LocOMP: algorithme localement orthogonal pour l'approximation parcimonieuse rapide de signaux longs sur des dictionnaires locaux

Boris Mailhé, Rémi Gribonval, Frédéric Bimbot, Pierre Vandergheynst

LocOMP: algorithme localement orthogonal pour l'approximation parcimonieuse rapide de signaux longs sur des dictionnaires locaux

Boris MAILHÉ[1], Rémi GRIBONVAL (INRIA)[1], Frédéric BIMBOT (CNRS)[1], Pierre VANDERGHEYNST[2]

[1] IRISA, Campus de Beaulieu
35042 Rennes CEDEX, France

[2] LTS2, station 11, EPFL 1015 Lausanne, Suisse

Prenom.Nom@irisa.fr
Pierre.Vandergheynst@epfl.ch

Résumé – Cet article présente LocOMP, un algorithme d'approximation parcimonieuse rapide pour les dictionnaires locaux (en particulier, les dictionnaires invariants par translation comme les dictionnaires temps-fréquence). Expérimentalement, LocOMP fournit des approximations aussi bonnes qu'OMP tout en restant dans la même classe de complexité que MP. C'est à notre connaissance le premier algorithme plus évolué que MP utilisable sur des signaux longs. LocOMP s'est montré expérimentalement 500 fois plus rapide qu'OMP.

Abstract – We propose a variant of Orthogonal Matching Pursuit (OMP), called LoCOMP, for scalable sparse signal approximation. The algorithm is designed for shift-invariant signal dictionaries with localized atoms, such as time-frequency dictionaries, and achieves approximation performance comparable to OMP at a computational cost similar to Matching Pursuit. Numerical experiments with a large audio signal show that, compared to OMP and Gradient Pursuit, the proposed algorithm runs in over 500 less time while leaving the approximation error almost unchanged.

1 Introduction

L'approximation parcimonieuse consiste, étant donné un signal s de dimension N et un *dictionnaire* Φ, à rechercher la meilleure approximation de s à $K < N$ termes de Φ appelés *atomes* :

$$\hat{x} = \text{argmin}_{\|x\|_0 \leq N} \|s - \Phi x\|_2 \quad (1)$$

C'est une étape préliminaire utile à de nombreux traitements comme la compression, la séparation de sources ou plus récemment l'échantillonnage compressé. Ce problème est NP-Complet mais de nombreux algorithmes polynomiaux sous-optimaux ont été proposés.

Parmi eux, les algorithmes de poursuite sélectionnent itérativement l'atome le plus corrélé au résidu courant et le soustraient. Cette famille comprend principalement la poursuite simple (MP) [5] et la poursuite orthogonale (OMP) [3], ainsi que des variantes comme la poursuite de gradient (GP) [4] ou ou l'algortihme glouton relaxé (RGA)[6].

A chaque itération i, un algorithme de poursuite effectue 2 opérations :

– sélectionner le meilleur atome φ_i dans le dictionnaire $\mathbf{\Phi}$ (le plus souvent, c'est l'atome le plus corrélé au résidu précédent r_{i-1})

$$\varphi_i = \text{argmax} |\mathbf{\Phi}^* r_{i-1}| \quad (2)$$

– mettre à jour le résidu r_i en retirant une approximation n'utilisant que des atomes déjà sélectionnés $\mathbf{\Phi}_i$

Plusieurs règles de mise à jour ont été proposées, parmi lesquelles :

1. MP :
$$r_i = r_{i-1} - \langle r_{i-1}, \varphi_i \rangle \varphi_i \quad (3)$$
2. OMP :
$$r_i = r_{i-1} - \mathbf{\Phi}_i (\mathbf{\Phi}_i^* \mathbf{\Phi}_i)^{-1} \mathbf{\Phi}_i^* r_{i-1} \quad (4)$$
3. GP :
$$r_i = r_{i-1} - \frac{\|\mathbf{\Phi}_i^* r_{i-1}\|_2^2}{\|\mathbf{\Phi}_i \mathbf{\Phi}_i^* r_{i-1}\|_2^2} \mathbf{\Phi}_i \mathbf{\Phi}_i^* r_{i-1} \quad (5)$$

MP est la moins coûteuse de ces règles mais elle peut produire des approximations significativement moins bonnes que celles calculées par OMP. Les variantes se situent entre les deux, tentant d'obtenir la qualité d'OMP avec la vitesse de MP.

On rencontre souvent la situation où un signal long (par exemple, de la musique échantillonnée à 44,1kHz qui compte presque 3 millions d'échantillons par minute) sur un dictionnaire *local*, c'est-à-dire dont les atomes sont support de longueur $L \ll N$. Dans ce cas, on sait que MP peut être implémenté très efficacement. Ceci n'est pas possible avec les autres algorithmes gloutons existants.

Cet article présente un algorithme qui calcule de meilleures approximations que MP tout en restant dans la même classe de complexité pour les dictionnaires locaux. Cet algorithme passe à l'échelle car il respecte une propriété de localité de mise à jour du résidu. Il est largement plus rapide qu'OMP, tout en calculant expérimentalement d'aussi bonnes approximations.

2 Complexité des algorithmes de poursuite

Pour mettre au point un algorithme plus rapide, la première tâche est d'identifer la partie coûteuse dans les algorithmes existants en fonction de N, L, le nombre d'atomes du dictionnaire $D = \alpha N$ et celui de la décomposition souhaitée K. Dans le cas des algorithmes de poursuites, on observe 2 comportements différents dans le cas général et dans le cas des dictionnaires locaux.

2.1 Cas général

Dans le cas général, la sélection du meilleur atome se fait de la même manière pour tous les algorithmes. Elle consiste à calculer le vecteur $\lambda = \mathbf{\Phi}^* r$, ce qui se fait en $O(DN) = O(\alpha N^2)$, puis $\operatorname{argmax}|\lambda|$ qui nécessite $\alpha N - 1$ comparaisons. C'est le coût de calcul des produits scalaires qui domine. Certains dictionnaires permettent de calculer les produits scalaires plus rapidement, par transformée de Fourier rapide par exemple.

La seule différence entre les algorithmes est dans la mise à jour du résidu. Pour MP, il s'agit d'une simple soustraction en $O(N)$. Pour OMP, il faut en plus calculer la dernière ligne et la dernière colonne de la matrice de Gram $G_i = \mathbf{\Phi}_i^* \mathbf{\Phi}_i$ (le reste était connu aux itérations précédentes), ce qui nécessite le calcul de $\mathbf{\Phi}_{i-1}^* \varphi_i$ en $O(iN)$, puis l'inverser, ce qui peut se faire en $O(i^2)$ par une descente de gradients conjugués. Pour GP, le gradient $\mathbf{\Phi}_i \mathbf{\Phi}_i^* r_{i-1}$ se calcule en $O(iN)$ ($\mathbf{\Phi}_i^* r_{i-1}$ est un sous-vecteur de λ) et son énergie $\|\mathbf{\Phi}_i \lambda\|_2^2$ qui détermine le pas de la descente en $O(N)$.

2.2 Dictionnaires locaux

Supposons maintenant que la longueur du support d'un atome est $L \ll N$.

Alors le coût de calcul de calcul d'un produit scalaire en $O(L)$ au lieu de $O(N)$. Donc le coût de l'étape de sélection est ramené à $O(\alpha N L)$ et celui du calcul de G_i à $O(iL)$. De plus, la matrice G_i est creuse : 2 atomes dont les supports de se chevauchent pas ont un produit scalaire nul. Si on suppose que les atomes sélectionnés sont répartis uniformément sur la longueur du signal, le dernier atome φ_i ne sera corrélé en moyenne qu'à $O(\frac{iL}{N})$ atomes précédents. Celà réduit encore le coût du calcul de G_i à $O(\frac{iL^2}{N})$, et celui de son inversion par descente de gradient conjugué à $O(\frac{i^2 L^2}{N})$.

Les atomes utiles être retrouvés rapidement en triant les atomes de $\mathbf{\Phi}_i$ par débuts de support croissants. Pour effectuer ce tri de manière efficace, on peut utiliser un arbre binaire de recherche auto-équilibré, par exemple les arbres rouge-noir qui sont utilisés dans les implémentations d'ensembles triés des librairies standard Java et C++. Cette implémentation garantit une complexité en $O(\log i)$ pour insérer un nouvel atome et pour extraire un intervalle : il suffit d'extraire le premier et le dernier élément de l'intervalle.

Pour GP la calcul du gradient ne demande plus que $O(iL)$ itérations mais le rapport d'énergies reste en $O(N)$.

Dans le cas particulier de MP, la sélection peut encore être accélérée. À chaque itération, le résidu n'est modifié que sur une longueur L. Tous les atomes dont le support ne chevauche pas le dernier conservent donc la même corrélation. Le coût de la mise à jour de corrélations est donc réduit à $O(\alpha L^2)$, et celui de la recherche du maximum à $O(\log \alpha N)$ grâce au stockage des comparaisons calculées précédemment dans un arbre de tournoi. Ces accélérations ont été implémentées dans la librairie MPTK [2] [1].

Le tableau 1 résume tous ces résultats. Quand N devient grand, l'étape la plus coûteuse n'est plus la projection mais le calcul de λ. Les approches prédécentes de projection rapide n'apportent aucune aide ici. La baisse de complexité de MP est due au fait que la mise à jour du résidu est locale. C'est pourquoi nous avons cherché un intermédiaire entre MP et OMP qui préserve cette propriété.

3 Poursuite localement orthogonale (LocOMP)

3.1 Algorithme

L'idée centrale de l'algorithme proposé est de sélectionner à chaque itération un voisinage $\mathbf{\Psi}_i \subset \mathbf{\Phi}_i$ du dernier atome φ_i et de n'orthogonaliser la décomposition que sur ce sous-dictionnaire. Le fonctionnement complet est détaillé dans l'algorithme 1. Le choix du voisinage définit une progression entre MP et OMP :

- si on ne sélectionne que le dernier atome ($\mathbf{\Psi}_i = \{\varphi_i\}$), alors on retombe sur MP.
- inversement, si on garde tous les atomes ($\mathbf{\Psi}_i = \mathbf{\Phi}_i$), alors c'est OMP qu'on obtient.

Le voisinage doit être choisi pour fournir un bon compromis entre la qualité de l'approximation et et la taille du support de mise à jour. Dans ce travail, il a été choisi de conserver tous les atomes $\varphi \in \mathbf{\Phi}_i$ dont le support recouvre partiellement celui du dernier atome φ_i. Nous avons vu à la section 2.2 cet ensemble est également celui qu'il faut extraire pour mettre à jour G_i, donc ce choix économise une recherche.

3.2 Complexité

LocOMP est légèrement moins local que MP. Avec le choix de voisinage proposé, à chaque étape le résidu est modifié sur un intervalle de longueur maximale $3L - 2$ au lieu de L pour MP. Il faut donc recalculer les corrélations pour tous les atomes dont le support est inclus dans un intervalle de longueur maximale $5L - 3$ pour tenir compte des effets de bord, contre $3L - 2$

[1] `http://mptk.irisa.fr`

width=88mm

TAB. 1 – Complexité d'une itération $i > 1$ de MP, OMP et GP, dans le ca général et avec des dictionnaires locaux

Étape	Cas général			Dictionnaires locaux		
	MP	OMP	GP	MP	OMP	GP
Sélection	αN^2			αL^2	αNL	
Projection	N	$iN + i^2$	iN	L	$\frac{i^2L^2}{N}$	$\frac{iL^2}{N} + N$

pour MP. Le coût de la sélection reste donc en $O(\alpha L^2)$, et est au maximum de l'ordre de $\frac{5}{3}$ de celui de MP.

D'autre part, un effet collatéral de la sélection du voisinage est que la matrice de Gram qu'on cherche à inverser est de dimension réduite par rapport à OMP. Si le calcul de ses coefficients demande toujours $O(\frac{iL^2}{N})$ opérations, la matrice est de taille $O(\frac{IL}{N})$, donc la projection se calcule en $O(\frac{i^2L^2}{N^2})$. Comme $L \ll N$, le coût de la projection devrait donc rester négligeable devant le coût de la sélection. On s'attend donc à observer des temps de calcul de l'ordre de $\frac{5}{3}$ de ceux de MP.

Algorithm 1 x = LocOMP(s, $\mathbf{\Phi}$)

$r_0 = s$
$\mathbf{\Phi}_0 = \emptyset$
$x_0 = 0$
for $i = 1$ to I **do**
  $\varphi_i = \text{argmax}_{\varphi \in \mathbf{\Phi}} |\langle r_{i-1}, \varphi \rangle|$ {recherche du meilleur atome}
  $\mathbf{\Phi}_i = \mathbf{\Phi}_{i-1} \cup \varphi_i$
  $\mathbf{\Psi}_i = \texttt{neighbour}(\mathbf{\Phi}_i, \varphi_i)$ {sélection du sous-dictionnaire}
  $\chi_i = (\mathbf{\Psi}_i^* \mathbf{\Psi}_i)^{-1} \mathbf{\Psi}_i^* r_{i-1}$ {calcul des coefficients de la projection du résidu sur le sous-dictionnaire}
  $x_i = x_{i-1} + \chi_i$ {mise à jour des coefficients}
  $r_i = r_{i-1} - \mathbf{\Psi}_i \chi_i$ {mise à jour du résidu}
end for
return x_I

4 Résultats expérimentaux

Des implémentations Matlab de LocOMP, MP, OMP et GP ont été testées sur un extrait de guitare jazz d'une minute extrait de la base RWC [7] et sous-échantillonné à 8kHz ($N = 4.8 \cdot 10^5$ échantillons). Les algorithmes ont bouclé pendant $I = 20000$ itérations pour décomposer le signal sur un dictionnaire d'atomes MDCT de taille $L = 32$ totalement invariant par translation (taille du dictionnaire $\approx 8 \cdot 10^6$ atomes).

La figure 1 montre le RSB atteint par les différents algorithmes à chaque itération. A cette échelle, on ne peut pas différencier OMP, GP et LocOMP. Le RSB atteint par LocOMP au bout des 20000 itérations n'est que 0.01dB plus bas que pour OMP et GP, alors que celui atteint par MP est 0.6dB plus faible.

Pour chaque algorithme, le temps de calcul observé d'une itération a augmenté au plus linéairement en cours d'exécution. Cette augmentation linéraire pour OMP est le signe que le calcul de G_i prend plus de temps que la descente de gradient.

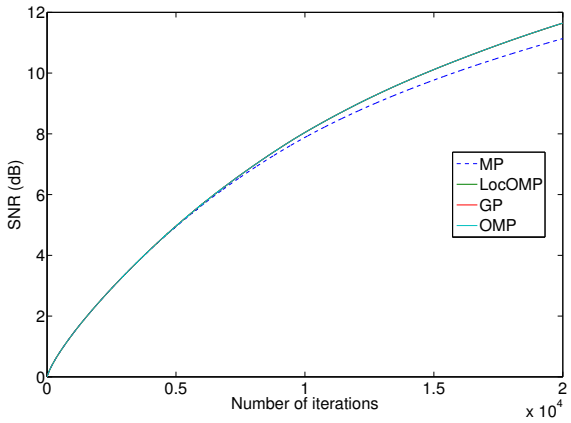

FIG. 1 – RSB en fonction de l'itération

C'est confirmé par la proximité des temps de calcul entre OMP et GP. Le tableau 2 donne le temps de calcul de la première itération (qui est plus coûteuse que les suivantes pour MP et LocOMP parce qu'elle nécessite le calcul exhaustif des corrélations), l'ordre de grandeur du coût d'une itération au début et à la fin de l'exécution, ainsi que la durée totale d'exécution.

Les algorithmes sont clairement séparés en 2 groupes. La chute du coût pour MP après la première itération montre que l'essentiel le calcul des corrélations représentait l'essentiel du coût de celle-ci. Jusqu'à la fin de l'exécution, les itérations restent nettement moins coûteuses que la première. Inversement, pour GP et OMP le coût de la première itération est une borne inférieure pour les suivantes. Même si on arriver à calculer la projection instantanément, ils resteraient largement plus coûteux que MP. De plus, la projection se fait sur le dictionnaire Φ_i plus gros que Ψ_i, l'augmentation du coût au cours de l'exécution est aussi plus importante. Le calcul total a pris de l'ordre de 5 minutes pour MP, 15 minutes pour LocOMP, et 5 *jours* pour OMP.

Le temps d'exécutation de LocOMP est 1.5 fois supérieur à celui de MP, ce qui est du même ordre, mais légérement inférieur à la borne supérieure théorique de $\frac{5}{3}$. Sur cet exemple, LocOMP a obtenu des performances similaires à OMP pour un temps de calcul 500 fois moindre.

4.1 Un exemple d'application : codage audio à débit variable

Dans cette expérience nous avons tenté d'intégrer LocOMP dans le système de codage audio à débit variable proposé par Ravelli [1]. Le signal à 8 kHz est décomposé sur l'union de 2

TAB. 2 – Temps d'exécution par itération (s)

Itération	MP	LocOMP	GP	OMP
Première ($i = 0$)	3.4	3.4	3.4	3.5
Début ($i \approx 1$)	0.028	0.033	3.4	3.4
Fin ($i \approx I$)	0.028	0.050	40.5	41
Durée totale	571	854	$4.50 \cdot 10^5$	$4.52 \cdot 10^5$

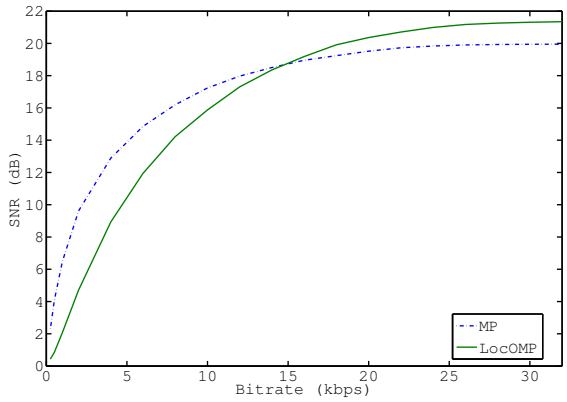

FIG. 2 – SNR depending on the decoding bit rate

bases MDCT d'échelles $L_1 = 32$ et $L_2 = 256$. Ces échelles sont proches de celles utilisées en codage AAC sur des signaux échantillonnés à 44.1kHz. Ensuite cette décomposition est codée en niveaux de bits, ce qui permet de ne transmettre que les niveaux es plus forts en fonction du débit autorisé.

La courbe débit/distortion 2 compare les performances obtenues en utilisant MP ou LocOMP pour la décomposition. LocOMP apporte un gain de jusqu'à 1.4dB à haut débit. Néanmoins MP reste meilleur à bas débit. L'avantage de MP à bas débit semble être un phénomène général, mais son ampleur semble être liée au choix d'un dictionnaire plus pauvre que le dictionnaire à 8 échelles utilisé par Ravelli [1].

5 Conclusion

LocOMP est un algorithme d'approximation parcimonieuse qui partage les même propriétés de passage que MP pour des signaux et des dictionnaires invariants par translation, tout en en procurant des approximation de meilleure qualité. C'est à notre connaissance le premier algorithme à cumuler ces 2 propriétés.

Ces gains ont été mesurés expérimentalement sur des signaux musicaux sous-échantillonnés et des dictionnaires MDCT d'échelle courte. Ces données ont été volontairement sous-dimensionnées pour pouvoir exécuter OMP et comparer. La gain de qualité par rapport à MP pourrait être plus important avec de meilleurs dictionnaires.

Une version localisée de GP qui exploite ces idées est actuellement en cours d'implémentation dans MPTK. Un prototype est déjà fonctionnel, mais sa vitesse d'exécution est encore loin de MP et des limites théoriques. De nouvelles applications pourront être envisagées quand ce développement sera achevé.

6 Remerciements

Les auteurs remercient Emmanuel Ravelli et Laurent Daudet du LAM à l'Université de Paris VI, pour leur aide dans l'application au codage audio.

Références

[1] E. Ravelli, G. Richard et L. Daudet. *Extending fine-grain scalable audio coding to very low bitrates using overcomplete dictionaries*. WASPAA, 2007.

[2] S. Krstulovic et R. Gribonval. *MPTK : Matching Pursuit made Tractable*. ICASSP, 2006.

[3] Y.C. Pati, R. Rezaiifar et P.S. Krishnaprasad. *Orthonormal Matching Pursuit : recursive function approximation with applications to wavelet decomposition*. Annual Asilomar Conf. on Signals, Systems and Computers, 1993.

[4] T. Blumensath et M.E. Davies. *In greedy pursuit of new directions : (nearly) orthogonal matching pursuit by directional optimisation*. EUSIPCO, 2008.

[5] S. Mallat et Z. Zhang. *Matching Pursuit With Time-Frequency Dictionaries*. IEEE transactions on Signal Processing, 1993.

[6] A.R. Barron, A. Cohen, W. Dahmen et R.A. DeVore. *Approximation and learning by greedy algorithms*. Annals of statistics, 2008.

[7] G. Masataka. *Development of the RWC Music Database*. International Congress on Acoustics, 2004