\left(\frac{1}{n}\right)$ et que cette vitesse est même géométrique si la fonction f est fortement convexe, dans le cas différentiable.

4.2 FISTA

L'accélération de la descente de gradient proposée Yurii Nesterov en 1984 et adapatée à FB sous le nom de FISTA par Beck et Teboulle en 2009 est d'une mise en oeuvre très simple.

Il suffit d'effectuer la descente de gradient en un point décalé de x_n avec un pas $h < \frac{1}{L}$ (attention cette borne est plus contraignante que pour la descente de gradient classique) :

$$x_{n+1} = T(x_n + \alpha_n(x_n - x_{n-1}))$$

où la suite α_n est bien choisie et T soit l'opérateur de descente de gradient explicite, ou l'opérateur associé au FB. On parle de méthode inertielle car cette méthode utilise un terme dit de "mémoire" ou inertiel qui exploite la dernière direction de descente.

On peut prendre $\alpha_n = \frac{n-1}{n+a-1}$ avec $a > 3$ dans ce cas, on a :

$$F(x_n) - F(x^*) \leq \frac{(a-1)^2\|x_0 - x^*\|^2}{2h(n+a)^2}$$

et en plus $F(x_n) - F(x^*) = o\left(\frac{1}{n^2}\right)$ et on a convergence de la suite $(x_n)_{n\geq 1}$. On peut noter que dans ce cas, la première étape est une simple sans inertie ($\alpha_1 = 0$) et donc $x_1 = T(x_0)$. L'inertie apparait pour le calcul de x_2. Le choix originel de Nesterov est très proche du choix $a = 3$.

La suite de terme général $F(x_n) - F(x^*)$ n'est pas nécessairemment décroissante comme dans le cas de FB ou de la descente de gradient. Les bornes données précédemment sont des bornes mais ne reflètent pas la décroissance réelle. Dans la pratique, FISTA est quand même plus rapide que FB.