Introduction au Compressed sensing, Complétion de matrices et systèmes de recommandations

Guillaume Lecué*

Résumé

On introduit dans ce chapitre le problème de complétion de matrices particulièrement étudié pour la construction de systèmes de recommandation. Une hypothèse de parcimonie naturelle dans ce problème est l'hypothèse de *faible rang*. La procédure naturellement associée à cette hypothèse est basée sur la minimisation de la fonction rang sur un espace de solutions $\{B \in \mathbb{R}^{u\times v} : \langle B, X_i\rangle = y_i, \forall i = 1,\dots,m\}$. Cette procédure est en générale NP-hard. On va alors utiliser l'approche de la *relaxation convexe* dans ce cadre. La procédure obtenue est celle qui consiste à minimiser la norme nucléaire (i.e. la somme des valeurs singulières) sur l'espace des solutions. On montre que cette procédure peut se réécrire comme un problème de *semi-definite programming* et peut donc être implémentée (plus ou moins efficacement selon la taille de la matrice à compléter).

On analyse ensuite la procédure de minimisation de la norme nucléaire par l'*approche RIP* puis par une approche plus générale basée sur l'étude de la taille de la sous-différentielle de la norme de régularisation et sur la minoration d'un processus quadratique.

1 Introduction

Soit $A^* \in \mathbb{R}^{u\times v}$. On suppose qu'on dispose de m mesures linéaires de A^* données par $y_i = \langle A^*, X_i\rangle$, $i = 1,\dots,m$. On souhaite pouvoir reconstruire exactement A^* à partir des mesures $(y_i)_{i=1}^m$ et des directions $X_i \in \mathbb{R}^{u\times v}$ de projections.

Étant donné qu'il y a uv inconnues (c'est le nombre d'entrées de A^*) et seulement $m << uv$ équations, on retrouve un problème hautement sous-déterminé comme dans les chapitres précédents. On sait néanmoins qu'il est possible de résoudre ce type de problème si 'le signal A^*' à reconstruire est structuré de telle sorte que 'sa vraie dimension' n'est en fait pas uv mais une quantité bien plus petite (en particulier, plus petite que m). La structure qui apparaît naturellement dans de nombreux problèmes de ce type (on le verra dans la suite pour la construction de systèmes de recommandation) est une structure de faible rang : on suppose que A^* est de rang $r << u \wedge v$. L'hypothèse de faible rang peut alors être vue comme notre hypothèse de parcimonie lorsqu'on souhaite reconstruire une matrice à partir d'un petit nombre de mesures linéaires.

A retenir : Pour les problème de reconstruction d'une matrice à partir d'un petit nombre de mesures linéaires de cette matrice, l'hypothèse de faible rang est une hypothèse qui permet de résoudre le problème.

*CREST, ENSAE, 5 avenue Henry Le Chatelier 91120 Palaiseau. Email : guillaume.lecue@ensae.fr

Systèmes de recommandation par complétion de la matrice users/items. Prenons, par exemple, le problème lié aux systèmes de recommandation. Il existe plusieurs manières de voir (et donc de résoudre) ce problème. Le point de vue qu'on adopte ici (et qui a eu beaucoup de succès, en particulier, suite au 'Netflix prize') est celui de la complétion de matrice ; la matrice qu'on cherche à compléter ici est la matrice **users/items**. La complétion de la matrice users/items donne une liste d'items à recommander en premier à chaque users.

Pour le 'Netflix Price', on disposait de $u = 480.189$ users pour $v = 17.770$ items (ici des films). Les users notent certains films avec une note allant de 1 à 5 (par exemple, 1 quand le user n'a pas aimé le film et 5 lorsqu'il l'a beaucoup apprécié). Tous les users n'ont pas noté tous les films ; en fait, sur le nombre maximal uv de notes possibles seulement 100.480.507 notes ont été obtenues soit $1,2\%$ du nombre total de notes.

Une manière utile (en tout cas du point de vue de la complétion de matrice) de représenter ces données est de considérer la matrice $u \times v$ appelée matrice users/items qui a pour entrée (i, j) la note du user i sur l'item j quand elle existe (càd quand le user i a noté l'item j) ou qui laisse cette case vide lorsqu'on ne connaît pas cette note (voir la Figure 1).

Item/ User	10	11	13	14	15	16	17	18	19
1	2	3	4	5					
2		6			6	1	10		
3		4	1		7			2	
4		8	1						5
5		2							
6							5	1	

⇓ Matrix Completion ⇓

Item/ User	10	11	13	14	15	16	17	18	19
1	2	3	4	5	2	4	2	3	1
2	4	6	3	4	6	1	10	9	3
3	7	4	1	3	7	7	2	2	9
4	6	8	1	4	6	10	2	3	5
5	1	2	7	2	9	2	9	9	2
6	7	8	2	10	2	3	5	1	3

FIGURE 1 – La matrice users/items complétée par un algorithme de complétion de matrice pour la construction d'un système de recommandation.

Une fois la matrice users/items construite pour un problème donné, on peut utiliser un algorithme de complétion de matrice sur cette matrice pour 'remplir' les cases vides. Une fois la matrice complétée, on peut identifier sur chaque ligne (càd pour chaque users), le ou les items qui ont les meilleures notes et ainsi recommander ce ou ces items à ce user. C'est de cette manière qu'on peut utiliser les algorithmes de complétion pour construire des systèmes de recommandation. Il existe de nombreux algorithmes de complétion de matrices, nous allons en voir quelques uns dans ce chapitre.

A retenir : On peut construire un système de recommandations en complétant la matrice users/items.

Hypothèse de faible rang. La construction de la matrice users/items n'est qu'une mise en forme des données ; elle permet de visualiser ce qu'on cherche à faire de manière concise et peut

donner des idées sur la manière de le faire. Une manière formelle d'écrire ces données est de les considérer comme des observations de projections linéaires d'une matrice A^* comme au début du chapitre. En effet, si le user p a noté l'item q par la note $y_{p,q}$, alors on peut voir $y_{p,q}$ comme étant la valeur du produit scalaire $\langle A^*, E_{p,q}\rangle$ où $E_{p,q}$ est la matrice ayant pour entrées 0 partout sauf en (p,q) où elle vaut 1. On a donc $A^*_{p,q} = \langle A^*, E_{p,q}\rangle = y_{p,q}$. Ainsi, pour

$$\Omega = \{(p,q) \in \{1,\dots,u\} \times \{1,\dots,v\} \text{ tel que le user } p \text{ a noté l'item } q\}$$

on observe toutes les entrées de A^* indexées par Ω et donc la matrice à compléter de la Figure 1 est représentée par $(A^*_{p,q} : (p,q) \in \Omega)$. Le problème de complétion de la matrice users/items est donc bien équivalent au problème de reconstruction d'une matrice A^* à partir d'un petit nombre de mesures linéaires de cette matrice.

Ce problème n'a pas de solution unique (il existe un espace affine de matrices A telles que $\langle A, X_i\rangle = y_i, i = 1,\dots,m$). Cependant, la matrice A^* qu'on souhaite reconstruire est structurée : on pense qu'elle est de faible rang (ou approximativement de faible rang, càd proche d'une matrice de faible rang). En effet, dans le cas du problème posé par Netflix (et des systèmes de recommandation en général), on imagine que les users peuvent être classés selon un relativement petit nombre de profils comme ceux qui sont fans de SF, de films policiers, de films d'aventure, etc. de même on suppose que les films peuvent être bien classés selon des thèmes. Ainsi les presque 500.000 users de Netflix forment des groupes (plus ou moins) homogènes qui ont donc tendance a attribuer les (plus au moins) même notes aux films de même catégories. Intuitivement, la matrice A^* qu'on cherche à reconstruire devrait être proche d'une matrice par blocs où tous les fans de SF ont mis un 5 aux bons films de SF, ceux fans d'aventure ont mis un 5 aux bons films d'aventure, etc... Cette matrice par bloc est de faible rang ; son rang est de l'ordre de grandeur du minimum entre le nombre de groupes homogènes de users et celui des items.

C'est de cette manière que hypothèse de faible rang est entrée dans la problématique des systèmes de recommandation : si on pense que les users et/ou les items ont cette structure par groupes homogènes alors on peut s'attendre à ce que la matrice A^* soit proche d'une matrice de faible rang et donc cette hypothèse semble légitimée. Dans la suite, on fera l'approximation que A^* est de faible rang (et pas seulement approximativement de faible rang) de même que dans les chapitre d'avant on avait fait l'hypothèse que les vecteurs à reconstruire étaient exactement sparse. L'étude plus général de matrice approximativement de faible rang, ainsi que de donnée bruitée ne nécessite pas d'idées nouvelles et c'est pourquoi on ne considère que ce cadre plus simple.

A retenir : La matrice users/items est approximativement de faible rang. Cette propriété est utile pour la construction d'algorithmes de complétion de cette matrice (et donc de systémes de recommandations).

2 Relaxation convexe en complétion de matrice

Le problème introduit précédemment qu'on souhaite résoudre est le suivant : étant donné m mesures linéaires $y_i = \langle A^*, X_i\rangle, i = 1,\dots,m$ d'une matrice $A^* \in \mathbb{R}^{u\times v}$ de faible rang dans les directions $X_1,\dots,X_m \in \mathbb{R}^{u\times v}$ comment reconstruire A^* exactement ? combien de mesures m faut-il ? et comment choisir les matrices de mesures $X_1,\dots,X_m$? et dans le cas particulier du problème de complétion de matrices comment choisir les X_i dans l'ensemble $\{E_{p,q} : 1 \le p \le u, 1 \le q \le v\})$?

Le but de cette section est de suivre la démarche utilisée dans les chapitres précédents sur la reconstruction d'un vecteur sparse pour le problème qu'on a ici càd le problème de reconstruction d'une matrice de faible rang à partir d'un petit nombre de mesures linéaires.

On commence par quelques **notations** utiles dans la suite. Sur l'espace $\mathbb{R}^{u\times v}$ des matrices de taille $u \times v$, on définit un produit scalaire par :

$$\langle A, B\rangle = \sum_{i,j} A_{ij}B_{ij}.$$

Pour toute matrice $A \in \mathbb{R}^{u\times v}$, on note σ_A son **spectre**, càd, le vecteur de $\mathbb{R}^{u\wedge v}$ des valeurs singulières (qui sont les racines carrées des valeurs propres de $A^\top A$) : $\sigma_A = (\sigma_j)_{j=1}^{u\wedge v}$ et $\sigma_1(A) \geq \ldots, \geq \sigma_{u\wedge v}(A) \geq 0$. On définit ensuite les **normes de Schatten** par :

$$\|A\|_{S_p} = \|\sigma_A\|_{\ell_p} = \Big(\sum_{j=1}^{u\wedge v} \sigma_j(A)^p\Big)^{1/p}$$

pour tout $A \in \mathbb{R}^{u\times v}$ et $p \geq 1$. Le rang d'une matrice A est la dimension de son image, c'est aussi le nombre de valeurs singulières non nulles et peut donc s'exprimer comme la 'norme' ℓ_0 de son spectre :

$$\mathrm{rang}(A) = \|\sigma_A\|_{\ell_0}.$$

La norme S_1 est aussi appelée **norme nucléaire** ou **norme trace**. La norme S_2 est la **norme de Frobenius**, c'est la norme Hilbertienne associée à $\langle\cdot,\cdot\rangle$ défini plus haut. La norme S_∞ est la **norme d'opérateur** de ℓ_2^v dans ℓ_2^u :

$$\|A\|_{S_\infty} = \sup_{\|x\|_2=1} \|Ax\|_2.$$

On rappelle les résultats de dualité :

$$\|A\|_{S_q} = \sup\big(\langle A, B\rangle : \|B\|_{S_p} \leq 1\big)$$

quand $q^{-1} + p^{-1} = 1$.

Pour tout entier p, on note

$$\mathcal{S}^p = \{A \in \mathbb{R}^{p\times p} : A^\top = A\}, \mathcal{S}^p_+ = \{A \in \mathcal{S}^p : A \succeq 0\} \text{ et } \mathcal{S}^p_{++} = \{A \in \mathcal{S}^p : A \succ 0\}$$

qui sont respectivement, l'espace des **matrices symétriques**, le cône des **matrices semi-définies** (càd telles que $x^\top Ax \geq 0$ pour tout x) et le cône des **matrices semi-définies positives** (càd telles que $x^\top Ax > 0$ pour tout $x \neq 0$). On note aussi le groupe des **matrices orthonormales** par

$$\mathcal{O}(p) = \{A \in \mathbb{R}^{p\times p} : A^\top A = AA^\top = I_p\}$$

où I_p est la matrice identité de $\mathbb{R}^p$.

2.1 Procédure de minimisation du rang.

Sous l'hypothèse de faible rang, une procédure naturelle pour la reconstruction de A^* à partir des $y_i = \langle X_i, A^*\rangle, i = 1, \ldots, m$ est celle consistant à rechercher une solution de rang minimal dans l'espace $\{A \in \mathbb{R}^{u\times v} : \langle A, X_i\rangle = y_i, i = 1, \ldots, m\}$ de toutes les solutions ayant même projections :

$$\min\Big(\mathrm{rang}(A) : \langle A, X_i\rangle = y_i, i = 1, \ldots, m\Big). \tag{2.1}$$

On pourrait étudier les propriétés théoriques de la procédure (2.1) de la même manière qu'on a étudié la procédure de minimisation ℓ_0 aux chapitres précédents. Cependant, la procédure de

minimisation ℓ_0 a été disqualifiée par ces propriétés algorithmiques. En effet, on a montré que la procédure de minimisation ℓ_0 est en général NP-hard. Ce dernier point a justifié qu'on laisse de côté cette procédure et c'est aussi ce dernier point qui nous fait abandonner de suite la procédure (2.1).

Proposition 2.1. *Trouver une solution au problème de minimisation du rang sur un espace affine est un problème NP-hard en général.*

Démonstration. On utilise un principe de réduction comme on l'a fait pour réduire le problème de la minimisation ℓ_0 en le problème de recouvrement par des ensembles de taille 3. Ici, on montre qu'on peut réduire le problème de minimisation du rang en le problème de minimisation de la norme ℓ_0 en temps polynomial. L'idée est que si on a un algorithme solutionnant les problèmes de la forme

$$\min_{\mathcal{A}(A)=y} \operatorname{rang}(A) \tag{2.2}$$

où $y \in \mathbb{R}^m$ et $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ est un opérateur linéaire alors on aura aussi un algorithme solutionnant les problèmes de la forme

$$\min_{Bt=b} \|t\|_0 \tag{2.3}$$

où $B \in \mathbb{R}^{m_0\times N}$ et $b \in \mathbb{R}^{m_0}$.

En effet, on considère un problème de la forme (2.3) pour une matrice $B \in \mathbb{R}^{m_0\times N}$ et un vecteur $b \in \mathbb{R}^{m_0}$. On pose $u = v = N$ et $m = m_0 + N^2$. On construit l'opérateur

$$\mathcal{A} : \begin{cases} \mathbb{R}^{u\times v} & \longrightarrow & \mathbb{R}^{m_0+u\times v} \\ A & \longrightarrow & \begin{pmatrix} B\operatorname{diag}(A) \\ A - \operatorname{diag}(A) \end{pmatrix} \end{cases}$$

où u et v sont tels que $u \wedge v = N$ et $\operatorname{diag}(A) = (a_{ii})_{1\leq i\leq N} \in \mathbb{R}^N$. On construit le vecteur de mesures

$$y = \begin{pmatrix} b \\ 0 \end{pmatrix} \in \mathbb{R}^{m_0+N^2}.$$

Pour cet opérateur $\mathcal{A}$ et ce vecteur y de mesures, on voit que l'espace des solutions satisfait

$$\big\{A \in \mathbb{R}^{u\times v} : \mathcal{A}(A) = y\big\} = \big\{\text{matrices diagonales de } \mathbb{R}^{u\times v} \text{ de diagonale } t : Bt = b\big\}$$

et comme le rang d'une matrice diagonale c'est la 'norme' ℓ_0 de sa diagonale on a que si $\hat{A}$ est solution de (2.2) pour le choix de $\mathcal{A}$ et y comme ci-dessus alors $\hat{A}$ est une matrice diagonale et sa diagonale $\hat{t} = \operatorname{diag}(\hat{A})$ est solution de (2.3). Ainsi, comme la construction de $\mathcal{A}$ et y à partir de B et b se fait en temps polynomial en fonction des dimensions d'entrées de B et b, on voit que l'existence d'un algorithme solutionnant (2.3) peut être réduit en temps polynomial en un algorithme solutionnant (2.3). Comme (2.3) est NP-hard, (2.2) l'est aussi. ∎

La procédure de minimisation du rang est donc NP-hard en général. Cela signifie qu'à moins de tomber dans une situation assez favorable, on ne peut pas trouver d'algorithme solutionnant ce problème exactement en un temps raisonable. On ne va donc pas l'utiliser en pratique et donc ne pas l'utiliser tout cour.

Néanmoins, on peut essayer de comprendre ce qui rend le problème (2.2) si difficile numériquement. On voit ici que la contrainte est un espace affine ; c'est donc une contrainte assez facile à gérer (par exemple on sait projeter dessus assez facilement). C'est donc la fonction objectif $A \to \operatorname{rang}(A)$ qui pose problème. En particulier, elle n'est pas convexe. Une approche naturelle similaire à celle utilisée pour la construction du Basis Pursuit est de remplacer la fonction objectif (non-convexe) $A \to \operatorname{rang}(A)$ par la "fonction convexe la plus proche". C'est l'approche dite de relaxation convexe qu'on va maintenant mettre en œuvre pour le problème (2.2).

2.2 Enveloppe convexe du rang

Dans cette section, on va remplacer la fonction objectif $A \to \text{rang}(A)$ du problème d'optimisation (2.2) de minimisation du rang sur l'espace des solutions au système linéaire par une fonction convexe qui lui est proche. On rappelle d'abord une manière de convexifier une fonction sur un ensemble.

Définition 2.2. *Soit E un espace vectoriel et C un sous-ensemble convexe de E. Soit $f : C \to \mathbb{R}$. On appelle* ***enveloppe convexe de f sur C*** *la plus grande fonction convexe plus petite que f sur C. On la note* $\text{conv}(f)$.

Autrement dit, on a pour tout $x \in C$,

$$\text{conv}(f) : x \in C \to \sup\big(g(x) : g \text{ est convexe sur } C \text{ et } g(y) \leq f(y), \forall y \in C\big).$$

Cette définition a bien du sens car la fonction maximale d'une famille de fonctions convexes est convexe. C'est donc bien le cas pour la fonction maximale de l'ensemble de toutes les fonctions convexes sur C plus petites que f uniformément sur C.

Quand E est un espace de Hilbert, on peut montrer par des arguments géométriques sur l'épigraphe de f ou en écrivant que la fonction convexe $\text{conv}(f)$ est la fonction maximale de ses tangentes, qu'on peut restreindre la définition de $\text{conv}(f)$ à toutes les fonctions affines plus petites que f : on a pour tout $x \in C$,

$$\text{conv}(f)(x) = \sup_{a \in H, b \in \mathbb{R}} \big(\langle a, x\rangle + b : \langle a, y\rangle + b \leq f(y), \forall y \in C\big).$$

Définition 2.3. *Soit H un espace de Hilbert et C un sous-ensemble convexe de H. Soit $f : C \to \mathbb{R}$. On appelle* ***transformée de Fenchel de f sur C*** *la fonction définie en tout point $x \in H$ par*

$$f^*(x) = \sup_{y \in C} \big(\langle x, y\rangle - f(y)\big).$$

La transformée de Fenchel de f est une fonction convexe (même si f n'est pas convexe) car c'est la fonction maximale d'une famille de fonctions linéaires (donc convexes). Elle est définie sur l'espace entier H. On peut alors définir la transformée de Fenchel de f^* sur H par : pour tout $y \in H$,

$$f^{**}(y) = \sup_{x \in H} \big(\langle x, y\rangle - f^*(x)\big).$$

La fonction f^{**} est appelée **bidual** de f. On notera que le supremum définissant f^{**} est pris sur tout H alors que celui définissant f^* est restreint à C.

Théorème 2.4. *Soit H un espace de Hilbert et C un sous-ensemble convexe de H. Soit $f : C \to \mathbb{R}$. On note* $\text{conv}(f)$ *l'enveloppe convexe de f sur C et par f^{**} la fonction bidual de f sur C. On a pour tout $x \in C$, $f^{**}(x) = \text{conv}(f)(x)$.*

Démonstration. On utilise la caractérisation de $\text{conv}(f)$ en terme de fonctions affines : pour tout $x \in C$,

$$\text{conv}(f)(x) = \sup_{a \in H, b \in \mathbb{R}} \big(\langle a, x\rangle + b : \langle a, y\rangle + b \leq f(y), \forall y \in C\big). \tag{2.4}$$

Etant donné $a \in H$, on voit que la contrainte de (2.4) est satisfaite pour tout $b \in \mathbb{R}$ tel que pour tout $y \in C$,

$$b \leq f(y) - \langle a, y\rangle.$$

Etant donné qu'on cherche à maximiser en a et b, on n'a pas d'autres choix que de prendre

$$b = \inf_{y \in C} \big(f(y) - \langle a, y \rangle \big) = -f^*(a).$$

Pour ce choix de b la contrainte est vérifiée et on a donc

$$\mathrm{conv}(f)(x) = \sup_{a \in H} \big(\langle a, x \rangle - f^*(a) \big) = f^{**}(x).$$

■

Il suffit donc de calculer le **bidual** de f pour connaître son enveloppe convexe. C'est la stratégie qu'on utilise pour déterminer l'enveloppe convexe du rang sur la boule unité de la norme d'opérateur.

Théorème 2.5. *L'enveloppe convexe de la fonction rang sur $B_{S_\infty} = \{A \in \mathbb{R}^{u \times v} : \|A\|_{S_\infty} \leq 1\}$ est la norme nucléaire.*

Démonstration. On commence par rappeler l'inégalité de von Neuman.

Proposition 2.6 (Inégalité de von Neuman)**.** *Soit $A, B \in \mathbb{R}^{u \times v}$, on a*

$$|\langle A, B \rangle| \leq \sum_i \sigma_i(A) \sigma_i(B)$$

où $\sigma_1(A) \geq \cdots \geq \sigma_{u \wedge v}(A)$ est la suite décroissante des valeurs singulières de A. Le cas d'égalité $\langle A, B \rangle = \sum_i \sigma_i(A)\sigma_i(B)$ a lieu uniquement si, étant donné la SVD $A = U_A D_A V_A^\top$ de A, on a $U_A^\top B V_A = D_B$ où $D_A = \mathrm{diag}(\sigma_A)$ et $D_B = \mathrm{diag}(\sigma_B)$.

De même on a $\|A - B\|_{S_2} \geq \|\sigma_A - \sigma_B\|_2$ et le cas d'égalité a lieu quand $B = U_A D_B V_A^\top$.

Partie 1 : On calcul la transformée de Fenchel de la fonction rang $f(A) = \mathrm{rang}(A)$ sur B_{S_∞}. Pour tout $A \in \mathbb{R}^{u \times v}$, on a

$$f^*(A) = \sup_{B \in B_{S_\infty}} \big(\langle A, B \rangle - f(B) \big).$$

Par l'inégalité de von Neumann, on a $\langle A, B \rangle \leq \sum \sigma_i(A)\sigma_i(B)$ et comme le rang et la norme S_∞ sont invariants par multiplication à gauche et à droite par les matrices orthogonales, on a

$$f^*(A) = \sup_{B \in B_{S_\infty}} \big(\sum_i \sigma_i(A)\sigma_i(B) - f(B) \big).$$

On note $q = u \wedge v$ et on partitionne ensuite B_{S_∞} en fonction du rang :

$$\begin{aligned} f^*(A) &= \sup_{B \in B_{S_\infty}} \big(\sum_i \sigma_i(A)\sigma_i(B) - f(B) \big) = \max_{0 \leq r \leq q} \sup_{B \in B_{S_\infty} : f(B) = r} \big(\sum_{i=1}^{r} \sigma_i(A)\sigma_i(B) - r \big) \\ &= \max_{0 \leq r \leq q} \sup_{B \in B_{S_\infty} : f(B) = r} \big(\sum_{i=1}^{r} \sigma_i(A) - r \big) = \sum_{i=1}^{q} (\sigma_i(A) - 1)_+ \end{aligned}$$

Partie 2 : On calcul maintenant le bidual de f. Pour tout $B \in \mathbb{R}^{u \times v}$, on a

$$f^{**}(B) = \sup_{A \in \mathbb{R}^{u \times v}} \Big(\langle A, B \rangle - f^*(A) \Big).$$

Il s'ensuit de l'inégalité de von Neumann et du cas d'égalité que

$$f^{**}(B) = \sup_A \Big(\sum \sigma_i(A)\sigma_i(B) - f^*(A) \Big).$$

Pour tout $\|A\|_{S_\infty} \leq 1$, on a

$$f^*(A) = \sum_i (\sigma_i(A) - 1)_+ = 0.$$

On a donc

$$\sup_{\|A\|_{S_\infty} \leq 1} \Big(\sum \sigma_i(A)\sigma_i(B) - f^*(A) \Big) = \sup_{\|A\|_{S_\infty} \leq 1} \sum \sigma_i(A)\sigma_i(B) = \sum \sigma_i(B) = \|B\|_{S_1}.$$

Par ailleurs, si $\|A\|_{S_\infty} \geq 1$ et si $\|B\|_{S_\infty} \leq 1$ on a

$$\begin{aligned}
&\sum \sigma_i(A)\sigma_i(B) - f^*(A) = \sum_i \sigma_i(A)\sigma_i(B) - \sum_i (\sigma_i(A) - 1)_+ \\
&= \sum_i \sigma_i(B) + \sum_i (\sigma_i(A) - 1)\sigma_i(B) - \sum_i (\sigma_i(A) - 1)_+ \\
&= \|B\|_{S_1} + \underbrace{\sum_{i:\sigma_i(A)\leq 1} (\sigma_i(A) - 1)\sigma_i(B)}_{\leq 0} + \underbrace{\sum_{i:\sigma_i(A)>1} (\sigma_i(A) - 1)(\sigma_i(B) - 1)}_{\leq 0} \leq \|B\|_{S_1}
\end{aligned}$$

et on conclut car $\sigma_i(B) \leq 1$ pour tout i. ∎

Conclusion : L'enveloppe convexe de la fonction rang sur B_{S_∞} est la norme nucléaire. Il est alors naturel de "remplacer" la fonction objectif rang dans le problème NP-hard de minimisation du rang par la norme nucléaire suivant le principe de relaxation convexe.

2.3 Procédure de minimisation de la norme nucléaire et problème SDP

La relaxation convexe du problème (2.1) est donc la procédure de minimisation de la norme nucléaire sur l'espace des solutions :

$$\hat{A} \in \operatorname*{argmin}_{A \in \mathbb{R}^{u \times v}} \big(\|A\|_{S_1} : \mathcal{A}(A) = y \big). \tag{NM}$$

où on note $\mathcal{A}(A) = (\langle A, X_i \rangle)_{i=1}^m$ et $y = (y_i)_{i=1}^m$.

Tout comme pour le Basis Pursuit on vérifie que la procédure de minimisation de la norme nucléaire sur l'espace des solutions peut être implémentée de manière efficace en réécrivant (NM) comme un problème de *semidefinite programming* **(SDP)** (on rappelle la définition d'un problème SDP sous forme standard dans la Définition 2.9).

Pour cela on considère le problème suivant

$$\begin{pmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \end{pmatrix} \in \operatorname*{argmin}_{X \in \mathbb{R}^{u \times v}, Y \in \mathcal{S}^u, Z \in \mathcal{S}^v} \operatorname{Tr}(Y) + \operatorname{Tr}(Z) \text{ tel que } \begin{matrix} \mathcal{A}(X) = y \\ \begin{bmatrix} Y & X \\ X^\top & Z \end{bmatrix} \succeq 0 \end{matrix} \tag{SDP}$$

Théorème 2.7. *Les deux problèmes (NM) et (SDP) sont équivalents :*

1. *si* $\hat{A}$ *est solution de* (NM) *alors* $(\hat{A}, \hat{Y}, \hat{Z})^\top$ *est solution de* (SDP) *pour certains* $\hat{Y}$ *et* $\hat{Z}$ *(dependent uniquement de* $\hat{A}$*, cf.* (2.5)*),*
2. *si* $(\hat{X}, \hat{Y}, \hat{Z})^\top$ *est solution de* (SDP) *alors* $\hat{X}$ *est solution de* (NM)*.*

Pour démontrer Théorème 2.7, on démontre d'abord le lemme suivant.

Lemme 2.8. *Soit* $A \in \mathbb{R}^{u\times v}$ *et* $t \geq 0$*. Il y a équivalence entre les deux assertions suivantes :*

a) $\|A\|_{S_1} \leq t$

b) il existe $Y \in \mathcal{S}^u, Z \in \mathcal{S}^v$ *tels que*

$$\mathrm{Tr}(Y) + \mathrm{Tr}(Z) \leq 2t \text{ et } \begin{bmatrix} Y & A \\ A^\top & Z \end{bmatrix} \succeq 0$$

Démonstration. On suppose que *b)* est vraie. On considère la SVD de A : $A = UDV^\top$ où $U \in \mathcal{O}(u)$, $D = \mathrm{diag}(\sigma_A) \in \mathbb{R}^{u\times v}$ et $V \in \mathcal{O}(v)$. On voit que pour $I_{u,v} = \mathrm{diag}((1)_{i=1}^q) \in \mathbb{R}^{u\times v}$ où $q = u \wedge v$, la matrice

$$\begin{pmatrix} I_u & -UI_{u,v}V^\top \\ -VI_{u,v}^\top U^\top & I_v \end{pmatrix}$$

est semi-définie car pour tout $x \in \mathbb{R}^u, y \in \mathbb{R}^v$, on a

$$\begin{pmatrix} x^\top & y^\top \end{pmatrix} \begin{pmatrix} I_u & -UI_{u,v}V^\top \\ -VI_{u,v}^\top U^\top & I_v \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \|x\|_2^2 + \|y\|_2^2 - 2\langle U^\top x, I_{u,v}V^\top y\rangle \geq 0.$$

Comme le produit de deux matrices semi-définies est aussi semi-défini, on a

$$\mathbf{Tr}\left[\begin{pmatrix} I_u & -UI_{u,v}V^\top \\ -VI_{u,v}^\top U^\top & I_v \end{pmatrix} \begin{pmatrix} Y & A \\ A^\top & Z \end{pmatrix}\right] \geq 0$$

càd

$$\mathbf{Tr}(Y - UI_{u,v}V^\top A^\top) + \mathbf{Tr}(-VI_{u,v}U^\top A + Z) \geq 0$$

et donc $2t \geq \mathbf{Tr}(Y) + \mathbf{Tr}(Z) \geq 2\mathbf{Tr}(D) = 2\|A\|_{S_1}$.

On suppose que *a)* est vrai. On considère la SVD de A sous la forme : $A = U_r D_r V_r^\top$ où $r = \mathrm{rang}(A)$, $U_r \in \mathbb{R}^{u\times r}$ est la matrice dont les colonnes sont les vecteurs singuliers gauches de A, $V_r \in \mathbb{R}^{v\times r}$ où les colonnes sont les vecteurs singuliers droits de A et $D_r \in \mathbb{R}^{r\times r}$ est la matrice diagonale des valeurs singulières non-nulles de A.

On pose

$$Y = U_r D_r U_r^\top + \gamma I_u \in \mathcal{S}^u \text{ et } Z = V_r D_r V_r^\top + \gamma I_v \in \mathcal{S}^v \tag{2.5}$$

pour un certain $\gamma > 0$ à déterminer. On a

$$\mathbf{Tr}(Y) + \mathbf{Tr}(Z) = 2\mathbf{Tr}(D_r) + \gamma(u+v) = 2\|A\|_{S_1} + \gamma(u+v) = 2t$$

pour $\gamma = 2(t - \|A\|_{S_1})/(u+v)$. De plus,

$$\begin{bmatrix} Y & A \\ A^\top & Z \end{bmatrix} = \begin{bmatrix} U_r D_r U_r^\top & U_r D_r V_r^\top \\ V_r D_r U_r^\top & V_r D_r V_r^\top \end{bmatrix} + \gamma \begin{bmatrix} I_u & 0 \\ 0 & I_v \end{bmatrix} = \begin{bmatrix} U_r \\ V_r \end{bmatrix} D_r \begin{bmatrix} U_r^\top & V_r^\top \end{bmatrix} + \gamma I_{u+v} \succeq 0$$

■

Preuve du Théorème 2.7. On démontre d'abord le premier point. Si $\hat{A}$ est solution de (NM) alors d'après Lemme 2.8, il existe $\hat{Y}, \hat{Z}$ tel que

$$\mathrm{Tr}(\hat{Y}) + \mathrm{Tr}(\hat{Z}) = 2\left\|\hat{A}\right\|_{S_1} \text{ et } \begin{bmatrix} \hat{Y} & \hat{A} \\ \hat{A}^\top & \hat{Z} \end{bmatrix} \succeq 0.$$

Comme $\mathcal{A}(\hat{A}) = y$, on a bien que $(\hat{A}, \hat{Y}, \hat{Z})$ est dans l'ensemble des contrainte de (SDP). De plus $\mathrm{Tr}(\hat{Y}) + \mathrm{Tr}(\hat{Z}) = 2\left\|\hat{A}\right\|_{S_1}$ donc si (X, Y, Z) est dans l'ensemble des contraintes de (SDP), on aura $\mathcal{A}(X) = y$ et donc, par optimalité de $\hat{A}$, $\left\|\hat{A}\right\|_{S_1} \leq \|X\|_{S_1}$, mais aussi $\mathrm{Tr}(Y) + \mathrm{Tr}(Z) = 2\|X\|_{S_1}$ et donc nécessairement

$$\mathrm{Tr}(\hat{Y}) + \mathrm{Tr}(\hat{Z}) = 2\left\|\hat{A}\right\|_{S_1} \leq 2\|X\|_{S_1} = \mathrm{Tr}(Y) + \mathrm{Tr}(Z).$$

Donc $(\hat{A}, \hat{Y}, \hat{Z})$ est bien solution de (SDP).

Pour démontrer le point *2)*, on suppose que $(\hat{X}, \hat{Y}, \hat{Z})$ est solution de (SDP). On a

$$\mathrm{Tr}(\hat{Y}) + \mathrm{Tr}(\hat{Z}) \leq 2t \text{ et } \begin{bmatrix} \hat{Y} & \hat{X} \\ \hat{X}^\top & \hat{Z} \end{bmatrix} \succeq 0$$

pour $t = (\mathrm{Tr}(\hat{Y}) + \mathrm{Tr}(\hat{Z}))/2$. Alors, d'après Lemme 2.8, on a

$$\left\|\hat{X}\right\|_{S_1} \leq \frac{\mathrm{Tr}(\hat{Y}) + \mathrm{Tr}(\hat{Z})}{2}.$$

Soit A dans l'espace des contraintes de (NM). Par le Lemme 2.8, il existe Y, Z tel que $\mathrm{Tr}(Y) + \mathrm{Tr}(Z) = 2\|A\|_{S_1}$ et (A, Y, Z) est dans l'espace des contrainte de (SDP). En particulier,

$$\left\|\hat{X}\right\|_{S_1} \leq \frac{\mathrm{Tr}(\hat{Y}) + \mathrm{Tr}(\hat{Z})}{2} \leq \frac{\mathrm{Tr}(Y) + \mathrm{Tr}(Z)}{2} = \|A\|_{S_1}$$

et comme $\mathcal{A}(\hat{X}) = y$, $\hat{X}$ est dans l'ensemble des contraintes de (NM). On en déduit donc que $\hat{X}$ est solution de (NM). ∎

On peut donc réécrire le problème de minimisation de la norme nucléaire comme un problème *SDP* dont on rappelle la forme générale maintenant.

Définition 2.9. *Un problème d'optimisation est un* ***semidefinite programming (SDP)*** *s'il peut se réécrire sous la forme*

$$\min_{X \in \mathcal{S}^p} \left(\langle C, X \rangle : \tilde{\mathcal{A}}(X) = b, X \succeq 0 \right) \tag{2.6}$$

où $C \in \mathcal{S}^p$, $\tilde{\mathcal{A}} : \mathcal{S}^p \to \mathbb{R}^m$ est une application linéaire et $b \in \mathbb{R}^m$.

On peut réécrire le problème (SDP) sous la forme d'un *semidefinite programming* :

$$\min \left(\langle C, \begin{bmatrix} Y & X \\ X^\top & Z \end{bmatrix} \rangle : \tilde{\mathcal{A}} \left(\begin{bmatrix} Y & X \\ X^\top & Z \end{bmatrix} \right) = y, \begin{bmatrix} Y & X \\ X^\top & Z \end{bmatrix} \succeq 0 \right)$$

où le minimum est pris sur toutes les matrices symétriques $\begin{bmatrix} Y & X \\ X^\top & Z \end{bmatrix}$,

$$C = I_{u+v}, \tilde{\mathcal{A}} \left(\begin{bmatrix} Y & X \\ X^\top & Z \end{bmatrix} \right) = \mathcal{A}(X) \text{ et } b = y.$$

Il existe ensuite des algorithmes similaires à ceux vus en *Linear programming* pour résoudre de manière approchée les problèmes SDP. Par exemple, sous `CVXOPT`, on résout (2.6) en transformant C et X en vecteur de taille p^2 par concaténation (par colonnes par exemple) par la fonction $\text{vec} : \mathbb{R}^{p\times p} \to \mathbb{R}^{p^2}$ et en réécrivant $\tilde{\mathcal{A}}$ comme un opérateur linéaire $A^\top : \mathbb{R}^{p^2}$ dans $\mathbb{R}^m$. On réécrit alors (2.6) sous la forme

$$\min_X \Big(\big\langle \text{vec}(C), \text{vec}(X) \big\rangle : A^\top(\text{vec}(X)) = b, X \succeq 0 \Big)$$

qui se résout avec la commande

```
> sol = cvxopt.solvers.conelp(-b, A, vec(C))
```

où $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times p^2}$ et $vec(C) \in \mathbb{R}^{p^2}$ sont des objets du type `matrix` de `CVXOPT`. On récupère ensuite $\text{vec}(X)$ par la commande

```
> print(sol['z']).
```

Conclusion : On peut réécrire le problème de minimisation de la norme nucléaire comme un problème SDP. Il existe des algorithme de type barrière primal-dual pour résoudre les SDP.

3 Etude de la reconstruction exacte par (NM)

L'objectif de cette section est de déterminer pour quels "vecteurs" de mesures $X_i \in \mathbb{R}^{u\times v}, i = 1, \dots, m$ et en quel nombre m on peut reconstruire exactement toutes les matrices A de rang r à partir des observations $y_i = \langle A, X_i \rangle, i = 1, \dots, m$ en utilisant la procédure (NM). En notant

$$\Sigma_r = \big\{ A \in \mathbb{R}^{u\times v} : \text{rang}(A) = r \big\}$$

on est alors intéressé par la propriété suivante :

Définition 3.1. *Soit $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ un opérateur linéaire de mesure $\mathcal{A}(A) = (\langle A, X_i \rangle)_{i=1}^m$. On dit que $\mathcal{A}$ satisfait la* ***propriété de reconstruction exacte d'ordre*** *r, notée RE(r) quand pour toute matrice $A \in \Sigma_r$,*

$$\operatorname*{argmin}_X \big(\|X\|_{S_1} : \mathcal{A}(X) = \mathcal{A}(A) \big) = \{A\}.$$

Autrement dit, un opérateur de mesures $\mathcal{A}(\cdot)$ vérifie $\text{RE}(r)$ quand pour tout $A \in \Sigma_r$, il y a une unique matrice de plus faible norme nucléaire dans l'espace des solutions $\{X \in \mathbb{R}^{u\times v} : \mathcal{A}(X) = \mathcal{A}(A)\}$ et cette matrice est A.

Il est possible de reprendre la même suite d'arguments que ceux développés pour l'étude du Basis Pursuit à partir de version matricielle de RIP et NSP. On présente brièvement ces arguments (sans preuve, car elles sont identiques au cas vectoriel) dans la section suivante. On mettra plutôt en avant dans la dernière section, une méthode de preuve alternative mettant en avant le rôle de la non-différentiation de la fonction objective pour induire de la parcimonie.

3.1 Argumentation basée sur l'approche RIP

On présente ici, dans le cas matriciel, les outils introduits précédemment pour la reconstruction exacte de vecteurs parcimonieux à partir d'un petit nombre de mesures. On commence par la *Null Spae property* qui donne une condition nécessaire et suffisante de reconstruction exacte.

Définition 3.2. *Soit $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ un opérateur linéaire. On dit que $\mathcal{A}$ satisfait la **rank-Null Space Property (rank-NSP)** de degrés r quand pour toutes matrices A dans le noyau de $\mathcal{A}$, on a*

$$\sum_{i=1}^{r} \sigma_i(A) < \sum_{i\geq r+1} \sigma_i(A)$$

où $\sigma_1(A) \geq \sigma_2(A) \geq \cdots$ est la suite décroissante des valeurs singulières de A.

Proposition 3.3. *Soit $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ un opérateur linéaire. Il y a équivalence entre les deux assertions suivantes :*

1. *$\mathcal{A}$ satisfait RE(r)*
2. *$\mathcal{A}$ satisfait rank-NSP(r).*

Il est cependant difficile de prouver rank-NSP directement. On peut alors considérer une version matricielle de RIP.

Définition 3.4. *Soit $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ un opérateur linéaire. On dit que $\mathcal{A}$ satisfait **rank Restricted Isometry Property (rank-RIP)** de degrés r et de constante d'isométrie δ_r quand pour toutes matrices A de rang r, on a*

$$(1-\delta_r)\|A\|_{S_2}^2 \leq \frac{\|\mathcal{A}(A)\|_{\ell_2^m}^2}{m} \leq (1+\delta_r)\|A\|_{S_2}^2 .$$

Proposition 3.5. *Il existe des constantes absolues $c_0 > 0, 0 < c_1 < 1$ pour lesquelles ce qui suit est vrai. Soit $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ un opérateur linéaire. Si $\mathcal{A}(\cdot)$ vérifie rank-RIP($c_0 r$) pour $\delta_r < c_1$ alors rank-NSP(r) est vérifiée.*

De plus, on peut montrer que dans le cas de mesures Gaussiennes, la condition rank-RIP(r) est vérifiée avec grande probabilité dès que $m \gtrsim r(u+v)$. On définit d'abord ce qu'on entend par mesures gaussiennes dans le cas matriciel. On dit que $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ où $\mathcal{A}(A) = (\langle X_i, A\rangle)_{i=1}^m$ est un **opérateur linéaire de mesures gaussiennes** quand pour tout $i = 1, \ldots, m$, $X_i = (g_{pq} : 1 \leq q \leq u, 1 \leq q \leq v)$ est une matrice Gaussienne, càd les g_{pq} sont i.i.d. $\mathcal{N}(0,1)$. De plus les X_i's sont indépendants entre eux.

Théorème 3.6. *Soit $\mathcal{A} : \mathbb{R}^{u\times v} \to \mathbb{R}^m$ un opérateur de mesures Gaussiennes et soit $0 < \delta < 1$. Il existe des constantes c_0, c_1 et c_2 dépendant uniquement de δ telles que, avec probabilité au moins $1 - c_0\exp(-c_1 m)$, $\mathcal{A}(\cdot)$ vérifie RIP(r) dès que $m \geq c_2 r(u+v)$.*

On peut aussi montrer par un argument de complexité que $c_3 r(uv+v)$ mesures sont nécessaires.

3.2 Preuve générale mettant l'accent sur la non-différentiation de la fonction objective

Dans cette section, on considère un espace de Hilbert muni de son produit scalaire $\langle\cdot,\cdot\rangle$. Soit $E \subset H$ un sous-espace vectoriel de H et $\|\cdot\|$ une norme sur E. Soient $X_1, \ldots, X_m$ des vecteurs de mesures éléments de H. On observe $y_i = \langle X_i, x^*\rangle, i = 1, \ldots, m$ pour un certain $x^* \in E$. On souhaite pouvoir reconstruire x^* exactement à partir des mesures $(y_i)_{i=1}^m$.

Pour cela, on considère la procédure

$$\hat{x} \in \operatorname{argmin}\big(\|x\| : \mathcal{A}(x) = y\big) \tag{3.1}$$

où $\mathcal{A}(x) = (\langle X_i, x\rangle)_{i=1}^m$ est l'opérateur de mesures et $y = (y_i)_{i=1}^m$ est le vecteur des mesures. On souhaite savoir quels sont les x^* qui peuvent être exactement reconstruit par (3.1), pour quels vecteurs de mesures $X_1, \dots, X_m$ et en quel nombre m ?

Pour répondre à ces questions, on va introduire une condition sur les $X_1, \dots, X_m$ et une condition sur la sous-différentielle de la fonction objective définie par la norme $\|\cdot\|$. On introduit quelques notations :

1. la norme duale associée à $\|\cdot\|$ est définie pour tout $x \in E$ par

$$\|x\|_* = \sup\big(\langle x, y\rangle : \|y\| \leq 1\big),$$

2. les boules et sphères unité associées à $\|\cdot\|$, $\|\cdot\|_*$ et $\|\cdot\|_2$ (où $\|x\|_2^2 = \langle x, x\rangle$) sont notées, respectivement par

$$\begin{aligned} &B = \{x \in E : \|x\| \leq 1\}, S = \{x \in E : \|x\| = 1\} \\ &B_* = \{x \in E : \|x\|_* \leq 1\}, S_* = \{x \in E : \|x\|_* = 1\} \\ &B_2 = \{x \in H : \|x\|_2 \leq 1\}, S_2 = \{x \in H : \|x\|_2 = 1\}. \end{aligned}$$

3. la sous-différentielle de $\|\cdot\|$ en x est

$$\partial \|\cdot\|(x) = \{g \in E : \|x + h\| \geq \|x\| + \langle g, h\rangle, \forall h \in E\} = \begin{cases} \{g \in S_* : \langle g, x\rangle = \|x\|\} & \text{si } x \neq 0 \\ B_* & \text{si } x = 0. \end{cases}$$

(cf. Exemple 2.5 du chapitre précédent sur les méthodes proximales).

On introduit d'abord une condition sur l'opérateur de mesures.

Définition 3.7. *Soit $X_1, \dots, X_m$ des vecteurs de H et $\mathcal{A} : H \to \mathbb{R}^m$ l'opérateur de mesures tel que $\mathcal{A}(A) = (\langle A, X_i\rangle)_{i=1}^m$. Pour tout $r^* > 0$, on introduit le cône*

$$\mathcal{C}_{r^*} = \{x \in E : \|x\|_2 \geq r^* \|x\|\}. \tag{3.2}$$

*On dit que $\mathcal{A}(\cdot)$ vérifie la **condition de valeur singulière restreinte de rayon** r^*, notée VSR(r^*), quand pour tout $x \in \mathcal{C}_{r^*}$, on a*

$$\frac{1}{m}\sum_{i=1}^m \langle x, X_i\rangle^2 \geq \frac{\|x\|_2^2}{2}. \tag{3.3}$$

Quand $H = \mathbb{R}^N$ et $\|\cdot\| = \|\cdot\|_1$, on montre que VSR(r^*) est impliquée par RIP(s) pour $r^* \sim 1/\sqrt{s}$. Par exemple quand les mesures sont sous-gaussiennes, on peut montrer qu'avec probabilité au moins $1 - 2\exp(-c_0 m)$, VSR(r^*) est vérifiée pour

$$r^* = \inf\big(r > 0 : \ell^*(B \cap rB_2) \leq c_1 r\sqrt{m}\big) \tag{3.4}$$

où $\ell^*(B \cap rB_2)$ est la **fenêtre Gaussienne moyenne** de $B \cap rB_2$. On rappelle la définition des fenêtres Gaussiennes.

Définition 3.8. *Soit T un sous-ensemble de l'espace de Hilbert H. On note par $(G_t)_{t\in T}$, le processus Gaussien canonique associé à T. La fenêtre Gaussienne de T est*

$$\ell^*(T) = \mathbb{E}\sup_{t\in T} G_t.$$

En dimension finie, le processus Gaussien $(G_t)_{t\in T}$ est donné par $G_t = \langle G, t\rangle$ où G est une variable aléatoire Gaussienne Standard. Par exemple, dans $\mathbb{R}^N$, on a pour tout $t = (t_j)_{j=1}^N \in \mathbb{R}^N$, $G_t = \sum_{j=1}^N g_i t_i$ où $g_1, \dots, g_N$ sont des $\mathcal{N}(0,1)$ i.i.d.. On a alors pour tout $T \subset \mathbb{R}^N$,

$$\ell^*(T) = \mathbb{E} \sup_{t\in T} \sum_{j=1}^N g_j t_j.$$

On introduit maintenant un paramètre relatif à la taille de la sous-différentielle de $\|\cdot\|$ en tout point.

Définition 3.9. *Soit $\|\cdot\|$ une norme sur E un sev de H. On note par S la sphère unité de $\|\cdot\|$ et par $\partial\|\cdot\|(x^*)$ sa sous-différentielle en x^*. Pour tout $x^* \in E$, on définit le coefficient de sparsité de $\|\cdot\|$ en x^* pour le rayon r^* par*

$$\Delta_{r^*}(x^*) = \inf_{h\in S\cap r^*B_2} \sup_{z\in\partial\|\cdot\|(x^*)} \langle z, h\rangle. \tag{3.5}$$

Comme $S\cap r^*B_2 \subset S$ et $\partial\|\cdot\|(x^*) \subset S_*$ quand $x^* \neq 0$, on a forcément pour tout $h \in S\cap r^*B_2$ et $z \in \partial\|\cdot\|(x^*)$, $\langle z, h\rangle \leq 1$. La condition qui va nous intéresser dans la suite est de demander que $\Delta_{r^*}(x^*) > 0$.

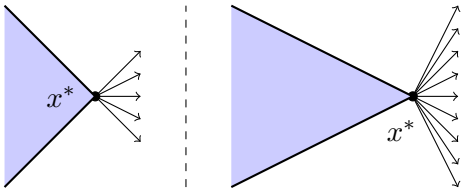

FIGURE 2 – La richesse de la sous-différentielle de $\|\cdot\|$ en les points d'intérêt (càd les points sparse en un certain sens) – ou dans un voisinage de ces points dans le cas bruité – joue un rôle central en statistiques en grandes dimensions.

Théorème 3.10. *On suppose qu'il existe $r^* > 0$ tel que l'opérateur de mesures $\mathcal{A}(\cdot)$ satisfait la condition VSR(r^*). Alors pour tout x^* tel que $\Delta_{r^*}(x^*) > 0$, on aura*

$$\operatorname*{argmin}_{x:\mathcal{A}(x)=\mathcal{A}(x^*)} \|x\| = \{x^*\}.$$

Démonstration. Soit $x^* \in E$ tel que $\Delta_{r^*}(x^*) > 0$. Soit $x \in E$ tel que $\mathcal{A}(x) = \mathcal{A}(x^*)$ et $x \neq x^*$. Montrons que $\|x\| > \|x^*\|$.

Si $x - x^* \in \mathcal{C}_{r^*}$ alors d'après VSR(r^*), on a

$$\|\mathcal{A}(x) - \mathcal{A}(x^*)\|_2^2 = \sum_{i=1}^m \langle X_i, x^* - x\rangle^2 \geq \frac{m\|x^* - x\|_2^2}{2} > 0$$

car $x \neq x^*$. Alors $\mathcal{A}(x) \neq \mathcal{A}(x^*)$ ce qui est une contradiction et donc $x - x^* \notin \mathcal{C}_{r^*}$. On a alors en posant $h = x - x^*$ que $\|h\|_2 < r^*\|h\|$ et donc $h/\|h\| \in S\cap r^*B_2$. Alors

$$\|x\| - \|x^*\| = \|x^* + h\| - \|x^*\| \geq \langle g, h\rangle = \langle g, \frac{h}{\|h\|}\rangle \|x - x^*\|$$

pour tout $g \in \partial \|\cdot\|(x^*)$. On passant au supremum sur $g \in \partial \|\cdot\|(x^*)$ et à l'infimum sur $h/\|h\| \in S \cap r^* \|h\|_2$, on obtient

$$\|x\| - \|x^*\| \geq \Delta_{r^*}(x^*) \|x - x^*\| > 0$$

car $x \neq x^*$ et $\Delta_{r^*}(x^*) > 0$. On a donc bien $\|x\| > \|x^*\|$ et donc x^* est l'unique solution du problème de minimisation

$$\min_{x:\mathcal{A}(x)=\mathcal{A}(x^*)} \|x\|.$$

■

On voit donc que la taille de la sous-différentielle de $\|\cdot\|$ en les points sparse joue un rôle central pour leur reconstruction. Pour illustrer ce propos, on peut calculer :

1. le sous-gradient de la norme ℓ_1 en les vecteurs s-sparse
2. le sous-gradient de la norme nucléaire en les matrices de rang r.

On observera alors que plus le vecteur (resp. la matrice) est sparse (resp. de faible rang) plus la sous-différentielle en ce point est riche et donc plus Δ_{r^*} a des chances d'être strictement positif.

On commence par le calcul de la sous-différentielle de la norme ℓ_1. En introduction, on peut visualiser cette différentielle en dimension $N = 2$, puis on étudiera le cas général.

La sous-différentielle de la fonction valeur absolue est donnée par définition pour tout $t \in \mathbb{R}$ par

$$\partial|\cdot|(t) = \{g \in \mathbb{R} : |t+h| \geq |t| + gh, \forall h \in \mathbb{R}\}.$$

Soit $t > 0$. Si $g \in \mathbb{R}$ est un sous-gradient de la valeur absolue en t alors pour tout $h \in \mathbb{R}$, on aura $|t+h| \geq |t|+gh$ en particulier pour tout $-t < h < 0$, $t+h \geq t+gh$ donc $h \geq gh$ et comme $h < 0$, on a $g \geq 1$. De même pour tout $h > 0$, on aura $t+h \geq t+gh$ donc $h \geq gh$ et comme $h > 0$, on a $g \leq 1$. Alors, nécessairement, $g = 1$ et comme $g = 1$ est bien un sous-gradient de $|\cdot|$ en $t > 0$ vu que $|t+h| \geq |t|+h$, on a bien $\partial|\cdot|(t) = \{1\}$. De même quand $t < 0$, l'unique sous-gradient de $|\cdot|$ en t est -1. Pour $t = 0$, g est un sous-gradient quand $|h| \geq gh$ pour tout $h \in \mathbb{R}$, c'est le cas pour tout $g \in [-1, 1]$. On conclut que

$$\partial|\cdot|(t) = \begin{cases} \{1\} & \text{si } t > 0 \\ \{-1\} & \text{si } t < 0 \\ [-1,1] & \text{si } t = 0. \end{cases}$$

On retrouve bien que la sous-différentielle d'une fonction en un point où elle est différentiable est réduite à un point qui est le sous-gradient de cette fonction en ce point.

Pour le calcul de la sous-différentielle de la norme $\|\cdot\|_1$ en dimension 2, on considère $x = (x_1, x_2)^\top \in \mathbb{R}^2$. On suppose $x_1 \neq 0, x_2 \neq 0$. Si $g = (g_1, g_2)^\top \in \mathbb{R}^2$ est un sous-gradient de $\|\cdot\|_1$ en x alors pour tout $h = (h_1, h_2)^\top \in \mathbb{R}^2$, on doit avoir

$$|x_1 + h_1| + |x_2 + h_2| \geq |x_1| + |x_2| + g_1 h_1 + g_2 h_2.$$

En particulier, pour $h_1 = 0$ on voit que g_2 est un sous-gradient de $|\cdot|$ en x_2. De même quand $h_2 = 0$, on trouve que g_1 est un sous-gradient de $|\cdot|$ en x_1. On en déduit que $g = \text{sign}(x)$. Si une des coordonnée de x est nulle alors on voit que pour cette coordonnée le sous-gradient peut prendre toute les valeurs dans $[-1, 1]$. On a donc par exemple

$$\partial \|\cdot\|_1 \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\} \text{ et } \partial \|\cdot\|_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \left\{ \begin{pmatrix} 1 \\ g \end{pmatrix} : |g| \leq 1 \right\}$$

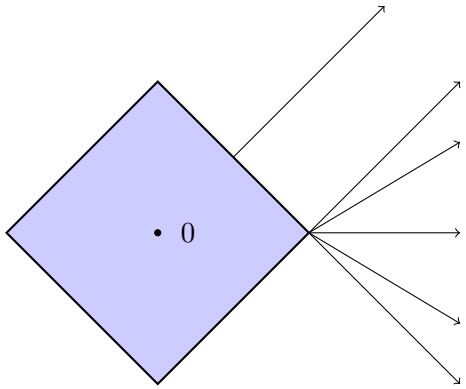

FIGURE 3 – Sous-différentielle de la norme $\|\cdot\|_1$ en dimension $N = 2$ en $(1/2, 1/2)^\top$ et $(1, 0)^\top$.

Lemme 3.11. *Soit $x^* \in \mathbb{R}^N$ et $1 \leq s \leq N$. On suppose que x^* est s-sparse et on note $I = \mathrm{supp}(x^*)$. On a, pour $x^* \neq 0$,*

$$\partial \|\cdot\|_1 (x^*) = \{z \in \mathbb{R}^N : \mathrm{sign}(z_i) = x_i^*, \forall i \in I \text{ et } |z_i| \leq 1, \forall i \notin I\}.$$

Si $x^* = 0$, $\partial \|\cdot\|_1 (x^*) = B_\infty^N$.

Démonstration. On utilise la caractérisation de la sous-différentielle pour $x^* \neq 0$:

$$\partial \|\cdot\|_1 (x^*) = \{g \in S_\infty^{N-1} : \langle g, x^* \rangle = \|x^*\|_1\}.$$

On voit facilement que g est normant pour x^* si et seulement si $\mathrm{sign}(g_i) = x_i^*$ pour tout $i \in I$ vu que

$$\langle g, x^* \rangle = \sum_{i \in I} g_i x_i^* = \sum_{i \in I} |x_i^*| = \|x\|_1$$

seulement si cette condition est satisfaite. Les coordonnées de g sur I^c sont elles choisie arbitrairement en assurant que $\|g\|_\infty \leq 1$. ∎

En d'autre termes, $\partial \|\cdot\|_1 (x^*)$ est isomorphe à une sphère S_∞^{N-s-1} en dimension $\mathbb{R}^{N-s}$. En particulier, plus x^* est sparse plus $\partial \|\cdot\|_1 (x^*)$ est riche. Les cas extrêmes étant donnés par :

$$\partial \|\cdot\|_1 (0) = B_\infty^N, \text{ ou } \partial \|\cdot\|_1 ((1, 0, \cdots, 0)) = \{1\} \times S_\infty^{N-2} \text{ et } \partial \|\cdot\|_1 ((1)_{i=1}^N) = \{(1)_{i=1}^N\}.$$

Dans le cas 1-sparse $x^* = (1, 0, \cdots, 0)$, on retrouve presque toute la sphère duale S_∞^{N-1} (seule la première coordonnée est choisie égale à 1) et dans le deuxième cas d'un vecteur *diagonal (ou well-spread)*, $\|\cdot\|_1$ est différentiable en $(1)_{i=1}^N$ et donc sa sous-différentielle est réduite à un point, qui est le gradient de la norme $\|\cdot\|_1$ en ce point.

On peut obtenir un résultat similaire pour la sous-différentielle de la norme nucléaire en une matrice de faible rang. Les notion de support et de signe peuvent être adaptées au cas matriciel.

Théorème 3.12. *Soit $A \in \mathbb{R}^{u \times v}$ où $u \geq v$. On considère une SVD de A : $A = UDV^\top$ où $D = \mathrm{diag}(\sigma_A) \in \mathbb{R}^{u \times v}$ où $\sigma_A = (\sigma_1, \ldots, \sigma_{u \wedge v})$ est le spectre de A tel que $\sigma_1 \geq \ldots \geq \sigma_{u \wedge v}(A)$. On suppose que A est de rang r. On considère la décomposition par blocks*

$$U = \begin{bmatrix} U^{(1)} & U^{(2)} \end{bmatrix} \in \mathcal{O}(u), D = \begin{bmatrix} \mathrm{diag}(\sigma_1, \ldots, \sigma_r) & 0_{r,v-r} \\ 0_{u-r,r} & 0_{u-r,v-r} \end{bmatrix} \text{ et } V = \begin{bmatrix} V^{(1)} & V^{(2)} \end{bmatrix} \in \mathcal{O}(v)$$

où $U^{(1)} \in \mathbb{R}^{u \times r}, V^{(1)} \in \mathbb{R}^{v \times r}$. La sous-différentielle de $\|\cdot\|_{S_1}$ en A est

$$\partial \|\cdot\|_{S_1} (A) = \left\{U^{(1)} {V^{(1)}}^\top + U^{(2)} W {V^{(2)}}^\top : W \in \mathbb{R}^{u-r, v-r}, \|W\|_{S_\infty} \leq 1\right\}$$

Pour faire l'analogie avec la sous-différentielle de la norme ℓ_1, on peux dire que le terme $U^{(1)}V^{(1)\top}$ joue le rôle du "signe" de A alors que le terme $U^{(2)}WV^{(2)\top}$ est le terme (induisant la richesse de la sous-différentielle) qui correspond aux matrices ayant un "support" différent de celui de A : on a pour tout $W \in \mathbb{R}^{u-r,v-r}$, $\langle A, U^{(2)}WV^{(2)\top}\rangle = 0$. On peut voir l'analogie en réécrivant la matrice A dans la base $(u_i v_j^\top)_{i,j}$ associée à une SVD de A :

$$A = UDV^\top = \sum_i \sigma_i u_i v_i^\top$$

où les u_i et v_i sont les colonnes de U et V respectivement. Ainsi, en regardant $(u_i v_i^\top)_i$ comme une famille orthogonale de matrice de rang 1 dans laquelle A se représente naturellement (on remarque que $\langle A, u_i v_j^\top\rangle = 0$ quand $i \neq j$ et $\langle A, u_i v_i^\top\rangle = \sigma_i$), on peut écrire tout sousgradient de $\|\cdot\|_{S_1}$ en A comme

$$\sum_{i=1}^r u_i v_i^\top + U^{(2)}WV^{(2)\top}$$

pour $W \in \mathbb{R}^{u-r,v-r}$ tel que $\|W\|_{S_\infty} \leq 1$ où "$U^{(2)}WV^{(2)\top}$" est vu comme un terme dont le support est différent de A.

Avant de démontrer Théorème 3.12, on introduit la notion suivante : on dit qu'une norme $\|\cdot\|$ sur $\mathbb{R}^{u\times v}$ est **orthogonalement invariante** quand pour tout $A \in \mathbb{R}^{u\times v}$, $U \in \mathcal{O}(u)$ et $V \in \mathcal{O}(v)$, on a $\|UAV\| = \|A\|$. Dans ce cas, on peut montrer qu'il existe une norme inconditionnelle $\phi(\cdot)$ sur $\mathbb{R}^{u\wedge v}$ telle que $\|A\| = \phi(\sigma_A)$ où $\sigma_A \in \mathbb{R}^{u\times v}$ est le spectre de A. On rappelle qu'une norme est **inconditionnelle** quand

$$\phi(\epsilon_1 x_1, \ldots, \epsilon_{u\wedge v} x_{u\wedge v}) = \phi(x_1, \ldots, x_{u\wedge v})$$

pour tout signes $\epsilon_1, \ldots, \epsilon_{u\wedge v} \in \{\pm 1\}$.

Pour démontrer Théorème 3.12, on va utiliser le résultat suivant sur la différentiation directionnelle d'une norme orthogonalement invariante :

$$\lim_{\epsilon\to 0^+} \frac{\|A+\epsilon R\| - \|A\|}{\epsilon} = \max_{d\in\partial\phi(\sigma_A)} \sum_{i=1}^{u\wedge v} d_i u_i v_i^\top \tag{3.6}$$

où $A = UDV^\top$, $U = [u_1|\cdots|u_u]$, $V = [v_1|\cdots|v_v]$ et $D = \mathrm{diag}(\sigma_A)$. On rappelle qu'on a toujours

$$\lim_{\epsilon\to 0^+} \frac{\|A+\epsilon R\| - \|A\|}{\epsilon} = \max_{G\in\partial\|\cdot\|(A)} \langle R, G\rangle. \tag{3.7}$$

On démontre maintenant le lemme suivant.

Lemme 3.13. *Soit $\|\cdot\|$ une norme sur $\mathbb{R}^{u\times v}$ orthogonalement invariante. On note $\phi(\cdot)$ une norme inconditionnelle sur $\mathbb{R}^{u\wedge v}$ telle que pour tout $A \in \mathbb{R}^{u\times v}$, $\|A\| = \phi(\sigma_A)$ où σ_A est le spectre de A. Soit $A \in \mathbb{R}^{u\times v}$, on a*

$$\partial\|\cdot\|(A) = \mathrm{conv}\big\{U\Sigma V^\top : A = UDV^\top, \Sigma = \mathrm{diag}(\sigma), \sigma \in \partial\phi(\sigma_A)\big\}$$

où toutes les SVD de A sont à considérer dans la construction de l'ensemble de droite.

Démonstration. On note $S(A) = \{U\Sigma V^\top : A = UDV^\top, \Sigma = \mathrm{diag}(\sigma), \sigma \in \partial\phi(\sigma_A)\}$. Soit $G \in \mathrm{conv}(S(A))$. Montrons que G est un sous-gradient de $\|\cdot\|$ en A. On rappelle que si $A \neq 0$ alors

$$\partial\|\cdot\|(A) = \big\{G \in S_* : \langle G, A\rangle = \|A\|\big\}$$

où S_* est la sphère unité de la norme duale de $\|\cdot\|$. Il suffit donc de montrer que $\langle G, A\rangle = \|A\|$ et que pour tout R tel que $\|R\| \leq 1$, on a $\langle G, R\rangle \leq 1$.

On écrit $G = \sum_i \lambda_i U_i \Sigma_i V_i^\top$ où $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$, $\Sigma_i = \mathrm{diag}(\sigma_i)$ où $\sigma_i \in \partial\phi(\sigma_A)$ et $A = U_i D V_i^\top$ est une SVD de A. On a

$$\langle G, A\rangle = \sum_i \lambda_i \langle U_i \Sigma_i V_i^\top, U_i D V_i^\top\rangle = \sum_i \lambda_i \langle \sigma_i, \sigma_A\rangle = \sum_i \lambda_i \phi(\sigma_A) = \phi(\sigma_A) = \|A\|$$

car $\langle \sigma_i, \sigma_A\rangle = \phi(\sigma_A)$ vue que $\sigma_i \in \partial\phi(\sigma_A)$.

Pour le second point, on prend $R \in \mathbb{R}^{u\times v}$ tel que $\|R\| \leq 1$. On considère σ_R le spectre de R. On a par l'inégalité de trace de von Neumann,

$$\langle G, R\rangle = \sum_i \lambda_i \langle U_i \Sigma_i V_i^\top, R\rangle \leq \sum_i \lambda_i \langle \sigma_i, \sigma_R\rangle \leq 1$$

car $\sigma_i \in S_*$ et $\sigma_R \in B$, la boule unité de $\|\cdot\|$, donc $\langle \sigma_i, \sigma_R\rangle \leq 1$. On en déduit donc que $G \in \partial \|\cdot\| (A)$.

Réciproquement, on suppose qu'il existe $G \in \partial \|\cdot\| (A)$ et $G \notin \mathrm{conv}(S(A))$. Par un argument de séparation des ensemble de convexe, il existe R tel que $\langle R, H\rangle < \langle R, G\rangle$ pour tout $H \in S(A)$. On a donc

$$\max_{H\in S(A)} \langle R, H\rangle < \max_{G\in\partial\|\cdot\|(A)} \langle R, G\rangle$$

alors

$$\max_{\sigma\in\partial\phi(\sigma_A)} \sum_i \sigma_i u_i^\top R v_i < \max_{G\in\partial\|\cdot\|(A)} \langle R, G\rangle.$$

Ceci contredit les égalités (3.6) et (3.7) ∎

Preuve du Théorème 3.12 On applique le Lemme 3.13 dans le cas où $\phi(\sigma) = \|\sigma\|_1$. Soit A de rang r et on considère une SVD de $A = UDV^\top$ avec la décomposition par blocks

$$U = \begin{bmatrix} U^{(1)} & U^{(2)} \end{bmatrix} \in \mathcal{O}(u), D = \begin{bmatrix} \mathrm{diag}(\sigma_1, \ldots, \sigma_r) & 0_{r,v-r} \\ 0_{u-r,r} & 0_{u-r,v-r} \end{bmatrix} \text{ et } V = \begin{bmatrix} V^{(1)} & V^{(2)} \end{bmatrix} \in \mathcal{O}(v)$$

où $U^{(1)} \in \mathbb{R}^{u\times r}, V^{(1)} \in \mathbb{R}^{v\times r}$.

Soit $G \in \partial \|\cdot\|_{S_1} (A)$. D'après le Lemme 3.13, il existe $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$ et $A = U_i D V_i^\top$ des SVD de A tels que

$$G = \sum_i \lambda_i U_i \Sigma_i V_i^\top$$

où $\Sigma_i = \mathrm{diag}(\sigma_i)$ et $\sigma_i \in \partial \|\cdot\|_1 (\sigma_A)$. Comme

$$\partial \|\cdot\|_1 (\sigma_A) = \{g \in \mathbb{R}^{u\wedge v} : g_i = 1, i = 1, \ldots, r \text{ et } |g_i| \leq 1, i \geq r+1\},$$

si $(U_i)_j$ et $(V_i)_j$ sont les colonnes de U_i et V_i alors

$$G = \sum_i \lambda_i \Big(\sum_j \sigma_{ij} (U_i)_j (V_i)_j^\top \Big) = U^{(1)} V^{(1)\top} + \sum_i \lambda_i U_i^{(2)} W_i V_i^{(2)\top}$$

où les W_i sont des matrices diagonale de taille $(u-r, v-r)$ dont les éléments diagonaux sont dans $[-1, 1]$. De plus, pour tout i, on peut trouver des matrices orthogonales $Y_i \in \mathcal{O}(u-r)$ et $Z_i \in \mathcal{O}(v-r)$ telles que $U_i^{(2)} = U^{(2)} Y_i$ et $V_i^{(2)} = V^{(2)} Z_i$. On a donc

$$G = U^{(1)} V^{(1)\top} + \sum_i \lambda_i U^{(2)} Y_i W_i Z_i^\top V^{(2)\top} = U^{(1)} V^{(1)\top} + U^{(2)} W V^{(2)\top}$$

où $W = \sum_i \lambda_i Y_i W_i Z_i^\top$. De plus, la plus grande valeur singulière de W est telle que

$$\sigma_1(W) = \sigma_1\Big(\sum_i \lambda_i Y_i W_i Z_i^\top\Big) \leq \sum_i \lambda_i \sigma_1(Y_i W_i Z_i \top) = \sum_i \lambda_i \sigma_1(W_i) \leq 1.$$

■

Une autre façon de caractériser la sous-différentielle $\partial \|\cdot\|_{S_1}(A)$ pour faire mieux ressortir la notion de "support" associée à une matrice est de l'écrire sous la forme suivante. Pour $A \in \mathbb{R}^{u\times v}$, on peut voir que A est de rang r si et seulement si il existe deux sous-espaces vectoriels $I \subset \mathbb{R}^u$ et $J \subset \mathbb{R}^v$ de dimension r tels que $A = P_I A P_J$ où P_I est la projection sur I et P_J est la projection sur J. Dans ce cas, $\partial \|\cdot\|_{S_1}(A)$ est l'ensemble de tout les $G \in \mathbb{R}^{u\times v}$ tels que pour tout $B \in \mathbb{R}^{u\times v}$,

1. $\langle G, A\rangle = \|A\|_{S_1}$ – càd G est normant pour A
2. $\langle G, P_{I^\perp} B P_{J^\perp}\rangle = \|P_{I^\perp} B P_{J^\perp}\|_{S_1}$ – càd G est normant pour toute matrice de "support" disjoint de A
3. $\langle G, P_I B P_{J^\perp}\rangle = 0$ et $\langle G, P_{I^\perp} B P_J\rangle = 0$ – càd dire le "support" de G est restreint à "$(I, J) \cup (I^\perp, J^\perp)$".

On peut aussi caractériser tout sous-gradient de $\partial \|\cdot\|_{S_1}(A)$ en l'écrivant dans la base $(u_i v_j^\top)_{i,j}$ de la SVD de $A = UDV^\top$ où les u_i et v_j sont les colonnes de U et V. En effet, $G \in \partial \|\cdot\|_{S_1}(A)$ si et seulement si

$$G = \sum_{i=1}^{r} u_i v_i^\top + \sum_{i=1}^{u\wedge v} u_i W v_i^\top \tag{3.8}$$

où $\|W\|_{S_\infty} \leq 1$.

En prenant n'importe quelle expression des sous-gradients de $\|\cdot\|_{S_1}$ en une matrice A^* de rang r, on voit que plus r est petit plus la sous-différentielle de S_1 en A^* est riche. Ce qui est la propriété recherchée pour avoir $\Delta_{r^*}(A^*) > 0$.

Conclusion : La sous-différentielle d'une fonction est plus grande en ces points de non-différentiation. C'est le cas pour les normes $\|\cdot\|_1$ et $\|\cdot\|_{S_1}$ en les vecteurs sparse et les matrices de rang faible. C'est une propriété essentielle de ces normes pour les procédures Basis Pursuit et Nuclear Norm Minimization qui cherchent à reconstruire ce type d'objets. L'idée essentielle pour la construction de normes de régularisation induisant de la sparsité est de créer des singularités en les objets qui sont le plus sparses pour le notion de sparsité désirée.

4 Descente de gradient proximale pour la complétion de matrice

On a vu dans les sections précédentes que la procédure de minimisation de la norme nucléaire peut se réécrire comme un problème d'optimisation SDP au Théorème 2.7 et peut donc être résolu (approximativement et pour des dimensions raisonnables) numériquement. En particulier, cette approche fournit une première solution numérique aux problème de complétion de matrices et peut donc être utilisée pour la construction de systèmes de recommandation. Dans cette section, on propose d'autres solutions algorithmiques au problème de complétion de matrice qui ont l'avantage d'éviter une reformulation SDP (qui peut être difficile à résoudre en grande dimension).

On introduit maintenant deux algorithmes de descente de gradient. Dans le premier cas, on parle de *descente de gradient projeté* et dans le deuxième cas *de descente de gradient proximal*. On considère ici les mesures obtenues dans ce problème, càd chaque observation est la donnée

d'une entrée de la matrice à reconstruire. On peut alors parler d'un **masque** qui ne révèle qu'une partie de la matrice :

$$P_\Omega : \left\{ \begin{array}{ccc} \mathbb{R}^{u\times v} & \to & \mathbb{R}^{u\times v} \\ A & \to & P_\Omega(A) \end{array} \right. \quad \text{où } (P_\Omega(A))_{i,j} = \left\{ \begin{array}{cc} A_{i,j} & \text{si } (i,j)\in\Omega \\ 0 & \text{sinon} \end{array} \right.$$

où $\Omega \subset \{1,\dots,u\}\times\{1,\dots,v\}$ est l'ensemble des indices des entrées révélées. On voit en particulier que P_Ω est un opérateur de projection (i.e. $P_\Omega^2 = P_\Omega$) et qu'il est auto-adjoint (i.e. $P_\Omega^\top = P_\Omega$).

Pour le problème de complétion de matrice, on peut écrire ces observations de manière condensée : on observe $P_\Omega(A^*)$ pour un certain ensemble Ω et une certaine matrice A^* à reconstruire càd à compléter (par exemple la matrice users/items). On suppose que A^* est de faible rang. C'est l'hypothèse centrale de 'faible dimension' pour ce problème.

On peut alors considérer deux autres procédures en plus de la *rank-minimization* et de sa relaxation convexe donnée par la *nuclear norm minimization* :

1.
$$\tilde{A} \in \operatorname*{argmin}_{\operatorname{rang}(A)\le r} \|P_\Omega(A^*) - P_\Omega(A)\|_{S_2} \tag{4.1}$$

2.
$$\bar{A} \in \operatorname*{argmin}_A \left(\frac{1}{2}\|P_\Omega(A^*) - P_\Omega(A)\|_{S_2}^2 + \lambda \|A\|_{S_1} \right). \tag{4.2}$$

Dans les deux cas, on voit un terme d'adéquation aux données : $A \to \|P_\Omega(A^*) - P_\Omega(A)\|_{S_2}$ et une contrainte de rang dans le premier cas et un terme de régularisation par la norme nucléaire dans la deuxième cas.

Pour implémenter ces deux procédures, on va considérer des analogues 'matricielles' aux fonctions vectorielles dites de "*hard-thresholding*" et de "*soft-thresholding*". Dans notre cas matriciel, on verra que ces procédures agissent sur le spectre de la matrice. On commence par décrire un algorithme pour l'implémentation d'une solution (approchante) à (4.1).

Un algorithme de gradient projeté pour (4.1). Une approche classique pour la résolution numérique d'un problème d'optimisation sous contrainte est de faire une méthode de descente de gradient projeté. Pour cela on doit savoir calculer le gradient de la fonction objectif et savoir projeter sur la contrainte. Le cadre canonique d'application de cette solution algorithmique est de demander à avoir une fonction objectif convexe et différentiable et une contrainte convexe. Pour le problème (4.1), la fonction objectif est différentiable et son gradient se calcul aisément (on remarque que minimiser $A \to \|P_\Omega(A^*) - P_\Omega(A)\|_{S_2}$ ou $A \to (1/2)\|P_\Omega(A^*) - P_\Omega(A)\|_{S_2}^2$ revient au même ici) : pour $F : A \in \mathbb{R}^{u\times v} \to (1/2)\|P_\Omega(A^*) - P_\Omega(A)\|_{S_2}^2$, on a

$$\nabla F(A) = -P_\Omega^\top(P_\Omega(A^*) - P_\Omega(A)) = P_\Omega(A) - P_\Omega(A^*) \tag{4.3}$$

car $P_\Omega^2 = P_\Omega$ et $P_\Omega^\top = P_\Omega$.

En revanche la contrainte de (4.1) est l'ensemble des matrices de rang au plus r ; ce n'est donc pas une contrainte convexe et, en général, il est difficile de projeter sur un ensemble qui n'est pas convexe. Cependant, il se trouve que dans le cas particulier de l'ensemble des matrices de rang au plus r, on sait projeter sur cet ensemble. C'est l'objet de la proposition suivante. On rappelle d'abord que l'ensemble des projections de A sur $\Sigma_r = \{B : \operatorname{rang}(B) \le r\}$ est donné par

$$\operatorname*{argmin}_{B\in\Sigma_r} \|A - B\|_{S_2} \tag{4.4}$$

On montre dans le résultat suivant que cet ensemble est un singleton et qu'il est obtenu en faisant la SVD de A puis du seuillage dur sur le vecteur spectre de A.

Proposition 4.1. *Soit $A \in \mathbb{R}^{u\times v}$. On considère la décomposition en valeurs singulières de A donnée par $A = PDQ^\top$ où $P \in \mathcal{O}(u)$, $Q \in \mathcal{O}(v)$ et $D = \mathrm{diag}(\sigma_1, \cdots, \sigma_{u\wedge v}) \in \mathbb{R}^{u\times v}$ où $\sigma_1 \geq \cdots \geq \sigma_{u\wedge v} \geq 0$ sont les valeurs singulières de A. La projection de A sur l'ensemble des matrices de rang au plus r est $PD_rQ^\top$ où D_r est la matrice $u \times v$ diagonale $\mathrm{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$. En d'autres termes, on a*

$$\operatorname*{argmin}_{B:\mathrm{rang}(B)\leq s} \|A - B\|_{S_2} = \{PD_rQ^\top\}.$$

Démonstration. On sait d'après l'inégalité de von Neuman (voir Proposition 2.6) que pour tout $B \in \mathbb{R}^{u\times v}$,

$$\|A - B\|_{S_2} \geq \|\sigma_A - \sigma_B\|_2 \tag{4.5}$$

où σ_A (resp. σ_B) est le spectre de A (resp. B), càd le vecteur de taille $u \wedge v$ des valeurs singulières de A (resp . B) rangées en ordre décroissant. Par ailleurs, il y a égalité dans (4.5) ssi B est de la forme $B = P\mathrm{diag}(\sigma_B)Q^\top$. L'ensemble des solutions à (4.4) est donc un sous-ensemble de l'ensemble des matrices de la forme $B = P\mathrm{diag}(\sigma_B)Q^\top$. Il reste alors à optimiser sur le spectre σ_B. On voit que si B est de rang r alors $\|\sigma_A - \sigma_B\|_2 \geq \sum_{j>r} \sigma_i(A)\sigma_i(B)$ qui lui est atteint pour $\sigma_B = (\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)^\top$. ∎

On voit donc que pour projeter une matrice A sur l'ensemble des matrices de rang au plus r, il suffit de savoir calculer sa SVD et de seuiller son spectre en ne gardant que ses r plus grandes valeurs singulières. Il y a des algorithmes qui permettent de calculer la SVD d'une matrice $u\times v$ en $\mathcal{O}(\min(u^2v, v^2u))$ et même dans notre cas, vu qu'on ne doit calculer que les r premiers vecteurs et valeurs singuliers, on peut le faire en $\mathcal{O}(r^2 \min(u, v))$. On sait donc projeter sur $\{B : \mathrm{rang}(B) \leq r\}$ pour des dimensions pas trop grandes.

On a donc les deux ingrédients pour la construction d'un algorithme de gradient projeté associé au problème (4.1) : le gradient de la fonction objectif et l'opérateur de projection sur la contrainte. Étant donnée une suite de *steps size* $(\gamma_k)_k$ (suite décroissante de $\mathbb{R}_+^*$), on obtient l'algorithme de descente de gradient projeté suivant

$$A^{k+1} = S_r^{hard}\big(A^k - \gamma_k \nabla F(A^k))\big) = S_r^{hard}\big(A^k + \gamma_k(P_\Omega(A^*) - P_\Omega(A^k))\big) \tag{4.6}$$

où S_r^{hard} est l'opérateur de seuillage spectral dur au rang r : si $A = PDQ^\top$ où $P \in \mathcal{O}(u)$, $Q \in \mathcal{O}(v)$ et $D = \mathrm{diag}(\sigma_1, \cdots, \sigma_{u\wedge v}) \in \mathbb{R}^{u\times v}$ alors

$$S_r^{hard}(A) = PD_r^{hard}Q^\top \text{ où } D_r^{hard} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \cdots, 0).$$

Au passage, on voit que si on prend $\gamma_k = 1$ pour tout $k \in \mathbb{N}$ dans (4.6) alors $A^{k+1} = S_r^{hard}\big(P_\Omega(A^*) + P_{\Omega^c}(A^k))\big)$ où Ω^c est le complémentaire de Ω. Ainsi, l'algorithme s'obtient en remplaçant à chaque itération les entrées de A^k indexées par Ω par celle de A^* (càd à faire $P_\Omega A^k \leftarrow P_\Omega A^*$) et à faire une ACP de rang r sur la matrice obtenue. On obtient A^{k+1} et on recommence. On reconnaît un algorithme utilisé en *données manquantes* qui est appelé ACPmiss ou missMDA. Cet algorithme fournit donc une solution (algorithmique) au problème de données manquantes basée sur l'hypothèse de faible rang.

Descente de gradient proximale pour (4.2). Pour l'implémentation de (4.2), on va construire un algorithme de descente de gradient proximal. Ce type d'algorithme peut être vu comme une généralisation du gradient projeté. On commence par quelques rappels sur cet algorithme.

On rappelle qu'une fonction convexe $f : U \to \mathbb{R}$ n'est pas forcément différentiable en tout point de son domaine $U \subset \mathbb{R}^n$ un ensemble convexe (on sait qu'elle est différentiable presque

partout). Cependant en tout point $x \in U$ elle admet une **sous-différentielle**, c'est l'ensemble

$$\partial^- f(x) = \left\{ g \in \mathbb{R}^n : f(y) \geq f(x) + \left\langle g, y - x \right\rangle, \forall y \in U \right\}. \tag{4.7}$$

Un élément de la sous-différentielle $\partial^- f(x)$ est appelé un **sous-gradient**. On a déjà vu des calcul de sous-gradient dans les sections précédentes. L'autre notion dont on aura besoin est la notion d'**opérateur proximal**. Pour introduire cette notion, on peut la voir comme une généralisation des opérateurs de projection. En effet, si $K \subset \mathbb{R}^n$ et $i_K : x \in \mathbb{R}^n \to 0$ si $x \in K$ et $+\infty$ si $x \notin K$ alors on voit que l'ensemble des projection de $x \in \mathbb{R}^n$ sur K satisfait

$$\operatorname*{argmin}_{y \in K} \|x - y\|_2 = \operatorname*{argmin}_{y \in \mathbb{R}^b} \left(\frac{1}{2} \|x - y\|_2^2 + i_K(y) \right). \tag{4.8}$$

On peut alors généraliser cette notion à n'importe quelle fonction convexe et pas seulement aux indicatrices d'ensembles comme i_K. En effet, si $g : \mathbb{R}^n \to \mathbb{R}$ est une fonction convexe, l'opérateur proximal de g est l'unique solution

$$\operatorname{prox}_g : x \in \mathbb{R}^n \to \operatorname*{argmin}_{u \in \mathbb{R}^n} \left(\frac{1}{2} \|u - x\|_2^2 + g(u) \right). \tag{4.9}$$

On fera dans la suite le calcul de l'opérateur proximale de la norme S_1 pour construire un algorithme de descente de gradient proximale il se base sur l'opérateur proximal de la norme ℓ_1 qu'on donne maintenant.

Exemple 4.2. *[**Fonction proximale de la norme** ℓ_1 **et de** ℓ_0] L'opérateur proximal de $\ell_1 : x \to \|x\|_1$ est l'opérateur de seuillage doux. En effet, on voit que la fonction à optimiser définissant l'opérateur proximal de $\gamma\ell_1$ (pour $\gamma > 0$) est décomposable : pour tout $u \in \mathbb{R}^N$,*

$$\frac{1}{2} \|u - z\|_2^2 + \gamma \|u\|_1 = \sum_{j=1}^{N} \Big(\frac{1}{2}(u_j - z_j)^2 + \gamma |u_j| \Big).$$

Il suffit alors de minimiser chaque terme de cette somme. On est donc amené à trouver une solution au problème (à une variable) : pour tout $s \in \mathbb{R}$,

$$\operatorname*{argmin}_{t \in \mathbb{R}} \Big(\frac{1}{2}(s - t)^2 + \gamma |t| \Big).$$

Pour obtenir ce minimum, on étudie $\varphi(t) = (s-t)^2/2 + \gamma|t|$ sur $(-\infty, 0)$, $\{0\}$ et $(0, +\infty)$. On voit que φ atteint son minimum en $s - \gamma$ sur $(0, \infty)$ quand $s - \gamma \geq 0$ et que ce minimum est plus petit que $\varphi(0)$ sous cette condition. De même, quand $s \leq -\gamma$, φ atteint son minimum en $s + \gamma$ sur $(-\infty, 0)$ et est plus petit qu'en zéro. Finalement, quand $-\gamma \leq s \leq \gamma$, on voit que le minimum de φ est atteint en 0. On a donc

$$\operatorname*{argmin}_{t \in \mathbb{R}} \Big(\frac{1}{2}(s - t)^2 + \gamma |t| \Big) = \begin{cases} s - \gamma & si\ s \geq \gamma \\ 0 & si\ -\gamma \leq s \leq \gamma \\ s + \gamma & si\ s \leq -\gamma \end{cases} = \operatorname{sgn}(s) \big(|s| - \gamma \big)_+$$

qui est la fonction de seuillage doux pour un seuil égal à γ. La fonctions de seuillage dur étant $s \to sI(|s| \geq \gamma)$ qui peut être obtenu comme l'opérateur proximal de l'indicatrice de 0 :

$$\operatorname*{argmin}_{t \in \mathbb{R}} \Big(\frac{1}{2}(s - t)^2 + \frac{\gamma^2}{2} I(t \neq 0) \Big) = sI(|s| \geq \gamma)$$

où $I(t \neq 0)$ vaut 1 quand $t \neq 0$ et 0 sinon. On a donc pour tout $z \in \mathbb{R}^N$,

$$\boxed{\text{prox}_{\gamma\|\cdot\|_1}(z) = \left(\text{sgn}(z_j)(|z_j| - \gamma)_+\right)_{j=1}^N} \tag{4.10}$$

qui est la fonction de seuillage doux par coordonnées et

$$\boxed{\text{prox}_{(\gamma^2/2)\|\cdot\|_0}(z) = \left(z_j I(|z_j| \geq \gamma)\right)_{j=1}^N} \tag{4.11}$$

qui est la fonction de seuillage dur par coordonnées.

L'opérateur proximal a plusieurs propriété intéressantes comme celle d'être contractant. La propriété qui nous intéresse ici est la suivante car elle permet de justifier les méthodes proximales.

Proposition 4.3. *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe différentiable et $g : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe. Soit $\gamma > 0$. Il y a équivalence entre les deux assertions suivantes :*

a) x^* est solution de $\min_{x\in\mathbb{R}^n}(f(x) + g(x))$

b) $x^* = \text{prox}_{\gamma g}(x^* - \gamma\nabla f(x^*))$.

Démonstration. Quitte à remplacer $f + g$ par $\gamma f + \gamma g$ dans *a)*, on voit qu'on peut prendre $\gamma = 1$. On montre d'abord que pour tout $z \in \mathbb{R}^n$, on a l'équivalence

$$u^* = \text{prox}_g(z) \text{ ssi } z - u^* \in \partial^- g(u^*). \tag{4.12}$$

En effet, par définition de l'opérateur proximal, on voit que $u^* = \text{prox}_g(z)$ ssi $0 \in (u^* - z) + \partial^- g(u^*)$ et donc (4.12) est bien vérifié. On a donc la suite d'équivalence suivante :

$$x^* = \text{prox}_g(x^* - \nabla f(x^*)) \Leftrightarrow x^* - \nabla f(x^*) - x^* \in \partial^- g(x^*) \Leftrightarrow 0 \in \nabla f(x^*) + \partial^- g(x^*).$$

Ce qui prouve le résultat. ∎

La Proposition 4.3 montre que pour trouver une solution à un problème de minimisation de la somme de deux fonctions convexes dont au moins une est différentiable (l'autre ne doit pas l'être forcément) alors il suffit de trouver un point fixe à la fonction $x \to \text{prox}_{\gamma g}(x - \gamma\nabla f(x))$. La Proposition 4.3 fait donc le lien entre un problème d'optimisation convexe et un problème de point fixe. En particulier, on peut construire un algorithme de point fixe pour résoudre un problème du type $\min_{x\in\mathbb{R}^n}(f(x) + g(x))$. Dans notre cas, pour trouver $x^* = G(x^*)$ où $G : x \to \text{prox}_{\gamma g}(x - \gamma\nabla f(x))$, cet algorithme s'écrit $x_{k+1} = G(x_k)$ et on sait qu'il converge bien vers un $x^* = G(x^*)$ par exemple quand G est contractant (i.e. $|G(u) - G(v)| < \|u - v\|_2, \forall u, v \in \mathbb{R}^n$). C'est cet algorithme qu'on va développer aussi ici sauf qu'à chaque itération on change la valeur de γ. On obtient l'algorithme suivant

$$A_{k+1} = \text{prox}_{\gamma_k\|\cdot\|_{S_1}}\Big(A_k + \gamma_k(P_\Omega(A^*) - P_\Omega(A_k))\Big)$$

où on a utilisé la formule du gradient de F donnée en (4.3). La dernière chose qu'il nous reste à faire pour lancer effectivement cet algorithme est de calculer l'opérateur proximal de $\gamma\|\cdot\|_{S_1}$. C'est ce qu'on fait maintenant.

Proposition 4.4. *Soit $\gamma > 0$. Soit $A \in \mathbb{R}^{u\times v}$ et on note $A = UDV^\top$ la SVD de A. L'opérateur proximal de $\gamma\|\cdot\|_{S_1}$ en A est l'opérateur de seuillage spectral doux :*

$$\text{prox}_{\gamma\|\cdot\|_{S_1}}(A) = S_\gamma^{soft}(A) = UD_\gamma V^\top$$

où $D_\gamma = \text{diag}((\sigma_1 - \gamma)_+, (\sigma_2 - \gamma)_+, \cdots, (\sigma_{u\wedge v} - \gamma)_+)$ est la matrice $u \times v$ diagonale dont les éléments diagonaux sont obtenus en appliquant la fonction de seuillage doux au niveau γ aux valeurs singulière de A.

Démonstration. Par définition,

$$\mathrm{prox}_{\gamma\|\cdot\|_{S_1}}(A) \in \operatorname*{argmin}_{B\in\mathbb{R}^{u\times v}} \Big(\frac{1}{2}\|A-B\|_{S_2}^2 + \gamma\|B\|_{S_1}\Big).$$

Par l'inégalité de von Neuman, on montre que le minimum est atteint en un point B de la forme $B = UD_BV^\top$ où $D_B = \mathrm{diag}(\sigma_B)$ et σ_B est le spectre de B. En effet, par von Neumann, on a

$$\|A-B\|_{S_2}^2 = \|A\|_{S_2}^2 + \|B\|_{S_2}^2 - 2\langle A, B\rangle \geq \|A\|_{S_2}^2 + \|B\|_{S_2}^2 - 2\langle \sigma_A, \sigma_B\rangle = \|\sigma_A - \sigma_B\|_2^2\,.$$

On a donc $\mathrm{prox}_{\gamma\|\cdot\|_{S_1}}(A) = U\mathrm{diag}(\hat{d})V^\top$ où

$$\hat{d} \in \operatorname*{argmin}_{\sigma\in\mathbb{R}^{u\wedge v}} \Big(\frac{1}{2}\|\sigma_A-\sigma\|_2^2 + \gamma\|\sigma\|_1\Big) = \mathrm{prox}_{\gamma\|\cdot\|_1}(\sigma_A).$$

En utilisant l'expression de la fonction proximale de la norme $\|\cdot\|_1$, on retrouve donc l'opérateur de seuillage doux appliqué en σ_A (qui a toutes ces coordonnées positives et ordonnées) alors

$$\mathrm{prox}_{\gamma\|\cdot\|_1}(\sigma_A) = \big((\sigma_1-\gamma)_+, (\sigma_2-\gamma)_+, \dots, (\sigma_{u\wedge v}-\gamma)_+\big)^\top.$$

On obtient bien le résultat. ∎

Une fois calculer l'opérateur proximal de $\gamma\|\cdot\|_{S_1}$ dans Proposition 4.4, on peut mettre en œuvre la méthode de descente de gradient proximal pour implémenter (4.2) :

$$A_{k+1} = S_{\gamma_k}^{soft}\Big(A_k - \gamma_k\big(P_\Omega(A_k) - P_\Omega(A^*)\big)\Big) \tag{4.13}$$

où $S_{\gamma_k}^{soft}$ est la fonction de seuillage spectral doux au niveau γ_k et $(\gamma_k)_k$ est une famille de step size.

On voit donc que la seule différence entre l'algorithme de descente de gradient projeté associé à (2.1) obtenu en (4.6) et l'algorithme de descente de gradient proximal obtenu en (4.13) pour approcher une solution de (4.2) est qu'au lieu de faire du seuillage dur on fait du seuillage moux. Les deux algorithmes ont donc le même coup computationnel. Ils s'implémentent tous les deux sans avoir recours aux SDP mais demande de faire des SVD (ou SVD seuillées) à chaque itération ; ce qui peut être coûteux.

A retenir : On peut projeter sur l'espace des matrices de faibles rang et calculer l'opérateur proximale de la norme nucléaire. On peut alors construire un algorithme de complétion de matrices en faisant une méthode de gradient projeté ou de gradient proximal.

5 Algorithme de factorisation de matrices et incorporation de meta-données pour les systèmes de recommandation.

L'idée de l'approche par factorisation de matrice est de considérer l'espace $\mathbb{R}^d$ comme un espace latent dans lequel on va représenter à la fois les *users* et les *items* par des vecteurs $(p_i)_{1\leq i\leq u}$ et $(q_j)_{1\leq j\leq v}$ où

1. p_i est la représentation du user i dans l'espace latent $\mathbb{R}^d$
2. q_j est la représentation de l'item j dans l'espace latent $\mathbb{R}^d$

tel que le produit scalaire $\langle p_i, q_j \rangle_{\mathbb{R}^d}$ mesure l'affinité entre l'*item* j et le *user* i.

Par exemple, un espace latent pour les données Netflix est représenté dans la Figure 4.

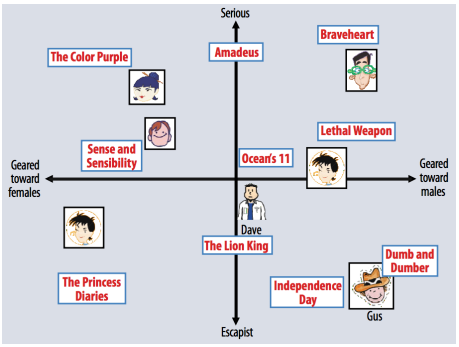

FIGURE 4 – Espace latent, ici $\mathbb{R}^2$, dans lequel sont représentés au même endroit les users et les items (ici les films dans le cas des données Netflix).

Factorisation de la matrice users/items pour la construction d'un système de recommandations. On dispose de données de notations d'item par des users de la forme $\{(user_i, item_j, grade_{ij}) : (i,j) \in \Omega\}$ où $\Omega \subset \{1,\dots,u\} \times \{1,\dots,v\}$. On cherche à construire les deux familles de vecteurs $(p_i)_{1\leq i\leq u}$ et $(q_j)_{1\leq j\leq v}$ représentant les users et les items dans l'espace latent $\mathbb{R}^d$.

L'idée est de construire $(p_i)_{1\leq i\leq u}$ et $(q_j)_{1\leq j\leq v}$ tels que les mesures de similarité $\langle p_i, q_j \rangle$ coïncident le plus possible avec les notes observées, càd, on cherche à avoir

$$\langle p_i, q_j \rangle \approx grade_{ij} \quad \forall (i,j) \in \Omega$$

Toutes les autres valeurs $\langle p_i, q_j \rangle$ pour $(i,j) \notin \Omega$ sont ensuite utilisées pour compléter la matrice users/items. Le problème de factorisation de matrice pour la construction d'un système de recommandation revient alors à écrire la matrice A^* comme le produit de deux matrice comme représenté dans la Figure 5.

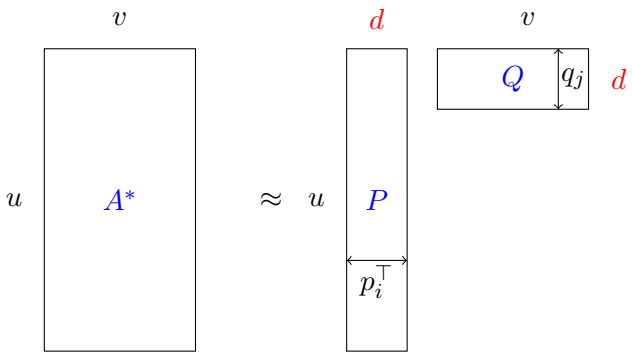

FIGURE 5 – Représentation matricielle du problème de factorisation de la matrice A^* en le produit PQ.

On voit en particulier que l'hypothèse de faible rang se retrouve ici de manière explicite vu que le produit PQ est au plus de rang d. Ainsi en cherchant à approcher A^* par le produuit PQ, on est en particulier en train d'approcher A^* par une matrice de faible rang. Cette approximation a une chance d'être bonne uniquement si A^* est bien approximativement de faible rang.

Comme indiqué plus haut, on ne connaît que quelques entrées de la matrice A^*. C'est sur ces entrées que nous allons calculer les deux familles de vecteurs $(p_i)_{1\leq i\leq u}$ et $(q_j)_{1\leq j\leq v}$. On se donne alors un critère à minimiser :

$$(p_i)_{1\leq i\leq u}, (q_j)_{1\leq j\leq v} \longrightarrow \sum_{(i,j)\in\Omega} \left(grade_{ij} - \langle p_i, q_j\rangle\right)^2 = \|P_\Omega(A^* - PQ)\|_2^2$$

où $A^* = \left(grade_{ij}\right)_{i,j}$, $P = (p_i^\top)_{1\leq i\leq u}$ et $Q = (q_i)_{1\leq i\leq v}$. En version matricielle, la fonction objective s'écrit

$$P \in \mathbb{R}^{p\times d}, Q \in \mathbb{R}^{d\times q} \to \|P_\Omega(G - PQ)\|_2^2 \,. \tag{5.1}$$

C'est cette fonction objective qu'on va chercher à minimiser. Néanmoins, le produit PQ étant invariant par changement $(P, Q) \to (\alpha P, Q/\alpha)$ pour tout $\alpha \neq 0$, on risque d'avoir des problèmes de normalisation juste à minimiser (5.1). On va donc forcer la norme des vecteurs (p_i) et (q_j) a être aussi petite en régularisant par leur norme : la fonction objective qu'on va minimiser est alors

$$P \in \mathbb{R}^{p\times d}, Q \in \mathbb{R}^{d\times q} \to f(P, Q) := (1/2)\,\|P_\Omega(A^* - PQ)\|_2^2 + \lambda\left(\|P\|_2^2 + \|Q\|_2^2\right) \tag{5.2}$$

où $\lambda > 0$ est un paramètre de régularisation à choisir.

Cependant, minimiser cette fonction f en (P, Q) conjointement est difficile car la fonction f n'est pas convexe. Néanmoins, on peut remarquer que

1. pour un P^* fixé, $Q \to f(P^*, Q)$ est convexe et son gradient est $\partial_2 f(P^*, Q) = (P^*)^\top P_\Omega(A^* - P^*Q) + 2\lambda P$
2. pour un Q^* fixé, $P \to f(P, Q^*)$ est convexe et son gradient est $\partial_1 f(P, Q^*) = P_\Omega(A^* - PQ^*)(Q^*)^\top + 2\lambda Q$

On peut donc alterner des étapes de descentes de gradient en P puis en Q. C'est cette approche qu'on appell parfois ARLS pour Alternating ridge least Squares.

Data: $\{A^*_{ij}, (i, j) \in \Omega\}$
Init : $P_0 \in \mathbb{R}^{u\times d}$, $Q_0 \in \mathbb{R}^{d\times v}$ et deux suites de step sizes $(\eta_k)_k$ et $(\mu_k)_k$
Output : $P \in \mathbb{R}^{u\times d}$ et $Q \in \mathbb{R}^{d\times v}$ telles que $A^* \approx PQ$
while *stopping criteria* **do**
    $P_{k+1} = P_k - \eta_k \partial_1 f(P_k, Q_k)$
    $Q_{k+1} = Q_k - \mu_k \partial_2 f(P_{k+1}, Q_k)$
end

FIGURE 6 – Algorithme de descente de gradient alterné pour la représentation de la matrice A^* en un produit PQ.

En sortie de l'algorithme 8, on obtient deux matrices P et Q. On peut utiliser ces matrices pour construire un système de recommendation, il suffit alors de calculer le produit PQ et d'identifier les entrées les plus grandes sur toutes les lignes. On peut aussi clusteriser les users et items dans l'espace latent simplement en 'clusterisant' les lignes de P pour trouver des clusters de users et les colonnes de Q pour trouver les clusters d'items. On peut se représenter cette idée avec la Figure 7.

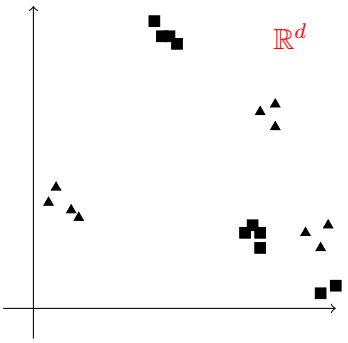

FIGURE 7 – En plus de la construction d'un système de recommandation, la représentation des users et des items dans l'espace latent peut aussi être utile pour clusteriser les users et les items en fonction de leurs notes. On peux ainsi trouver des groupes de users ayant les mêmes goûts relativement à leurs appréciations sur ces items (et similairement pour les items).

Non-Negative Matrix Factorization (NMF). Dans l'approche standard de factorisation de matrice décrite plus haut, les coordonnées des vecteurs latents p_i et q_j peuvent être négatives et donc le produit scalaire $\langle p_i, q_j \rangle$ aussi. Dans notre approche le produit scalaire correspond à une similitude entre le user et l'item. On peut alors se dire qu'un produit scalaire négatif signifie que le user ne va probablement pas apprécier cet item. Ce qui reviens à ramener ce produit scalaire à la note la plus basse, càd à 1 dans le cas de Netflix. Une autre stratégie est d'imposer la positivité sur les coordonnées des vecteurs p_i et q_j. On ajoute alors une contrainte de positivité sur les entrées des matrices P et Q dans notre problème d'optimisation. On obtient alors la procédure de minimisation suivante

$$\min_{(P,Q)\in\mathbb{R}^{u\times d}\times\mathbb{R}^{d\times v}} (\|P_\Omega(A^* - PQ)\|_2 : P \geq 0, Q \geq 0)$$

où on note '$P \geq 0$' quand $P_{ij} \geq 0, \forall (i,j)$.

De même que dans ARLS, la fonction objective n'est pas convexe en (P,Q) mais elle l'est en P à Q fixe et aussi en Q à P fixe. On va donc alterner des descentes en P et puis en Q. Sauf que contrairement à l'algorithme précédent, on doit respecter les contraintes '$P \geq 0$' et '$Q \geq 0$'. Pour ce faire on a le choix entre faire du **gradient projeté** ou faire du **gradient conditionel (appelé aussi algorithme de Frank-Wolfe)**. Comme la projection sur l'espace des matrices à entrées positives est facile, c'est proj : $P \to (\max(P_{ij}, 0))_{i,j}$, on va faire du gradient projeté.

Data: $\{A^*_{ij}, (i,j) \in \Omega\}$
Init : $P_0 \in \mathbb{R}^{u\times d}$, $Q_0 \in \mathbb{R}^{d\times v}$ et deux suites de step sizes $(\eta_k)_k$ et $(\mu_k)_k$
Output : $P \in \mathbb{R}^{u\times d}$ et $Q \in \mathbb{R}^{d\times v}$ telles que $A^* \approx PQ$
while *stopping criteria* **do**
    $R_{k+1} = P_k - \eta_k \partial_1 f(P_k, Q_k)$
    $P_{k+1} = \text{proj}(R_{k+1})$
    $S_{k+1} = Q_k - \mu_k \partial_2 f(P_{k+1}, Q_k)$
    $Q_{k+1} = \text{proj}(S_{k+1})$
end

FIGURE 8 – Algorithme de descente de gradient projeté alterné pour la représentation de la matrice A^* en un produit PQ où P et Q ont toutes leurs entrées positives.

On peut trouver d'autres algorithmes pour le NMF dans l'article de Lee et Seung, NIPS 2001.

Ajout de méta-données dans l'algorithme de factorisation de matrices. Dans de nombreux problèmes, en plus des données d'appréciations de users sur des items on peut avoir des données supplémentaires qu'on appelle les méta-données (car elles viennent en plus des données de bases de notations). On peut, par exemple, rencontrer des données sur les users (age, sexe, CSP, etc.), sur les items (prix, catégories, etc.) ou des relations entre users (membres de communautés,likes, etc.) et de même des relations entre items (même catégories, date de sortie d'un film, présence d'acteurs, etc.). On imagine que l'ajout de ces données lors de la construction d'un système de recommandations ne peut que améliorer la qualité de ses recommandations. L'objectif de cette section est de donner des idées de modélisations pour ces données et de voir comment les inclure dans les problèmes d'optimisation pour que la construction de notre système de recommandation en tienne compte.

L'idée derrière la factorisation de matrice pour le construction de systèmes de recommandations évoquée plus haut est d'expliquer les notes observées $(A^*_{ij} : (i,j) \in \Omega)$ par la construction de vecteurs (p_i) représentant les users et de vecteurs (q_j) représentant les items dans le même espace latent $\mathbb{R}^d$ tels que pour toutes les notes observées, càd pour tout $(i,j) \in \Omega$, on a $A^*_{ij} \approx \langle p_i, q_j \rangle$. Les métadonnées peuvent aussi aider à expliquer les notes $(A^*_{ij} : (i,j) \in \Omega)$. On va donc ajouter au modèle $A^*_{ij} \approx \langle p_i, q_j \rangle$ les meta-données. On considère des métadonnées sous la forme suivante :

a) x_i est le vecteur des features du user i, pour tout $i \in [u]$ (auto-encodage, embedding, one-hot encoding, etc.) ;
b) z_j est le vecteur des features de l'item j, pour tout $j \in [v]$ (auto-encodage, embedding, one-hot encoding, etc.) ;
c) $U \in \mathbb{R}^{u \times u}$ une matrice users/users qui représente les affinités/influences entre users (par exemple, $U_{ij} = 1$ si i aime bien j et $U_{ij} = -1$ quand i n'aime pas j ou le nombre de messages 'positifs' échangés entre i et j, ou le nombre de message de j retweeté par i, etc.). La matrice users/users est ici pour indiquer les influences entre users ; on souhaite encoder dans cette matrice une information du type 'si le user r aime bien ce film alors ça risque aussi d'être le cas pour le user i'. Ainsi si j est un fort 'influenceur' alors U_{ij} sera un grand nombre positif pour une majorité d'autres users i sur lesquels j a de l'influence,
d) $I \in \mathbb{R}^{v \times v}$ une matrice items/items représentant les affinités entre items (par exemple, basé sur le casting, le producteur, l'année de sortie, le genre de deux films, etc.).

L'idée est maintenant d'utiliser ces données pour expliquer au mieux les notes observées. La première approche la plus naturelle est de faire des modèles linéaires. On va alors expliquer les $(A^*_{ij} : (i,j) \in \Omega)$ de la manière suivante : pour tout $(i,j) \in \Omega$

$$A^*_{ij} \approx \langle p_i, q_j \rangle + (a_i + \langle b_i, x_i \rangle) + (c_j + \langle d_j, y_j \rangle) + \alpha \sum_{r=1}^{u} U_{ir} \langle p_r, q_j \rangle + \beta \sum_{s=1}^{v} I_{js} \langle p_i, q_s \rangle. \tag{5.3}$$

Le modèle (5.3) tient en compte les influences entre users dans le terme '$\sum_{r=1}^{v} U_{ir} \langle p_r, q_j \rangle$' (si r influence i alors U_{ir} est un grand nombre positif et si r aime bien l'item j alors $\langle p_r, q_j \rangle$ sera aussi un grand nombre positif ; in fine, $U_{ir} \langle p_r, q_j \rangle$ sera un grand nombre positif et donc la note que i donnera à j sera d'autant plus grande), entre items dans le terme '$\sum_{s=1}^{u} I_{js} \langle p_i, q_s \rangle$', les features propres aux users dans le terme '$a_i + \langle b_i, x_i \rangle$' et les features propres aux items dans le terme '$c_j + \langle d_j, y_j \rangle$'.

Les paramètres du modèle (5.3) qu'on doit apprendre à partir de de nos données sont les vecteurs latents $(p_i)_i$ et $(q_j)_j$ et les paramètres des termes linéaires expliquant la note finale à partir des features des users et des items, càd a_i, b_i pour $i \in [u]$ et c_j, d_j pour $j \in [v]$. L'algorithme

d'apprentissage qu'on va lancer se base aussi sur des minimisations alternées entre les vecteurs (p_i) et (q_j) et à une mise à jour des coefficients a_i, b_i, c_j, d_j à chaque itérations. Ensuite on peut soit régulariser par la norme ℓ_2^d des p_i et q_j (comme dans ARLS) soit imposé une contrainte de positivité sur les coordonnées des p_i et q_j comme dans NMF. On donne ci-dessous la version régularisée de l'algorithme. On suppose que $U_{ii} = 0$ et $I_{jj} = 0$ tel que l'information $\langle p_i, q_j \rangle$ ne soit pas redondante dans le modèle (5.3). On note par $F\big((p_i, a_i, b_i)_i, (q_j, c_j, d_j)_j\big)$ la quantité

$$\sum_{(i,j)\in\Omega} \left[A^*_{ij} - \left(\langle p_i, q_j \rangle + (a_i + \langle b_i, x_i \rangle) + (c_j + \langle d_j, y_j \rangle) + \alpha \sum_{r=1}^{v} U_{ir} \langle p_r, q_j \rangle + \beta \sum_{s=1}^{u} I_{js} \langle p_i, q_s \rangle \right) \right]^2 + \lambda \left(\sum_i \|p_i\|_2^2 + \sum_j \|q_j\|_2^2 \right)$$

qu'on souhaite minimiser. A $(p_i)_i$ et $(q_j)_j$ fixes, on peut minimiser explicitement F en (a_i, b_i, c_j, d_j) vu qu'on retombe sur un problème de moindre carré.

6 Approches 'benchmark' : collaborative et content-based filterings

Les approches vues plus haut sont basées sur l'hypothèse de faible rang de la matrice users/items. C'est cette hypothèse de faible dimension sur le signal A^* à reconstruire qu'on cherche à mettre en forme soit directement dans l'espace des solutions (en cherchant la matrice de rang le plus petit parmi toutes les matrice A telle que $A_{ij} = A^*_{ij}$ pour $(i, j) \in \Omega$), soit en factorisant A^* en un produit PQ de faible rang. Cette approche qui peut paraître naturelle au vu des similitudes entre users et celles entre items n'est cependant pas la plus naturelle et il y a en fait des algorithmes qui viennent plus naturellement (et qui ont été introduites au tout début des systèmes de recommandations avant ceux introduits plus haut). Ces algorithmes originels qui sont aussi basés sur les similitudes entre users et entre items sont ceux qu'on appelle maintenant **algorithmes benchmark** car c'est bien à ces algorithmes que nous devons comparer les performances de tout nouvel algorithme de construction de systèmes de recommandations. Ces algorithmes sont les plus simples et ne nécessitent que très peu de connaissance en math pour les comprendre et les mettre en oeuvre. Ils sont donc aussi les plus rapides à s'exécuter, ce qui les rend d'autant plus 'benchmark'. On présente ces algorithmes dans cette section.

Il existe deux types d'algorithmes benchmark pour la construction de systèmes de recommandation :

1. les algorithmes de type **collaborative filtering** qui utilisent une mesure de similarité entre users
2. les algorithmes de type **content-based filtering** qui utilisent une mesure de similarité entre items.

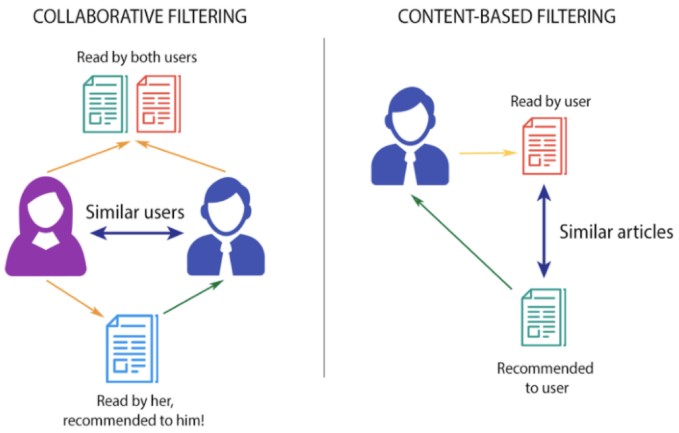

FIGURE 9 – Deux types d'approches 'benchmark' pour la construction de systèmes de recommandations. Le collaborative filtering est basé sur les similitudes entre users et l'approche content-based filtering est basée sur les similitudes entre items.

Tout l'enjeu du collaborative et du content-based filtering est de construire une mesure de similarité, soit entre users (pour l'approche collaborative filtering) soit entre items (pour l'approche content-based filtering) :

a) pour deux users $i, i' \in [u]$, on note $\mathrm{sim}(i, i')$ une mesure de similarité entre le user i et le user i'
b) pour deux items $j, j' \in [v]$, on note $\mathrm{sim}(j, j')$ une mesure de similarité entre l'item j et l'item j'.

Un système de recommandation de type 'collaborative filtering' va remplir la matrice users/items avec

$$\hat{A}_{ij} = \frac{\sum_{i' \in [u]} \mathrm{sim}(i, i') \hat{A}_{i'j}}{\sum_{i' \in [u]} \mathrm{sim}(i, i')}$$

et un système de recommednations 'content-based filtering' va la remplir avec

$$\hat{A}_{ij} = \frac{\sum_{j' \in [v]} \mathrm{sim}(j, j') \hat{A}_{ij'}}{\sum_{j' \in [v]} \mathrm{sim}(j, j')}.$$

Tout l'enjeu est donc bien de définir une mesure de similarité entre users ou entre items. Au passage, on remarque que dans les deux formules ci-dessus la definition de $\hat{A}_{ij}$ dépend des autres entrées $\hat{A}_{i'j}$ ou $\hat{A}_{ij'}$ et peut donc imaginer un certain algorithme itératif. Cependant dans la majorité des filtres collaboratif ou basé sur le contenu on ne fera qu'une seule étape et dans ce cas on ne somme que sur les données observées A^*_{ij} pour $(i, j) \in \Omega$.

La contruction de mesure de similarités se prette aussi très bien à l'ajout de métadonnées users, items, users/users ou items/items. On ne va pas entrer dans ces détails et on ne donne ici que des exemples de mesures de similarité basés sur les notes observées de la matrice users/items. Pour deux users $i, i' \in [u]$, on utilise la mesure de similarité cosinus entre les vecteurs de notations masqués des deux users :

$$\mathrm{sim}(i, i') = \frac{\langle (P_\Omega A^*)_{i\bullet}, (P_\Omega A^*)_{i'\bullet} \rangle}{\|(P_\Omega A^*)_{i\bullet}\|_2 \, \|(P_\Omega A^*)_{i'\bullet}\|_2} = \frac{\sum_{j \in N(i,i')} A^*_{ij} A^*_{i'j}}{\sqrt{\sum_{j \in N(i,i')} (A^*_{ij})^2} \sqrt{\sum_{j \in N(i,i')} (A^*_{i'j})^2}}$$

où $N(i,i') = \{j \in [v] : (i,j),(i',j) \in \Omega\}$ et $(P_\Omega A^*)_{i\bullet}$ est le i-ième vecteur ligne de $P_\Omega A^*$. Pour une mesure de similarité entre items on peut définir de la même manière une mesure de similarité cosinus.

7 Systèmes de recommandation par apprentissage profond

Une autre approche qui a reçu beaucoup d'attention ces dernières années est la construction de systèmes de recommandations par des modèles de réseaux neuronaux aussi appelé *Deep learning*. On présente un exemple d'architecture de réseau de neurones pour la construction d'un système de recommandations dans cette section. Il en existe beaucoup d'autres. Celui qu'on présente ici est probablement un des plus simples (le plus simple étant celui construisant un modéle linéaire de factorisation de matrices). On peut voir ce réseau comme une 'complexification' de l'approche par factorisation de matrice vue plus haut car il utilise des embeddings des users et des items sauf qu'au lieu de plonger users et items dans le même espace latent et d'en prendre le produit scalaire pour prédire la note, on peut plonger users et items dans deux espaces latents différents et la note sera prédite par une fonction plus compliquée qu'un simple produit scalaire.

L'embedding des users et des items transforme un user représenté par un vecteur de one-hot encoding et le transforme en un vecteur d'une certaine taille comme représenté dans la Figure 10. C'est la première étape / couche du réseau de neurones qu'on va construire.

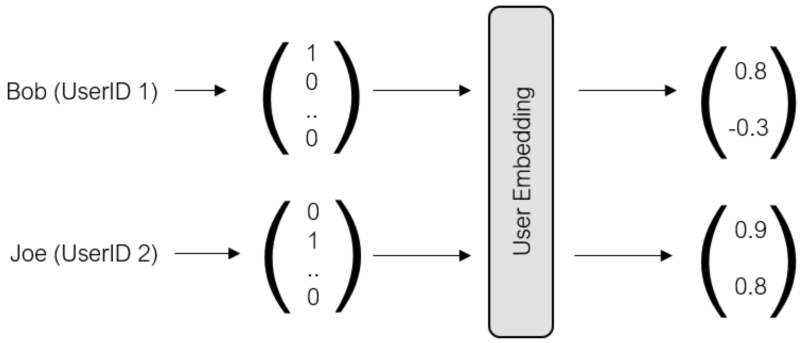

FIGURE 10 – Représentation vectorielle d'un user : '*users embedding*'. C'est une application linéaire qui prend en entrée un vecteur 'one-hot encoding' (un vecteur unité) de $\mathbb{R}^u$ et qui va dans $\mathbb{R}^d$ où d est la dimension de l'espace latent à choisir. Il y a $u \times d$ poids à apprendre dans ce layer.

L'embedding est une composante importante pour tout problème d'apprentissage. Il peut se faire à la main, c'est ce qu'on a coutume d'appeler le 'featuring' ou il se fait grâce à un noyau ou il s'apprend à partir des données. C'est cette dernière approche qu'on a prise plus haut dans le cas de la factorisation de matrices : les vecteurs p_i et q_i sont des embeddings des users et des items qu'on apprend en résolvant un problème d'optimisation construit à partir des données. C'est aussi cette approche qu'on va utiliser dans le cadre de l'apprentissage profond ; la première couche de notre réseau de neurones sera une couche de deux embeddings, un pour les users et un autre pour les items.

Le réseau de neurones qu'on va construire ci-dessous commence donc par une couche d'embeddings du user et de l'item ; il est ensuite suivi par des couches *fully connected* aussi appelé *dense layer* ou *deep fully connected*. Le fully connected layer est la structure de base des réseaux neuronaux car elle ne contient aucune structure (on dit qu'elle est 'structured agnostic'). On en représente une en Figure 11.

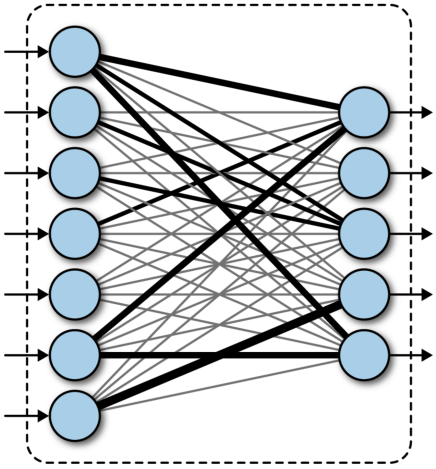

FIGURE 11 – Une couche de neurones 'fully connected' ayant 9 unités en entrées et 5 en sortie, soit un total de 45 poids qui devront être appris.

Les couches fully connected sont utilisées quand on n'a aucune connaissance a priori sur la structure du problème qu'on souhaite résoudre. C'est ce type de réseau qu'on utilise lorsqu'on ne sait pas quoi faire d'autre. Elles sont néanmoins coûteuses en temps d'apprentissage car elles ont le nombre maximum de poids à fitter et elles sont en général moins performantes que les couches dédiées à un problème structuré. Mathématiquement, une couche fully connected à m neurones en entrées et p neurones en sortie est une fonction paramétrée par la matrice de poids $W = (w_{ij} : 1 \leq i \leq p, 1 \leq j \leq m)$ qui a un vecteur en entrée $x \in \mathbb{R}^m$ renvoie un vecteur $y \in \mathbb{R}^p$ tel que pour tout $i \in [p]$

$$y_i = \sigma\left(w_{i1}x_1 + \cdots + w_{im}x_m\right)$$

où $\sigma : \mathbb{R} \to \mathbb{R}$ est une fonction d'activation comme la fonction sigmoïd $\sigma : t \to (1+\exp(-t))^{-1}$ ou la fonction ReLU (Rectified Linear Unit) $\sigma : t \to \max(0, t)$. Les couches fully connected peuvent s'enchaîner (= stack) les unes aux autres si on souhaite apprendre des structures de dépendance compliquées entre les entrées et les sorties. On représente dans la Figure 12 un réseau de 4 fully connected networks.

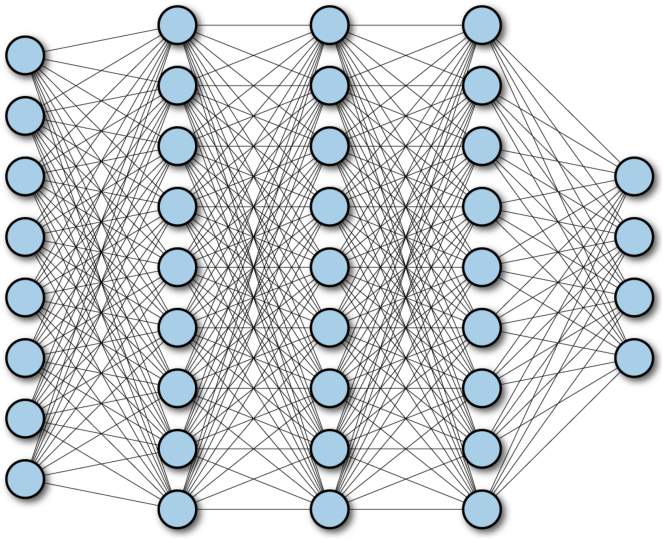

FIGURE 12 – Une architecture réseau de 4 fully connected networks avec respectivement 8 neurones en entrée, 4 en sortie et 9, 9 et 9 neurones pour les couches intermédiaires. Soit un total de $8 \times 9 \times 9 \times 9 \times 4 = 23328$ poids à apprendre.

Le réseau de neurone utilisé dans le NCF (neural collaborative filtering) est représenté dans la Figure 14. Il prend en entrée des couples (user, item) et en sortie il renvoie un vecteur de probabilité sur (ici) les deux choix possibles 'interaction' (i.e. l'output vaut 1) ou 'pas d'interaction' (càd l'output vaut 0). Ce vecteur de sortie est ensuite appliqué à une sigmoïd qui renvoie la classe la plus probable, par ici la classe 1 est renvoyée car $0.8 > 0.2$.

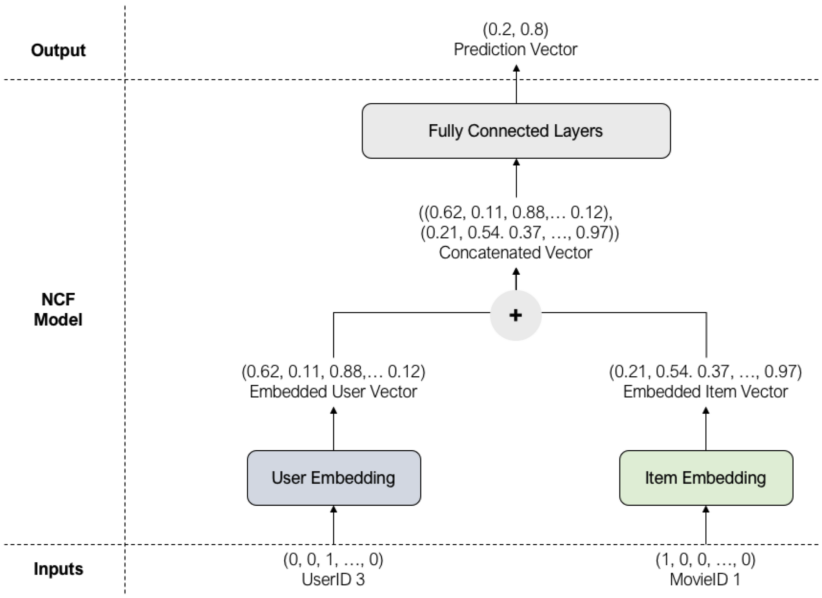

FIGURE 13 – Architecture du *Neural Collaborative Filtering* : une couche d'embedding suivie d'un réseau totalement connecté donnant le vecteur de probabilité d'appartenance au deux classes (interaction ou pas).

Maintenant qu'on a décidé la structure de réseau de neurone qu'on souhaite utiliser, on doit

apprendre les poids qui la compose à partir des données. On doit d'abord choisir une fonction de perte. Ici on prend la **Binary Cross Entropy** (BCE) qui a un batch $(user_i, item_i, y_i)_{i=1}^N$ associe une perte

$$BCE(\hat{y}, y) = -\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

où $\hat{y} = (\hat{y}_i)_{i=1}^N$ est le vecteur des prédictions en sortie du réseau pour les entrées successives $(user_i, item_i)$ pour $i = 1, \ldots, N$. La perte $BCE(\hat{y}, y)$ est une fonction de tous les poids qui compose le réseau de neurones NCF. On note ces poids w. On a donc

$$BCE(\hat{y}, y) = f(w, (user_i, item_i)_{i=1}^N) = f(w).$$

On va apprendre les poids w en minimisant la fonction $w \to f(w)$ en faisant plusieurs étapes d'une méthode d'optimisation inspirée de la descente de gradient. Le solver actuellement le plus utilisé pour apprendre les poids des réseaux de neurones est appelé ADAM (Adaptive moment estimation) qui est une méthode du premier ordre. ADAM fait la mise à jour suivante sur les poids. On pose $m_0 = 0$ et $v_0 = 0$ et on choisit $\beta_1, \beta_2, \eta, \varepsilon \in (0, 1)$, on calcul des estimés pour les deux premiers moments :

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1) \nabla f(w^{(k)}) \quad \text{and} \quad v_k = \beta_2 v_{k-1} + (1 - \beta_2)(\nabla f(w^{(k)}))^2 \,,$$

puis, on corrige le biais

$$\hat{m}_k = \frac{m_k}{1 - \beta_1^k} \quad \hat{v}_k = \frac{v_k}{1 - \beta_2^k} \,,$$

et on fait la mise à jour suivante des poids avec

$$w^{(k+1)} = w^{(k)} - \frac{\eta}{\sqrt{\hat{v}_k} + \varepsilon} \hat{m}_k. \tag{7.1}$$

Le point clef de toute méthode du premier ordre est de savoir calculer efficacement le gradient d'une fonction qui peut être très compliqué (par exemple f dépend d'au moins 23328 variables si le réseau de la Figure 12 est utilisé dans le NCF). C'est ici que l'algorithme de **backpropagation** entre en jeux. C'est un algorithme qui provient de la théorie de l'**autodifférentiation** et qui permet principalement grâce à la chain rule de calculer des gradients de fonction compliquée en un point donné. C'est cet algorithme de retropropagation qui est utilisé pour calculer les gradients pour n'importe quel solver, en particulier pour ADAM.

Finalement, il y a plein de paramètres à choisir quand on entraîne un réseau de neurones comme le nombre de couches, le nombre de neurones, les dimensions des espaces des embeddings, le nombre d'**epochs** (c'est le nombre de fois où on passe entièrement sur la base de données (découpée en batch) pour apprendre les poids du réseau), la taille des batchs, le learning rate (c'est le paramètre η dans (7.1)). Logiquement il faudrait les apprendre avec des critères de validation croisé, cependant le temps de d'apprentissage d'un seul réseau est trop couteux pour mettre en oeuvre cette stratégie. On utilise alors des approches de type 'best practice' qui nous viennent des intutions ou des pratiques des experts qui nous disent par exemple de prendre des batch size de 512, pour 5 epochs, de prendre une couche dense en entrée 16 neurones, 64 intermédiaires et 32 en sortie.

Une fois appris le réseau $f(\hat{w}, (\cdot, \cdot))$, on peut remplir la matrice users/items en requettant toutes les entrées non observées par $f(\hat{w}, (user_i, item_j))$ pour $(i, j) \notin \Omega$.

Negative sample. Il y a néanmoins un point important qu'on rencontre systématiquement pour la construction de systèmes de recommandations via un réseau neuronal. En fait, dans l'exemple qu'on a considérer plus haut, la base de données est de la forme $(user_i, item_i)_{i=1}^n$, ici il n'y a pas de note puisqu'on ne regarde que les users et items qui ont interagit. En particulier, on n'a pas d'exemple de couple (user,item) qui n'ont pas interagit par goût car on ne peut pas différencier une absence d'interaction parce que l'user n'a pas eu l'idée d'interagir avec cet item (alors que s'il l'avait eu, il aurait interagit avec) d'une absence d'interaction parce que le user ne souhaite pas interagir avec cet item. Il manque donc ce qu'on appelle des **negative sample**, càd des exemple d'interactions avec des items volontairement évitées par le user. Il faut alors trouver un moyen de générer des exemples négatifs. Sinon on ne donnera que des exemple positif à apprendre au réseau et donc il apprendra à répondre systématiquement 1 pour chaque couple $(user, item)$. La manière la plus simple quoique largement discutable est simplement de prendre au hasard des items parmi ceux avec lequels le users n'a pas interagit et de l'ajouter comme negative sample. En faisant celà, on retombe sur le problème vu précédemment sur le fait qu'on n'a pas interagit par choix ou par absence de connaissance de l'existence de ce choix. Néanmoins, c'est lapproche la plus simple et c'est celle qu'on va mettre en oeuvre (on peut aussi regarder les items les moins populaires par batch ou d'autre stratégie plus collaborative pour la création des negative sample). Ici on peut par exemple choisir 4 fois plus de negative sample que de positive sample par user pour compenser la 'mauvaise' qualité de nos negative sample.

Dropout. Il y a encore beaucoup à faire concernant la compréhension du phénomène d'over-fitting dans les réseau de neurones (comment un algorithme faisant zéro erreur sur l'echantillon d'apprentissage peut-il encore avoir de très bonne capacité à généraliser ?). Dans l'esprit (toujours persistant) d'éviter l'over-fitting des réseaux de neurones, l'idée du drop-out a fait son chemin. Le principe est de considérer de manière aléatoire une certaine proportion des unités / neurones durant l'apprentissage (à chaque étape de descente de gradient, on en choisit de nouveaux de manière alétoire) et de ne pas optimiser les poids associés à ces unités. Cela fait apparaître un paramètre supplémentaire : la proportion p d'unités à choisir de manière aléatoire à chaque itération, qu'il faut aussi choisir en amont. On représente le principe du drop-out en Figure

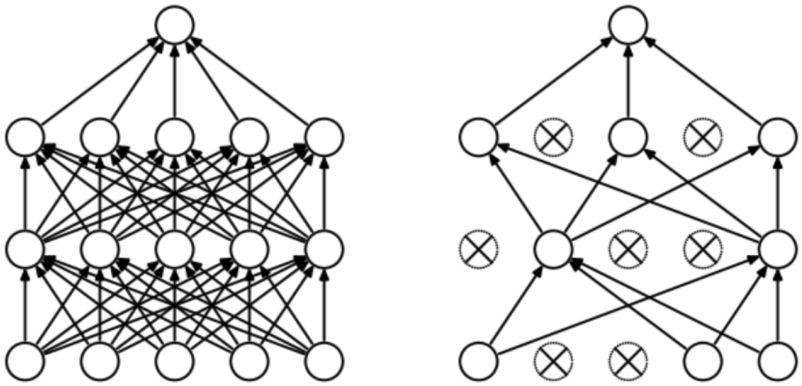

FIGURE 14 – Le dropout : une idée pour éviter l'overfitting des réseaux de neurones (voir Srivastava, Nitish, et al., JMLR 2014).

8 Critères de performance

Étant observé un sous-ensemble de notes observées $(A^*_{ij} : (i,j) \in \Omega)$, on mets de côté un certain pourcentage de cette base de données qu'on va ustilisé comme 'test set' ; le reste des notes étant utilisé comme 'train test'. On ne choist pas complétement de manière aléatoire les notes mises de côté pour le test set dans toute la matrice, on doit faire en sorte que chaque ligne (càd chaque user) à au moins une cinquantaine de notes présentes dans le test set.

On suppose construit un système de recommandations. Pour chaque user, on identifie l'ensemble des items qu'il a effectivement bien apprécié dans le test set et on compte parmi eux le nombre de recommendations qu'on lui fait. On divise ce nombre par le nombre total d'items rééllement appréciés. Ceci donne le **recall**. Pour la **précision**, on fait la même chose sauf qu'on divise par la nombre d'items recommandés. Ces deux quantités, sont données pour chaque users. On peut les agréger de multiple manière (en en prenant la moyenne et des quantiles par exemple) ou on peut calculer le F1 donné pour chaque user par

$$F1 = \frac{2 \times recall \times precision}{recall + precision}.$$

On peut ensuite représenter la distribution de ces score F1.

On peut aussi considérer des métriques du type **MAE** (mean absolute error) qui est la norme ℓ_1 entre les notes estimées et les notes du test set ou la norme ℓ_2 qui donne la **RMSE** (root mean square error). En fait, n'importe quelle distance entre les notes estimées et les notes observées du test set peuvent être utilisées comme critère de performance. Il est toujours bon de vérifier ces critères de performance à la fois sur le test set et sur le learning set pour voir s'il n'y a pas eu trop de sur-apprentissage (qui a lieu quand l'erreur sur le train est beaucoup plus petite que l'erreur sur le test set). Cette 'mesure' du sur-apprentissage a néanmoins été remis en cause assez récemment avec les méthodes de deep learning qui arrivent à ne faire aucune erreur sur l'échantillon d'apprentissage (ce qui était considéré comme un cas typique de sur-apprentissage avant) tout en généralisant bien, càd à faire une erreur raisonnable sur l'échantillon de test.