Inpainting et apprentissage de dictionnaire

L'*inpainting* en image s'attache à la reconstruction d'images détériorées ou au remplissage de parties manquantes (éliminer une personne ou un objet d'une image par exemple). Ce TME est consacré à l'implémentation d'une technique d'inpainting présentée dans [1] utilisant le Lasso (régression linéaire pénalisée L1) et sa capacité à trouver une solution parcimonieuse en termes de poids. Ces travaux sont inspirés des recherches en *Apprentissage de dictionnaire et représentation parcimonieuse* pour les signaux [2]. L'idée principale est de considérer qu'un signal complexe peut être décomposé comme une somme pondérée de signaux élémentaires, appelés *atomes*. L'analyse en composante principale (ACP) est un exemple de ce type de décomposition, mais du fait des contraintes d'orthogonalité, les atomes (ou la base canonique) ainsi inférés ne sont pas redondants - chacun représente le plus d'information possible et la reconstitution du signal est unique.
Dans le cas de l'inpainting, l'objectif est au contraire d'obtenir un dictionnaire fortement redondant. La difficulté est donc double : réussir à construire un dictionnaire d'atomes - les signaux élémentaires - et trouver un algorithme de décomposition/recomposition d'un signal à partir des atomes du dictionnaire, i.e. trouver des poids parcimonieux sur chaque atome tels que la combinaison linéaire des atomes permette d'approcher le signal d'origine. Dans la suite, nous étudions essentiellement une solution pour la deuxième problématique - la reconstruction - à l'aide de l'algorithme du LASSO.

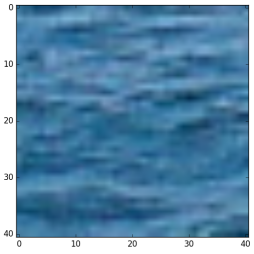
Formalisation
Pour une image en 2 dimensions, nous noterons $p_{i,j} \in [0,1]^3$ le pixel aux coordonnées (i,j) de l'image exprimée sur 3 canaux : ces 3 canaux sont usuellement le pourcentage de rouge, de vert et de bleu (rgb) de la couleur, ou sous format teinte, saturation, luminosité (hsv), plus proche de notre perception visuelle. Nous appellerons dans la suite un *patch* un petit carré de l'image de longueur de côté h (cette longueur sera une constante fixée au préalable). Nous noterons $\Psi^{p_{i,j}}$ le patch dont le centre est le pixel $p_{i,j}$. Ce patch correspond à une matrice 3d (un tenseur) de taille $h \times h \times 3$ que l'on peut voir comme un vecteur colonne de taille $3h^2$ (en applatissant le tenseur). A partir d'une image de taille $(width, height)$, il est possible de construire l'ensemble des patchs $\mathbf{\Psi}$ constituant cette image : $\mathbf{\Psi} = \{\Psi^{p_{i,j}} \in [0,1]^{3h^2} \; i \in \{\frac{h}{2} \dots width - \frac{h}{2}\}, j \in \{\frac{h}{2} \dots height - \frac{h}{2}\}\}$.
De manière générale en inpainting, une hypothèse fondamentale est qu'une image a une cohérence spatiale et de texture : un patch Ψ^p appartenant à une région cohérente en termes de texture doit pouvoir être reconstruit à partir d'une pondération des patchs environnants : $\Psi^p = \sum_{\Psi^{p_k} \in \mathbf{\Psi} \setminus \Psi^p} w_k \Psi^{p_k}$.

Dans la suite, nous allons considérer qu'une grande partie de l'image n'a pas de pixels manquants. Les patchs issus de cette partie de l'image constitueront notre dictionnaire d'atomes $\mathbf{\Psi}$ sur lesquels porteront nos combinaisons linéaires afin de reconstituer les parties manquantes.
Soit Ψ^p un patch avec des pixels manquants, soit I^p l'ensemble des pixels manquants dans le patch/vecteur Ψ^p et $\bar{I^p}$ son complémentaire (les pixels exprimés). Nous noterons ainsi $\Psi^p[I^p]$ le vecteur restreint aux pixels manquants et $\Psi^p[\bar{I^p}]$ le vecteur restreint aux pixels exprimés du patch. La reconstruction consiste à faire l'hypothèse que si une combinaison linéaire est performante pour approximer $\Psi^p[\bar{I^p}]$,

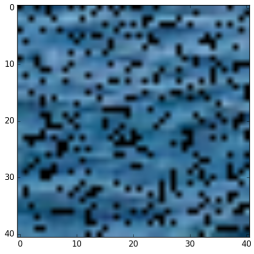
elle sera capable de généraliser aux valeurs manquantes $\Psi^p[I^p]$. Le problème d'optimisation consiste dans ce cas à trouver pour le patch Ψ^p le vecteur $\hat{\mathbf{w}}^p$ tel que :

$$\hat{\mathbf{w}}^p = \text{argmin}_{\mathbf{w}} \|\Psi^p[\bar{I}^p] - \sum_{\Psi^k \in \mathbf{\Psi}} w_k \Psi^k[\bar{I}^p]\|^2 + \lambda \|\mathbf{w}\|_1$$

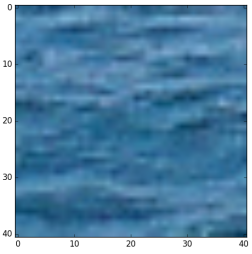
Une fois que la combinaison linéaire qui approche le mieux les pixels exprimés est trouvée, les pixels manquants peuvent être prédits en les complétant par ceux de la combinaison linéaire trouvée.



Patch origine



Patch bruité



Patch débruité

1 Expérimentation

Les fonctions suivantes vous sont données dans le squelette du code :

- `read_im(fn)` qui permet de lire une image et de la renvoyer sous forme d'un `numpy.array` en couleurs hsv, normalisé et centré en 0 .
- `show(img,fig)` qui permet d'afficher une image telle que rendue précédement sur une figure fig (ou création d'une nouvelle figure si `None`).
- une fonction `get_patch(i,j,im,h)` qui permet de retourner le patch centré en (i, j) et de longueur $2h + 1$ d'une image `im` ;
- les fonctions `patch2Vec(patch)` et `vec2patch(v)` qui convertissent (reshape) un patch $h \times h \times 3$ en vecteur et inversement ;
- `noise_patch(patch,prc)` qui permet d'ajouter aléatoirement des pixels manquants (*morts*) (avec une valeur de -100), et `remove_band(img,i,j,height,width)` qui permet de supprimer un rectangle dans une image `img`, de diagonale (i, j) à $(i + height, j + width)$;
- `build_patches(im,step)` qui permet de renvoyer la liste des patchs (sous forme de vecteurs) de l'image qui ne contiennent pas de pixels manquants (en découpant l'image tous les `step` pixels) ;

Q 1.1 Faites une fonction qui prend un patch et un dictionnaire de patchs en entrée, et qui rend le vecteur de poids sur le dictionnaire qui approxime au mieux le patch (restreint aux pixels exprimés) en utilisant l'algorithme du LASSO. Dans ce contexte, la matrice X des données est constitué des patchs aplatis en vecteur colonne (chaque colonne est un patch, représentant une dimension), chaque ligne de la matrice représente un pixel, et la cible à régresser est la valeur du pixel correspondant du patch cible. Les poids trouvés seront les coefficients de mélange des patchs du dictionnaire pou reconstruire le patch cible.
Testez en particulier sur un dictionnaire issu de l'image complète sans bruit, puis ensuite sur un dictionnaire issu d'une image dont vous avez bruité un certain nombre de pixels, sur des patchs bruités et non bruités. Expérimentez en fonction des valeurs de λ, la pénalisation du LASSO.
Vous pouvez utiliser la MSE pour évaluer la reconstruction.

Q 1.2 Supposons maintenant que c'est toute une partie de l'image qui est manquante. En considérant des patchs centrés sur les pixels manquants, on commence par remplir en partant des bords puis en remplissant au fur et à mesure vers le centre de l'image. A votre avis, l'ordre de remplissage a-t-il une

importance ? Implémentez une fonction qui permet de compléter l'image. Proposez des heuristiques pour remplir de manière intelligente (voir [3]).

Références

[1] Bin Shen and Wei Hu and Zhang, Yimin and Zhang, Yu-Jin, *Image Inpainting via Sparse Representation* Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '09)

[2] Julien Mairal *Sparse coding and Dictionnary Learning for Image Analysis* INRIA Visual Recognition and Machine Learning Summer School, 2010

[3] A. Criminisi, P. Perez, K. Toyama *Region Filling and Object Removal by Exemplar-Based Image Inpainting* IEEE Transaction on Image Processing (Vol 13-9), 2004