Projet de Compressed Sensing

L. Jacques, D. K. Hammond, M. J. Fadili (2009)
Dequantizing Compressed Sensing: When Oversampling and Non-Gaussian Constraints Combine

DEBRAY Alexandre
ELBAZ Nathan
LECOCQ Thomas

Mars 2016

Contents

Introduction

En Compressed Sensing, on cherche à reconstruire un vecteur s-sparse x de taille N à partir d'un certain nombre m de mesures linéaires $< \phi_i, x >$ de ce vecteur. Si l'on pose ϕ la matrice de taille mxN dont les vecteurs ϕ_i sont les lignes, alors il s'agit de reconstruire x à partir de $y = \phi x$. On utilise en général le Basis Pursuit (BP), c'est-à-dire que l'on résout le programme suivant :

$$x^* = \arg\min ||u||_1$$

sous contrainte :

$$\phi u = y$$

Ce programme est en fait la relaxation convexe du programme plus naturel où la norme 1 est remplacée par la norme 0.

Ici, on se place dans la situation où les mesures sont bruitées, ce qui a pour conséquence essentielle, le fait qu'il est alors illusoire de chercher des conditions sur ϕ pour que BP (ou tout autre programme) puisse reconstruire exactement tous les signaux s-sparses. Par contre, on pourra rechercher des conditions similaires pour que le signal reconstruit x^* soit proche en norme 2 du signal initial x.

Ensuite, le programme dans le cas bruité ne peut plus être résolu par programmation linéaire. Il faudra donc utiliser des méthodes plus générales d'optimisation convexe.

Nous allons dans un premier temps présenter les évolutions du Basis Pursuit nécessaires à traiter des mesures bruitées (Basis Pursuit DeNoiser) et nous verront que, dans le cas où le "bruit" considéré est issu de l'imprécision d'un appareil de mesure, on est amené à résoudre des programmes encore différents (Basis Pursuit DeQuantizer). On verra que, sous des conditions similaires à celles qui per mettaient la reconstruction exacte de signaux non-bruités avec Basis Pursuit, on peut obtenir des erreurs de reconstruction réduite sur les signaux bruités. Dans un second temps, on présentera les méthodes de résolution numérique de ces programmes.

Enfin, ces programmes ont été mis en oeuvre sur des données simulées, expérience que l'on pourra trouver dans le notebook IPython joint à ce rapport, avec des commentaires sur les résultats.

1 Le Programme Basis Pursuit DeNoiser

On se place donc dans la situation où les mesures sont bruitées, autrement dit :

$$y = \phi x + \kappa$$

avec $\kappa \in \mathbb{R}^m$ une réalisation d'une variable aléatoire centrée. On fait une hypothèse sur les bruits, et en général une idée naturelle est de supposer que $||\kappa||_2 \leq \epsilon$, évènement qui possède

une grande probabilité de survenance si ϵ est suffisamment grand. On résout alors le programme suivant, une variante du Basis Pursuit appelée Basis Pursuit DeNoiser ($BPDN$).

$$x^* = \arg\min ||u||_1$$

sous contrainte :

$$||y - \phi u||_2 \leq \epsilon$$

Un choix raisonnable de ϵ, qui assure $||\kappa||_2 \leq \epsilon$ avec une grande probabilité (c'est-à-dire la "faisabilité" de la reconstruction de x par BPDN) est nécessaire, mais on peut intuiter que de trop grandes valeurs de ϵ conduiront à une erreur trop importante sur x puisque l'on tiendrait alors de moins en moins compte des mesures ϕx à notre disposition (autrement dit, le décodeur ne serait pas "consistant").

Contrairement au Basis Pursuit, les données sont bruitées, il est donc vain de chercher des conditions sur ϕ pour que la reconstruction soit exacte pour tous les vecteurs s-sparses. Par contre, on peut vouloir obtenir des conditions pour que le vecteur reconstruit x^* soit suffisamment proche de x en norme 2 pour tous les vecteurs s-sparses. Comme pour Basis Pursuit, cela va nécessiter que la matrice ϕ vérifie une condition du type Restricted Isometry Property (RIP) pour tous les vecteurs $2s$-sparses :

$$(1-\delta)^{\frac{1}{2}}||x||_2 \leq \frac{||\phi x||_2}{\mu} \leq (1+\delta)^{\frac{1}{2}}||x||_2$$

quelque soit x $2s$-sparse, avec $\delta \in]0, \sqrt{2}-1[$ (le rayon) et $\mu > 0$ des constantes.

Si ϕ vérifie cette propriété, on a alors pour tous les vecteurs x s-sparses :

$$||x^* - x||_2 \leq C\frac{\epsilon}{\mu}$$

avec C une constante qui dépend de ϕ et de s.

On retrouve l'idée, évoquée plus haut, qu'il vaut mieux choisir ϵ petit pour réduire l'erreur effectuée (mais on est contraint par l'hypothèse de faisabilité : $||\kappa||_2 \leq \epsilon$, en tout cas avec une grande probabilité).

2 Le programme Basis Pursuit DeQuantizer

Un cas intéressant de "bruitage" des mesures est le cas où la mesure ϕx se fait avec précision $\alpha > 0$ finie, autrement dit, pour chaque composante de ϕx, on récupère l'élément de $\frac{\alpha}{2} + \alpha\mathbb{Z}$ le plus proche. Formellement, on observe $\alpha E(\frac{\phi x}{\alpha}) + \frac{\alpha}{2}$ au lieu de ϕx, E désignant la partie entière (coordonnée par coordonnée). On parlera plus volontiers ici de mesures "quantifiées".

En terme d'information, dans le cas de mesures quantifiées, on est face à une situation intéressante : si on dispose bien de m mesures, celles-ci ne sont pas de précision infinie, et ne

peuvent prendre qu'un certain nombre de valeurs discrètes. On pressent donc qu'il va falloir augmenter m pour compenser.

Cependant, tout se passe alors, au vu de l'information disponible, comme si on avait bien affaire à un bruit $\kappa = \left(\alpha E(\frac{\phi x}{\alpha}) + \frac{\alpha}{2}\right) - \phi x \in [-\frac{\alpha}{2}, +\frac{\alpha}{2}]^m$ suivaient des uniformes sur $[-\frac{\alpha}{2}, +\frac{\alpha}{2}]$.

Supposons donc maintenant que les composantes de κ suivent des uniformes sur l'intervalle $[-\frac{\alpha}{2}, +\frac{\alpha}{2}]$, alors quelque soit $i \in \{1, ..., n\}$, $|\kappa_i| \leq \frac{\alpha}{2}$. La bonne condition à imposer pour la reconstruction est alors $||\kappa||_\infty \leq \epsilon$, toutes les valeurs de κ la vérifiant étant équiprobables.

$$\arg\min ||u||_1$$

sous contrainte :

$$||y - \phi u||_\infty \leq \epsilon$$

Ce programme est cependant difficile à résoudre, et on ne possède pas de résultats précis sur ses performances de reconstruction.

On peut utiliser $BPDN$ pour traiter de tels bruits, par exemple en choisissant grossièrement $\epsilon \geq \frac{\alpha}{2}\sqrt{m}$ pour assurer la faisabilité avec probabilité 1 (ou, plus finement, en étudiant la loi de $||\kappa||_2^2$ afin de choisir un ϵ plus petit assurant tout de même une grande probabilité de faisabilité). Cependant, la boule unité de la norme 2 a une forme très différente de celle de la norme ∞, ce qui ne permet pas une grande marge de manoeuvre pour réduire ϵ tout en conservant une grande probabilité de faisabilité. On peut alors chercher à utiliser des contraintes en norme p pour $p > 2$. On résout alors le programme suivant, le Basis Pursuit DeQuantizer d'ordre p ($BPDP_p$) :

$$x^* = \arg\min ||u||_1$$

sous contrainte :

$$||y - \phi u||_p \leq \epsilon$$

où la norme ∞ est remplacée par la norme p. La norme p d'un vecteur converge, lorsque $p \to \infty$, vers la norme ∞ de ce vecteur, ce qui constitue l'intuition de l'utilisation de ce programme (avec p suffisamment grand) dans le cas où κ suit une uniforme. On retrouve le $BPDN$ pour $p = 2$.

Pour obtenir des résultats sur la qualité de la reconstruction en norme 2, on va une fois encore imposer une condition sur ϕ, la Restricted Isometry Property, d'ordre p cette fois, ou RIP_p, et ce pour tous les vecteurs $3s$-sparses (au lieu de $2s$-sparses précédemment) :

$$(1-\delta)^{\frac{1}{2}}||x||_2 \leq \frac{||\phi x||_p}{\mu} \leq (1+\delta)^{\frac{1}{2}}||x||_2$$

quelque soit x $3s$-sparse, avec $\delta \in]0, \sqrt{2} - 1[$ (le rayon) et $\mu > 0$ des constantes.

Si ϕ vérifie cette propriété, on a alors pour tous les vecteurs x s-sparses :

$$||x^* - x||_2 \leq C\frac{\epsilon}{\mu}$$

avec C une constante qui dépend de ϕ et de s.

Cette inégalité n'est pas valable pour $p = +\infty$, d'où le recours aux programmes pour p fini afin de traiter les bruits uniformes.

$$||x^* - x||_2 < B\frac{\epsilon}{\mu}$$

3 Considérations pratiques

3.1 Choisir ϵ

Si on choisit ϵ trop petit, par exemple $\epsilon < \frac{\alpha}{2}$, alors comme la boule unité de la norme p est entièrement contenue dans la boule unité de la norme ∞ et la faisabilité est assurée avec une probabilité médiocre, même si augmenter p améliore les choses. Si on choisit ϵ trop grand, par exemple $\epsilon > \sqrt[p]{m}\frac{\alpha}{2}$, alors on s'assure de la faisabilité du programme avec probabilité 1, mais l'erreur effectuée en norme 2 sera grande. Pour choisir plus finement ϵ, on peut étudier la loi de $||\kappa||_p^p$ pour obtenir le résultat suivant :

$$\mathbb{P}(||\kappa||_p^p \leq \zeta_p + \eta\sqrt{m}\frac{\alpha^p}{2^p}) \geq 1 - \exp(-2\eta^2)$$

avec $\zeta_p = \mathbb{E}(||\kappa||_p^p) = m\frac{\alpha^p}{2^p(p+1)}$

Pour assurer la faisabilité $||\kappa||_p^p \leq \epsilon^p$ avec une probabilité plus grande que $1 - \exp(-2\eta^2)$, on peut donc choisir :

$$\epsilon = \frac{\alpha}{2(p+1)^{\frac{1}{p}}}\left(m + \eta + \sqrt{m}(p+1)\right)^{\frac{1}{p}}$$

Par ailleurs, un tel choix de ϵ permet bien de se rapprocher de la contrainte limite voulue en norme ∞ car $\left(\zeta_p + \eta\sqrt{m}\frac{\alpha^p}{2^p}\right)^{\frac{1}{p}} \to \frac{\alpha}{2}$ lorsque $p \to +\infty$.

3.2 Construire ϕ vérifiant RIP_p

Comme dans le cas du Basis Pursuit, on peut construire des matrices aléatoires qui seront RIP_p avec une grande probabilité pour peu qu'elles aient suffisamment de lignes, c'est-à-dire que m soit grand. En particulier, les matrices aléatoires gaussiennes, i.e. les matrices sont les coefficients sont générés indépendamment selon des normales centrées réduites, vérifient la propriété suivante pour $p \geq 2$ et $m \geq 2^{p+1}(p-1)$:

Il existe une constante $c > 0$ telle que si :

$$m \geq \frac{c}{\delta^p}\left(s \log\left(\frac{eN}{s}\left(1+\frac{12}{\delta}\right)\right)+\eta\right)^{\frac{p}{2}}$$

alors, avec une probabilité plus grande que $1-2\exp(-\eta)$, une matrice gaussienne aléatoire de m lignes sera RIP_p de rayon δ pour les vecteurs s-sparses. Par ailleurs, la constante μ sera l'espérance de la norme p d'un vecteur aléatoire gaussien de taille m (pour $p = 2$, on a $\mu = \sqrt{m}$).

En ordre de grandeur, cela correspond à générer une matrice aléatoire gaussienne avec un nombre de ligne m de l'ordre de :

$$m \approx\geq \left(s \log\left(\frac{eN}{s}\right)\right)^{\frac{p}{2}}$$

Pour $p = 2$, on retrouve $m \approx\geq s \log\left(\frac{eN}{s}\right)$.

3.3 Choisir p

On voudrait choisir p le plus grand possible, pour que la norme p soit proche de la norme ∞. Cependant, choisir p trop grand impose des conditions plus restrictives sur m pour que la matrice vérifie RIP_p. Comme on l'a vu, à s et N constants, m doit alors croître exponentiellement avec p, ce qui est coûteux.

4 Résolution numérique du programme

La résolution numérique du programme dit Basis Pursuit DeQuantizer requiert plusieurs étapes clefs dont trois méthodes de calculs itératives. Il convient de rappeler d'abord que ce programme est une généralisation de la classe de problèmes suivants :

$$\arg\min_{x \in H} f_1(x) + f_2(x) \tag{P}$$

avec

$$f_1(x) = ||x||_1,$$

$$f_2(x) = i_{T^p(\epsilon)}(x) \text{ (= 0 si x } \in T^p(\epsilon), +\infty \text{ sinon)}$$

et

$$T^p(\epsilon) = \left\{x \in \mathbb{R}^{\mathbb{N}} : ||y_q - \Phi x|| \leq \epsilon\right\}$$

On peut montrer que les solutions de (P) sont charactérisés par :

$$\text{x est solution de (P)} \Leftrightarrow x = (\mathbb{1}_H + \beta\partial(f_1 + f_2))^{-1}, \beta > 0$$

L'opérateur $J_{\beta\partial f} = (\mathbb{1}_H + \beta\partial(f))^{-1}$ n'admet généralement pas de solution de forme fermée, il faut donc calculer séparément $J_{\beta\partial f_1}$ et $J_{\beta\partial f_2}$. Dans le cadre de $BPDQ_p$, f_1 etf_2 sont toutes deux non différenciables, l'auteur a donc choisi d'utiliser la méthode de séparation de Douglas Rachford :

$$x^{(t+1)} = (1 - \alpha_t/2)x^{(t)} + \frac{\alpha_t}{2} S_\gamma^\circ \circ P_{T^p(\epsilon)}^\circ(x^{(t)}) \tag{1}$$

avec $A^\circ = 2A - \mathbb{1}$, $S_\gamma = prox_{\gamma f_1}$, l'opérateur de seuil mou (soft-tresholding) et $P_{T^p(\epsilon)} = prox_{f_2}$, la projection orthogonale sur le tube $T^p(\epsilon)$. La séquence converge vers x^* et $P_{T^p(\epsilon)}(x^*)$ est solution de $BPDQ_p$.
Voyons maintenant comment calculer $P_{T^p(\epsilon)}(x^*)$ efficacement. Il faut calculer $prox_{f_2}$ à chaque étape de l'équation (1), et, pour cela, on utilise une méthode itérative.
Si on note B^p la boule unité au sens de l^p, on peut remarquer que :

$$f_2(x) = i_{T^p(\epsilon)}(x) = (i_{B^p} \circ A_\epsilon)(x)$$

avec

$$A_\epsilon(x) = (\Phi' x - y)/\epsilon$$

En utilisant le Lemme 4 de l'auteur, calculer $prox_{f_2}$ est réduit à appliquer la projection $prox_{i_{B^p}} = P_{B^p}$ en fixant $f = i_{B^p}$, $\Phi' = \Phi/\epsilon$ et $y = y_q/\epsilon$ dans le système de la méthode de calcul itérative suivante :

$$u^{(t+1)} = \beta_t(\mathbb{1} - prox_{\beta_t^{-1} f})(\beta_t^{-1} u^{(t)} + A(p^{(t)}))$$

$$p^{(t+1)} = x - \Phi'^* u^{(t+1)}$$

Malheureusement, aucune forme fermée du projecteur P_{B^p} n'est connue pour $2 < p < \infty$. A la place, on utilise une dernière méthode itérative.
On note:

$$f_y(u) = \frac{||u - y||_2^2}{2} \text{ et } g(u) = ||u||_p^p.$$

Pour $||y||_p \leq 1$, $P_{B^p}(y) = y$, et si $||y||_p > 1$, $P_{B^p}(y)$ est la solution du problème de minimisation:

$$u^* = \arg\min_u f_y(u)$$

sous la contrainte

$$g(u) = 1.$$

Celui-ci peut être résolu en appliquant la méthode de Newton au système de Karush-Kuhn-Tucker correspondant à notre problème de minimisation. Pour cela, il faut partir d'un point initial z^0 et calculer les $z^{n+1} = z^n - V(z^n)^{-1}F(z^n)$ successifs jusqu'à ce que $||F(z^n)||_2$ tombe en dessous d'une certaine valeur.

Conclusion

Les algorithmes présentés dans la partie précedente requièrent un temps de calcul très long: plusieurs heures sont à prévoir pour une exécution de 500 itérations comme le conseille l'article (On estime que la durée peut atteindre les 6-7h avec une paramétrisation peu exigeantes). Nous avons néanmoins réussi à mettre en place une implementation fonctionelle de l'algoritme. Cependant, celui-ci possède un temps d'execution très long et un grand nombre de paramètres dont parfois l'article fait à peine référence, nous n'avons donc pas pu trouver la parametrisation optimale. Tout le détail sur notre approche algorithmique est expliqué dans le notebook joint à notre mémoire.