Rapport du projet de Compressive Sensing

Dean BAH - Mathilde RAMSAMY–AGEORGES - Marwa TOURABI

March 17, 2024

1 Introduction

2 Apprentissage d'un dictionnaire (k-SVD)

3 Présentation d'algorithmes de codage parcimonieux

4 Comparaison d'algorithmes de codage parcimonieux

5 Matrices de mesure

6 Conclusion

7 Annexes

1. Introduction

Dans le cadre du quatrième semestre de notre formation d'Ingénieur en Mathématiques Appliquées, nous avons entrepris un projet portant sur l'application du Compressive Sensing, dans le cadre du cours dédié à cette thématique.
Ce rapport exposera tout d'abord le processus d'apprentissage d'un dictionnaire (k-SVD), le choix judicieux d'une matrice de mesure, ainsi que la mise en œuvre d'algorithmes de codage parcimonieux. Nous procéderons ensuite à une explication détaillée des méthodes mathématiques employées dans ce projet, avant d'analyser de manière critique les résultats obtenus. Enfin, nous formulerons nos conclusions.

2. Apprentissage d'un dictionnaire par k-SVD

Question 1

L'apprentissage par K-SVD est une technique utilisée en Compressive Sensing pour construire un dictionnaire parcimonieux. En pratique, il s'agit d'un processus itératif qui ajuste à la fois les coefficients de représentation et les éléments du dictionnaire. Ces ajustements visent à capturer de manière optimale les structures sous-jacentes des données observées. Le but est de minimiser l'erreur de reconstruction tout en maximisant la capacité du dictionnaire à représenter efficacement les données avec un nombre réduit de coefficients significatifs.

Question 2

Afin de réaliser cette étape, nous avons importer le fichier *DonneesProjets.xlsx* grâce à la bibliothèque *pandas* de Python. En utilisant ce package, les données importées sont sous forme de dataframe. Il a donc fallu convertir le dataframe en *np.array* pour pouvoir appliquer les différents traitements mathématiques. Après avoir appris le dictionnaire grâce à l'algorithme *ksvd.py*, nous avons utilisé le package *numpy* pour enregistrer le dictionnaire appris en fichier *Dico-appris.csv*.
Pour une analyse détaillée de l'apprentissage du dictionnaire k-SVD à partir des 100 premières colonnes du fichier "DonneeProjet.xlsx", veuillez consulter les fichiers intitulés *ksvd.py* et *Dico-appris.csv*.

3. Présentation d'algorithmes de codage parcimonieux

Question 1

Les algorithmes OMP (Orthogonal Matching Pursuit), StOMP (Stagewise Orthogonal Matching Pursuit) et CoSaMP (Compressive Sampling Matching Pursuit) sont tous utilisés dans le domaine du traitement du signal et de la com-

pression de données pour reconstruire des signaux à partir d'échantillons compressés. Malgré leur partage d'un objectif commun, ils se distinguent nettement dans leur approche et leurs résultats.
Tout d'abord, ces trois algorithmes emploient une méthode itérative appelée MP pour Matching Pursuit (en français poursuite de correspondance) afin de choisir les atomes du dictionnaire qui reflètent le signal en cours de reconstruction de manière optimale. Ils peuvent également être adaptés à divers dictionnaires en fonction des caractéristiques du signal.
Cependant, leurs méthodes de sélection d'atomes diffèrent :

- OMP sélectionne à chaque itération l'atome du dictionnaire qui maximise la corrélation avec le résidu actuel
- StOMP utilise une approche stagiaire où plusieurs atomes potentiels sont ajoutés et éliminés à chaque itération en fonction de leur corrélation
- CoSaMP effectue une sélection en deux étapes : d'abord en choisissant un ensemble potentiel d'atomes, puis en affinant cette sélection pour ne garder que les plus significatifs.

En ce qui concerne la complexité temporelle, OMP est généralement plus rapide que StOMP et CoSaMP en raison de sa méthode de sélection plus simple. Cependant, StOMP et CoSaMP peuvent avoir de meilleures performances dans des scénarios où le signal est fortement bruité ou compressé, bien que leur complexité temporelle soit plus élevée.
En termes de convergence, OMP converge vers une solution exacte si le signal est suffisamment parcimonieux et le nombre d'échantillons est suffisamment grand. StOMP et CoSaMP peuvent également converger vers une solution exacte, mais ils sont souvent utilisés dans des situations où une approximation de haute qualité est suffisante.
En résumé, OSMP, StOMP et CoSaMP partagent un objectif commun et une méthode de base similaire. Leurs différences dans la sélection d'atomes, la complexité temporelle et les performances dans différents scénarios les rendent adaptés à différents types de problèmes de reconstruction de signal.

Question 2

Si aucune contribution ne dépasse le seuil lors d'une itération dans l'algorithme StOMP (Stagewise Orthogonal Matching Pursuit), cela signifie qu'aucun atome du dictionnaire n'a une corrélation adéquate avec le résidu actuel pour être ajouté à la solution. Dans cette situation, StOMP a la possibilité d'adopter différentes méthodes :

- Méthode 1 : Si aucun atome ne dépasse le seuil, l'algorithme StOMP peut toujours sélectionner l'atome du dictionnaire qui est le plus lié au résidu d'origine. Bien que cette relation ne soit pas assez forte pour dépasser le seuil, l'incorporation de cet atome peut aider à améliorer la reconstruction du signal.

- Méthode 2 : Il est possible de diminuer le seuil afin de favoriser l'inclusion d'atomes moins liés. En modifiant le seuil, l'algorithme élargit la zone de recherche, ce qui peut entraîner une sélection plus efficace d'atomes et donc une reconstruction plus efficace du signal. Toutefois, si l'ajout d'atomes significativement corrélés n'est pas réalisable même après avoir diminué le seuil, StOMP peut être interrompu lorsque l'algorithme atteint un nombre maximum d'itérations ou si les itérations ne se rapprochent pas d'une solution correcte. Dans cette situation, on considère que la reconstruction du signal est totale, même si elle peut ne pas être optimale.

Question 3

Présentation du problème à travers le sujet.

Question 4

Soit A une matrice rectangulaire d'ordre (n, p) avec $n < p$. Le problème $\min\|x_2\|$ sous la contrainte $Ax = b$ se résout avec la solution donnée par les moindres carrés qui est :

$$x = A^+ b = (A^T A)^{-1} A^T b.$$

Or l'objectif d'IRLS est de détourner cette méthode pour résoudre :

$$\min\|\alpha\|_2 \text{ sous la contrainte } D\alpha = x \text{ avec } 0 < p < 1.$$

Ainsi:

$$\|\alpha\|_p^p = \sum_{i=1}^k |\alpha_i|^{p-2} = \sum_{i=1}^k |a_i|^2 = \omega^2 \sum_{i=1}^k |a_i|^2 = \sum_{i=1}^k |\omega_i a_i|^2 = \|W\alpha\|_2^2$$

avec W la matrice diagonale de coefficients égaux à $w_i = |\alpha_i|^{\frac{p}{2}-1}$.

Question 5

La recherche du minimum de $\|\alpha\|_p^p$ revient à rechercher $\min\|W\alpha\|_2^2$. Étant donné qu'il y a une norme 2, nous allons utiliser la méthode des moindres carrés.

On va donc poser $z = W\alpha$, vecteur dont on cherche à minimiser la norme. Avec comme contrainte $D\alpha = x \Leftrightarrow DW^{-1}z = x$.

Nous avons donc à résoudre : $\min\|z\|_2$ sous la contrainte $DW^{-1}z = x$.

La méthode des moindres carrés donne alors :
$z = (DW^{-1})^T((DW^{-1})^T DW^{-1})^{-1}x$
$z = W^{-1}D^T(DW^{-1}W^{-1}D^T)^{-1}x$
$z = W^{-1}D^T(DW^{-2}D^T)^{-1}x$

On en déduit ensuite α par :

$\alpha = W^{-1}z = W^{-2}D^T(DW^{-2}D^T)^{-1}x$

On appelle Q la matrice W^{-2}. C'est elle que l'on retrouve dans l'algorithme.
$\alpha = QD^T(DQD^T)^{-1}x$
On obtient ainsi α en fonction de matrice dépendante pour certaines valeurs de α.

Question 6

Ce qu'on peut écrire : $\alpha = f(\alpha)$.

L'obtention de α se fait donc à l'aide d'une suite récurrente $\alpha^{(k)} = f(\alpha^{(k-1)})$.

Le passage de α à z et inversement nécessite l'inversibilité de W. Ce qui ne peut être garanti. Voilà pourquoi on ajoute un $c > 0$ aux coefficient sur w_i^2.

On remplace donc : $w_i^2(|\alpha_i|^2 + \epsilon)^{\frac{p}{2}-1}$ pour être sûre que W soit inversible.

Ce même ϵ est utilisé pour le critère d'arrêt. Le critère retenu pour IRLS est le suivant : $\|\alpha^{(k)} - \alpha^{(k-1)}\| < \frac{\sqrt{\epsilon}}{100}$ et $\epsilon < 10^{-8}$.

Question 7

L'implémentation de l'alogirithme IRLS se trouve dans le fichier nommé *irls.py*.

4. Comparaison des algorithmes de codages parcimonieux

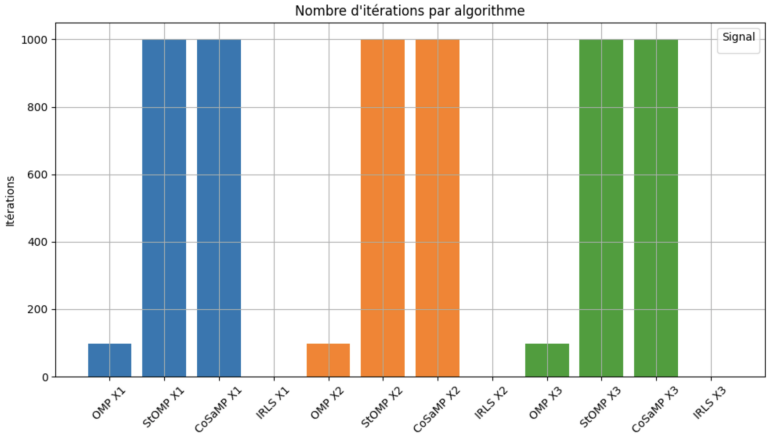

Figure 1: Nombre d'itérations par algorithme.

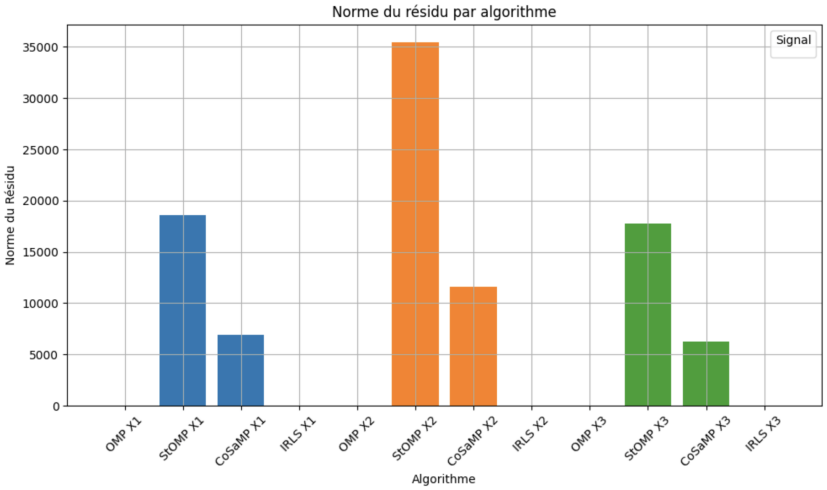

Figure 2: Norme du résidu par algorithme.

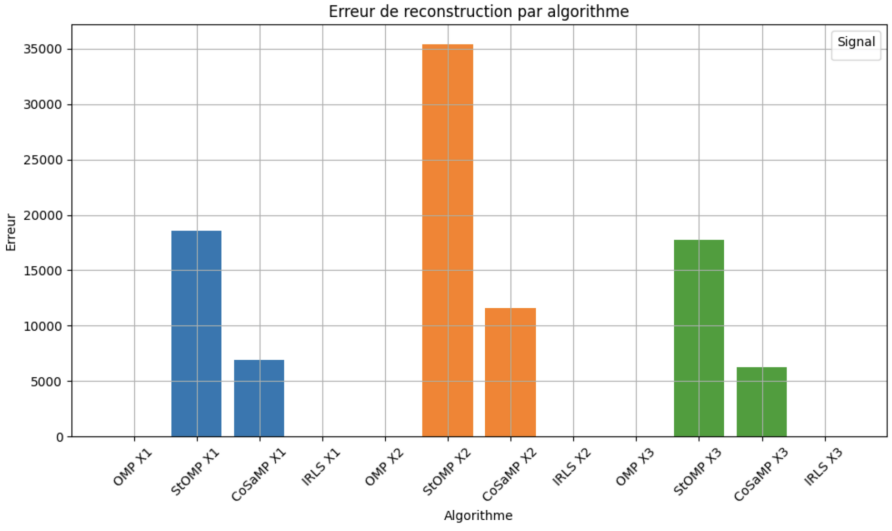

Figure 3: Erreur de reconstruction par algorithme.

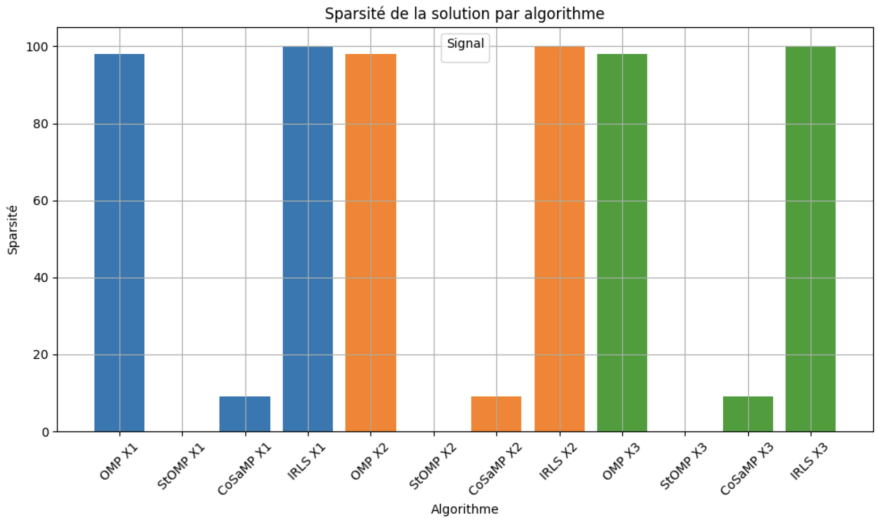

Figure 4: Sparcité de la solution par algorithme.

Dans le cadre de la comparaison des performances des quatre algorithmes de reconstruction de signaux parcimonieux - OMP, StOMP, CoSaMP, et IRLS - Nous avons appliqués à trois signaux différents (X1, X2, et X3), une analyse approfondie sur la base de trois critères principaux : le nombre d'itérations nécessaires pour atteindre la convergence, la norme du résidu post-reconstruction et la sparsité.
Les résultats obtenus démontrent que l'algorithme OMP présente une tendance à converger avec un nombre d'itérations relativement faible pour tous les signaux testés, suggérant une efficacité notable en termes de vitesse de calcul.
En revanche, l'algorithme CoSaMP, tout en nécessitant un nombre d'itérations légèrement supérieur, a montré une capacité à minimiser la norme du résidu, indiquant potentiellement une précision de reconstruction supérieure. Les performances de StOMP et d'IRLS se situent entre ces deux extrêmes, offrant un compromis entre vitesse et précision de reconstruction.

5. Matrices de mesure

Question 1

Nous avons construit les matrices de mesure Φ_1, Φ_2, Φ_3 et Φ_4, puis calculé leur cohérence mutuelle avec le dictionnaire appris par k-SVD.

- Φ_1 : matrice aléatoire obtenue par une loi uniforme $U(0,1)$
- Φ_2 : matrice aléatoire obtenue par une loi de Bernoulli avec des valeurs -1 ou 1.

- Φ_3 : matrice aléatoire obtenue par une loi de Bernoulli avec des valeurs 0 ou 1.
- Φ_4 : matrice aléatoire obtenue par une loi normale $N(0, \frac{1}{M})$

La cohérence mutuelle est une mesure statistique utilisée pour quantifier la corrélation entre deux variables aléatoires ou séries temporelles. Elle évalue la similarité des variations entre ces deux ensembles de données. Une cohérence mutuelle élevée indique une forte corrélation entre les deux ensembles de données, tandis qu'une cohérence mutuelle faible ou nulle suggère une faible corrélation. Ici, notre objectif est de trouver la matrice de mesure pour laquelle sa cohérence mutuelle avec notre dictionnaire appris par k-SVD est la plus faible. Ainsi, nous assurons une dispersion maximum des données.
La cohérence mutuelle se calcule de la manière suivante :

$$\mu(\Phi, D) = \sqrt{N} \max_{i,j} \frac{|<\Phi_{i,.}, D_{.,j}>|}{||\Phi_{i,.}|| \times ||D_{.,j}||}$$

Après avoir calculé les cohérences mutuelles pour chacune des mesures $M = [15\%, 20\%, 25\%, 30\%, 50\%]$. Nous obtenons des résultats respectant l'intervalle suivant :

$$1 \leq \mu(\Phi, D) \leq \sqrt{N}$$

Voici les résultats obtenus pour les différentes matrices de mesure :

Pourcentage	Φ_1	Φ_2	Φ_3	Φ_4
15%	8.256	6.522	5.672	2.609
20%	8.307	6.587	5.809	2.952
25%	8.348	6.533	6.059	2.842
30%	8.326	7.358	6.071	3.481
50%	8.502	6.665	6.144	2.769

Nous avons retenu la matrice de mesure ayant la plus petite cohérence mutuelle pour éviter au maximum la redondance. Ici, c'est Φ_4, la matrice générée par la loi Normale, qui a la plus petite cohérence avec une mesure de 25% et qu'il faut conserver.

Question 2

Tout d'abord, explicitons la procédure de Compressive Sensing :

1. Apprentissage du dictionnaire par k-SVD.
2. Détermination de l'algorithme de codage parcimonieux à utiliser.
3. Détermination de la matrice de mesure Φ.
4. Calcul de $y = \Phi x$.
5. Reconstruction du signal avec $A = \Phi D$ pour obtenir $x = D\alpha$.

Dans les parties précédentes, l'algorithme de codage parcimonieux sélectionné est CoSaMP et la matrice de mesure sélectionnée est Φ_4 avec une mesure de 25%.

Cependant, lors de l'application de la procédure de Compressive Sensing, nous n'avons pas réussi à appliquer l'algorithme CosaMP aux signaux. Nous avons donc décidé d'appliquer un autre algorithme *omp-compressed* (le code de cet algorithme se trouve dans le fichier *omp.py*) pour pouvoir reconstruire les signaux.

Après reconstruction des signaux, nous avons obtenu une MSE (Mean Square Error) de 0 pour chacun de nos signaux.

Signal 1	0.0000
Signal 2	0.0000
Signal 3	0.0000

Table 1: Valeurs des signaux

Cela signifie qu'il y a un problème lors de la reconstruction des signaux car chaque MSE vaut 0. Il se peut que nous aurions trouvé des résultats différents si nous avions réussi à appliquer l'algorithme choisi originellement (CosaMP).

6. Conclusion

Ce projet nous a permis de mettre en application les concepts et les applications pratiques de Compressive Sensing vus en cours.

Notre étude a montré que la cohérence mutuelle entre le dictionnaire k-SVD et la matrice de mesure est un facteur déterminant pour assurer une bonne reconstruction des signaux.

De plus, cela a aussi été l'occasion de développer nos compétences en Python et en gestion de projets.