Séminaire de Modélisation Statistique

Restauration parcimonieuse d'un signal multidimensionnel : algorithme K-SVD

Alain Soltani, Jérémie Dona

Sous la supervision de
Joseph Salmon, Eric Gautier

16 juin 2014

ENSAE ParisTech

Résumé

Les représentations parcimonieuses connaissent une évolution croissante depuis quelques années au sein de la communauté du traitement du signal et des images. On considère un signal comme parcimonieux dans une certaine base s'il peut y être décrit par un faible nombre de coefficients non nuls ; l'objectif étant de former des approximations concises et aisément manipulables des signaux multidimensionnels étudiés, pour en résumer le contenu informatif ou en extraire une partie.
De telles représentations se prêtent particulièrement bien à la résolution de problèmes inverses : étant donné un signal bruité multidimensionnel, nous cherchons à l'approximer à l'aide d'atomes issus d'un dictionnaire spécifique, pour obtenir *in fine* une reconstruction adaptée et sans bruit.

Nous introduisons ici l'algorithme K-SVD : une nouvelle méthode permettant d'entraîner le dictionnaire employé dans la reconstruction, pour former la meilleure représentation possible. Itérativement, l'algorithme approxime le signal multidimensionnel de manière parcimonieuse, puis met à jour le dictionnaire employé. La première étape est réalisée à l'aide de l'algorithme glouton *Matching Pursuit Orthogonal*, tandis que la mise à jour emploie une *décomposition en valeurs singulières*.

Nous présenterons ici son application à la restauration d'images bruitées ; diverses améliorations, d'ordre théorique ou algorithmique, seront introduites au fil de la présentation.

Table des matières

Résumé i

Table des matières ii

1 Définition du problème 1
- 1.1 Introduction au traitement de l'image . . . 1
  - 1.1.1 Définition d'une image . . . 1
  - 1.1.2 Problème de débruitage . . . 2
- 1.2 Données employées & cadre de l'étude . . . 3
- 1.3 Problème inverse multidimensionnel . . . 3

2 Reconstruction parcimonieuse 4
- 2.1 Restauration d'un signal unidimensionnel . . . 4
  - 2.1.1 Matching Pursuit . . . 5
  - 2.1.2 Matching Pursuit Orthogonal . . . 6
  - 2.1.3 Complément : Matching Pursuit Orthogonal - Cholesky . . . 8
- 2.2 Restauration d'un signal multidimensionnel . . . 8

3 Mise à jour du dictionnaire explicatif 9
- 3.1 Problème général . . . 9
- 3.2 Décomposition en valeurs singulières . . . 9
- 3.3 Dictionnaire initial . . . 10

4 Algorithme K-SVD 11
- 4.1 Récapitulatif . . . 11
- 4.2 Application & recherche des paramètres optimaux . . . 12
  - 4.2.1 Influence du pas p . . . 12
  - 4.2.2 Influence du nombre d'itérations N_{iter} . . . 14
- 4.3 Influence du ratio d'apprentissage R . . . 15
- 4.4 Conclusion . . . 15

Bibliographie 16

Chapitre 1

Définition du problème

1.1 Introduction au traitement de l'image

Dans de nombreux domaines d'application (imagerie médicale, imagerie astrophysique, contrôle industriel, etc.), les images sont omniprésentes et leur traitement est essentiel pour une exploitation judicieuse. En l'occurrence, ces images peuvent être sujettes à des dégradations comme des perturbations aléatoires communément qualifiées de *bruit*.
Le débruitage d'images vise à supprimer ce bruit en vue d'obtenir des images aussi propres que possible. Ce challenge, à la frontière des mathématiques et de l'informatique, a fait l'objet de nombreuses études. Des algorithmes efficaces ont été élaborés, parmi lesquels ceux basés sur l'emploi de sous-images (*patches*), qui font l'objet de cette étude. Il s'agira de former, à l'aide d'un catalogue de *patches* (appelé *dictionnaire*), la meilleure approximation possible, sans bruit, d'une image donnée.

1.1.1 Définition d'une image

Une image I est une fonction de $\mathbb{R}^2$ dans $\mathbb{R}^d$ à support compact. Il n'est en général pas possible de travailler sur des objets continus ; aussi une image sera-t-elle vue comme une fonction de $\mathbb{Z}^2$ dans $\mathbb{R}^d$ à support compact.

Le paramètre d dépend du type d'encodage de l'image : pour des images couleur codées en format *RVB (Rouge Vert Bleu)*, $d = 3$; pour une image en niveaux de gris - auxquelles nous nous restreindrons dans cette étude - $d = 1$.
En outre, nous travaillons avec des images rectangulaires, ce qui permet de représenter I comme une matrice, dont chaque coefficient i, j correspond à la valeur du niveau de gris du pixel issu de I, au point (i, j). Ces valeurs de niveaux de gris sont entières, comprises entre 0 et 255, par souci de simplification ; la valeur 0 correspondant au noir, 255 au blanc.

1.1.2 Problème de débruitage

En pratique, nous formerons nous-même l'image bruitée à partir d'une image initiale :

$$I = I^* + \epsilon \tag{1.1}$$

où la matrice ϵ est formée à partir d'un vecteur gaussien de taille adéquate :

$$\epsilon \sim \mathcal{N}(0, \sigma^2 I_{MN}), \tag{1.2}$$

que nous pouvons ré-écrire en une matrice de taille $M \times N$; le terme d'erreur peut alors être compris comme les réalisations de variables aléatoires gaussiennes *i.i.d.*

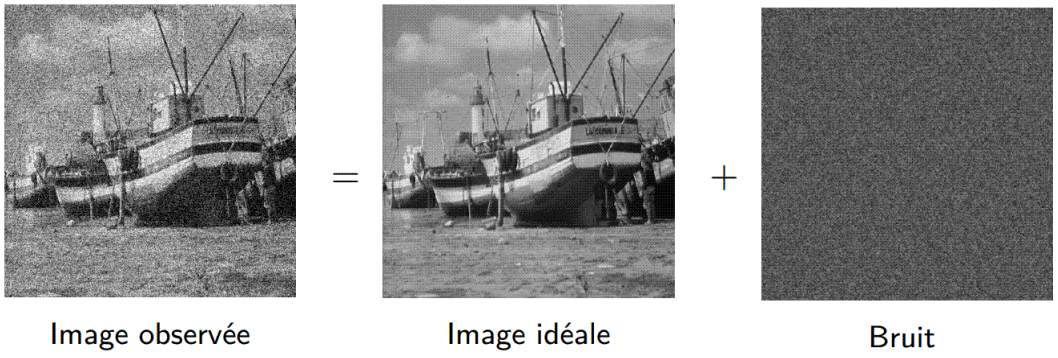

FIGURE 1.1 – *Simulation d'une image bruitée.*

Le problème de débruitage peut alors être formulé comme suit :

$$Y = Y^* + E, \tag{1.3}$$

où Y, formée à partir d'une image initiale bruitée I, contient l'ensemble des *patches* de I à débruiter : pour une image en entrée $I \in \mathcal{M}_{M,N}(\mathbb{R})$, on forme des sous-images de I de taille $\sqrt{n} \times \sqrt{n}$, avec $\sqrt{n} < \min(M, N)$.
Chacune de ces sous-images peut être vue comme un vecteur de $\mathcal{M}_{n,1}(\mathbb{R})$; Y contient ainsi, en colonne, tous les *patches* formés à partir de I - de taille $n \times r$, avec r le nombre de *patches* employés pour recouvrir I.
En pratique, ce recouvrement est fait de manière discrète, selon un certain pas p.

Précisons enfin que chaque *patch* carré i, j est obtenu en prenant la sous-image carrée dont le coin haut, gauche se trouve à la position (i, j) dans l'image de départ. Ainsi, on rencontre certaines difficultés à former les *patches* aux bords bas, droit de l'image bruitée.
Pour remédier à ce problème, on considèrera dans ce qui suit, sauf mention explicite, que $I \in \mathcal{M}_{M,N}(\mathbb{R})$ représente l'image bruitée *agrandie*, dont les bords ont été symétrisés.

La matrice idéale, sans bruit additif, est notée Y^*, à partir de laquelle on peut théoriquement reconstruire l'image idéale I^*. E constitue le terme d'erreur dont on veut s'affranchir.

Une fois l'approximation finale des *patches* $\hat{Y}$ obtenue, l'image débruitée est reformée en $\hat{I}$. Nous mesurons alors la qualité de notre débruitage à l'aide de la mesure de distorsion *Peak Signal to Noise Ratio* (PSNR), définie comme suit :

$$PSNR(I, \hat{I}) = 10 \log_{10} \frac{MAX_I^2}{MSE(I, \hat{I})}, \tag{1.4}$$

où $I \in \mathcal{M}_{M,N}(\mathbb{R})$ est l'image bruitée, $\hat{I}$ son approximation débruitée, et MSE l'erreur quadratique moyenne, définie par :

$$MSE(A, B) = \frac{1}{MN} \sum_{i=0}^{M} \sum_{j=0}^{N} [A(i,j) - B(i,j)]^2, \tag{1.5}$$

pour $A, B \in \mathcal{M}_{M,N}(\mathbb{R})$.

1.2 Données employées & cadre de l'étude

Ce projet a entièrement été programmé sous Python ; il est disponible à l'adresse suivante : https://github.com/Parveez/Seminaire_Modelisation_Statistique.
Les images employées furent traitées à l'aide des libraries PIL et SCIKIT IMAGE ; les décompositions matricielles et les distributions avec SCIPY et NUMPY.

1.3 Problème inverse multidimensionnel

Notre problème de débruitage s'apparente à une problème inverse multidimensionnel :

$$Y = D\Lambda + E, \tag{1.6}$$

$Y = [y_1 ... y_r] \in \mathcal{M}_{n,r}(\mathbb{R})$ étant la concaténation des r *patches* formés, $D \in \mathcal{M}_{n,q}(\mathbb{R})$ un *dictionnaire* de variables explicatives - rangées en colonne - appelées *atomes*,
$\Lambda \in \mathcal{M}_{q,r}(\mathbb{R})$ une matrice dite *parcimonieuse* contenant les poids accordés aux différents atomes pour chaque *patch*.

Nous chercherons au cours de cette étude, à former à la fois une matrice parcimonieuse Λ et un dictionnaire explicatif D permettant la meilleure reconstruction possible ; i.e. fournissant des valeurs de PSNR les plus hautes possibles (une gamme acceptable de valeurs allant de 30 dB à 50 dB).

Si le PSNR est utile pour mesurer la proximité de l'image compressée par rapport à l'originale au niveau du signal, il ne prend pas en compte la qualité visuelle de reconstruction et ne peut être considéré comme une mesure objective de la qualité visuelle d'une image.
Toutes nos analyses quantitatives seront donc suivies d'une analyse visuelle pour attester de la qualité de nos méthodes.

Chapitre 2

Reconstruction parcimonieuse

L'algorithme central de cette étude, dit algorithme K-SVD, est basé sur l'implémentation de deux étapes successives : la formation d'une approximation parcimonieuse des *patches*, et la mise à jour du dictionnaire. Dans les chapitres 2 et 3, nous allons détailler ces deux procédures, pour ensuite étudier l'algorithme final dans son ensemble.

Commençons par la première étape de l'algorithme K-SVD : la reconstruction parcimonieuse, qui consiste à approximer un signal initial comme combinaison linéaire d'atomes d'un dictionnaire fixé.
Introduisons tout d'abord le problème d'optimisation relatif au débruitage d'un signal unidimensionnel - dans notre cas, *un seul patch* - que nous étendrons ensuite au cas multidimensionnel.

2.1 Restauration d'un signal unidimensionnel

Dans ce qui suit, nous appelerons pseudo-norme l^0 :

$$\|\lambda\|_0 = \text{Card}\{i \in \{1; q\} \,|\lambda_i \neq 0\}.$$

Lors de cette première étape à dictionnaire fixé, notre but est de minimiser la distance entre notre observation $y \in \mathcal{M}_{n,1}(\mathbb{R})$ et son approximation $D\lambda$, sous une certaine contrainte de parcimonie - dans la mesure où nous cherchons à décrire efficacement nos données de manière concise.

Ceci revient à résoudre le problème d'optimisation sous contrainte suivant :

$$\min_{\|\lambda\|_0 \leqslant s} \|y - D\lambda\| \tag{2.1}$$

ou son équivalent :

$$\min_{\|y - D\lambda\| \leqslant \eta} \|\lambda\|_0 \,. \tag{2.2}$$

Les paramètres de parcimonie s ou d'erreur η sont à spécifier au vu du niveau de bruit inhérent au signal ; dans le cas particulier où il n'y a aucun bruit additif - i.e. $\epsilon = 0$ - on

fixera $\eta = 0$, et résoudra le problème simplifié :

$$\min_{y=D\lambda} \|\lambda\|_0 \,. \tag{2.3}$$

Ces problèmes d'optimisation sont en général très difficiles à résoudre, car non-convexes ; l'obtention et l'unicité d'un minimum global ne sont en outre pas assurées.

Nous proposons dans ce qui suit une méthode de résolution s'appuyant sur des algorithmes heuristiques dits "gloutons", comme le *Matching Pursuit* (MP).

2.1.1 Matching Pursuit

Tout d'abord introduit par S. Mallat et S. Zhang [1], l'algorithme du Matching Pursuit est une procédure gloutonne qui construit itérativement un vecteur parcimonieux en recherchant les atomes du dictionnaire les plus corrélés au résidu courant.

Etant donné notre signal initial y et un dictionnaire fixé D, l'objectif est d'approcher le premier en formant une combinaison linéaire des atomes du second.
L'algorithme MP offre une approche simple :

· Nous fixons le résidu initial à la valeur $R^{(0)}y = y$;
· A l'étape l de l'algorithme, nous recherchons l'atome $D_{i_0} \in \mathcal{M}_{n,1}(\mathbb{R})$ le plus corrélé au résidu courant $R^{(l)}y$:

$$D_{i_0} = \underset{D_i \in D}{\operatorname{argmax}} |\langle R^{(l)}y, D_i \rangle|. \tag{2.4}$$

· Nous soustrayons ensuite à $R^{(l)}y$ sa projection sur D_{i_0} pour former le nouveau résidu $R^{(l+1)}y$:

$$R^{(l+1)}y = R^{(l)}y - \langle R^{(l)}y, D_{i_0} \rangle D_{i_0} \tag{2.5}$$

jusqu'à ce que notre combinaison linéaire approche suffisamment bien notre signal initial.

Data: Signal unidimensionnel bruité y.
Result: Reconstruction débruitée $D\lambda$.
Initialization : $\lambda^{(0)} = 0$;
while *l'approximation formée n'est pas suffisamment bonne* **do**
  $\lambda^{(l+1)} = \lambda^{(l)} + \mu$
  avec μ vecteur à une composante non nulle, minimisant le terme d'erreur
  $\min_{\|\mu\|_0=1} \left\|y - D(\lambda^{(l)} + \mu)\right\|$.
end

Algorithm 1: Matching Pursuit classique.

Cet algorithme se ré-écrit de la manière suivante : à chaque étape, nous formons toutes les corrélations entre les atomes de D et le résidu courant. L'indice de corrélation maximale est alors déterminé comme suit :

Data: Signal unidimensionnel bruité y.
Result: Reconstruction débruitée $D\lambda$.
Initialization : $\lambda^{(0)} = 0$;
while *l'approximation formée n'est pas suffisamment bonne* **do**
  $[\lambda^{(l+1)}]_{i_0} = [\lambda^{(l)}]_{i_0} + [c]_{i_0}$
  avec
  $c = D^T(y - D\lambda^{(l)})$ vecteur de corrélation, $i_0 = \underset{i}{\text{argmax}}|c_i|$.
end

Algorithm 2: Matching Pursuit, sous forme matricielle.

Il est alors possible de montrer que l'erreur courante $\left\|y - D\lambda^{(l)}\right\|$ converge vers zéro quand le nombre d'atomes inclus dans le modèle l augmente.
Cependant, inclure trop d'atomes peut signifier approximer plus que le signal recherché, i.e. du bruit. De ce fait, la question du critère d'arrêt de cet algorithme est essentielle.

Ici, nous avons considéré une critère simple, permettant un court temps de calcul tout en gardant sa légitimité statistique.

Etant donné le n-échantillon $y = (y_1 \dots y_n)^T \sim \mathcal{N}(D\lambda^{(l)}, \sigma^2 I_n)$,
nous pouvons introduire l'estimateur non-biaisé de la variance suivant :

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \mathbb{E}[y_i])^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - [D\lambda^{(l)}]_i)^2. \tag{2.6}$$

Nous cherchons à tester l'hypothèse $\mathcal{H}_0 : \sigma = \sigma_0$ contre l'alternative $\mathcal{H}_1 : \sigma > \sigma_0$.
Nous pouvons alors former la statistique de test suivante :

$$T_n = (n-1)\frac{s^2}{\sigma^2} = \frac{1}{\sigma^2}\|y - D\lambda^{(l)}\|^2, \tag{2.7}$$

qui suit, sous $\mathcal{H}_0$, une loi du χ^2 à $n-1$ degrés de liberté.
Ainsi, tant que nous nous trouvons dans la zone de rejet $\{T_n > q_{0.995}^{\chi^2(n)}\}$,
où $q_{0.995}^{\chi^2(n)}$ est le fractile d'ordre 0.05 d'une loi χ^2 à n degrés de liberté, nous continuons d'inclure des variables dans notre modèle, afin d'atteindre la forme de l'erreur voulue.
En outre, dans la mesure où son coût calculatoire est très faible - il nous suffit de considérer la norme des résidus courants à chaque étape - nous avons privilégié ce critère pour la suite de notre étude.

2.1.2 Matching Pursuit Orthogonal

De nombreuses améliorations peuvent être apportées à l'algorithme originel du Matching Pursuit ; nous nous focaliserons ici sur une variante appelée *Matching Pursuit Orthogonal* (OMP) [2] ; c'est cet algorithme que nous emploierons en outre dans la méthode K-SVD finale, présentée au chapitre suivant.

L'amélioration apportée par l'OMP consiste en une mise à jour des coefficients non-nuls dans le vecteur parcimonieux λ ; de la même manière que précédemment, nous

déterminons itérativement quels atomes inclure dans notre modèle, et fixons la valeur leurs poids dans λ à la projection du signal sur les atomes préalablement sélectionnés.

A l'étape l de l'algorithme OMP, nous commençons par implémenter l'étape classique du Matching Pursuit - la recherche de l'atome de plus forte corrélation : nous formons $\tilde{\lambda}$ tel que

$$[\tilde{\lambda}]_{i_0} = [\lambda^{(l)}]_{i_0} + [c]_{i_0}.$$

Nous poursuivons en recherchant la projection orthogonale $D\lambda$ sur le support $I(\tilde{\lambda})$:

$$\lambda^{(l+1)} = \underset{I(\lambda)=I(\tilde{\lambda})}{\operatorname{argmin}} \|y - D\lambda\|, \tag{2.8}$$

où $I(\lambda) = \{i \in \langle 1; q\rangle \mid \lambda_i \neq 0\}$ est le support de la solution.
Matriciellement, ceci s'écrit de la manière suivante [3] :

$$\lambda_I^{(l+1)} = D_I^+ y, \tag{2.9}$$

où D_I est la sous-matrice des colonnes de D indexées par I, $\lambda_I^{(l+1)}$ le sous-vecteur indexé par I et $D_I^+ = (D_I^T D_I)^{-1} D_I^T$ la pseudo-inverse de Moore-Penrose de D.

Ceci revient bien à projeter l'approximation $D\lambda = D_I \lambda_I$ sur le support formé par les atomes sélectionnés à l'étape l. De cette façon, les résidus et l'approximation du signal sont orthogonaux à chaque étape de l'algorithme.

L'algorithme OMP est implémenté à l'aide du critère statistique présenté précédemment.

Data: Signal unidimensionnel bruité y.
Result: Reconstruction débruitée $D\lambda$.
Initialization : $\lambda^{(0)} = 0$;
while $\|y - D\lambda^{(l)}\|^2 > q_{0.995}^{\chi^2(n)} \sigma^2$ **do**

$[\lambda^{(l+1)}]_{i_0} = [\lambda^{(l)}]_{i_0} + [c]_{i_0}$
with
$c = D^T(y - D\lambda^{(l)})$, $i_0 = \underset{i}{\operatorname{argmax}} |c_i|$;
$\lambda_I^{(l+1)} = D_I^+ y$
où I support de $\lambda^{(l+1)}$.

end

Algorithm 3: Matching Pursuit Orthogonal.

Cette projection additionnelle justifie bien souvent l'emploi répandu de l'OMP : la projection orthogonale assure une converge de l'erreur plus rapide que le Matching Pursuit. En pratique, l'OMP peut fournir de meilleurs résultats qu'un MP standard, avec la contrepartie de susciter des étapes de calcul supplémentaires lors de la projection [1] ; c'est pourquoi nous présentons brièvement un second algorithme OMP, basé sur la décomposition de Cholesky.

1. Pour plus de détails concernant le coût calculatoire, se reporter à [4].

2.1.3 Complément : Matching Pursuit Orthogonal - Cholesky

En pratique, l'étape d'inversion de la matrice $D_I^T D_I$ (symétrique, définie positive) est celle qui cause la majeure partie du long temps de calcul observé à l'emploi de l'OMP. En effet, il s'agit à chaque étape de l'algorithme OMP, de recalculer et inverser cette matrice pour former l'inverse de Moore-Penrose D_I^+.

Une solution est alors de ré-écrire $D_I^T D_I$ à l'aide de la décomposition de Cholesky :

$$D_I^T D_I = LL^T. \tag{2.10}$$

La mise à jour de la matrice L est bien plus aisée : si $D_I^{(l+1)} = \begin{pmatrix} D_I^{(l)} & v \\ v^T & c \end{pmatrix}$, alors

$$L^{(l+1)} = \begin{pmatrix} L^{(l)} & 0 \\ w^T & \sqrt{c - \|w\|^2} \end{pmatrix}, \tag{2.11}$$

avec $w = (L^{(l)})^{-1} v$.
Cette méthode a l'avantage de produire un temps de calcul bien meilleur que l'algorithme OMP standard, mais se trouve sujette aux éventuelles erreurs d'inversion de $L^{(l)}$, ce qui demande de distinguer des cas (comme $l > 1$) lors de l'implémentation de l'algorithme.

2.2 Restauration d'un signal multidimensionnel

Une fois la procédure d'approximation parcimonieuse détaillée pour un signal unidimensionnel $y \in \mathcal{M}_{n,1}(\mathbb{R})$ - qui correspond, dans l'application qui suit, à un unique *patch* pris comme vecteur colonne - détaillons la reconstruction d'un signal multidimensionnel $Y \in \mathcal{M}_{n,r}(\mathbb{R})$ - une concaténation de r *patches*.

Nous considérons ici le problème inverse associé au cas multidimensionnel, détaillé au chapitre 1. Immédiatement, on peut penser à une généralisation du problème d'optimisation précédent, avc tous nos sous-signaux de Y (également appelés *canaux*) soumis à la même contrainte de parcimonie :

$$\min_{\substack{\Lambda \\ \forall i,j \in \langle 1;n \rangle, \|\Lambda^i\|_0 = \|\Lambda^j\|_0 \leq s}} \|Y - D\Lambda\| \tag{2.12}$$

où Λ^j représente la $j^{\text{ème}}$ colonne de Λ.
On détermine alors, dans ce cas, un unique ensemble d'atomes de D expliquant l'ensemble des *patches*. Cependant, cette approche est trop réductrice : il se peut que certains patches contiennent très peu d'information (ex. portion de ciel bleu, dans un paysage), quand d'autres en possèderont beaucoup plus (immeubles, personnes, etc.).

C'est pourquoi, dans ce qui suit, nous considèrerons le problème de restauration beaucoup plus libre où tous les sous-signaux ne vérifient pas une même contrainte de parcimonie, mais un même *critère statistique* : après avoir réalisé un *Matching Pursuit* sur chaque canal y_i, selon la procédure de test présentée en 2.1.1., nous obtenons [2] :

$$\forall i \in \langle 1; n \rangle, \mathbb{P}[||y_i - D\Lambda^i||^2 \leq q^{\chi^2(n)} \sigma^2] = 0.995. \tag{2.13}$$

2. Notons enfin que les vecteurs Λ^i sont construits indépendamment les uns des autres. Il est alors avantageux de réaliser cette étape à l'aide d'outils de calcul parallèle, pour gagner en temps de calcul.

Chapitre 3

Mise à jour du dictionnaire explicatif

Comme mentionné précédemment, chaque itération de notre algorithme total se décompose en deux étapes ; la première consistait à former une approximation parcimonieuse du signal multidimensionnel, à dictionnaire fixé. Présentons maintenant l'étape suivante.

3.1 Problème général

Une fois l'approximation parcimonieuse Λ formée, nous optimisons le dictionnaire en mettant à jour chacun de ses atomes ; il s'agit ici d'obtenir itérativement, à partir d'un dictionnaire initial, une nouvelle version permettant d'approcher au mieux les données non-bruitées. La question du choix du dictionnaire initial est essentielle, et sera détaillée ultérieurement.

Le problème d'optimisation concerne donc cette fois uniquement le dictionnaire D [1] :

$$\min_{D} \|Y - D\Lambda\| \tag{3.1}$$

sous la contrainte : « D est une famille orthonormale de $\mathcal{M}_{n,1}(\mathbb{R})$ ».

L'algorithme employé est une méthode d'apprentissage basée sur la décomposition en valeurs singulières, appelée K-SVD [4]. C'est en réalité une généralisation de la méthode de partitionnement des « k-moyennes ».

3.2 Décomposition en valeurs singulières

La mise à jour du dictionnaire se déroule donc comme suit : nous mettons à jour tous les atomes du dictionnaire D les uns après les autres. Soit k l'indice de la colonne à modifier.

1. Il est possible de minimiser, de manière équivalente, le terme $\|Y - D\Lambda\|^2$.

Ré-écrivons le terme d'erreur à minimiser, en séparant les termes liés à la colonne k :

$$\begin{aligned} \|Y - D\Lambda\|^2 = \|Y - \sum_{j=1}^{q} D_j\Lambda^j\|^2 = \|Y - \sum_{j=1, j\neq k}^{q} D_j\Lambda^j - D_k\Lambda^k\|^2 \\ = \|Err - D_k\Lambda^k\|^2, \end{aligned} \tag{3.2}$$

avec D_k la $k^{\text{ème}}$ colonne, Λ^k la $k^{\text{ème}}$ ligne et Err le terme d'erreur ôté de l'influence de D_k.
Nous pouvons maintenant résoudre notre problème de minimisation en approximation Err avec une matrice de rang 1, en utilisant une décomposition en valeurs singulières (SVD), et ainsi mettre à jour D_k.

En pratique, pour assurer que les contraintes imposées à Λ soient bien respectées après cette décomposition, nous considérons seulement des vecteurs restreints : seuls les indices correspondants à des atomes inclus dans le modèle sont conservés.

Par exemple, le vecteur ligne Λ^k devient Λ^k_R, ôté de ses coefficients nuls ; la matrice Err devient Err_R, où l'on a retiré les colonnes correspondants aux atomes n'intervenant pas dans le modèle.

Le problème de minimisation devient alors $\|Err_R - D_k\Lambda^k_R\|^2$ que nous pouvons cette fois bien résoudre direcement, à l'aide d'une décomposition SVD. Err_R s'écrit alors $U\Delta V^T$, où U, V sont unitaires et Δ une matrice diagonale de coefficients positifs.

Nous mettons à jour D_k en la remplaçant par la première colonne de U, puis Λ^k_R en remplaçant par la première colonne de V, multipliée par $[\Delta]_{1,1}$.

Une fois tous les atomes mis à jour, l'étape se termine et nous commençons une nouvelle itération de l'algorithme K-SVD total.

3.3 Dictionnaire initial

Une fois spécifié un dictionnaire de départ $D^{(0)}$, nous sommes maintenant en mesure de construire un dictionnaire optimisé, étape par étape. Mais reste à choisir correctement le dictionnaire initial.
Plusieurs choix sont possibles : on peut s'intéresser à des matrices de décomposition dans certaines bases de fonctions adaptées (transformée de Fourier, ondelettes, etc.).

Dans cette étude, nous avons privilégié des matrices de *patches*, formées à l'aide d'une image extérieure au problème de débruitage ; il est également possible d'employer l'image sans bruit associée, ou encore l'image bruitée elle-même.
Cependant, d'un point de vue calculatoire, il n'est pas intéressant de conserver l'ensemble des *patches* - en particulier si le pas entre chaque patch est relativement petit.

Nous avons alors introduit un ratio R, qui correspond au pourcentage d'atomes sélectionnés [2] parmi tous les *patches* disponibles : si $R = 0.05$, 5% d'atomes sont retenus aléatoirement pour former le dictionnaire ; si $R = 1$, tous les atomes sont employés.
Le choix des valeurs de R (ainsi que des valeurs de pas p, de taille de *patch* $\sqrt{n}$, et de nombre d'itérations de l'algorithme total N_{iter}) sera détaillé ultérieurement.

2. Le tirage des atomes se fait alatoirement, selon une loi uniforme discrète.

Chapitre 4

Algorithme K-SVD

4.1 Récapitulatif

Prenons du recul sur les méthodes détaillées aux deux chapitres précédents, pour revenir au problème de débruitage général.

Le problème d'optimisation associé est plus large qu'auparavant : il porte maintenant sur le dictionnaire D et le vecteur parcimonieux Λ :

$$\min_{D,\Lambda} \|Y - D\Lambda\| \tag{4.1}$$

sous les contraintes :

1. « D est une famille orthonormale de $\mathcal{M}_{n,1}(\mathbb{R})$ »,
2. (2.13) : « $\forall i \in \langle 1; n \rangle, \mathbb{P}[||E^i||^2 \leq q^{\chi^2(n)}\sigma^2] = 0.995$ ».

On atteint l'optimum de manière heuristique, en décomposant le problème en deux sous-problèmes successifs. Chaque itération de l'algorithme K-SVD total se décompose donc en deux étapes :

1. A D fixé, nous formons une approximation parcimonieuse Λ de Y, à l'aide de l'*Orthogonal Matching Pursuit* sur l'ensemble des canaux (Chapitre 2).
2. Après le codage parcimonieux, à Λ fixé, nous optimisons le dictionnaire D à l'aide de la décomposition en valeurs singulières (Chapitre 3).

Le tout étant répété un nombre N_{iter} de fois. Il n'existe pas de valeur théorique idéale pour N_{iter}, mais on considère en général que 10 à 15 itérations suffisent pour obtenir une image débruitée acceptable.

Notons également que le nombre d'atomes inclus à chaque itération de l'algorithme total à tendance à diminuer ; ceci est logique, car chaque dictionnaire optimisé D décrit plus succintement les données qu'auparavant.
Une manière de contourner le problème est alors de fixer le nombre d'atomes à inclure lors du codage parcimonieux ; cependant, on perd la légitimité statistique que fournissait la procédure de test des atomes.

4.2 Application & recherche des paramètres optimaux

Nous pouvons désormais passer à l'application pratique de notre algorithme ; ici, nous débruiterons l'image *Lenna* (Image A), bien connue en traitement de l'image, à laquelle nous superposerons un bruit d'écart-type $\sigma = 10$ (Image B). Le dictionnaire initial sera formé à partir de l'image *Barbara* (Image C).
Toutes les images seront réduites à la taille 256×256, dans un souci de vitesse d'implémentation des algorithmes.

(A) Image originale. (B) Image bruitée. (C) Image d'apprentissage.

FIGURE 4.1 – Images employées dans le problème de débruitage.

Différents paramètres sont accessibles à l'utilisateur pour l'emploi de la méthode K-SVD :

1. la taille des *patches* $\sqrt{n}$,
2. le pas entre chaque *patch* p,
3. le ratio d'apprentissage R,
4. le nombre total d'itérations de l'algorithme K-SVD N_{iter}.

Nous nous intéresserons dans les sections suivantes à l'influence de ces paramètres sur le résultat débruité final.

4.2.1 Influence du pas p

Nous fixons dans cette section les autres paramètres aux valeurs suivantes : $N_{iter} = 20$ (qui suffit généralement pour observer une influence du paramètre sur nos résultats), $R = 0.5$ (ceci permet de conserver un temps de calcul acceptable), $\sqrt{n} = 16$ (nous formons des *patches* recouvrant $\frac{1}{16}^{\text{ème}}$ de l'image).

La Figure 4.2 présente les images débruitées pour $p \in \{3, 6, 12\}$, ainsi que les différences entre l'image bruitée et son approximation ; ici, la remarque faite sur l'inaptitude du PSNR à traduire parfaitement la qualité visuelle d'un débruitage prend tout son sens. Bien que les valeurs de PSNR semblent décroître en diminuant le pas, on observe - notamment sur le visage - des irrégularités accentuées avec un fort pas.
Un faible pas offre un meilleur niveau de détail ; en contrepartie, le temps de calcul et les ressources nécessaires en mémoire augmentent drastiquement. L'implémentation des algorithmes avec un pas de 1, 2 requiert des moyens et une infrastructure informatique n'étant pas à notre disposition ; pour autant, nous sommes convaincus qu'une implémentation en parallèle sur différents processeurs, avec des ressources suffisantes en mémoire

fournissent des résultats idéaux. Pour préserver un temps de calcul acceptable, nous emploierons dans ce qui suit un pas de 6.

(A) Image bruitée. PSNR : 30.00 dB.

(B) Image débruitée : $p = 12$. PSNR : 29.93 dB.

(C) Image débruitée : $p = 6$. PSNR : 29.00 dB.

(D) Image débruitée : $p = 3$. PSNR : 28.39 dB.

(E) Différence : $p = 6$.

(F) Différence : $p = 3$.

FIGURE 4.2 – Influence du pas sur le débruitage K-SVD.

4.2.2 Influence du nombre d'itérations N_{iter}

Dans cette section $p = 6$, $R = 0.5$, $\sqrt{n} = 16$, et nous faisons varier le nombre d'itérations de l'algorithme total. Les résultats sont présentés en Figure 4.3.

(A) Image bruitée.

(B) Image débruitée : $N_{iter} = 2$. PSNR : 29.11 dB.

(C) Image débruitée : $N_{iter} = 5$. PSNR : 28.91 dB.

(D) Image débruitée : $N_{iter} = 20$. PSNR : 29.00 dB.

(E) Différence : $N_{iter} = 5$.

(F) Différence : $N_{iter} = 20$.

FIGURE 4.3 – Influence du nombre d'itérations sur le débruitage K-SVD.

On peut clairement voir qu'augmenter le nombre d'itérations améliore drastiquement la qualité du débruitage : l'apprentissage du dictionnaire est amélioré à chaque itération.

En revanche, si le pas choisi est trop grand, comme l'algorithme est susceptible de choisir un plus faible nombre d'atomes à chaque itération, on peut observer des irrégularités et la présence de « blocs » dans l'image débruitée. Ce cas de figure advient principalement pour $p = \sqrt{n}$, longueur d'un *patch* carré.
Comme décrit précédemment, un pas $p \leq 2$ et un nombre d'itérations $N_{iter} \geq 20$ sont susceptibles de fournir d'excellents résultats.

4.3 Influence du ratio d'apprentissage R

Dans cette section $p = 6$, $N_{iter} = 20$, $\sqrt{n} = 16$, et nous faisons varier le ratio d'apprentissage R. Rappelons qu'il correspond au pourcentage d'atomes sélectionnés aléatoirement parmi les *patches* de l'image d'apprentissage. Les résultats sont présentés en Figure 4.4.

Il n'existe pas de *ratio* idéal ; pour cause, suivant la diversité en niveaux de gris de l'image employée, on peut diminuer le ratio sans perdre en qualité visuelle, jusqu'à un certain seuil - tant que les atomes sélectionnés contiennent suffisamment d'information, vis-à-vis de l'image entière. Cependant, un ratio trop faible donne directement de très mauvais résultats, à la fois en terme de PSNR et d'analyse visuelle, comme en attestent les résultats suivants.

(A) Image débruitée : $R = 0.01$. PSNR : 25.43 dB.

(B) Image débruitée : $R = 0.5$. PSNR : 29.00 dB.

FIGURE 4.4 – Influence du ratio d'apprentissage sur le débruitage K-SVD.

4.4 Conclusion

L'algorithme K-SVD offre une méthode de débruitage efficace et concise ; conjointement basée sur l'approximation parcimonieuse de l'image bruitée, ainsi que l'optimisation du dictionnaire employé, elle permet d'obtenir une reproduction fidèle de l'image d'origine à l'aide d'un nombre réduit de variables explicatives.
Ce type de représentation, implémentée à l'aide d'une infrastructure adaptée, constitue une nouvelle alternative aux méthodes existantes en traitement du signal, débruitage et compression des images.

Bibliographie

[1] S. Mallat and S. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 1993.

[2] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit : Recursive function approximation with applications to wavelet decomposition. *Proceedings of the* 27^{th} *Annual Asilomar Conference on Signals, Systems, and Computers*, 1993.

[3] G. Peyre. Sparse regularization with matching pursuits. *A Numerical Tour of Signal Processing, Ceremade.*

[4] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *Technion - Computer Science Department - Technical Report*, 2008.