Une approche générique pour la factorisation booléenne de matrices binaires

Rodrigo CABRAL FARIAS[1], Sebastian MIRON[2]

[1]Université Côte d'Azur, CNRS, I3S, 06900 Sophia-Antipolis, France

[2]Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France

cabral@i3s.unice.fr, sebastian.miron@univ-lorraine.fr

Résumé – Dans cet article, nous proposons une approche générique pour les décompositions booléennes de matrices de données binaires. Dans notre approche, les matrices binaires de rang 1 du modèle de factorisation peuvent être combinées par une fonction booléenne arbitraire, généralisant ainsi le modèle standard de factorisation booléenne, où la combinaison est limitée à la fonction logique 'OR'. L'approche proposée repose sur une relaxation réelle des contraintes de binarité sur les facteurs du modèle et sur une représentation polynomiale de la fonction booléenne considérée. Un algorithme d'estimation basé sur la descente de gradient est également proposé et comparé aux algorithmes de l'état de l'art.

Abstract – In this paper, we propose an approach for fitting generalized Boolean matrix factorization models to binary data. In our generalized setting, the binary rank-1 components of the underlying model can be combined by any Boolean function, thus extending the standard Boolean matrix factorization model, where the combination is restricted to the logical 'OR' function. The proposed approach rely on a relaxation of the binary constraints on the factors of the model and on a polynomial representation of the Boolean function that combines the rank-1 components. An estimation algorithm based on the gradient descent optimization method is also proposed and compared to the state-of-the-art methods.

1 Introduction

Les matrices de données binaires constituent l'un des moyens les plus naturels de coder les données dans de nombreuses applications. Pour cette raison, un nombre important d'algorithmes reposant sur des factorisations de matrices binaires sont apparus au cours des deux dernières décennies, avec des applications dans divers domaines tels que l'analyse des données textuelles, les systèmes de recommandation, les télécommunication, etc.

Différents modèles de factorisation binaires ont été proposés dans la littérature, le plus populaire étant la *factorisation en matrices binaires (bMF)* [1], qui factorise directement la matrice de données en deux matrices binaires. Ce modèle est équivalent à une décomposition de la matrice de données en une somme algébrique de matrices binaires de rang 1. Il a été démontré que l'estimation du modèle bMF est un problème NP-difficile [2], et par conséquent, les algorithmes pour la bMF ne visent pas à résoudre exactement le problème initial, mais une approximation de celui-ci. Une autre classe de méthodes, dont font partie l'algorithme ASSO [3] ou l'analyse par concepts formels (FC) [4], repose sur des règles itératives, peu complexes, qui extraient de manière gloutonne des termes binaires de rang 1 se rapprochant le plus des composantes optimales. D'autres approches [1] visent à résoudre une version relaxée du problème d'estimation initial. Ainsi, les entrées des facteurs de la bMF sont relaxées sur l'orthant non-négatif, tandis qu'un terme de pénalisation force les facteurs à être proches de la binarité.

Les limites du modèle bMF apparaissent lorsque les termes binaires de rang-1 ont des supports qui se superposent. Dans ce cas, la somme des termes de rang 1 ne conduit pas à une matrice binaire. Une façon de contrer ce problème est de remplacer les sommes arithmétiques, dans le modèle de décomposition, par des fonctions logiques 'OR'. Une telle approche est appelée *factorisation booléenne de matrices binaires (BMF)*. Un algorithme explicitement adapté pour la BMF a été proposé dans [5]. Les auteurs appliquent une fonction de seuillage au mélange de valeurs binaires, de sorte que les éléments de la matrice résultante soient 0 ou 1. Comme la fonction de seuillage n'est pas différentiable, afin d'utiliser un algorithme de gradient multiplicatif, une approximation lisse de la fonction a été utilisée. L'algorithme résultant, utilisant cette fonction de pénalité post-nonlinéaire, a été appelé PNL-PF.

Néanmoins, l'approche dans [5] est limitée aux opérations logiques 'OR' entre les termes binaires de rang 1. Dans cet article, nous proposons une approche générique dans laquelle le 'OR' logique peut être remplacé par une fonction booléenne arbitraire avec une table de vérité connue, *e.g.,* 'XOR', la fonction majoritaire, etc. Comme pour PNL-PF, notre approche est fondée sur la relaxation des contraintes binaires et l'ajout d'un terme de pénalisation au problème d'optimisation sous-jacent pour favoriser les solutions binaires. Au lieu de représenter le comportement de la fonction booléenne par un seuillage, nous le représentons en utilisant un polynôme multivarié. Puisqu'un

polynôme est une fonction différentiable, une telle représentation nous permet de proposer un algorithme de gradient pour estimer ce modèle BMF généralisé, sans avoir besoin de recourir à des approximations lisses, comme c'est le cas pour PNL-PF.

2 Factorisation booléenne généralisée d'une matrice binaire

Nous nous intéressons à la décomposition exacte ou approximative d'une matrice binaire $\boldsymbol{Y}$ de taille $I \times J$ en $R \geqslant 2$ matrices binaires de rang 1 : $\boldsymbol{X}^{1:R} = \left\{\boldsymbol{X}^1, \boldsymbol{X}^2, \cdots, \boldsymbol{X}^R\right\}$. Chaque matrice $\boldsymbol{X}^r$, avec $r \in \{1, 2, \cdots, R\}$ s'écrit comme $\boldsymbol{X}^r = \boldsymbol{a}_r \boldsymbol{b}_r^\mathsf{T}$, avec $\boldsymbol{a}_r$ et $\boldsymbol{b}_r$ des vecteurs binaires de tailles respectives I et J. Les vecteurs $\boldsymbol{a}_r$ et $\boldsymbol{b}_r$ peuvent être rangés dans des matrices $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_R]$ et $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_R]$. Comme présenté dans [5] et [6], différentes décompositions peuvent être obtenues selon la façon dont les matrices $\boldsymbol{X}_r$ sont combinées pour obtenir $\boldsymbol{Y}$. Si nous spécifions une fonction $f : \{0, 1\}^R \to \mathcal{Y}$ définie sur R entrées binaires et résultant en une valeur sur un sous-ensemble fini $\mathcal{Y}$ des entiers, la factorisation booléenne généralisée correspond à approximer les éléments de $\boldsymbol{Y}$ par

$$y_{ij} \approx f\left(x_{i,j}^1, x_{i,j}^2, \cdots, x_{i,j}^R\right) = f\left(x_{i,j}^{1:R}\right) = f\left(a_{i,1:R} b_{j,1:R}\right), \tag{1}$$

où $a_{i,1:R}b_{j,1:R} = \{a_{i,1}b_{j,1}, a_{i,2}b_{j,2}, \cdots, a_{i,R}b_{j,R}\}$. En désignant la matrice résultant de l'application de $f(\cdot)$ élément par élément aux matrices de rang 1 $\boldsymbol{X}^{1:R}$, par $f(\boldsymbol{X}^{1:R})$, le problème d'approximation (1) peut être formulé comme la minimisation par rapport à $\boldsymbol{A} \in \{0, 1\}^{I \times R}$ et $\boldsymbol{B} \in \{0, 1\}^{J \times R}$ de la fonction de coût ci-dessous :

$$\mathcal{F}(\boldsymbol{A}, \boldsymbol{B}) = \frac{1}{2}\|\boldsymbol{Y} - f(\boldsymbol{X}^{1:R})\|_F^2, \tag{2}$$

où $\boldsymbol{X}^{1:R} = \left\{\boldsymbol{X}^1 = \boldsymbol{a}_1\boldsymbol{b}_1^\mathsf{T}, \cdots, \boldsymbol{X}^R = \boldsymbol{a}_R\boldsymbol{b}_R^\mathsf{T}\right\}$.

L'existence d'une solution à ce problème d'approximation est garantie, puisque le cardinal de l'ensemble des solutions admissibles est fini. Si une solution $\boldsymbol{A}^\star, \boldsymbol{B}^\star$ du problème d'approximation est telle que $\mathcal{F}(\boldsymbol{A}^\star, \boldsymbol{B}^\star) = 0$, on dit qu'il s'agit d'une factorisation exacte de $\boldsymbol{Y}$.

Représentation polynomiale d'une fonction booléenne générique : nous nous concentrons dans cet article sur la résolution du problème (2) où $f(\cdot)$ est une fonction booléenne générique $f : \{0, 1\}^R \to \{0, 1\}$.

Pour ce faire, on propose une approche en deux étapes :

— Dans la première étape nous appliquons un algorithme d'optimisation pour résoudre une version relaxée de (2) où les contraintes de binarité sont abandonnées. Les éléments de $\boldsymbol{A}$ et $\boldsymbol{B}$ sont autorisés à se trouver dans $\mathbb{R}$.

— Dans la deuxième étape, les approximations de $\boldsymbol{A}$ et $\boldsymbol{B}$, désignées par $\hat{\boldsymbol{A}}$ et $\hat{\boldsymbol{B}}$, sont projetées sur l'ensemble des valeurs binaires. Cette projection $\mathcal{P}_\mathbb{B}(\cdot)$ correspond simplement à une opération de seuillage. Par exemple, pour les éléments de $\hat{\boldsymbol{A}}$ on a

$$\left[\mathcal{P}_\mathbb{B}(\hat{\boldsymbol{A}})\right]_{i,r} = \begin{cases} 0 & \text{, pour } \hat{a}_{i,r} < 0.5, \\ 1 & \text{, pour } \hat{a}_{i,r} \geqslant 0.5. \end{cases} \tag{3}$$

Pour mettre en oeuvre cette méthode, la fonction booléenne $f(\cdot)$ doit être remplacée par une autre fonction $\bar{f}$ définie pour des entrées réelles. La fonction $\bar{f}$ doit être définie de manière à ce que les deux fonctions soient égales pour des valeurs exactement binaires. Dans ce travail, nous proposons de définir $\bar{f}(\cdot)$ en s'appuyant sur le fait que toute fonction booléenne de R variables $\mathbf{x} = [x_1, x_2, \cdots, x_R]^\mathsf{T}$ peut être écrite comme un polynôme multivarié. Pour tout $\mathbf{x} \in \{0, 1\}^R$, le polynôme suivant donne les mêmes valeurs que $f(\cdot)$:

$$\bar{f}(\mathbf{x}) = \sum_{\mathbf{w} \in \mathcal{W}^1} \left\{ \prod_{i|w_i=1} x_i \prod_{j|w_j=0} (1 - x_j) \right\}, \tag{4}$$

avec $\mathcal{W}^1 = \left\{\mathbf{w} \in \{0,1\}^R \,|\, f(\mathbf{w}) = 1\right\}$.

Quelques exemples de représentations polynomiales de fonctions booléennes élémentaires sont donnés ci-dessous :

— Fonction 'OR' logique avec $R = 2$, $(x_1 \vee x_2)$:

$$\bar{f}_{\text{OR}}(x_1, x_2) = (1 - x_1)x_2 + x_1(1 - x_2) + x_1x_2. \tag{5}$$

— Fonction 'XOR' logique avec $R = 2$, $(x_1 \oplus x_2)$:

$$\bar{f}_{\text{XOR}}(x_1, x_2) = (1 - x_1)x_2 + x_1(1 - x_2). \tag{6}$$

— Fonction majoritaire à 3 termes :

$$\bar{f}_{\text{MAJ}}(x_1, x_2) = x_1x_2 + x_2x_3 + x_1x_3 - 2x_1x_2x_3. \tag{7}$$

Il faut noter qu'avec cette représentation, le nombre de termes du polynôme dépend du cardinal de $\mathcal{W}^1$ (nombre de combinaisons d'entrées telles que $f(\mathbf{x}) = 1$). Si l'ensemble $\mathcal{W}^0 = \left\{\mathbf{w} \in \{0,1\}^R \,|\, f(\mathbf{w}) = 0\right\}$ a un cardinal plus petit que $\mathcal{W}^1$, il peut être plus pratique d'utiliser une autre forme équivalente :

$$\bar{f}(\mathbf{x}) = 1 - \sum_{\mathbf{w} \in \mathcal{W}^0} \left\{ \prod_{i|w_i=1} x_i \prod_{j|w_j=0} (1 - x_j) \right\}. \tag{8}$$

Notez que dans le cas de $x_1 \vee x_2$, la représentation ci-dessus conduit à $\bar{f}_{\text{OR}}(x_1, x_2) = 1 - (1 - x_1)(1 - x_2)$ et la version correspondante de 'OR' pour R termes a une expression simple :

$$\bar{f}_{\text{OR}}(x_1, x_2, \cdots, x_R) = 1 - \prod_{r=1}^{R} (1 - x_r). \tag{9}$$

Dans la suite de l'article, nous utiliserons cette deuxième représentation polynomiale (8) car elle permet une formulation plus aisée de l'algorithme que nous proposons pour la BMF.

3 Un algorithme de descente de gradient pour la BMF

Puisque $\bar{f}(\cdot)$ est différentiable, la descente de gradient peut être appliquée pour tenter de résoudre une version relaxée de la

minimisation de (2). Nous proposons dans cette section un algorithme qui utilise une version relaxée de (2), où les éléments des facteurs sont autorisés à prendre des valeurs dans $\mathbb{R}$. Pour forcer la solution à être proche de la binarité, nous introduisons un terme de pénalité $\mathcal{G}(\boldsymbol{A}, \boldsymbol{B})$ dans la fonction de coût, comme dans [1, 5]. L'expression de ce terme de pénalité est

$$\mathcal{G}(\boldsymbol{A}, \boldsymbol{B}) = \frac{1}{2}\left\{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{r=1}^{R}\left[a_{i,r}^2(1-a_{i,r})^2 + b_{j,r}^2(1-b_{j,r})^2\right]\right\}.$$

Observez que cette pénalité est minimale lorsque toutes les entrées des matrices facteurs sont binaires. Le nouveau problème d'optimisation que nous devons résoudre correspond à la fonction de coût suivante :

$$\mathcal{H}(\boldsymbol{A}, \boldsymbol{B}; \lambda) = \mathcal{F}(\boldsymbol{A}, \boldsymbol{B}) + \lambda\mathcal{G}(\boldsymbol{A}, \boldsymbol{B}), \tag{10}$$

où $\lambda > 0$ est une valeur fixée. Puisque ce terme de pénalité est différentiable, si nous l'ajoutons à $\mathcal{F}(\boldsymbol{A}, \boldsymbol{B})$, la fonction de coût globale reste différentiable. Par conséquent, nous pouvons appliquer l'algorithme standard de descente de gradient pour trouver les points critiques de $\mathcal{H}(\boldsymbol{A}, \boldsymbol{B}; \lambda)$.

Dans les approches standard de descente de gradient, tous les paramètres du modèle sont estimés en même temps via le vecteur $\boldsymbol{\theta} = \left[\text{vec}(\boldsymbol{A})^\mathsf{T}\ \text{vec}(\boldsymbol{B})^\mathsf{T}\right]^\mathsf{T}$, avec

$$\text{vec}(\boldsymbol{A}) = \left[\boldsymbol{a}_1^\mathsf{T}, \boldsymbol{a}_2^\mathsf{T}, \cdots, \boldsymbol{a}_R^\mathsf{T}\right]^\mathsf{T}; \text{vec}(\boldsymbol{B}) = \left[\boldsymbol{b}_1^\mathsf{T}, \boldsymbol{b}_2^\mathsf{T}, \cdots, \boldsymbol{b}_R^\mathsf{T}\right]^\mathsf{T}.$$

L'estimation $\hat{\boldsymbol{\theta}}_k$ du vecteur des paramètres à l'itération k est donnée par $\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} - \gamma_k\ \nabla_{\boldsymbol{\theta}}\mathcal{H}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{k-1}}$, où γ_k est le pas de descente et $\nabla_{\boldsymbol{\theta}}\mathcal{H}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{k-1}}$ est le gradient de $\mathcal{H}(\cdot)$ évalué en $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_{k-1}$.

Calcul du gradient : l'expression du gradient complet en fonction des gradients partiels par rapport à $\boldsymbol{A}$ et $\boldsymbol{B}$ peut être donnée par $\nabla_{\boldsymbol{\theta}}^\mathsf{T}\mathcal{H}(\boldsymbol{\theta}) = \left[\nabla_{\text{vec}(\boldsymbol{A})}^\mathsf{T}\mathcal{H}(\boldsymbol{A}, \boldsymbol{B})\quad \nabla_{\text{vec}(\boldsymbol{A})}^\mathsf{T}\mathcal{H}(\boldsymbol{A}, \boldsymbol{B})\right]$, où les éléments de $\nabla_{\text{vec}(\boldsymbol{A})}\mathcal{H}(\boldsymbol{A}, \boldsymbol{B})$ sont

$$\frac{\partial\mathcal{H}(\boldsymbol{A}, \boldsymbol{B})}{\partial a_{i',r'}} = -\left\{\sum_{j=1}^{J}\left[y_{i',j} - \bar{f}(x_{i',j}^{1:R})\right]\frac{\partial\bar{f}(x_{i',j}^{1:R})}{\partial x_{i',j}^{r'}}b_{j,r'}\right\} + \lambda g(a_{i,r'}) \tag{11}$$

pour $i' \in \{1, \cdots, I\}$ et $r' \in \{1, \cdots, R\}$, où $g(a_{i',r'}) = a_{i',r'}(1-a_{i',r'})(1-2a_{i',r'})$ et la dérivée partielle de $\bar{f}(\cdot)$ est

$$\frac{\partial\bar{f}(x_{i',j}^{1:R})}{\partial x_{i',j}^{r'}} = \sum_{\substack{\boldsymbol{w}\in\mathcal{W}^0,\\ w_{r'}=0}}\left[\prod_{s|w_s=1}x_{i',j}^{s}\right]\left[\prod_{\substack{s'|w_s'=0,\\ s'\neq r'}}(1-x_{i',j}^{s'})\right]$$
$$- \sum_{\substack{\boldsymbol{w}\in\mathcal{W}^0,\\ w_{r'}=1}}\left[\prod_{\substack{s|w_s=1,\\ s\neq r'}}x_{i',j}^{s}\right]\left[\prod_{s'|w_s'=0}(1-x_{i',j}^{s'})\right]$$

Les éléments de $\nabla_{\text{vec}(\boldsymbol{B})}\mathcal{H}(\boldsymbol{A}, \boldsymbol{B})$ sont donnés par

$$\frac{\partial\mathcal{H}(\boldsymbol{A}, \boldsymbol{B})}{\partial b_{j',r'}} = -\left\{\sum_{i=1}^{I}\left[y_{i,j'} - \bar{f}(x_{i,j'}^{1:R})\right]\frac{\partial\bar{f}(x_{i,j'}^{1:R})}{\partial x_{i,j'}^{r'}}a_{i,r'}\right\} + \lambda g(b_{j',r'}) \tag{12}$$

pour $j' \in \{1, \cdots, J\}$ and $r' \in \{1, \cdots, R\}$, où $g(b_{j',r'}) = b_{j',r'}(1-b_{j',r'})(1-2b_{j',r'})$.

Les gradients partiels peuvent être écrits sous forme vectorielle en fonction de $\boldsymbol{A}$ et $\boldsymbol{B}$ selon

$$\nabla_{\text{vec}(\boldsymbol{A})}\mathcal{H}(\boldsymbol{A}, \boldsymbol{B}) = -\text{Diag}\left(\boldsymbol{E}\boxdot\mathbf{P}_1, \cdots, \boldsymbol{E}\boxdot\mathbf{P}_R\right)\text{vec}(\boldsymbol{B})$$
$$+\lambda\text{vec}(\boldsymbol{A})\boxdot(\mathbf{1}_{IR\times 1} - \text{vec}(\boldsymbol{A}))\boxdot(\mathbf{1}_{IR\times 1} - 2\text{vec}(\boldsymbol{A}))$$
$$\nabla_{\text{vec}(\boldsymbol{B})}\mathcal{H}(\boldsymbol{A}, \boldsymbol{B}) = -\text{Diag}\left(\boldsymbol{E}^\mathsf{T}\boxdot\mathbf{P}_1^\mathsf{T}, \cdots, \boldsymbol{E}^\mathsf{T}\boxdot\mathbf{P}_R^\mathsf{T}\right)\text{vec}(\boldsymbol{A})$$
$$+\lambda\text{vec}(\boldsymbol{B})\boxdot(\mathbf{1}_{JR\times 1} - \text{vec}(\boldsymbol{B}))\boxdot(\mathbf{1}_{JR\times 1} - 2\text{vec}(\boldsymbol{B})),$$

où $(\cdot\boxdot\cdot)$ est le produit d'Hadamard, $\text{Diag}\left(\boldsymbol{Z}_1, \cdots, \boldsymbol{Z}_N\right)$ est une matrice bloc-diagonale avec $\boldsymbol{Z}_1, \cdots, \boldsymbol{Z}_N$ comme blocs diagonaux, $\mathbf{1}_{L\times M}$ est une matrice de '1' de taille $L \times M$, $\boldsymbol{E}$ est la matrice d'erreur : $\boldsymbol{E} = \boldsymbol{Y} - \bar{f}\left(\boldsymbol{X}^{1:R}\right) = \boldsymbol{Y} - \mathbf{1}_{I\times J} + \sum_{\boldsymbol{w}\in\mathcal{W}^0}\left[\boxdot_{s|w_s=1}\boldsymbol{X}^s\right]\boxdot\left[\boxdot_{s'|w_s'=0}(\mathbf{1}_{I\times J} - \boldsymbol{X}^{s'})\right]$ et les $\boldsymbol{P}_{r'}$ sont des matrices $I \times J$ données par

$$\boldsymbol{P}_{r'} = \sum_{\substack{\boldsymbol{w}\in\mathcal{W}^0,\\ w_{r'}=0}}\left[\boxdot_{s|w_s=1}\boldsymbol{X}^s\right]\boxdot\left[\boxdot_{\substack{s'|w_s'=0,\\ s'\neq r'}}(\mathbf{1}_{I\times J} - \boldsymbol{X}^{s'})\right]$$
$$- \sum_{\substack{\boldsymbol{w}\in\mathcal{W}^0,\\ w_{r'}=1}}\left[\boxdot_{\substack{s|w_s=1,\\ s\neq r'}}\boldsymbol{X}^s\right]\boxdot\left[\boxdot_{s'|w_s'=0}(\mathbf{1}_{I\times J} - \boldsymbol{X}^{s'})\right].$$

Dans la version la plus simple de l'algorithme, la valeur du pas γ_k est fixée à une petite valeur constante. Le paramètre de pénalité λ peut être choisi comme variable au fil des itérations : λ est fixé à une valeur proche de zéro dans les premières itérations et incrémenté jusqu'à une valeur maximale, à mesure que k augmente. Comme la fonction de coût à minimiser est hautement non convexe, la descente de gradient risque de converger vers des minima locaux. Pour cette raison, il est important de tester différentes initialisations de l'algorithme et de choisir la solution qui donne la meilleure adéquation aux données. Pour chaque initialisation, les éléments de $\hat{\boldsymbol{A}}$ et $\hat{\boldsymbol{B}}$ peuvent être tirés aléatoirement à partir des distributions uniformes : $\hat{a}_{i,r} \sim \mathcal{U}[0, 1]$, $\hat{b}_{j,r} \sim \mathcal{U}[0, 1]$, pour $i \in \{1, \cdots, I\}$, $j \in \{1, \cdots, J\}$ et $r \in \{1, \cdots, R\}$.

4 Résultats numériques

Dans cette section, nous présentons des résultats de simulation pour l'algorithme proposé, ainsi que son application à l'analyse d'un jeu de données réelles. Nous nous concentrons sur la BMF, c'est-à-dire, le cas où la fonction $f(\cdot)$ est le 'OR' logique.

Données simulées : nous comparons les performances de notre algorithme de descente de gradient (GD) avec trois autres méthodes de la littérature sur des données simulées. Les données sont générées selon des modèles BMF aléatoires et perturbées par un bruit de basculement ('XOR') binaire. La valeur de R est maintenue constante et les résultats sont présentés pour différentes valeurs de la probabilité de basculement binaire, *i.e.,* différentes valeurs d'« intensité » du bruit. Les performances de l'algorithme proposé (GD) sont comparées aux

méthodes de l'état suivantes : ASSO [3], FC [4], et PNL-PF [5]. Nous considérons que $\boldsymbol{Y}$ est une matrice de données binaires 20×20 correspondant à une version perturbée d'une BMF : $\boldsymbol{X} = \bigvee_{r=1}^{R} \boldsymbol{a}_r \boldsymbol{b}_r^{\mathsf{T}}$. La matrice de bruit $\boldsymbol{N}$ est également binaire, et ses éléments sont tirés i.i.d. à partir d'une distribution de Bernoulli $n_{i,j} \sim \mathcal{B}(p_n)$, avec $p_n = \mathbb{P}(n_{i,j} = 1)$. Les éléments de $\boldsymbol{Y}$ peuvent s'écrire, en utilisant le 'XOR' logique, comme : $y_{i,j} = x_{i,j} \oplus n_{i,j}$. Nous avons fixé $R = 3$, et p_n varie de 0 à 0.3 par incréments de 0.02. La Fig. 1 montre les performances de notre algorithme comparé à ceux de l'état de l'art, en termes d'erreur quadratique moyenne normalisée (EQMN). L'EQMN est calculée sur 100 matrices $\boldsymbol{Y}$ générées à partir de tirages aléatoires de $\boldsymbol{A}$, $\boldsymbol{B}$ et $\boldsymbol{N}$. Comme attendu, les performances se dégradent lorsque p_n augmente, mais la méthode proposée (GD) semble fournir des meilleures résultats comparés aux autres méthodes de la littérature pour toutes les valeurs d'intensité de bruit.

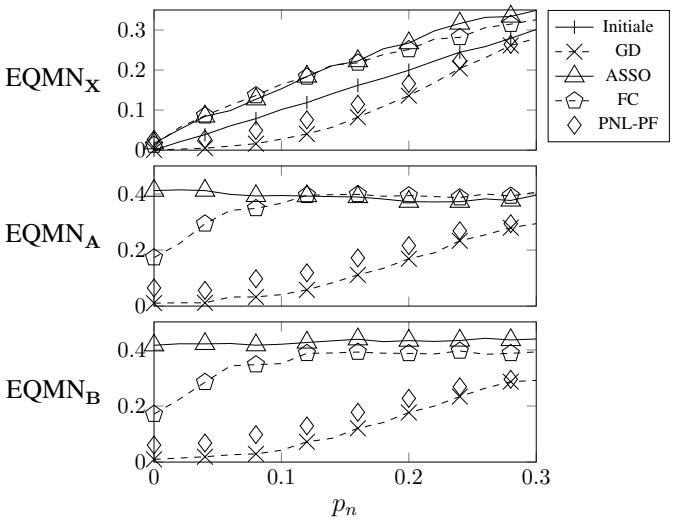

FIGURE 1 – EQMN pour la prédiction de $\boldsymbol{X}$ (20×20) et pour l'estimation des facteurs $\boldsymbol{A}$ and $\boldsymbol{B}$ de la BMF avec $R = 3$ termes. Les algorithmes de décomposition ont été testés sur une version bruité $\boldsymbol{Y}$ de $\boldsymbol{X}$. La probabilité p_n du bruit binaire qui fait basculer les éléments de $\boldsymbol{X}$ varie de 0 à 0.3 avec un pas de 0.02.

Observons que lorsque p_n se rapproche de 0.3, l'EQMN de la méthode proposée est proche de 0.3 et pour des valeurs inférieures à $p_n < 0.3$ l'EQMN semble être plus petite que p_n. Notons également, sur la figure d'en haut, que la prédiction de $\boldsymbol{X}$ en utilisant les données bruitées $\boldsymbol{Y}$ (performance indiquée par la courbe « Initiale ») est aussi efficace que l'utilisation de la méthode proposée pour $p_n = 0.3$. Pour cette valeur de p_n, aucun effet de débruitage des données n'est obtenu par l'utilisation du modèle BMF de rang faible. Enfin, on peut voir que, lorsque $p_n = 0$, l'EQMN pour l'estimation des facteurs est non-nulle. Cela peut être dû à la convergence des algorithmes vers des minima locaux de (10) ou à la non-unicité de la factorisation pour certaines réalisations. Dans ce dernier cas, les algorithmes peuvent trouver des factorisations qui sont des minima globaux de (10), mais néanmoins très éloignés de $\boldsymbol{A}$ et $\boldsymbol{B}$ originaux. Comme les erreurs EQMN$_{\boldsymbol{A}}$ et EQMN$_{\boldsymbol{B}}$ sont très petites pour $p_n = 0$, il semblerait que l'occurrence de tels problèmes reste tout de même très rare.

Données réelles : nous appliquons notre algorithme à un jeu de données contenant les votes des représentants du congrès américain sur 16 propositions importantes votées en 1984 [1]. Les votes des 435 représentants sont codés par des valeurs binaires : 1 pour un vote « pour », 0 pour un vote « contre ». Ce qui permet de coder le jeu de données en une matrice binaire de taille 435×16. Le jeu de données contient aussi les partis des représentants (Démocrate ou Républicain). Notre algorithme a été appliqué pour retrouver la BMF correspondant à ces données avec $R = 2$. En comparant la matrice $\boldsymbol{A}$ retrouvée par l'algorithme avec celle qui correspondrait à l'encodage des partis de chaque candidat, nous trouvons que l'algorithme est capable de prédire correctement le parti des représentants dans 77.7% des cas. Ce résultat est très similaire à celui obtenu avec l'algorithme de l'état de l'art [5] PNL-PF.

5 Conclusion

Nous introduisons dans cette communication une approche flexible, généralisant les modèles classiques de factorisations booléennes de matrices binaires basés sur des « sommes »'OR' logiques, à des fonctions logiques arbitraires. L'approche est fondée sur une représentation réelle des fonctions logiques sous forme de polynômes multivariés. Cela permet des implémentations efficaces d'algorithmes d'estimation en s'appuyant sur les outils standard d'optimisation pour les fonctions réelles lisses.

Références

[1] Z. Zhang, T. Li, C. Ding, and X. Zhang, "Binary matrix factorization with applications," in *Seventh IEEE international conference on data mining (ICDM 2007)*. IEEE, 2007, pp. 391–400.

[2] N. Gillis and S. A. Vavasis, "On the complexity of robust PCA and L1-norm low-rank matrix approximation," *Mathematics of Operations Research*, vol. 43, no. 4, pp. 1072–1084, 2018.

[3] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila, "The discrete basis problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 10, pp. 1348–1362, 2008.

[4] R. Belohlavek and V. Vychodil, "Discovery of optimal factors in binary data via a novel method of matrix decomposition," *Journal of Computer and System Sciences*, vol. 76, no. 1, pp. 3–20, 2010.

[5] S. Miron, M. Diop, A. Larue, E. Robin, and D. Brie, "Boolean decomposition of binary matrices using a post-nonlinear mixture approach," *Signal Processing*, vol. 178, p. 107809, 2021.

[6] D. DeSantis, E. Skau, and B. Alexandrov, "Factorizations of binary matrices–rank relations and the uniqueness of boolean decompositions," *arXiv preprint arXiv :2012.10496*, 2020.

1. `https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`.