Matching Pursuit et Orthogonal Matching Pursuit

Ch. Dossal

Mars 2012

1 Introduction

Le Matching Pursuit (MP) et sa variante orthogonale, le Matching Pursuit Orthogonal (OMP) sont deux méthodes classiques largement utilisées dans différents domaines du traitement du signal : approximation, débruitage, problèmes inverses, détection, séparation de sources etc ...
Le problème de base où apparaît le MP est le suivant : On dispose d'un signal X et d'un dictionnaire de signaux $A = (a_i)_{i \leqslant p}$ de la taille de X et on souhaite écrire X comme une combinaison linéaire d'éléments a_i de A ou au moins approcher X par une telle combinaison linéaire.
Le MP propose une approche simple : On estime l'atome a_{i_0} le plus corrélé à X, on soustrait à X sa projection sur a_{i_0} et on recommence l'opération sur le résidu $X - \lambda a_{i_0}$ jusqu'à un critère d'arrêt fixé par l'utilisateur. Dans l'OMP on sélectionne également les atomes a_i un à un mais à chaque nouvelle sélection d'atome, on recalcule la projection de X sur l'espace vectoriel engendré par les vecteurs sélectionnés. Quand X s'écrit vraiment comme une combinaison linéaire des a_i, l'OMP peut converger rapidement vers une erreur nulle, l'erreur commise par le MP sera elle rarement nulle.
Nous proposons ici deux applications du MP. Dans un premier temps nous verrons comment cette tehnique permet d'effectuer de la séparation sur des petits signaux 1D et nous comparerons le MP et l'OMP. Dans un second temps nous utiliserons le MP pour de la compression et du débruitage de signaux musicaux.

2 MP et OMP pour la séparation

Le but de cette section est de séparer un signal S somme de Diracs et de sinusoides en une somme de Diracs d'un part et et une somme de sinusoides d'autre part. Nous évaluerons également la sensibilité au bruit.

1. Ecrire un programme GenereMatSinussoide

   ```
   A=GenereMatSinusoide(N)
   ```

   qui génère une matrice de taille $N \times N$ contenant une base de cosinus discrets **On pourra utiliser la dct.**

2. A l'aide de la fonction précédente et par concaténation avec la matrice identité I_N, créer une matrice **avec des colonnes normées**, de taille $N \times 2N$ une matrice dont les colonnes sont des Diracs ou des sinusoides.

3. Ecrire un programme

`X=Genevect(d,N1)`

qui génère un vecteur colonne de longueur $N1$, ayant exactement d composantes non nulles. Chaque composante étant la réalisation d'une variable aléatoire gaussienne.

4. A l'aide des deux fonctions précédentes créer un mélange d'un petit nombre de Diracs et de sinusoides. Afficher sur trois figures différentes la partie Diracs, la partie sinusoides et la sommes. On prendra de préférence des signaux de taille $N = 1024$.

5. Ecrire un programme

   `X=MP(Y,A,er)`

   qui prend en entrée, un vecteur Y de taille N, une matrice de taille $N \times 2N$ et un réel positif er qui renvoie un vecteur X de taille X tel que $\|Y - AX\|_2^2 \leqslant er$ en utilisant le MP.

6. Ecrire un programme utilisant le MP qui à partir d'un mélange de sinusoides et de Diracs, sépare et affiche les deux composantes et qui calcule l'erreur de reconstruction.

7. Tester la fonction précédente à partir d'un mélange de 3 Diracs ou sinusoides. Puis augmenter la taille du mélange pour voir jusqu'à quel niveau de mélange, le MP arrive à faire la séparation.

8. Ecrire un programme

   `[er1,er2]=MPtest(Y,A,X,NBmax)`

   prenant en entrée un vecteur Y de taille N, une matrice A de taille $N \times 2N$, un vecteur X de taille $2N$ et un nombre d'étapes NBmax et qui renvoie et affiche les vecteurs er1 d'erreur de reconstruction par rapport aux observations Y et er2 d'erreur de reconstruction par rapport aux données *réelles* X en fonction du nombre d'étapes.

9. Tester la fonction précédente avec plusieurs tailles de mélanges et différents niveaux de bruit. On utilisera un modèle de bruit gaussien sur les observations.

10. A partir de ces courbes, peut on déduire un critère d'arrêt pertinent ?

11. Proposer un critère d'arrêt théorique et une manière pratique de le mettre en place. Ce critère permet il d'avoir de bon résultat ?
    D'une manière générale dans les méthodes itératives de reconstruction il est important de savoir quand s'arrêter. Si on s'arrête trop tôt on ne récupère pas assez de signal, si on s'arrête trop tard on a une représentation moins compacte et surtout quand on fait du débruitage on risque de récupérer trop de bruit.

12. Faire le travail correspondant pour l'OMP et comparer les deux méthodes à divers niveaux de parcimonie et différents niveaux de bruit. On commencera par rédiger la fonction

    `X=OMP(Y,A,er)`

    Pour l'OMP on a besoin de la pseudo inverse d'une matrice ...

13. Proposez un algorithme de séparation par un seuillage brut des données et comparer ses performances à celles du MP et de l'OMP.

14. Qu'en concluez vous ?

3 Matching Pursuit pour la compression et le débruitage de sons

Nous allons maintenant exploiter le fait qu'un son musical peut être approché par un petit nombre de *sinusoides locales*, c'est à dire d'atomes de Gabors, produit d'une fenêtre de Hanning et d'une sinusoide. (Rq :Les atomes de Gabor sont souvent fenêtrés avec des gaussiennes mais nous utiliserons ici un fenêtrage de Hanning.)
Une première idée serait de calculer tous les atomes du dictionnaire et d'effectuer les produits scalaires un à un comme nous l'avons fait dans l'exemple précédent. Si on considère un son échantillonné à 44000 Hz même avec un recouvrement de moitié, on a 88000 atomes. Calculer tous les produits scalaires n'est pas l'approche optimale ici. On peut utiliser la puissance de la fft pour limiter les calculs.
Nous allons plus précisément utiliser une transformée de Fourier à fenêtres pour estimer les produits scalaires avec les différents atomes temps-Fréquences.

15. Ecrire un programme Analyse.m qui prend pour entrée un vecteur colonne S et un entier N et qui renvoie un tableau de transformée de Fourier à fenêtres. La première ligne du programme est

    ```
    function [Tfct]=Analyse(S,N)
    ```

    Dans l'analyse comme dans la synthèse on utilisera des fenêtres de taille N, décalées de $N/8$. En chaque point passeront ainsi 8 fenêtres.
    On tronquera le signal à un nombre entier de fenêtres, en pratique on prendra souvent $N = 1024$.
    On pourra utiliser les notations suivantes

    - NS taille de S.
    - Nf nombre de fenêtres d'analyse.
    - H fenêtre de Hanning de taille N.

    Remarques :

    - $Nf = floor(8 * NS/N) - 7$.
    - La fenêtre d'indice k commence à l'indice $1 + (k-1)N/8$ et se termine à l'indice $(k-1)N/8 + N$.
    - Ce programme peut être écrit proprement en moins de 15 lignes de matlab.

16. La transformée de Fourier à fenêtres permet d'estimer le produits scalaires du signal sonore avec chacun des atomes temps-Fréquence. Ecrire un programme AtomeGabor

    ```
    a=AtomeGabor(Tfct,I1,I2,h)
    ```

    Prenant en entrée une transformée de Fourier à fenêtre Tfct, deux indices entiers I1 et I2 désignant un coefficient de la Tfct, une fenêtre h et qui renvoie la projection du signal sur l'atome de Gabor associé. Le vecteur a est alors un signal de même taille que le signal d'origine, formée d'une sinusoide dont la fréquence est donnée par I1, l'amplitude par Tfct, la position par I2.
    Remarque : On peut aussi programmer une fonction qui à une taille de fenêtre et une

fréquence, renvoie un atome de Gabor local de la taille de la fenêtre. En utilisant ensuite le Tfct et la position de l'atome on peut calculer la projection du signal sur cet atome. La fonction s'insèrera alors différemment dans le code de MP.

17. Ecrire un code de Matchnig Pursuit

    ```
    [Srec,Residu]=MP1(S,N)
    ```

    prenant un vecteur S, un nombre d'atomes N et renvoyant une approximation et un résidu de S avec N atomes en utilisant la fonction précédente.

18. Tester la fonction précédente sur un extrait d'une seconde de son et afficher le temps de calcul en fonction du nombre d'atomes sélectionnés. Cette méthode est elle viable pour un son de plus de 10 secondes ? Peut on se passer totalement de recalculer la tansformée de Fourier à fenêtres à chaque étape ?

19. Dès que le son est un peu long (plusieurs secondes) et qu'on souhaite réaliser une approximation avec un grand nombre d'atomes, l'algorithme qui consiste à recalculer le Transformée De Fourier À Fenêtres à chaque étape est trop lent. Si certains atomes sont assez correlés comme c'est le cas dès qu'on utilise un recouvrement quelconque, la projection sur un atome modifie les produits scalaires avec les atomes voisins c'est pourquoi il est nécessaire de recalculer ces produits scalaires avant d'essayer de sélectionner un de ces atomes voisins. Une première méthode consiste à sélectionner plusieurs atomes à chaque étape en prenant soin de délimiter une zone de sureté autour de chaque atome sélectionné. Dans ce cas on peut ne recalculer le transformée de Fourier à fenêtres que toutes les 10, 20 ou 30 sélections d'atomes. On peut aussi ne recalculer qu'une petite partie du transformée de Fourier à fenêtres à chaque fois, celle correpsondant aux atomes potentiellement modifiés par la sélection. Programmer une version plus rapide de l'algorithme de MP.

20. Tester le programme précédent sur des données bruitées. Quel critère d'arrêt du MP peut on proposer ?

21. On peut aussi débruiter le signal audio en effectuant un seuillage simple de le transformée de Fourier à fenêtres. Ecrire un code d'approximation/Débruitage par seuillage de la transformée de Fourier à fenêtres.

22. Comparer à l'oreille, en norme ℓ_2 et en temps de calcul les deux méthodes sur différents exemples et différents niveaux de bruit.