Restauration d'image à l'aide de l'ADMM

Céline Meillier

Ce TP concerne la problématique de restauration d'images dans le contexte d'un problème de minimisation à résoudre. Le choix de l'algorithme s'est porté sur l'ADMM (alternating direction method of multipliers) qui permet de résoudre les problèmes d'optimisation convexe en décomposant le problème en une somme de problèmes de minimisation plus simples que l'on résout simultanément en alternant les étapes de mise à jour des variables à estimer.

Quelques conseils avant de commencer

- On rappelle qu'un compte rendu de TP comporte une introduction et une conclusion et que cette conclusion ne doit pas être la copie de l'introduction au passé.
- On rappelle également que les figures doivent comporter un titre, et que, le cas échéants, les axes des graphes doivent avoir un nom (et une unité quand elle est connue).
- Créer un script python (notebook ou script .py au choix), ne faites pas tout le TP en lignes de commandes.

Le compte-rendu de TP est à déposer sur EOLE (site pédagogique) au plus tard le 2 juillet 2017 à 23h59.

1 Altération d'une image

1.1 Flou de bougé

On utilisera tout au long du TP la scène présentée sur la figure 1. Il s'agit d'une image en niveaux de gris (normalisés entre 0 et 1) de taille 64×64.

FIGURE 1 – Illustration de la scène observée.

On suppose que l'on prend en photo la scène illustrée sur la figure 1. Lorsque l'on prend une photo, l'obturateur de l'appareil photo laisse passer la lumière durant un petit intervalle de temps. Si durant cet intervalle de temps, l'appareil (ou la scène) bouge, cela entraîne le phénomène bien connu de la photo floue. On souhaiterait pouvoir restaurer l'image afin de la rendre le plus net possible. L'effet d'un flou de bougé peut s'exprimer comme une convolution par un filtre contenant l'information sur le mouvement de l'appareil photo lors de l'acquisition. La réponse spatiale de ce filtre est illustrée sur la figure 2.

1.2 Modélisation mathématique

On notera $\mathbf{X} \in \mathcal{R}^{K \times L}$ l'image 2D originale que l'on cherche à reconstruire, avec $K = L = 64$ pixels dans le cas de la scène observée présentée sur la figure 1. On notera $P = K \times L$ le nombre total de pixels dans

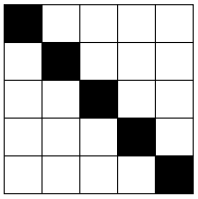

FIGURE 2 – Réponse spatiale du filtre représentant le mouvement de l'appareil photo ayant entraîné le flou de bougé.

l'image. De même $\mathbf{Y}$ représente l'observation en 2D, *i.e.* l'image $\mathbf{X}$ détériorée par le flou de bougé. On peut modéliser mathématiquement l'effet du flou de bougé sur l'image observée par l'équation :

$$\mathbf{Y} = \mathcal{H} * \mathbf{X} \tag{1}$$

où $\mathcal{H} \in \mathcal{R}^{M \times N}$ est la réponse spatiale du filtre de flou de bougé. Dans le cas du filtre utilisé dans ce TP, illustré sur la figure 2, on a $M = N = 5$ pixels. On préférera utiliser la notation matricielle pour exprimer ce produit de convolution sur les images vectorisées :

$$\mathbf{y} = \mathbf{H}\mathbf{x} \tag{2}$$

où $\mathbf{H} \in \mathbb{R}^{P \times P}$ est la matrice de convolution, traduisant l'opération de convolution par $\mathcal{H}$ après vectorisation de l'image.

1.3 Illustration

Manipulation 1

Toutes les données à utiliser dans ce TP sont stockées dans le fichier "data.pk" qui contient :

- 'original' : tableau 2D contenant l'image originale que l'on va chercher à reconstruire.
- 'filtre' : tableau 2D contenant le filtre de flou de bougé.
- 'H' : représentation matricielle de la convolution par le filtre de flou de bougé (matrice $\mathbf{H}$ précalculée pour traiter une image de taille 64×64 pixels).
- 'observations' : tableau 2D contenant les observations.
- 'T' : tableau 2D contenant la matrice $\mathbf{T}$ qui intervient dans le modèle (3).

Le format *pickle* (.pk) permet de conserver la structure de l'objet python sauvegardé (tout type d'objet, y compris une instanciation de classe définie par l'utilisateur, peut être stocké dans un fichier .pk). Les instructions suivantes pourront être utile durant ce TP :

```
import pickle

# Ouvrir des données stockées dans un fichier .pk
data = pickle.load(open('data.pk','rb'), encoding='latin1')

# Charger par exemple la matrice H
H = data.get('H')

# Sauvegarder des données dans un fichier .pk
pickle.dump({'obj1': obj1, 'obj2':obj2}, open('filename.pk','wb'), pickle.HIGHEST_PROTOCOL)
```

Observer l'effet du flou de bougé sur les observations avec `matplotlib.pyplot.imshow(image, cmap='gray')`.

2 Restauration d'image

2.1 Formulation du problème de restauration comme un problème d'optimisation sous contrainte

Le problème de restauration d'image est un problème mal posé, il faut donc avoir recours à une régularisation de la solution. Dans ce TP nous allons introduire deux termes de régularisation : une régularisation de type variation totale anisotrope et une contrainte de positivité puisque l'on mesure des intensités (grandeurs homogènes à des énergies, donc positives). On obtient alors le problème suivant :

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y}-\mathbf{H}\mathbf{x}\|_2^2 + \beta\|\mathbf{T}\mathbf{x}\|_1 + \imath_{[0,1]^P}(\mathbf{x}), \tag{3}$$

où $\imath_{[0,1]^P}(\mathbf{x})$ traduit la contrainte que les pixels de l'image $\mathbf{x}$ à reconstruire ont une valeur entre 0 et 1. S'il existe un élément $x_i < 0$ ou un élément $x_i > 1$ dans le vecteur $\mathbf{x}$ alors $\imath_{[0,1]^P}(\mathbf{x}) = +\infty$, sinon, si tous les éléments de $\mathbf{x}$ sont dans $[0,1]$, $\imath_{[0,1]^P}(\mathbf{x}) = 0$. La matrice $\mathbf{T}$ code la variation totale définie par :

$$\|\mathbf{T}\mathbf{x}\|_1 = \sum_{i,j} \Big(\Big|\mathbf{X}[i+1,j] - \mathbf{X}[i,j]\Big| + \Big|\mathbf{X}[i,j+1] - \mathbf{X}[i,j]\Big| \Big) \tag{4}$$

La matrice $\mathbf{T}$ est rectangulaire, $\mathbf{T} \in \mathbb{R}^{(2P-K-L)\times P}$, elle contient des coefficients égaux à -1, 0 et 1 pour coder les décalages horizontaux et verticaux. Cette matrice est fournie dans le fichier "data.pk".

2.2 Algorithme ADMM

L'algorithme ADMM permet de minimiser la somme de deux fonctions convexes $f + g$ par duplication de la variable $\mathbf{x}$ (ou d'une transformée de $\mathbf{x}$) que l'on cherche à estimer. On rappelle que l'algorithme ADMM permet de résoudre des problèmes de la forme :

$$\begin{aligned} &\min_{\mathbf{x}} \; f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{subject to :} \;\; \mathbf{A}\mathbf{x} = \mathbf{z} \end{aligned} \tag{5}$$

où $g(\mathbf{z})$ résume les contraintes sur $\mathbf{x}$ et la variable $\mathbf{z}$ permet de dupliquer $\mathbf{x}$ (ou une transformée de $\mathbf{x}$).

Préparation 1

Le problème de restauration d'image (3) contient une contrainte sur l'image $\mathbf{x}$ à reconstruire : $\imath_{[0,1]^P}(\mathbf{x})$ et une pénalisation sur $\mathbf{T}\mathbf{x}$. En introduisant les variables $\mathbf{z}_1 = \mathbf{x}$ et $\mathbf{z}_2 = \mathbf{T}\mathbf{x}$ avec $\mathbf{z} = [\mathbf{z}_1^T, \mathbf{z}_2^T]^T$, montrer que le problème de restauration d'image (3) peut s'écrire sous la forme (5).

Exprimer alors les fonctions, f et g, le vecteur $\mathbf{z}$ et la matrice $\mathbf{A}$.

L'ADMM consiste à minimiser le Lagrangien augmenté du problème (5) en fonction des variables $\mathbf{x}$ et $\mathbf{z}$ successivement. Le Lagrangien augmenté associé au problème (5) s'écrit :

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}^k, \boldsymbol{\gamma}^k) = f(\mathbf{x}) + g(\mathbf{z}) + \boldsymbol{\gamma}^T(\mathbf{A}\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{z}\|_2^2 \tag{6}$$

où $\boldsymbol{\gamma}$ est la variable duale (le multiplicateur de Lagrange). Dans notre cas $\boldsymbol{\gamma} = [\gamma_1, \gamma_2]^T$.

Préparation 2

Ecrire le Lagrangien augmenté $\mathcal{L}_\rho$ du problème de restauration d'image (3).

L'ADMM est un algorithme itératif permettant de mettre à jour successivement les variables $\mathbf{x}$, $\mathbf{z}$ et $\boldsymbol{\gamma}$. A l'itération $k+1$:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}}{\operatorname{argmin}}\ \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}^k, \boldsymbol{\gamma}^k) \tag{7}$$

$$\begin{aligned}\mathbf{z}^{k+1} &= [{\mathbf{z}_1^{k+1}}^T, {\mathbf{z}_2^{k+1}}^T]^T \\ &= \underset{\mathbf{z}}{\operatorname{argmin}}\ \mathcal{L}_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \boldsymbol{\gamma}^k)\end{aligned} \tag{8}$$

$$\boldsymbol{\gamma}^{k+1} = \boldsymbol{\gamma}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}). \tag{9}$$

Préparation 3

Ecrire l'étape de minimisation de $\mathcal{L}_\rho$ selon la variable $\mathbf{x}$.

Préparation 4

Ecrire l'étape de minimisation de $\mathcal{L}_\rho$ selon la variable $\mathbf{z}_1$.

Ecrire l'étape de minimisation de $\mathcal{L}_\rho$ selon la variable $\mathbf{z}_2$.

Préparation 5

Ecrire l'étape (9) de mise à jour des variables duales $\boldsymbol{\gamma}_1$ et $\boldsymbol{\gamma}_2$.

3 Implémentation de l'ADMM pour résoudre le problème de restauration d'image

Nous allons maintenant implémenter l'algorithme ADMM pour restaurer l'image flou contenue dans le fichier `observations.pk`. Dans un premier temps, on fixera les différents paramètres de l'ADMM aux valeurs suivantes :

- $\rho = 0.05$,
- $\beta = 0.01$,
- $k_{max} = 100$.

Manipulation 2

A partir des équations développées dans la section précédente, écrire un script python permettant de résoudre le problème (3) à l'aide de l'ADMM.

On rappelle que la vectorisation d'une image de taille $K \times L$ en python s'effectue de la façon suivante :

```
K,L = im.shape
vec = numpy.reshape(im, K*L)
```
ou
```
vec = numpy.ravel()
```

Pour la résolution des systèmes linéaires avec une matrice définie positive et creuse, il peut être intéressant d'utiliser la méthode du gradient conjugué implémentée dans `scipy.sparse.linalg.cg`.

L'ADMM minimise le Lagrangien augmenté associé au problème de restauration d'image (3). Afin de caractériser la qualité de la restauration, on peut s'intéresser à l'erreur quadratique moyenne à l'itération k :

$$\mathrm{EQM}^k = \frac{1}{P}\sum_{p=1}^{P}\left(\mathbf{x}^k - \mathbf{x}\right) \tag{10}$$

ou encore le biais $\mathrm{B}(\mathbf{x}^k)$:

$$\mathrm{B}(\mathbf{x}^k) = \mathbf{x}^k - \mathbf{x} \tag{11}$$

Manipulation 3

Evaluer à chaque itération la valeur du Lagrangien augmenté et de l'erreur quadratique moyenne. Commenter les courbes obtenues.

Afficher le biais obtenu pour l'estimation finale.

Nous souhaitons maintenant évaluer l'influence des paramètres ρ et β sur la convergence et sur la qualité de la reconstruction.

Manipulation 4

Faire varier le paramètre ρ dans l'intervalle $[0, 0.1]$ et β dans $[0, 0.02]$.
Analyser l'influence de chacun de ces paramètres sur la solution (qualité de la reconstruction, biais, EQM, etc) et sur le problème d'optimisation (convergence, rapidité, etc).

Pour aller plus loin (facultatif)

Ajouter un bruit additif gaussien de variance 0.1 à l'image et étudier son influence sur la reconstruction.
