Instructions pour T.P. déconvolution d'image

F. Soulez

FOCUS, Nov 2020

Environnements de travail

Le TP consiste à compléter les ligne manquante d'un notebook Jupyter en Python.

Pour la déconvolution d'image, vous avez besoin d'une image de l'objet (*e.g.*, `saturn.fits`) et d'une image d'une source de référence (*e.g.*, `saturn_psf.fits`).

1 Filtre de Wiener approché

Dans un premier temps, nous allons utiliser une version simplifiée du filtre de Wiener :

$$\widehat{g}_k = \frac{\widehat{h}_k^\star}{\left|\widehat{h}_k\right|^2 + \widehat{q}_k} \quad \text{avec} \quad \widehat{q}_k = \frac{\mathbb{E}|\widehat{n}_k|^2}{\mathbb{E}|\widehat{x}_k|^2} > 0$$

1. Expliquer ce que représentent les différentes variables de la formule ci-dessus.
2. On modelise $\widehat{\boldsymbol{q}}$ comme une loi de puissance:

   $$\widehat{q}_k = \alpha \left|u_k\right|^\beta$$

   Quel est la forme attendue de $\mathbb{E}|\widehat{x}_k|^2$, à quels *a priori* sur l'objet cela corespond-il, quel doit être le signe de β?
3. Coder la fonction `wiener_filter(data,psf,q)` dans le fichier `wiener.py` effectuant ce filtrage et régler visuellement la valeur de α de façon optimale en fixant $\beta = 2$.

2 Approche inverse par méthode itérative

Nous allons résoudre le problème de la déconvolution en minimisant la fonction de coût :

$$\phi(\boldsymbol{x}) = \phi_{\mathsf{data}}(\boldsymbol{x}) + \alpha\, \phi_{\mathsf{prior}}(\boldsymbol{x}) \tag{1}$$

avec

$$\phi_{\mathsf{data}}(\boldsymbol{x}) = (\mathbf{H}\cdot\boldsymbol{x} - \boldsymbol{y})^{\mathrm{t}} \cdot \mathbf{W} \cdot (\mathbf{H}\cdot\boldsymbol{x} - \boldsymbol{y}) \tag{2}$$

$$= \sum_i w_i \left[(\boldsymbol{h} \star \boldsymbol{x})_i - y_i\right]^2 \tag{3}$$

où $\mathbf{H}$ est l'opérateur de convolution par $\boldsymbol{h}$ et

$$\phi_{\mathsf{prior}}(\boldsymbol{x}) = \|\mathbf{D}\cdot\boldsymbol{x}\|^2 \tag{4}$$

$$= \sum_{i,j} \left[x_{i+1,j} - x_{i,j}\right]^2 + \sum_{i,j} \left[x_{i,j+1} - x_{i,j}\right]^2 \tag{5}$$

avec $\mathbf{D}$ un opérateur linéaire (de type différences finies).

1. En utilisant les expressions algébriques, montrer que le minimum de la fonction de coût est la solution d'un système linéaire d'équations de la forme :

$$\mathbf{A} \cdot \boldsymbol{x} = \boldsymbol{b} \tag{6}$$

   Donner les expressions de $\mathbf{A}$ et de $\boldsymbol{b}$.

2. Écrire la fonction `A(x)` qui calcule $\mathbf{A} \cdot \boldsymbol{x}$ étant donné $\boldsymbol{x}$. Les opérateur $\mathbf{H}$, $\mathbf{H}^{\mathrm{t}}$ et $\mathbf{D}^{\mathrm{t}} \cdot \mathbf{D}$ sont déjà implémentés (la fonction $\boldsymbol{x} \mapsto \mathbf{D}^{\mathrm{t}} \cdot \mathbf{D} \cdot \boldsymbol{x}$ est fournie par le code `DtD(x)`).

3. Calculer $\boldsymbol{b}$.

La méthode des gradients conjugués linéaires est un algorithme simple et efficace (rapide et peu gourmand en mémoire) pour résoudre de façon itérative un système linéaire d'équations de la forme :

$$\mathbf{A} \cdot \boldsymbol{x} = \boldsymbol{b} \tag{7}$$

où le « vecteur » $\boldsymbol{x}$ contient les inconnues et la matrice $\mathbf{A}$ est symétrique définie positive. L'algorithme des gradients conjugués linéaires est donné par la Figure 1. Les ingrédients de la méthode sont :

- une fonction réalisant l'opération $\boldsymbol{x} \mapsto \mathbf{A} \cdot \boldsymbol{x}$;
- une estimation initiale de $\boldsymbol{x}$ (une image remplie de zéros ou le résultat d'une reconstruction précédente feront l'affaire) ;
- le « vecteur » $\boldsymbol{b}$;

La fonction `conjgrad` en NumPy permet d'appliquer les gradients conjugués.

4. Deconvoluer avec les gradients conjugués et déterminer la valeur de α par essais et erreurs.

Un problème a détruit 99% des pixels de la caméra. Les pixels défectueux sont indiqués par des 0 dans le tableau `pixMap`.

5. Appliquer à ces nouvelles données le filtre de Wiener avec les paramètres determinés à la section 1.

6. Qu'y a-t-il a changer dans les définitions de $\mathbf{A}$ et $\boldsymbol{b}$ pour prendre en compte ces pixels morts?

7. Deconvoluer avec les gradients conjugués ces nouvelles données.

3 Déconvolution Myope

La résolution des images prises au T80 de l'OHP est dominée par le seeing. La PSF peut être approchée par une distribution de Moffat paramètres η et ν:

$$h(x, y; \eta, \nu) = \left[1 + \left(\frac{x^2 + y^2}{\eta^2}\right)\right]^{-\nu} .$$

Si l'on utilise le filtre de Wiener défini à la section 1, la solution dépend donc de 2 couples de paramètres:

Gradients conjugués

initialisation:
  compute $\boldsymbol{r}_0 = \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{x}_0$ for some initial guess $\boldsymbol{x}_0$
  let $k = 0$
until convergence do
  $\rho_k = \boldsymbol{r}_k^{\mathrm{t}} \cdot \boldsymbol{r}_k$
  if $k = 0$, then
    $\boldsymbol{p}_k = \boldsymbol{r}_k$
  else
    $\beta_k = \rho_k / \rho_{k-1}$
    $\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_k \, \boldsymbol{p}_{k-1}$
  endif
  $\boldsymbol{q}_k = \mathbf{A} \cdot \boldsymbol{p}_k$
  $\alpha_k = \rho_k / (\boldsymbol{p}_k^{\mathrm{t}} \cdot \boldsymbol{q}_k)$ (optimal step size)
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \, \boldsymbol{p}_k$
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \, \boldsymbol{q}_k$
  $k \leftarrow k + 1$
done

Figure 1: Algorithme des gradients conjugués linéaires.

- $\theta = (\alpha, \beta)$ pour $\widehat{q}_{k,\theta} = \alpha \, |u_k|^{\beta}$,
- $\gamma = (\eta, \nu)$ décrivant la PSF.

Estimer ces 4 paramètres par essais erreur est très fastidieux voir impossible. Nous proposons donc ici un réglage automatique basé sur le maximum de vraisemblance généralisé.

Maximum de vraisemblance généralisé

On suppose

- l'objet *a priori* centré: $\bar{\boldsymbol{x}} = 0$
- l'inverse covariance *a priori* est $\mathbf{C}_{\boldsymbol{x}\,|\,\boldsymbol{\theta}}^{-1} = \mathbf{F}^{-1} \, \mathrm{diag}(\widehat{\boldsymbol{q}}_{\boldsymbol{\theta}}) \, \mathbf{F}$
- le bruit est Gaussien i.i.d. de variance 1: $\mathbf{W}_{\mathrm{e}} \propto \mathbf{I}$.

Dans ce cas le critère GML est:

$$\mathrm{GML}(\gamma, \alpha) = \frac{\boldsymbol{y}^{\mathrm{t}} \cdot \mathbf{W}_{\boldsymbol{y}\,|\,\gamma,\alpha} \cdot \boldsymbol{y}}{M_+ \, |\mathbf{W}_{\boldsymbol{y}\,|\,\gamma,\alpha}|_+^{1/M_+}}$$

$$\mathbf{W}_{\boldsymbol{y}\,|\,\gamma,\alpha} = \mathbf{I} - \mathbf{H}_\gamma \cdot \left(\mathbf{H}_\gamma^{\mathrm{t}} \cdot \mathbf{H}_\gamma + \mathbf{C}_{\boldsymbol{x}\,|\,\boldsymbol{\theta}}^{-1} \right)^{-1} \cdot \mathbf{H}_\gamma \, .$$

Le calcul du déterminant de $\mathbf{W}_{\boldsymbol{y}\,|\,\gamma,\alpha}$ semble impossible mais $\mathbf{W}_{\boldsymbol{y}\,|\,\gamma,\alpha}$ est diagonal dans le domaine

de Fourier.

$$\begin{aligned}\hat{w}_u &= 1 - \frac{|\hat{h}_u|^2}{|\hat{h}_u|^2 + \widehat{q}_u} \\ &= \frac{\widehat{q}_u}{|\hat{h}_u|^2 + \widehat{q}_u} \\ \mathrm{GML}(\gamma, \alpha) &= \frac{\sum_u \widehat{w}_u \, |\widehat{y}_u|^2}{\left[\prod_{\widehat{w}_u > 0} \widehat{w}_u\right]^{\frac{1}{M_+}}}\end{aligned}$$

avec M_+ le nombre de valeurs strictement positive de $\hat{w}$. On peut facilement calculer ce produit en utilisant un logarithme.

Traitement des données du T80

On a observé la nébuleuse de la Lyre (M57) au T80. Les données qui ont été réduites par Gilles Duvert sont présentées dans le répertoire `data`.

Pour les déconvoluer, les 4 paramètres vont être estimés avec GML sur la zone centrale.

1. Dans le fichier `GML.py`, coder la fonction `GML(dataDSP,moffat_psf, [eta,nu], powerLaw, [alpha, beta])` donnant la valeur de critère GML étant donné la DSP des données, les modèles de PSF et d'a priori et les 4 paramètres.
2. On minimise ce critère GML à l'aide de l'algorithme du Simplex (Nelder-Mead).
3. Utiliser les paramètres estimés pour déconvoluer les données sur tout le champ avec le filtre de Wiener.

4 Utilisation de NumPy

4.1 Installation de Python et de Jupyter

Pour faire le T.P. en NumPy (*Numerical Python*) il vous faut installer un interpréteur Python et un certain nombre d'extensions ou de modules ainsi que le notebook Jupyter. Deux possibilités :

1. Vous pouvez installer une distribution comme Anaconda (https://www.anaconda.com/products/individual). Inutile de télécharger Anaconda, si le répertoire `soft` contient déjà une version qui convient à votre type d'ordinateur. Sous Linux, l'installation d'Anaconda se fait par :

   ```
   bash Anaconda-2.4.0-Linux-x86_64.sh
   ```

2. Si vous utilisez une autre distribution de Python que celle établie par Anaconda, vous aurez besoin de NumPy, de Scipy, de MatplotLib et du module AstroPy (http://www.astropy.org/), par exemple sous Ubuntu :

   ```
   sudo apt-get install python-numpy python-matplotlib python-astropy python-scipy jupyt
   ```

4.2 Lire et afficher les données avec NumPy

Le court exemple ci-dessous indique comment lire les fichiers qui contiennent les images de l'objet et d'une source de référence et comment les afficher.

```
# Chargement des routines de base
from tp_utils import *

# Lecture de l'image
y = fits_read("../data/saturn.fits")
h = fits_read("../data/saturn_psf.fits")

# Affichage
figure()
imshow(y, vmin=0)     # affichage avec coupure a zero

# Affichage avec une table de couleur differente
clf()           # on efface la figure
imshow(y, vmin=0, cmap="gist_stern")

# Affichage de la PSF dans une autre fenetre
figure()            # nouvelle fenetre
imshow(h, vmin=0, cmap=...) # où ... est le nom d'une table de couleurs
```

Vous êtes prêt pour attaquer la suite... À toutes fins utiles, vous trouverez ci-après un bref récapitulatif de la syntaxe de base de Python et des fonctions les plus utiles.

5 Fonctions utiles pour le TP

Numpy et le fichier `python/tp_utils.py` définit pour vous un certain nombre de *raccourcis* pour vous faciliter la vie.

```
from tp_utils import *  # initialization
x = expr                # donner une valeur a une variable

# Pour les variables stockant des tableaux multi-dimensionnels :
x.size          # nombre d'elements dans x
x.shape         # dimensions de x
x[i]            # le i-eme element (a partir de 0) de x
x[i,j]          # un element d'un tableau 2D
x[:,j]          # la j-eme colonne
x[i1:i2,j1:j2] # une sous-image (i1≤i<i2 et j1≤j<j2)
x[test]         # le sous-tableau pour lequel `test` est vrai
x[x <= 0]       # les elements de `x` negatifs ou nuls

# Creation d'un tableau :
a = np.zeros((dim1, dim2, ...), dtype=dtype)
a = np.ones(shape, dtype=dtype)
a = np.empty(shape, dtype=dtype)

# Fonction numpy utiles pour le TP
np.sum(x)             # calcule la somme de x
np.log(x)             # calcule le log
np.exp(x)             # calcule l'exponentielle

# Fonctions utiles pour le T.P. :
zeropad(x, shape)     # remplissage de zeros a une taille donnee
z = fft(x)            # FFT appliquee a x
x = ifft(z)           # FFT inverse
fftshift(x)           # recentrage FFT
ifftshift(x)          # recentrage inverse FFT
real(z)               # partie reelle de z
imag(z)               # partie imaginaire de z
conj(z)               # complexe conjugue de z
abs2(z)               # module carre de z
normalize(x)          # normalise x tel que  ∑x = 1
conjgrad(A, b, ...)   # gradients conjugues
DtD(x)                # calcule D^t · D · x
powerLaw(shape, [alpha, beta]) # retourne la loi de puissance 10^α |u|^β
moffat_psf(shape, [eta nu]) # retourne une moffat de taille `shape`
                            # de parametres η et ν
fft_dist(shape)       # retourne le tableau de taille `shape`
                      # des distances avec le pixel (1,1)

data = fits_read("filename")    # lire une image FITS
fits_write("filename", data)    # lire une image FITS

def f(arg1, arg2, ...):   # definir une fonction f
    ...                   # le corps de la fonction est indente
    return expr           # résultat

%run file.py          # execute le contenu du fichier file.py
execfile("file.py")   # idem sans IPython
```