

Introduction au Compressed sensing.

Notions de complexité algorithmique et relaxation convexe¹

Guillaume Lécué²

Résumé

Dans ces notes de cours nous introduisons la procédure de minimisation ℓ_0 pour laquelle on montre son optimalité d'un point de vue théorique. Nous montrons aussi qu'elle n'est pas utilisable en pratique car c'est une procédure NP-hard en général.

Puis nous introduisons la procédure du Basis Pursuit par relaxation convexe. Nous montrons qu'elle peut s'écrire comme un problème de programmation linéaire. Finalement, nous montrons que si le Basis Pursuit résout le problème de CS pour l'ensemble des vecteurs s -sparse alors nécessairement le nombre de mesures doit être plus grand que $s \log(eN/s)$.

1 Introduction

Le problème de CS s'énonce de la manière suivante : retrouver un vecteur $x \in \mathbb{R}^N$ à partir des mesures $y (= Ax)$ et de la matrice de vecteurs mesure $A \in \mathbb{R}^{m \times N}$ sachant que x a un support de petite taille.

On souhaite donc résoudre un système de la forme donné dans la Figure 1 où il y a beaucoup plus d'inconnues (ici N) que d'équations (ici m).

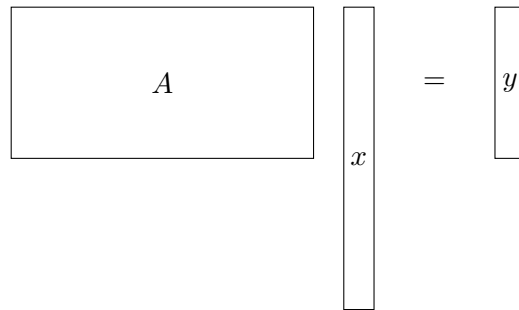

$$\boxed{A} \boxed{x} = \boxed{y}$$

FIGURE 1 – Trouver x à partir de $y = Ax$ et $A \in \mathbb{R}^{m \times N}$ où $m < N$ mais sachant que x est sparse.

L'hypothèse centrale en CS est que le signal x à reconstruire est **parcimonieux**. Une idée naturelle est donc de rechercher l'élément dans l'espace affine des solutions au système d'équations linéaires $Ax = y$:

$$\{t \in \mathbb{R}^N : At = y\} = x + \text{Ker}(A)$$

qui est le plus parcimonieux, c'est-à-dire celui ayant le support de plus petite taille. Cette procédure est appelée la procédure de minimisation ℓ_0 qui est introduite maintenant.

1. Notes du cours du 2 février 2016

2. CNRS, CREST, ENSAE. Bureau E31, 3 avenue Pierre Larousse. 92 245 Malakof. Email : guillaume.lecue@ensae.fr.

Définition 1.1 (Procédure de minimisation ℓ_0). La **procédure de minimisation** ℓ_0 est la fonction qui à tout $y \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times N}$ renvoie

$$\hat{x}_0 \in \underset{t \in \mathbb{R}^N : At=y}{\operatorname{argmin}} \|t\|_0 \quad (P_0)$$

si l'équation $Ax = y$ admet une solution et \emptyset sinon. On rappelle que $\|t\|_0 = |\operatorname{supp}(t)|$ est la taille du support de t .

Le système linéaire présenté dans la Figure 1 admet tout l'espace affine $x + \operatorname{Ker}(A)$ pour ensemble de solutions. Comme $m < N$, A a un noyau de taille au moins $N - m \geq 1$. Il y a donc une infinité de solutions à ce système d'équations.

D'un autre côté, on dispose d'une information supplémentaire : le vecteur x qu'on cherche à retrouver a un support de petite taille. La procédure de minimisation ℓ_0 est donc construite pour aller chercher dans l'espace des solutions du système de la Figure 1, celle ayant le plus petit support.

L'espoir étant que le noyau de A sera "bien positionné" de telle sorte que l'espace des solutions $x + \operatorname{Ker}(A)$ intersecte l'ensemble des vecteurs s -sparse Σ_s où $s = \|x\|_0$ uniquement en x :

$$(x + \operatorname{Ker}(A)) \cap \Sigma_s = \{x\} \text{ pour } s = \|x\|_0$$

où on note

$$\Sigma_s = \{x \in \mathbb{R}^N : \|x\|_0 \leq s\}.$$

La proposition suivante montre que cette condition est nécessaire et suffisante pour que la procédure de minimisation ℓ_0 reconstruise exactement le vecteur x .

Proposition 1.2. Soit $x \in \mathbb{R}^N$. On note $s = \|x\|_0$. On a équivalence entre les deux assertions suivantes :

1. x est l'unique solution du problème de minimisation ℓ_0 , c'est-à-dire $\hat{x}_0 = x$
2. $(x + \operatorname{Ker}(A)) \cap \Sigma_s = \{x\}$.

Démonstration. Si x est l'unique solution du problème de minimisation ℓ_0 alors pour tout $t \neq x$ dans l'espace des solutions $(x + \operatorname{Ker}(A))$, on a $\|t\|_0 > s$ donc $(x + \operatorname{Ker}(A))$ intersecte Σ_s seulement en x .

Réciproquement, si $(x + \operatorname{Ker}(A))$ intersecte Σ_s seulement en x alors tout vecteur $t \neq x$ dans $(x + \operatorname{Ker}(A))$ on aura $\|t\|_0 > s$ alors le vecteur ayant le plus petit support dans $(x + \operatorname{Ker}(A))$ est x et il est le seul à avoir cette propriété. Donc l'algorithme de minimisation ℓ_0 admet une unique solution qui est x , c'àd $\hat{x}_0 = x$. ■

Proposition 1.2 est le premier résultat qu'on rencontre où le rôle de "l'orientation" du noyau $\operatorname{Ker}(A)$ vis-à-vis de l'ensemble des vecteurs s -sparse est mis en avant.

L'idée derrière la Proposition 1.2 – qui sera présente dans l'ensemble des résultats de reconstruction de ce cours – est que l'espace $\operatorname{Ker}(A)$ doit être "éloigné" de l'ensemble Σ_s . Géométriquement, on souhaite que $\operatorname{Ker}(A)$ n'intersecte Σ_s que en 0 et qu'il soit "proche d'éléments diagonaux de \mathbb{R}^N " (c'àd des éléments les "moins sparse" de \mathbb{R}^N).

Proposition 1.2 n'est par contre par un cas typique de résultat en CS. En effet, Proposition 1.2 donne une condition nécessaire et suffisante pour la reconstruction exacte par une procédure (ici la minimisation ℓ_0) d'un seul vecteur x . Dans la suite du cours, on ne s'intéressera qu'à des propriétés de la matrice de mesure qui permettent de reconstruire tous les vecteurs de Σ_s .

2 Nombre minimal de mesures

On s'intéresse ici à la question suivante : étant donné $A \in \mathbb{R}^{m \times N}$, le système de la Figure 1 est hautement sous-déterminé car $m < N$ (et même $m \ll N$), néanmoins, sachant qu'on cherche une solution x qui est s -sparse, quel est le nombre minimal de mesures m (càd de lignes de A) nécessaires pour qu'on puisse reconstruire tout vecteur de Σ_s à partir seulement de m mesures ?

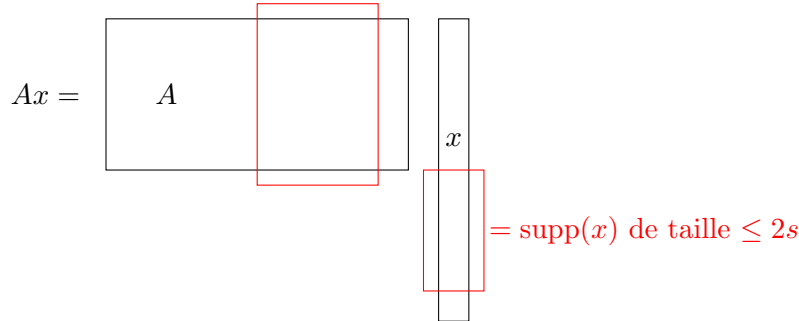
On insiste sur le fait qu'on souhaite pouvoir reconstruire tout vecteur dans Σ_s (et pas seulement un seul vecteur comme dans la Proposition 1.2). Il est en particulier nécessaire d'avoir $Ax \neq Ay$ pour tout $x, y \in \Sigma_s$ si on veut avoir une chance de retrouver x (ou y) à partir de Ax (ou Ay). Cette condition nécessaire a un coût en terme de nombre de mesures et "d'orientation" de $\text{Ker}(A)$.

Proposition 2.1. *Les assertions suivantes sont équivalentes :*

1. pour tout $x, y \in \Sigma_s$, si $x \neq y$ alors $Ax \neq Ay$
2. $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$ (et donc $m \geq 2s$)
3. toutes les sous-matrices de A de taille $m \times (2s)$ sont injectives (et donc $m \geq 2s$).

Démonstration. Si pour tout $x, y \in \Sigma_s$, $Ax \neq Ay$ quand $x \neq y$ alors $A(x - y) \neq 0$ et donc le seul vecteur $2s$ -sparse dans $\text{Ker}(A)$ est le vecteur nul (car tout vecteur $2s$ -sparse peut s'écrire comme la différence de deux vecteurs s -sparse). En d'autres termes $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$. Cette condition implique bien que $m \geq 2s$ car si on extrait de A la matrice de taille $m \times (2s)$ faite des $2s$ premières colonnes de A , on voit qu'elle est injective, en particulier, son nombre de lignes doit être plus grand que son nombre de colonnes, càd $m \geq 2s$.

On suppose maintenant que $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$. Soit $J \subset \{1, \dots, N\}$ tel que $|J| = 2s$. Soit $x \in \mathbb{R}^N$ non nul et supporté dans J . En particulier, x est $2s$ -sparse. Alors par hypothèse $Ax \neq 0$ càd $A_{\cdot J} x_J \neq 0$ (cf. Figure 2) et donc pour tout $y \in \mathbb{R}^J$ non nul, $A_{\cdot J} y \neq 0$. En d'autres termes, $A_{\cdot J}$ est injective. Ceci étant vrai pour tout $J \subset \{1, \dots, N\}$ tel que $|J| = 2s$, on obtient bien que toutes les sous-matrices extraites $m \times (2s)$ de A sont injectives.



Finalement, on suppose que toutes les matrices $m \times (2s)$ extraites de A sont injectives. Soit $x, y \in \Sigma_s$ tels que $x \neq y$. On note $J_x = \text{supp}(x)$ et $J_y = \text{supp}(y)$ alors $|J_x \cup J_y| \leq 2s$. En particulier, $A_{J_x \cup J_y}$ est injective et comme $x - y \neq 0$ et $\text{supp}(x - y) \subset J_x \cup J_y$, on a $A(x - y) = A_{J_x \cup J_y}(x - y) \neq 0$. On a donc bien $Ax \neq Ay$. ■

Une conclusion de Proposition 2.1 est que si on veut avoir une chance de pouvoir reconstruire exactement les vecteurs de Σ_s alors il est nécessaire d'avoir au moins $2s$ mesures. On montre maintenant que cette condition est suffisante pour l'algorithme de minimisation ℓ_0 si le noyau de la matrice A est bien positionné.

Proposition 2.2. *Soit $A \in \mathbb{R}^{m \times N}$. Il y a équivalence entre les deux assertions suivantes :*

1. $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$
2. la procédure de minimisation ℓ_0 permet de reconstruire exactement tout vecteur de Σ_s :

$$\underset{t \in \mathbb{R}^N : At = Ax}{\text{argmin}} \|t\|_0 = \{x\} \text{ pour tout } x \in \Sigma_s.$$

Démonstration. Soit $x \in \Sigma_s$. Soit $t \in \mathbb{R}^N$ tel que $At = Ax$. On a $A(x-t) = 0$ or $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$ donc soit $x = t$ soit $|\text{supp}(x-t)| > 2s$. Comme $|\text{supp}(x)| \leq s$, la condition $|\text{supp}(x-t)| > 2s$ implique forcément que $|\text{supp}(t)| \geq s+1$ càd $\|t\|_0 > \|x\|_0$. Donc soit $x = t$ ou $\|t\|_0 > \|x\|_0$, ce qui implique que la vecteur ayant le plus petit support dans l'espace des solutions $x + \text{Ker}(A)$ est x .

Pour la réciproque, on prend $x \in \text{Ker}(A) \cap \Sigma_{2s}$. On raisonne par contra-positée en supposant que $x \neq 0$. Soit $I_1 \cup I_2 = \text{supp}(x)$ une décomposition du support de x telle que $|I_1| = |I_2| \leq s$ (I_1 et I_2 ne sont pas nécessairement disjoints). On a pour $u = x_{I_1-I_2} + (1/2)x_{I_1 \cap I_2}$ et $v = x_{I_2-I_1} + (1/2)x_{I_2 \cap I_1}$, $u + v = x$ alors $Au = A(-v)$ et $\|u\|_0 = \|v\|_0$. Donc u ne peut pas être l'unique solution de $\underset{t \in \mathbb{R}^N : At = Au}{\text{argmin}} \|t\|_0$ (si u est solution de ce problème alors $-v$ l'est aussi). ■

On voit donc que la condition $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$ est centrale pour la faisabilité du problème (cf. Proposition 2.1) et sa réalisation par l'algorithme de minimisation ℓ_0 (cf. Proposition 2.2). La question qui vient naturellement à l'esprit concerne l'existence de telles matrices : est-il possible de construire des matrices $A \in \mathbb{R}^{m \times N}$ telles que $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$? Par ailleurs, on souhaite devoir prendre le moins de mesures possibles. On aimerait donc pouvoir construire de telles matrices pour un nombre minimal de mesures m . On sait déjà qu'on ne peut pas avoir cette propriété pour $m < 2s$ (cf. Proposition 2.1). On voit dans le résultat qui suit que ce nombre est en fait suffisant.

Proposition 2.3. *Pour tout $N \geq 2s$, il existe une matrice $A \in \mathbb{R}^{m \times N}$ vérifiant :*

1. $m = 2s$
2. $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$.

Démonstration. On utilise les matrices de Vandermonde pour cette construction. Soient $t_N > \dots > t_1 > 0$. On pose

$$A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_N \\ \vdots & \vdots & \vdots & \vdots \\ t_1^{2s-1} & t_2^{2s-1} & \dots & t_N^{2s-1} \end{pmatrix}.$$

Soit $J \subset \{1, \dots, N\}$ de taille $|J| = 2s$. On note $J = \{j_1, \dots, j_{2s}\}$. Pour démontrer que la matrice extraite $A_{\cdot J}$ est inversible il suffit de calculer son déterminant. On a

$$\det(A_{\cdot J}) = \begin{vmatrix} 1 & 1 & \dots & 1 \\ t_{j_1} & t_{j_2} & \dots & t_{j_N} \\ \vdots & \vdots & \vdots & \vdots \\ t_{j_1}^{2s-1} & t_{j_2}^{2s-1} & \dots & t_{j_N}^{2s-1} \end{vmatrix} = \prod_{k < l} (t_{j_k} - t_{j_l}) > 0.$$

La matrice $A_{\cdot J}$ est donc bien inversible. Ceci étant vrai pour tout $|J| = 2s$, on en déduit grâce à Proposition 2.1 que $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$. ■

Proposition 2.3 nous renseigne sur le choix des mesures qu'on doit effectuer pour pouvoir résoudre le problème de CS en un nombre minimal d'observations. Si A est une matrice de Vandermonde (cf. preuve de Proposition 2.3) de taille $(2s) \times N$ alors l'algorithme de minimisation ℓ_0 permet de reconstruire tous les vecteurs de Σ_s à partir de $2s$ mesures si la matrice de mesures est A .

***Conclusion** : $m = 2s$ est le nombre nécessaire et suffisant pour pouvoir reconstruire exactement tout vecteur de Σ_s à partir de m mesures linéaires. La procédure de minimisation ℓ_0 fournit dans ce cas une procédure de reconstruction exacte quand la matrice de mesure $A \in \mathbb{R}^{m \times N}$ vérifie $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$. De telles matrices existent pour $m = 2s$.*

La démarche suivie dans cette section est typique en CS :

1. on se donne une procédure (ici la méthode par minimisation ℓ_0)
2. on identifie une ou des propriétés de la matrice de mesure $A \in \mathbb{R}^{m \times N}$ qui assurent que la procédure permet la reconstruction exacte des vecteurs de Σ_s
3. on construit des matrices A vérifiant cette ou ces propriétés qui ne nécessitent qu'un nombre minimal de lignes – c-à-d un nombre minimal de mesures m .

On étudiera dans la suite d'autres procédures (en particulier, le Basis Pursuit) pour lesquels on identifiera des conditions sur A qui permettent à ces procédures de résoudre le problème de CS dans Σ_s (en particulier, la condition RIP). Ensuite, on construira des matrices vérifiant ces conditions pour un nombre minimal de mesures (les matrices aléatoires vérifieront RIP pour $m \sim s \log(N/s)$).

Au vue des résultats de cette section et de la conclusion qu'on en a tiré, il semble que le problème du CS (retrouver tout vecteur $x \in \Sigma_s$ à partir de Ax et A) semble résolu grâce à la méthode de minimisation ℓ_0 pour un nombre minimal de mesures $m = 2s$ et, par exemple, grâce à des vecteurs mesures construits à partir des matrices de Vandermonde de taille $(2s) \times N$.

Ce n'est cependant pas le cas, car la procédure de minimisation ℓ_0 ne peut pas être implémentée en pratique, en général. On a donc proposer une procédure mais celle-ci a seulement un intérêt théorique et n'est pas utilisée en pratique. C'est ce qu'on va étudier dans la section suivante.

3 La propriété NP-hard de la minimisation ℓ_0

Dans cette section, on s'intéresse à un des aspects le plus important du CS et, plus généralement, des statistiques modernes : l'aspect "computationnel" des procédures. On va donc s'intéresser à la mise en application et à la faisabilité des procédures.

Pour l'instant, nous avons seulement rencontré la procédure de minimisation ℓ_0 . Comme vue dans la section précédente, cette procédure résout de manière optimale le problème de CS mais est-elle utilisable en pratique ?

Remarque 3.1 (Procédures et algorithmes). *Dans l'ensemble du cours, on fait la distinction entre procédures et algorithmes. Une procédure est, d'une manière générale, une statistique, c'est-à-dire une fonction (mesurable) des observations. Cette notion ne contient aucun aspect informatique et ne se soucie pas de son utilisation pratique. Un algorithme est associé à une procédure, c'est une routine informatique (généralement décrite par du pseudo-code) qui donne une manière concrète d'implémenter (c-à-d coder) la procédure.*

Il existe deux sortes d'algorithmes : ceux qu'on peut utiliser en pratique, c-à-d qui ne demandent pas un temps trop important de calcul (ou une capacité de calcul trop grande) et ceux qu'on ne peut pas utiliser en pratique pour des limitations en terme de puissance / temps de calcul.

L'idéal pour un statisticien est de proposer une procédure qui a des propriétés d'optimalité d'un point de vue statistique / théorique (par exemple, d'être minimax) et pour laquelle, il existe un algorithme qui peut être utilisé en pratique.

Une manière algorithmique de résoudre le problème de minimisation ℓ_0 est de procéder de la manière qui suit.

```

Data:  $y \in \mathbb{R}^m, A \in \mathbb{R}^{m \times N}$ 
1 Output : solution  $\hat{x}_0$  pour  $(P_0)$ 
2 for  $s = 1, 2, 3, \dots$ , do
3   for  $J \subset \{1, \dots, N\}$  de taille  $|J| = s$  do
4     Résoudre le système linéaire  $A_{.J}x = y$  de taille  $m \times s$ .
5     if Il y a une solution then
6       enregistrer cette solution
7     sortir des boucles
8     else
9       continuer
10    end
11  end
12 end

```

Algorithm 1: Algorithme pour la résolution de la procédure de minimisation ℓ_0 .

Dans l'hypothèse de "fonctionnement" de la procédure de minimisation ℓ_0 , c'est-à-dire quand $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$, le premier ensemble J pour lequel il va y avoir une solution au système $A_{.J}x = y$ est $J = \text{supp}(x)$ où $x \in \Sigma_s$ est la solution la plus sparse dans $x + \text{Ker}(A)$, c'est donc l'unique solution du problème de minimisation ℓ_0 qu'on retrouvera en sortie de Algorithm 1.

Cet algorithme permet donc d'implémenter la procédure de minimisation ℓ_0 dans le cadre d'application de cette procédure, c'est-à-dire sous l'hypothèse que $\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$.

En quelques mots, Algorithm 1 procède de la manière suivante : étant donné un nombre s de colonnes, il fait une recherche parmi tous les ensembles $J \subset \{1, \dots, N\}$ de taille s du premier ensemble J tel que $A_{.J}x = y$ admet une solution – à s fixé, si un tel J existe alors Algorithm 1 résout le système $A_{.J}x = y$ et renvoie la solution. S'il n'y a pas de solution pour s colonnes alors l'algorithme itère sur s (c'est-à-dire $s \rightarrow s + 1$) et ceci jusqu'à trouver un support J tel que $A_{.J}x = y$ (un tel support existe vu que $y = Ax^*$ pour un certain x^*).

Le point faible de cet algorithme est l'étape de recherche parmi tous les ensembles J de taille s dans $\{1, \dots, N\}$. Il y a

$$\binom{N}{s}$$

choix possibles. Par exemple, pour $N = 1000$ et $s = 10$, on aura

$$\binom{N}{s} = \frac{N(N-1) \cdots (N-s+1)}{s(s-1) \cdots 1} \geq \left(\frac{N}{s}\right)^s = 10^{20}$$

choix possibles. Pour chaque ensemble $J \subset \{1, \dots, N\}$, l'algorithme 1 résout le système d'équations linéaires $A_{.J}x = y$. Si cela prend 10^{-10} secondes pour résoudre un tel système alors le temps de calcul pour résoudre l'étape $s = 10$ de l'algorithme sera de 10^{10} secondes, soit 300 ans.

Cette algorithme n'est donc pas exécutable en un temps raisonnable. Il n'est donc pas utilisé en pratique. Son point faible est cette étape de recherche de tous les sous-ensembles d'un cardinal donné dans un plus grand ensemble. Ce type d'exploration est de type recherche combinatoire qui est au coeur de beaucoup d'algorithmes qui s'effectuent en un temps exponentiel, c'est-à-dire pour

lesquels il faut un nombre exponentiel d'étapes en la dimension des entrées pour pouvoir être exécuté.

C'est le cas de l'algorithme 1, son exécution nécessite un nombre d'étapes de l'ordre de

$$\sum_{j=1}^s \binom{N}{j} \sim \left(\frac{N}{s}\right)^s$$

qui est exponentiel en s . Même pour des valeurs raisonnables de s et N (telles que $s \sim 10$ et $N \sim 1000$), ce nombre est déjà trop grand.

On peut maintenant se poser la question de l'efficacité de l'algorithme 1. Peut-être existe-il une manière plus intelligente d'implémenter la procédure de minimisation ℓ_0 que celle employée par l'algorithme 1 (qui utilise une recherche combinatoire) ?

C'est le résultat principale de cette section que de montrer que la méthode de minimisation ℓ_0 est fondamentalement difficile à implémenter (et donc inutilisable en pratique). L'algorithme 1 n'est donc pas si trivial en général (cf. Remarque 3.3 pour un cas particulier où la méthode de minimisation ℓ_0 peut être implémentée de manière plus rapide). Pour cela, on va devoir introduire (de manière informelle) quelques notions de complexité algorithmique.

Il existe plusieurs classes de problèmes de décisions :

1. \mathfrak{P} : ensemble de tous les problèmes pour lesquels il existe un algorithme exécutable en temps polynomiale.
2. \mathfrak{NP} : ensemble de tous les problèmes pour lesquels il existe un algorithme en temps polynomial certifiant une solution.
3. \mathfrak{NP} -hard : ensemble de tous les problèmes pour lesquels il existe un algorithme le solutionnant qui peut être transformé en temps polynomial en un autre algorithme solutionnant n'importe quel problème NP. Intuitivement, c'est l'ensemble de tous les problèmes qui sont aussi difficile à solutionner que n'importe quel problème NP.
4. \mathfrak{NP} -complet : ensemble de tous les problèmes qui sont dans la classe \mathfrak{NP} et \mathfrak{NP} -hard. C'est donc la classe de tous les problèmes NP qui sont aussi difficiles à résoudre que n'importe quel autre problème NP.

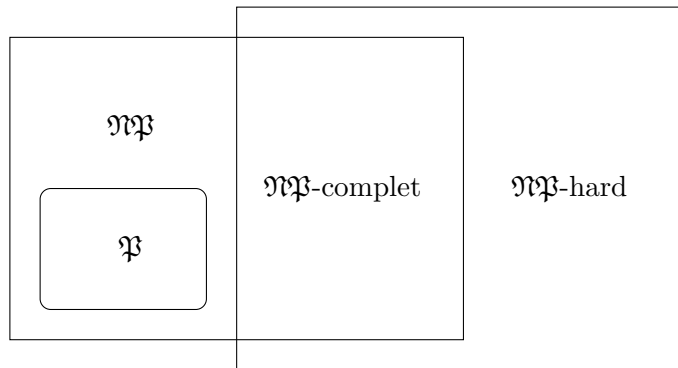


FIGURE 2 – Classes de problèmes en théorie de la décision

Il existe un catalogue de problèmes NP-complet. Ces problèmes sont particulièrement utiles car il "suffit" de trouver un algorithme solutionnant pour l'un d'entre eux pour solutionner tous les autres de la classe \mathfrak{NP} . Ils sont en quelques sorte considérés comme les problèmes NP les plus compliqués. En particulier, si on arrive à réduire (en temps polynomial) un problème donné en un

problème NP-complet alors ce problème est NP-hard. C'est la stratégie que nous allons utiliser pour montrer que la minimisation ℓ_0 est NP-hard.

On va donc réduire le problème de minimisation ℓ_0 en un problème NP-complet. Pour cela, on donne quelques exemples de problèmes NP-complet dont celui qui sera utilisé dans notre réduction du problème (P_0).

Probablement le problème NP-complet le plus connu est celui du **voyageur de commerce** : étant donné une liste de villes et leurs distances mutuelles, trouver le plus court chemin qui passe par toutes les villes et qui revient en son point de départ.

Un autre problème NP-complet assez connu est celui de la **recherche de clique maximale** : étant donné un graphe non orienté, trouver une clique (c'est-à-dire un sous-graphe où deux sommets quelconques sont adjacents) de taille maximale. D'autres problèmes NP-complet connus sont **la coloration d'un graphe, le remplissage d'un sac-à-dos, etc.**

Le problème NP-complet qui va nous intéresser ici est celui du **partitionnement par des ensembles de cardinal 3** : Soit $\{\mathcal{C}_i : i \in \{1, \dots, N\}\}$ une collection de sous-ensembles de $\{1, \dots, m\}$ de taille 3. Existe-t'il une partition de $\{1, \dots, m\}$ fait uniquement d'éléments de la collection $\{\mathcal{C}_i : i \in \{1, \dots, N\}\}$?

On admet que le problème du partitionnement par des ensembles de cardinal 3 est NP-complet. On va réduire le problème de minimisation ℓ_0 à ce problème. Ici "réduire" signifie transformer par une procédure en temps polynomiale un problème en un autre.

Théorème 3.2. *La procédure de minimisation ℓ_0 est NP-hard en général.*

Démonstration. On réduit le problème de minimisation ℓ_0 au problème de partitionnement par des ensembles de cardinal 3.

Soit $\{\mathcal{C}_i : i \in \{1, \dots, N\}\}$ une collection de sous-ensembles de $\{1, \dots, m\}$ de taille 3. On définit des vecteurs a_1, \dots, a_N de \mathbb{R}^m par

$$(a_i)_j = \begin{cases} 1 & \text{si } j \in \mathcal{C}_i \\ 0 & \text{sinon.} \end{cases}$$

Ainsi a_j est le vecteur de \mathbb{R}^m "indicatrice" de \mathcal{C}_j : $a_j = I_{\mathcal{C}_j}$ et A a pour colonnes les indicatrices des \mathcal{C}_j pour $j = 1, \dots, N$. On définit ensuite la matrice $A = [a_1 | \dots | a_N] \in \mathbb{R}^{m \times N}$ ayant pour colonnes les vecteurs $a_j, j = 1, \dots, N$. Finalement, on construit un vecteur $y = (1, \dots, 1)^\top \in \mathbb{R}^m$.

La construction de A et y peut se faire en un temps polynomial vu que $N \leq \binom{m}{3}$ (le cardinal de la collection $\{\mathcal{C}_i : i \in \{1, \dots, N\}\}$ est nécessairement plus petit que celui de tous les sous-ensembles de cardinal 3 dans $\{1, \dots, m\}$).

On commence par observer que si $x \in \mathbb{R}^N$ est tel que $Ax = y$ alors comme $Ax = \sum_{j=1}^N x_j a_j$ et $\|a_j\|_0 = 3$ pour tout $j = 1, \dots, N$, on a nécessairement que Ax a au plus 3 $\|x\|_0$ coordonnées non nulles et donc, comme $\|Ax\|_0 = \|y\|_0 = m$, on a $\|x\|_0 \geq m/3$.

Maintenant, on lance l'algorithme de minimisation ℓ_0 pour la matrice de mesure A et le vecteur des mesures y . On obtient un \hat{x}_0 de cardinal minimal parmi tous les $x \in \mathbb{R}^N$ tel que $Ax = y$. On a vu précédemment que nécessairement $\|\hat{x}_0\|_0 \geq m/3$, on explore deux cas :

1. si $\|\hat{x}_0\|_0 = m/3$ alors la collection $\hat{\mathcal{C}}_0 := \{\mathcal{C}_j : j \in \text{supp}(\hat{x}_0)\}$ forme une partition de $\{1, \dots, m\}$. En effet, il y a exactement $m/3$ ensembles (de cardinal 3) dans $\hat{\mathcal{C}}_0$. Donc, si deux ensembles de $\hat{\mathcal{C}}_0$ n'étaient pas disjoints alors nécessairement une des coordonnées de $A\hat{x}_0$ serait nulle. Hors ce n'est pas le cas. Donc $\hat{\mathcal{C}}_0$ est formé de $m/3$ ensembles disjoints de cardinal 3. Elle forme donc une partition de $\{1, \dots, m\}$.
2. si $\|\hat{x}_0\|_0 > m/3$ alors il n'y a pas de partition de $\{1, \dots, m\}$ dans la collection $\{\mathcal{C}_j : j \in \{1, \dots, N\}\}$. En effet, si une telle partition $\{\mathcal{C}_j : j \in J\}$ existait alors $z \in \mathbb{R}^N$ définit par

$z_j = 1$ si $j \in J$ et $z_j = 0$ si $j \notin J$ serait tel que $Az = y$ et $\|z\|_0 = m/3 < \|\hat{x}_0\|_0$ ce qui contredit la ℓ_0 -minimalité de \hat{x}_0 .

On peut donc réduire en temps polynomial un algorithme solutionnant (??) en un algorithme solutionnant le problème de partitionnement par des ensembles de cardinal 3. Or ce problème est NP-complet donc (??) est NP-hard. ■

***Conclusion** : La procédure de minimisation ℓ_0 est optimale d'un point de vue théorique. Elle permet de résoudre le problème de CS pour un nombre minimal de mesures ($m = 2s$) et sous l'hypothèse minimale sur la matrice de mesure ($\text{Ker}(A) \cap \Sigma_{2s} = \{0\}$). Cependant, elle ne peut pas être utilisée en pratique, en générale, car c'est une procédure NP-hard. Elle n'est donc pas entièrement satisfaisante.*

Remarque 3.3. Le terme “en général” utilisé dans le Théorème 3.2 signifie qu'en général le problème (P_0) est NP-hard. Cependant, il se peut que dans certains cas particuliers, ce problème puisse être résolu de manière efficace. Le but de cette remarque est de donner un tel exemple.

Dans le cas particulier des mesures de Fourier, il est possible d'implémenter l'algorithme de minimisation ℓ_0 de manière très rapide (en temps linéaire). C'est l'algorithme “de Prony”.

On introduit maintenant le cadre dans lequel l'algorithme “de Prony” est effectif. Nous ne présenterons pas cet algorithme car sa compréhension nécessite quelques outils de l'analyse de Fourier qui sortent du cadre de ce cours.

Etant donné un signal $x \in \mathbb{C}^N$ et un ensemble $I \subset \{1, \dots, N\}$ de fréquences, il est en général impossible de reconstruire x uniquement à partir des coefficients de Fourier ($\hat{x}_i : i \in I$) à moins que $I = \{1, \dots, N\}$ (car la transformée de Fourier est une bijection de \mathbb{C}^N). Mais quand on sait que x est parcimonieux alors ce problème devient faisable même pour des ensembles I de cardinal $2s$. Ce type de mesures dites “structurées” sont très utilisées en pratique.

Pour fixer les notations, on introduit la matrice de Fourier (transformée de Fourier discrète) :

$$\Gamma : \begin{cases} \mathbb{C}^N & \rightarrow & \mathbb{C}^N \\ x & \rightarrow & \Gamma x = \hat{x} \end{cases} \quad \text{où } \Gamma = \left(\frac{w^{(p-1)(q-1)}}{\sqrt{N}} \right)_{1 \leq p, q \leq N} \quad \text{et } w = \exp(-2i\pi/N). \quad (3.1)$$

On note par $\bar{\Gamma}_1, \dots, \bar{\Gamma}_N$ les vecteurs lignes de Γ . On a donc $\hat{x}_i = \langle \bar{\Gamma}_i, x \rangle$ pour tout $i = 1, \dots, N$.

Pour un sous-ensemble $I \subset \{1, \dots, N\}$ de fréquences, on dispose des données ($\hat{x}_i : i \in I$). En terme matriciel, on dispose de la donnée $\Gamma_I x$ où

$$\Gamma_I : \begin{cases} \mathbb{C}^N & \rightarrow & \mathbb{C}^{|I|} \\ x & \rightarrow & (\hat{x}_i : i \in I). \end{cases}$$

Cette matrice est donc la matrice de vecteurs mesures en Compressed Sensing.

L'algorithme “de Prony” permet de retrouver x à partir des $2s$ premiers coefficients de Fourier de x et ceci pour tout $x \in \Sigma_s$. Cette procédure résout donc le problème de CS de manière optimale (pour un nombre de mesures égale à $2s$) et peut être implémenté par un algorithme en temps linéaire.

C'est donc un exemple de situation idéale en statistique (procédure optimale théoriquement et implémentable en temps linéaire). Cependant l'algorithme de Prony n'est ni robuste (par rapport au bruit) ni stable (quand le signal n'est pas exactement sparse, cet algorithme ne marche pas). C'est pour cela, qu'il n'est utilisé en pratique que dans certains cas de données très fiables (comme les DVD).

4 Relaxation convexe et la procédure du Basis Pursuit

La procédure de minimisation ℓ_0 n'est pas utilisable en pratique à cause de sa propriété NP-hard. Il faut donc trouver une autre procédure pour laquelle un algorithme exécutable en temps polynomial est constructible et pour lequel on peut prouver des propriétés théoriques proches de celles de (P_0) .

Pour les personnes familières avec les problèmes d'optimisation, le point faible de la procédure de minimisation ℓ_0 est que la fonction qu'on cherche à minimiser – la **fonction objectif** – n'est pas convexe.

Minimiser une fonction qui n'est pas convexe même sur un espace affine comme $x + \text{Ker}(A)$ est un problème difficile en général. Une idée qui peut alors venir naturellement à l'esprit (des personnes qui sont habituées à minimiser des fonctions convexes) est de convexifier la fonction objective qui est ici la fonction $\ell_0 : x \in \mathbb{R}^N \rightarrow \|x\|_0$. On est donc amené à chercher la fonction convexe la plus proche de la fonction ℓ_0 . Ce type de fonction est appelé *enveloppe convexe* dont nous rappelons la définition maintenant.

Définition 4.1. ([4]) Soient $N \geq 1$ un entier et C une partie convexe de \mathbb{R}^N . Soit $f : C \rightarrow \mathbb{R}$ une fonction. On appelle **enveloppe convexe** de f , la plus grande fonction convexe g telle que $g(x) \leq f(x)$ pour tout $x \in C$. On note par $\text{conv}(f)$, l'enveloppe convexe de f .

Cette définition a bien un sens car si \mathcal{C} représente l'ensemble de toutes les fonctions convexes plus petite que f alors $\sup_{g \in \mathcal{C}} g$ est toujours une fonction convexe et donc égale à $\text{conv}(f)$.

Théorème 4.2. La norme ℓ_1 est l'enveloppe convexe de la fonction ℓ_0 sur B_∞^N , la boule unité de $(\mathbb{R}^N, \ell_\infty^N)$.

Démonstration. On ne montre le résultat que sur B_1^N (la boule unité de (\mathbb{R}^N, ℓ_1^N)). Pour obtenir le résultat sur B_∞^N , on a recours au bi-conjugué convexe de ℓ_0 qui est une notion sortant du cadre de ce cours. On renvoie le lecteur intéressé à [4].

Pour tout $x \in B_1^N$, on a $\|x\|_1 \leq \|x\|_0$ donc si F est l'enveloppe convexe de ℓ_0 sur B_1^N , on aura : $\|x\|_1 \leq F(x)$ pour tout $x \in B_1^N$. Notamment, si on note par (e_1, \dots, e_N) la base canonique de \mathbb{R}^N , on voit que pour tout $i = 1, \dots, N$, $1 = \|\pm e_i\|_1 \leq F(\pm e_i) \leq \|\pm e_i\|_0 = 1$, donc $F(\pm e_i) = 1$. Soit $\lambda = (\lambda_1, \dots, \lambda_N)^\top \in \mathbb{R}^N$ tel que $\sum_{i=1}^N |\lambda_i| \leq 1$. On a, par convexité de F et comme $F(0) = 0$ (car $\|0\|_1 \leq F(0) \leq \|0\|_0$),

$$\begin{aligned} \|\lambda\|_1 &= \left\| \sum_{i=1}^N |\lambda_i| \text{sign}(\lambda_i) e_i \right\|_1 \leq F\left(\sum_{i=1}^N |\lambda_i| \text{sign}(\lambda_i) e_i + (1 - \|\lambda\|_1) 0 \right) \\ &\leq \sum_{i=1}^N |\lambda_i| F(\text{sign}(\lambda_i) e_i) + (1 - \|\lambda\|_1) F(0) = \sum_{i=1}^N |\lambda_i| = \|\lambda\|_1. \end{aligned} \quad (4.1)$$

Donc pour tout $\lambda \in B_1^N$, $F(\lambda) = \|\lambda\|_1$. Ceci prouve que l'enveloppe convexe de ℓ_0 sur B_1^N est bien ℓ_1 . ■

Remarque 4.3. L'enveloppe convexe de ℓ_0 sur tout \mathbb{R}^N est la fonction constante égale à zéro ; ce qui n'apporte pas beaucoup d'information. C'est pourquoi, on regarde l'enveloppe convexe de ℓ_0 seulement sur B_∞^N ou B_1^N ici.

C'est donc par relaxation convexe que la norme ℓ_1 apparaît dans le problème de Compressed Sensing. Plus généralement, la norme ℓ_1 joue un rôle centrale pour les statistiques en grandes

dimensions. Ici, pour le problème de CS, la procédure qui a été la plus étudiée est connue sous le nom de Basis Pursuit.

Définition 4.4 ([3, 1]). *Etant donné une matrice $A \in \mathbb{R}^{m \times N}$ et $y \in \mathbb{R}^m$ la procédure de **Basis Pursuit** (BP) renvoie*

$$\hat{x} \in \underset{t \in \mathbb{R}^N : At=y}{\operatorname{argmin}} \|t\|_1, \quad (\text{BP})$$

s'il existe une solution au système $Ax = y$ et \emptyset sinon.

Le problème BP est la relaxation convexe du problème de minimisation ℓ_0 . Avant de passer à l'étude théorique de cette procédure (càd de déterminer les conditions sur A sous lesquelles cette procédure permet de reconstruire exactement tout vecteur de Σ_s et de déterminer le nombre minimal de mesures et des matrices A qui satisfont ces conditions), on s'intéresse avant tout à ces propriétés algorithmiques (qui était le principal défaut de la minimisation ℓ_0).

Conclusion : La relaxation convexe est un outil très utilisé en statistiques en grandes dimensions. Elle permet dans certains cas de transformer des problèmes qui n'ont pas de solution algorithmique raisonnable en des procédures qui peuvent être implémentées efficacement grâce aux outils de l'optimisation convexe. C'est donc un outil essentiel à connaître et un bon réflexe à avoir lorsqu'on cherche à retrouver des structures qui semblent nécessiter une recherche combinatoire dans des ensembles de grandes tailles.

Remarque 4.5. *Les méthodes mettant en oeuvre la norme ℓ_1 comme fonction d'objectif à minimiser ou comme fonction de régularisation sont pléthores en statistiques en grande dimensions. Elles ont été utilisées empiriquement dans divers domaines comme la géologie/géophysique ([2], Taylor et al. (1979), Levy and Fullager (1981), Oldenburg et al. (1983), Santosa and Symes (1988)) en radioastronomie (Högbom (1974), Schwarz (1978)) en spectroscopie (Kawata et al. (1983), Mammone (1983), Minami et al. (1985), Barkhuijsen (1985), Newman (1988)), en imagerie médicale (Papoulis and Chamzas (1979)) etc.. Les autres méthodes célèbres utilisant la norme ℓ_1 sont le LASSO, le Dantzig sélecteur ou encore le Matching Pursuit.*

4.1 Une procédure de programmation linéaire pour le Basis Pursuit

La procédure de Basis Pursuit est très populaire en CS parce qu'elle a, en particulier, de bonnes propriétés algorithmiques. En effet, le problème (BP) peut être réécrit comme un problème de programmation linéaire. Ce type de problème est en quelque sorte le plus facile à résoudre en optimisation convexe : c'est un problème de minimisation d'une fonction objectif linéaire sous des contraintes linéaires.

On considère le problème de programmation linéaire suivant : étant donnés $A \in \mathbb{R}^{m \times N}$ et $y \in \mathbb{R}^m$,

$$(\hat{z}^+, \hat{z}^-) \in \underset{(z^+, z^-) \in \mathbb{R}^{2N}}{\operatorname{argmin}} \sum_{j=1}^N z_j^+ + z_j^- \text{ tel que } [A] - A \begin{bmatrix} z^+ \\ z^- \end{bmatrix} = y \text{ et } \begin{bmatrix} z^+ \\ z^- \end{bmatrix} \geq 0. \quad (\text{LP})$$

C'est un problème d'optimisation convexe dans \mathbb{R}^{2N} où la fonction objectif est linéaire ainsi que les contraintes. C'est donc bien un problème de programmation linéaire. On montre dans ce qui suit que les problèmes (BP) et (LP) sont équivalents. On rappelle d'abord la notation

suivante : pour tout $x \in \mathbb{R}^N$, on note $x^+ \in \mathbb{R}^N$ et $x^- \in \mathbb{R}^N$ deux vecteurs de \mathbb{R}^N dont les coordonnées sont, pour tout $j = 1, \dots, N$, données par

$$(x^+)_j = \max(0, x_j) \text{ et } (x^-)_j = \max(0, -x_j).$$

Proposition 4.6. *Il y a équivalence entre les deux problèmes (BP) et (LP) :*

1. Si \hat{x} est solution de (BP) alors (\hat{x}^+, \hat{x}^-) est solution de (LP)
2. Si (\hat{z}^+, \hat{z}^-) est solution de (LP) alors $\hat{z}^+ - \hat{z}^-$ est solution de (BP).

Démonstration. Supposons \hat{x} solution de (BP) et considérons sa partie positive \hat{x}^+ et négative \hat{x}^- . On a

$$[A] - A \begin{bmatrix} \hat{x}^+ \\ \hat{x}^- \end{bmatrix} = A\hat{x} \text{ et } \begin{bmatrix} \hat{x}^+ \\ \hat{x}^- \end{bmatrix} \geq 0 \quad (4.2)$$

donc (\hat{x}^+, \hat{x}^-) est bien dans l'ensemble de contrainte de (LP). Il suffit maintenant de montrer que $\sum_{j=1}^N \hat{x}_j^+ + \hat{x}_j^-$ est minimal dans cet ensemble.

Pour cela, on prend $(z^+, z^-) \in \mathbb{R}^{2N}$ dans l'ensemble de contrainte de (LP). On note $z = z^+ - z^-$. Ainsi $y = [A] - A \begin{bmatrix} z^+ \\ z^- \end{bmatrix} = Az$, donc z est dans l'ensemble de contrainte de (BP). Par minimalité de la norme ℓ_1 de \hat{x} sur l'espace de contrainte de (BP) (auquel appartient z), on a

$$\sum_{j=1}^N \hat{x}_j^+ + \hat{x}_j^- = \|\hat{x}\|_1 \leq \|z\|_1 \leq \sum_{j=1}^N z_j^+ + z_j^-.$$

Ceci étant vrai pour tout z dans l'espace de contrainte de (LP), on en déduit que (\hat{x}^+, \hat{x}^-) est bien solution de (LP).

Réciproquement, soit $(\hat{z}^+, \hat{z}^-) \in \mathbb{R}^{2N}$ une solution de (LP). On note $z = \hat{z}^+ - \hat{z}^-$. De la même manière que dans (4.2), on voit que z est dans l'ensemble de contrainte de (BP). Par ailleurs, si $x \in \mathbb{R}^N$ est dans l'ensemble de contrainte de (BP) alors (x^+, x^-) est dans celui de (LP) donc

$$\|z\|_1 \leq \sum_{j=1}^N \hat{z}_j^+ + \hat{z}_j^- \leq \sum_{j=1}^N x_j^+ + x_j^- = \|x\|_1.$$

Donc z est bien de norme ℓ_1 minimale sur l'ensemble de contrainte de (BP), c'est bien une solution de ce problème. ■

La proposition 4.6 nous assure donc que la procédure de Basis Pursuit est utilisable en pratique. C'est même plus que ça vu que (BP) peut se réécrire comme un programme linéaire, un cadre favorable en optimisation convexe. On est donc passé d'un problème NP-hard (P_0) à un problème de programmation linéaire par relaxation convexe. Ce qui constitue un gain important d'un point de vue computationnel. On passe ensuite aux propriétés théoriques du (BP).

4.2 Nombre minimal de mesures pour le Basis Pursuit

Il est difficile d'imaginer que le gain très important d'un point de vue computationnel que nous avons réalisé par relaxation convexe ("transformation" d'un problème NP-hard en un problème de programmation linéaire) ne se "paie" pas à un autre niveau. On va donc dans ce qui suit étudier les propriétés du Basis Pursuit et en particulier la propriété qui nous intéresse le plus, celle de résoudre le problème du compressed sensing sur Σ_s .

On a vu dans les sections précédentes que la procédure de minimisation ℓ_0 peut résoudre le problème du CS avec seulement $m = 2s$ mesures (sous certaines conditions sur la matrice de mesure). On aimerait savoir maintenant si le Basis Pursuit peut aussi résoudre ce problème et on aimerait déterminer le nombre minimal de mesures qui lui sont nécessaires et suffisantes pour le résoudre. On formalise cette propriété dans la définition qui suit.

Définition 4.7. Soit $A : \mathbb{R}^N \rightarrow \mathbb{R}^m$ une matrice telle que $m \leq N$ et s un entier plus petit que N . On dit que A vérifie la propriété de **reconstruction exacte d'ordre s** quand pour tout vecteur s -sparse x , on a

$$\operatorname{argmin}_{t \in \mathbb{R}^N} (\|t\|_1 : At = Ax) = \{x\}.$$

On dit alors que A vérifie $RE(s)$.

La propriété de reconstruction exacte d'ordre s caractérise donc les matrices de mesures A qui permettent de reconstruire exactement les vecteurs s -sparse x à partir de la donnée Ax grâce à l'algorithme de Basis Pursuit. Un des objectifs de ce cours est de déterminer les matrices vérifiant la propriété de reconstruction exacte d'ordre s pour un nombre minimal de mesures (càd de lignes). Dans la suite du cours, nous verrons des conditions suffisantes de reconstruction exacte d'ordre s .

Pour le moment, on peut déjà établir une borne inférieure sur le nombre m de lignes des matrices vérifiant cette propriété. C'est-à-dire, la propriété de reconstruction exacte de la Définition 4.7 implique nécessairement un nombre minimum de mesures à prendre. On peut mettre ce résultat en parallèle avec celui de la Proposition 2.2 montrant que la propriété de reconstruction de la procédure de minimisation ℓ_0 sur Σ_s implique que $m \geq 2s$. Ici, pour l'algorithme (BP), on aura besoin d'un peu plus de mesures.

Proposition 4.8. Soit $A : \mathbb{R}^N \mapsto \mathbb{R}^m$ une matrice vérifiant $RE(2s)$ alors

$$m \geq \frac{1}{\log 3} \left\lfloor \frac{s}{2} \right\rfloor \log \left(\left\lfloor \frac{N}{8es} \right\rfloor \right).$$

Démonstration. Soit $A : \mathbb{R}^N \mapsto \mathbb{R}^m$. On considère l'espace vectoriel quotient $\ell_1^N / \operatorname{Ker}(A)$, la fonction quotient et la norme quotient :

$$Q : \ell_1^N \mapsto \ell_1^N / \operatorname{Ker}(A) \text{ et } \|Qx\|_1 = \min_{h \in \operatorname{Ker}(A)} \|x + h\|_1. \quad (4.3)$$

On rappelle que $\mathbb{R}^N / \operatorname{Ker}(A)$ est l'ensemble de toutes les classes d'équivalence $Qx = x + \operatorname{Ker}(A)$ pour $x \in \mathbb{R}^N$. On note $\ell_1^N / \operatorname{Ker}(A)$ l'espace $\mathbb{R}^N / \operatorname{Ker}(A)$ muni de la norme quotient introduite dans (4.3).

Si A vérifie $RE(2s)$ alors pour tout $x \in \Sigma_{2s}$, on a $\|Qx\|_1 = \|x\|_1$, c'est-à-dire Q préserve les normes sur Σ_{2s} . Comme $\Sigma_s - \Sigma_s \subset \Sigma_{2s}$, on a pour tout $x, y \in \Sigma_s$,

$$\|Qx - Qy\|_1 = \|x - y\|_1. \quad (4.4)$$

C'est-à-dire Q est une isométrie sur Σ_s . On va pouvoir bénéficier de cette isométrie pour transporter la complexité métrique de l'ensemble (Σ_s, ℓ_1^N) dans l'espace quotient $\ell_1^N / \operatorname{Ker}(A)$ qui est au plus de dimension m (bien plus petite que la dimension N de l'espace ambiant) et donc en déduire une borne inférieure sur m (car la boule unité de $\ell_1^N / \operatorname{Ker}(A)$ va devoir contenir un ensemble 1/2-séparé de grande taille au vue de l'isométrie de Q sur Σ_s).

On commence d'abord par étudier la complexité de (Σ_s, ℓ_1^N) . On utilise le lemme suivant (qui peut être obtenu par une énumération successive par exemple et dont la preuve est donnée dans la feuille d'exercices corrigés jointe).

Lemme 4.9 (Varshamov-Gilbert). *Soit $s \leq N/2$. Il existe une famille \mathcal{S} d'ensembles de $\{1, \dots, N\}$ telle que*

1. $\forall S \in \mathcal{S}, |S| = s,$
2. $\forall S_1, S_2 \in \mathcal{S}, S_1 \neq S_2 \Rightarrow |S_1 \cap S_2| \leq \lfloor s/2 \rfloor,$
3. $\log |\mathcal{S}| \geq \lfloor \frac{s}{2} \rfloor \log \left(\lfloor \frac{N}{8es} \rfloor \right).$

Démonstration de la Proposition 4.8. On considère un ensemble \mathcal{S} comme défini dans Lemma 4.9 et pour tout $S \in \mathcal{S}$, on note $x(S) = s^{-1} \sum_{i \in S} e_i \in B_1^N$. On a pour tout $S_1 \neq S_2 \in \Sigma_s$

$$\|x(S_1) - x(S_2)\|_1 = \left\| \frac{1}{s} \sum_{i \in S_1 \Delta S_2} e_i \right\|_1 = \frac{|S_1 \Delta S_2|}{s} \geq 1$$

où $S_1 \Delta S_2 = S_1 \cup S_2 - S_1 \cap S_2$.

Par ailleurs, pour tout $S \in \mathcal{S}$, $\|Qx\|_1 = 1$, donc $\{x(S) : S \in \mathcal{S}\}$ est un sous-ensemble de la boule unité de $\ell_1^N / \text{Ker}(A)$. Il s'ensuit de la propriété d'isométrie (4.4) que la la boule unité de $\ell_1^N / \text{Ker}(A)$ contient un ensemble 1-séparé pour sa norme naturelle. Cet ensemble est de cardinal $|\mathcal{S}|$. Or, le cardinal d'un tel ensemble peut être majoré par un argument volumique comme suit.

Lemme 4.10 (Argument volumique). *Soit $\|\cdot\|$ une norme sur \mathbb{R}^n . On note par B sa boule unité. Soit $0 < \epsilon \leq 1$ et $\Lambda \subset B$ tels que pour tout $x, y \in \Lambda$ $\|x - y\| \geq \epsilon$. Alors nécessairement, le cardinal de Λ est tel que*

$$|\Lambda| \leq \left(1 + \frac{2}{\epsilon}\right)^n$$

Ainsi, d'après l'argument volumique du Lemme 4.10, un tel ensemble est de cardinal au plus $3^{\text{rang} A}$ (où $\text{rang} A = \dim(\ell_1^N / \text{Ker}(A))$). Donc $|\mathcal{S}| \leq 3^{\text{rang} A} \leq 3^m$ et le résultat s'obtient grâce au contrôle du cardinal de $|\mathcal{S}|$ dans Lemme 4.9. ■

Conclusion : On en déduit que pour pouvoir reconstruire tout vecteur s -sparse x à partir de m mesures Ax grâce au Basis Pursuit, on doit nécessairement avoir m plus grand que $s \log(eN/s)$ (à une constante absolue près). On verra par la suite qu'il existe bien des matrices vérifiant $RE(2s)$ avec m de l'ordre de $s \log(eN/s)$. En particulier, on ne “perd” qu'un terme logarithmique comparé à la condition “ $m \geq 2s$ ” de la procédure de minimisation ℓ_0 . Le gain très important d'un point de vue computationnel obtenu par relaxation convexe ne va donc être payé que par une perte logarithmique sur le nombre de mesures. Ce qui est presque rien.

Références

- [1] Emmanuel Candes and Terence Tao. Reflections on compressed sensing. *IEEE Information Theory Society Newsletter*, 2008.
- [2] J. Claerbout and F. Muir. Robust modeling of erratic data. *Geophysics*, 38 :826–844, 1973.
- [3] David L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4) :1289–1306, 2006.
- [4] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms. II*, volume 306 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1993. Advanced theory and bundle methods.