Symmetric nonnegative matrix trifactorization

Etude du cas orthogonal pour le clustering

Alexandra Dache

Scientific computing

University of Mons, Belgium

Symmetric Non-Negative Matrix Trifactorization

Soient

- une matrice symétrique $X \in \mathbb{R}+^{n \times n}$,
- un rang $r \in \mathbb{N}$,

trouver $W \in \mathbb{R}_+^{n \times r}$ et $S \in \mathbb{R}_+{}^{r \times r}$ tq $X \approx WSW^T$

Symmetric nonnegative matrix trifactorization (TriSymNMF)

$$\min_{W \geq 0, S \geq 0} \|X - WSW^T\|_F^2 \quad \text{s.t.} \quad S^T = S$$

TriSymNMF for clustering

- X étant la matrice d'adjacence d'un graphe, où l'entrée $X(i, j) = X(j, i)$ indique le poids de la connexion entre le nœud i et j.

TriSymNMF for clustering

- X étant la matrice d'adjacence d'un graphe, où l'entrée $X(i,j) = X(j,i)$ indique le poids de la connexion entre le nœud i et j.
- la factorisation permet d'identifier r communautés et leurs interactions. Pour chaque élément de la matrice X, on a :

$$X(i,j) \approx W(i,:)SW(:,j)^T = \sum_{k=1}^{r}\sum_{l=1}^{r} W(i,k)S(k,l)W(l,j)^T \quad (1)$$

TriSymNMF for clustering

- X étant la matrice d'adjacence d'un graphe, où l'entrée $X(i,j) = X(j,i)$ indique le poids de la connexion entre le nœud i et j.
- la factorisation permet d'identifier r communautés et leurs interactions. Pour chaque élément de la matrice X, on a :

$$X(i,j) \approx W(i,:)SW(:,j)^T = \sum_{k=1}^{r}\sum_{l=1}^{r} W(i,k)S(k,l)W(l,j)^T \quad (1)$$

  - Les colonnes de W identifient les communautés cad un ensemble de noeuds très connectés. Si $W(j,k)$ est non nul, l'élément j appartient à la communauté k.

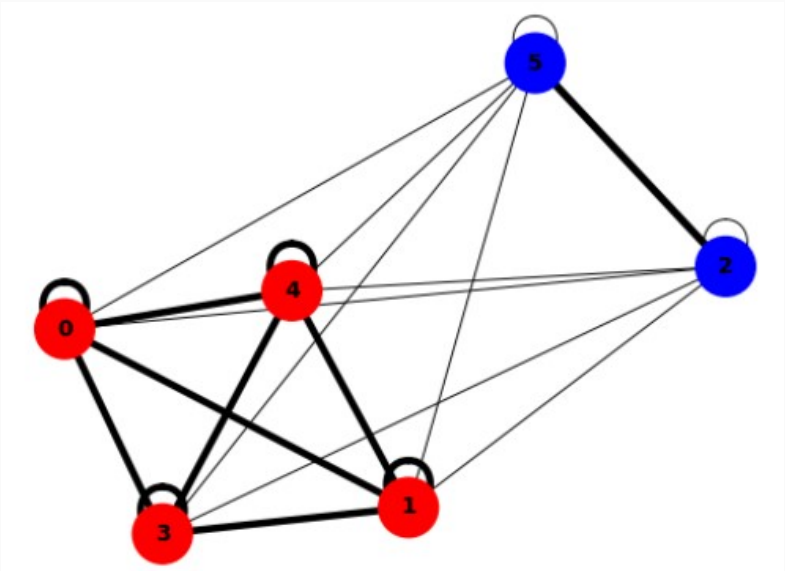
TriSymNMF for clustering

- X étant la matrice d'adjacence d'un graphe, où l'entrée $X(i,j) = X(j,i)$ indique le poids de la connexion entre le nœud i et j.
- la factorisation permet d'identifier r communautés et leurs interactions. Pour chaque élément de la matrice X, on a :

$$X(i,j) \approx W(i,:)SW(:,j)^T = \sum_{k=1}^{r}\sum_{l=1}^{r} W(i,k)S(k,l)W(l,j)^T \quad (1)$$

  - Les colonnes de W identifient les communautés cad un ensemble de noeuds très connectés. Si $W(j,k)$ est non nul, l'élément j appartient à la communauté k.
  - La matrice S donne les interactions entre les communautés, où l'entrée $S(k,l)$ représente la force de la connexion entre les communautés k et l.

TriSymNMF for clustering



Graphe de la matrice d'adjacence X

$$X, \quad W = \begin{bmatrix} 0.5 & 0 \\ 0.5 & 0 \\ 0 & \frac{1}{\sqrt{2}} \\ 0.5 & 0 \\ 0.5 & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \quad , S = \begin{bmatrix} 13 & 2 \\ 2 & 7 \end{bmatrix}$$

Contrainte de l'orthogonalité

- Un cas particulier est de trouver des communautés disjointes, c-a-d que chaque noeud appartient à une et une seule communauté.
- Mathématiquement, on peut imposer que $W^T W = I$, les colonnes de W sont orthogonales entre elles.

Contrainte de l'orthogonalité

- Un cas particulier est de trouver des communautés disjointes, c-a-d que chaque noeud appartient à une et une seule communauté.
- Mathématiquement, on peut imposer que $W^T W = I$, les colonnes de W sont orthogonales entre elles.

Propriété

Si $W \in \mathbb{R}_+^{n \times r}$ tq $W^T W = I$ alors chaque ligne de W contient un seul élément non nul.

Proof.

$W^T W = I$ implique que les colonnes de W sont orthogonales entre elles. Or, deux vecteurs positifs sont orthogonaux ssi ils ont des supports disjoints.Donc les colonnes de W doivent avoir des supports disjoints cad que W contient un seul élément non nul par ligne. □

Orthogonal symmetric nonnegative matrix trifactorization

Orthogonal symmetric nonnegative matrix trifactorization (OTriSymNMF)

$$\min_{W>=0,S>=0} \|X - WSW^T\|_F^2 \quad \text{s.t.} \quad W^T W = I,$$

Orthogonal symmetric nonnegative matrix trifactorization

Orthogonal symmetric nonnegative matrix trifactorization (OTriSymNMF)

$$\min_{W>=0,S>=0} \|X - WSW^T\|_F^2 \quad \text{s.t.} \quad W^T W = I,$$

$$X(i,j) \approx W(i,:)SW(:,j)^T = \sum_{k=1}^{r}\sum_{l=1}^{r} W(i,k)S(k,l)W(l,j)^T$$

Orthogonal symmetric nonnegative matrix trifactorization

Orthogonal symmetric nonnegative matrix trifactorization (OTriSymNMF)

$$\min_{W>=0,S>=0} \|X - WSW^T\|_F^2 \quad \text{s.t.} \quad W^T W = I,$$

$$X(i,j) \approx W(i,:)SW(:,j)^T = \sum_{k=1}^{r}\sum_{l=1}^{r} W(i,k)S(k,l)W(l,j)^T$$

En posant k_i et l_j comme les indices des éléments non nuls de la ligne i et j de W, on obtient :

$$X(i,j) \approx W(i,k_i)S(k_i,l_j)W(l_j,j)^T \tag{2}$$

Unicité de la factorisation

Propriété

Soit $X = WSW^T$ où $W \in \mathbb{R}+^{n \times r}$ satisfait $W^T W = I$ et les colonnes de $S \in \mathbb{R}+^{r \times r}$ sont linéairement indépendantes. Alors la trifactorisation orthogonale exacte non négative symétrique de X de taille r est unique, aux permutations près des colonnes de W (avec les permutations correspondantes des lignes et des colonnes dans S)."

Proof.

$X = FG$ tq $GG^T = Ir$, et les colonnes de $F \in \mathbb{R}+^{m \times r}$ ne sont pas colinéaires. Dans ce cas, la factorisation exacte non négative de matrice orthogonale (ONMF) de X avec une taille de r est garantie d'être unique [Gillis 2020, p. 136]. En posant que $F = WS$ et $G = W^T$, il est nécessaire que les colonnes du produit WS ne soient pas des multiples les unes des autres. Puisque $W^T W = I$, la condition est équivalente à s'assurer que S n'a pas de colonnes multiples. En effet, les colonnes de W ne sont pas des multiples les unes des autres, et chaque ligne ne contient qu'un seul élément non nul. □

Choix de la méthode

- Ding et al. 2006 propose un algorithme itératif en mettant à jour W, puis S, selon des formules de mise à jour. Les formules impliquent le calcul de produits matriciels.
- Paramètres pour prendre en compte l'othogonalité de W. Problème du choix des paramètres et descente de gradient couteux
- Coordinate descent ? Facile à mettre en pratique
- La méthode itérative proposée tire parti des propriétés de W en mettant à jour W ligne par ligne. Les calculs sont simplifiés en travaillant uniquement avec les éléments non nuls de W.

Pseudocode de la coordinate descent

Algorithm 1: Coordinate descent for OTriSymNMF

Input: Input matrix $X \in \mathbb{R}^{n \times n}$, rank $r \in \mathbb{N}$
Output: Matrix $W \in \mathbb{R}^{n \times r}$ and a matrix $S \in \mathbb{R}^{r \times r}$

$W_0, S_0 \leftarrow$ initialisation(X,r)
for $t = 1$ to maxiter **do**
  $W_t \leftarrow UpdateW(r, W_{t-1}, S_{t-1})$ see Algorithm 2
  $S_t \leftarrow UpdateS(r, W_t, S_{t-1})$ see Algoritm 3
  $error = \|X - WSW^T\|_F^2$
  if $error \simeq 0$ **then**
    break;
  end if
  if $previous\_error - error \simeq 0$ **then**
    break;
  end if
end for
for $j = 1$ to r **do**
  $W(i,j) = \frac{W(i,j)}{\|W(:,j)\|_2}, \forall i$
  $S(k,l) = S(k,l) * \|W(:,j)\|_2$
  $S(l,k) = S(l,k) * \|W(:,j)\|_2$
end for

Update de W

- Pour chaque ligne i de W, on va trouver le meilleur élément non nul de $W(i,:)$ qui minimise l'erreur tout en fixant les autres variables:

$$\min_{k, W(i,k) \geq 0} \|X - WSW^T\|_F^2 \text{s.t.} \quad W(i,j) = 0, \forall j \neq k$$

$$\Longleftrightarrow$$

$$\min_{k, W(i,:) \geq 0} [(X(i,i) - W(i,k)S(k,k)W(i,k))^2$$

$$+2 \sum_i (X(i,j) - W(i,k)SW(j,:))^2 + constantes]$$

Update de W

- Pour chaque ligne i de W, on va trouver le meilleur élément non nul de $W(i,:)$ qui minimise l'erreur tout en fixant les autres variables:

$$\min_{k, W(i,k) \geq 0} \|X - WSW^T\|_F^2 \text{s.t.} \quad W(i,j) = 0, \forall j \neq k$$

$$\Longleftrightarrow$$

$$\min_{k, W(i,:) \geq 0} [(X(i,i) - W(i,k)S(k,k)W(i,k))^2$$

$$+2 \sum_i (X(i,j) - W(i,k)SW(j,:))^2 + constantes]$$

- k fixé, cela revient à minimiser un polynôme d'ordre 4, pouvant être résolut de manière exacte avec les formules de Cardan:

$$\min_{z \geq 0} az^4 + bz^2 + cz$$

- On calcule l'erreur pour chaque $k = 1 : r$ et on garde le meilleur résultat

Update de S

- On met à jour chaque élément de la matrice S en fixant les autres variables. Pour mettre à jour $S(k, l)$:

$$\min_{S(k,l)\geq 0} \sum_{i \in cluster(k)} \sum_{j \in cluster(l)} (X(i,j) - W(i,k)S(k,l)W(j,l))^2$$

- On trouve les noeuds i appartenant au cluster k en regardant les éléments non nuls de la colonne $W(:, k)$.
- Le problème revient à minimiser un polynome d'ordre 2.

Initialisation

- Random :
  - Indice non nul de chaque ligne de W choisi aléatoirement.
  - S aléatoire puis rendue symétrique
- Kmeans :
  - K-means est une méthode classique de clustering où chaque point de données est assigné à un cluster avec le centre le plus proche.
  - K-means est utilisé pour trouver les éléments non nuls de W, qui sont ensuite normalisés. S est déterminé en utilisant la mise à jour de S de notre méthode.

Tests synthétiques

- $n = [10, 20, 50, 100, 150, 200]$ avec $r = [2, 3, 4, 5, 6, 7]$, 100 tests.
- matrice aléatoire W tq $W^T W = I$.
- matrice aléatoire sparse S (densité=0.3) avec $S(k, k) = 1, \forall k$.
- calcul $X = WSW^T$,

Tests synthétiques

- $n = [10, 20, 50, 100, 150, 200]$ avec $r = [2, 3, 4, 5, 6, 7]$, 100 tests.
- matrice aléatoire W tq $W^T W = I$.
- matrice aléatoire sparse S (densité=0.3) avec $S(k, k) = 1, \forall k$.
- calcul $X = WSW^T$,
- Notre algorithme tente de retrouver la factorisation de X
  - Initiation random et kmeans clustering
  - maxiter $= 1000$
  - $\delta error > 10^{-5}$.

Tests synthétiques

- code en Julia
- Précision = Pourcentage de nœuds correctement classés.
- Soit π_k un vecteur booléen tel que $\pi_k[i] = \text{true}$ si et seulement si le nœud i appartient au cluster k. De plus, en définissant π'_k comme le cluster k trouvé, la précision peut être calculée :

$$\text{Précision} = \max_{P \in \text{Permutations}} \frac{1}{n} \sum_{k=1:r} (\pi'_k[P] \wedge \pi_k).$$

Tests synthétiques

- code en Julia
- Précision = Pourcentage de nœuds correctement classés.
- Soit π_k un vecteur booléen tel que $\pi_k[i] = \text{true}$ si et seulement si le nœud i appartient au cluster k. De plus, en définissant π'_k comme le cluster k trouvé, la précision peut être calculée :

$$\text{Précision} = \max_{P \in \text{Permutations}} \frac{1}{n} \sum_{k=1:r} (\pi'_k[P] \wedge \pi_k).$$

- Un succès = lorsque notre modèle classe correctement tous les nœuds, précision de 100%.

Tests synthétiques

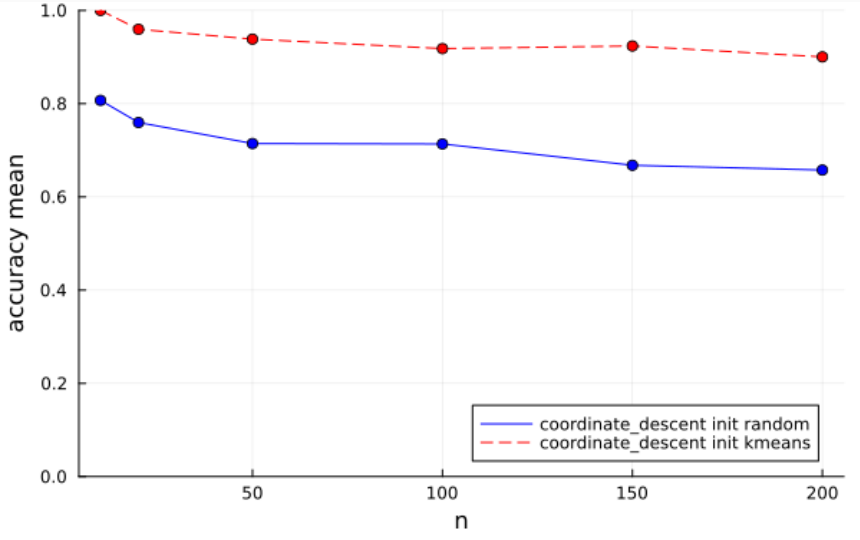
- code en Julia
- Précision = Pourcentage de nœuds correctement classés.
- Soit π_k un vecteur booléen tel que $\pi_k[i] = \text{true}$ si et seulement si le nœud i appartient au cluster k. De plus, en définissant π'_k comme le cluster k trouvé, la précision peut être calculée :

$$\text{Précision} = \max_{P \in \text{Permutations}} \frac{1}{n} \sum_{k=1:r} (\pi'_k[P] \wedge \pi_k).$$

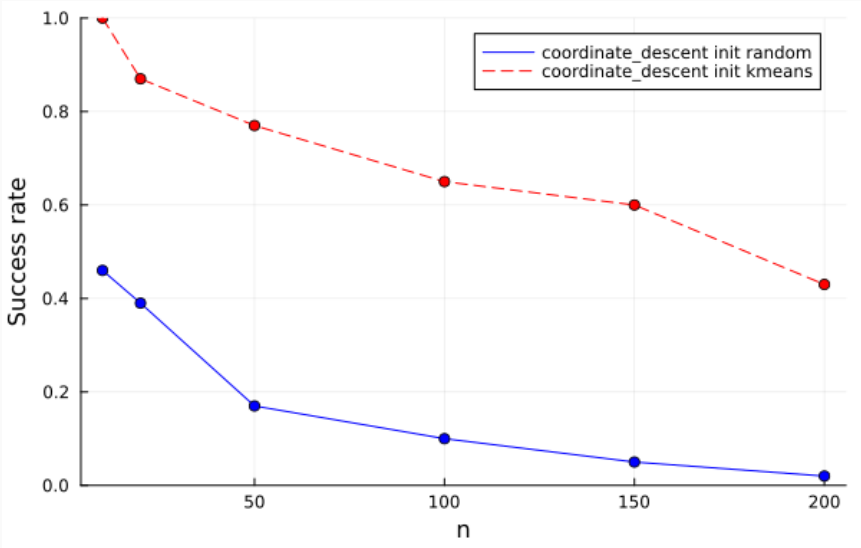
- Un succès = lorsque notre modèle classe correctement tous les nœuds, précision de 100%.
- la précision moyenne et le taux de succès et l'erreur relative :

$$\text{erreur} = \frac{\|X - WSW^T\|_F}{\|X\|_F}.$$

Tests synthétiques - précision

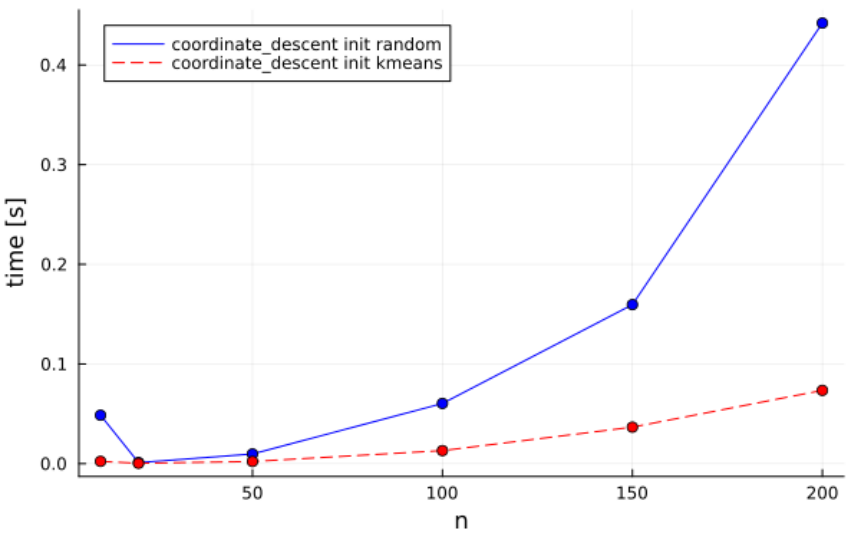


Evolution de la précision moyenne

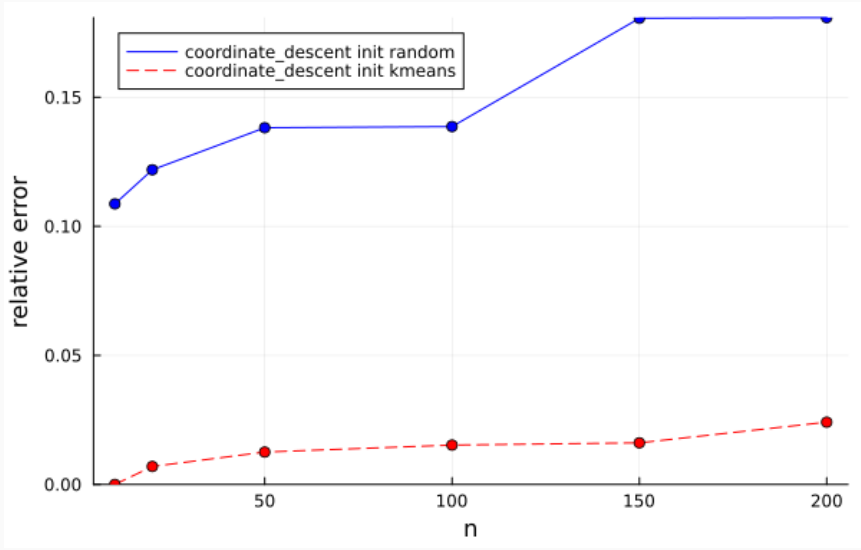


Evolution du taux de succès

Tests synthétiques - erreur & temps



Evolution du temps de résolution



Evolution de l'erreur relative

Tests synthétiques bruit et MU

- Comparaison avec les **multiplicatives updates MU** :

$$W_{ik} \leftarrow W_{ik} \sqrt{\frac{(X^T WS)_{ik}}{(WW^T X^T WS)_{ik}}}$$

$$S_{kl} \leftarrow S_{kl} \sqrt{\frac{(W^T XW)_{kl}}{(W^T WSW^T W)_{kl}}}.$$

Tests synthétiques bruit et MU

- Comparaison avec les **multiplicatives updates MU** :
  $W_{ik} \leftarrow W_{ik} \sqrt{\frac{(X^T WS)_{ik}}{(WW^T X^T WS)_{ik}}}$
  $S_{kl} \leftarrow S_{kl} \sqrt{\frac{(W^T XW)_{kl}}{(W^T WSW^T W)_{kl}}}$.
- Soient $\epsilon > 0$ et N une matrice de bruit gaussien $n \times n$:

$$X = X + \epsilon N \frac{\|X\|_F}{\|N\|_F}.$$

- Pour différentes valeurs de ϵ, nous testons les algorithmes 200 fois pour vérifier s'ils parviennent à trouver les clusters même en présence de bruit.
- $n = 200$ et $r = 8$.
- initialisation avec k-means
- Critères d'arrêts : (Δerreur $> 10^{-5}$) ou temps max de 2s (pas de maxiter)

Tests synthétiques bruit et MU

Results of the multiplicative updates (MU) and our coordinate descent algorithm on noisy data

	noise ϵ	0	0.25	0.5	0.75	1
Score	MU	34%	23.5%	10%	9%	4%
rate	Our algo	**42.5%**	**26%**	**17.5%**	**10%**	**15%**
Accuracy	MU	91.1%	86.7%	82.2%	81.7%	78.9%
mean	Our algo	90.5%	86.4%	84.1%	81.0%	82.1%
Time [s]	MU	0.0104	0.0086	0.089	0.0091	0.0088
	Our algo	0.135	0.126	0.131	0.136	0.148

- La méthode MU converge vers une solution plus rapidement que notre algorithme.
- Notre algorithme identifie plus fréquemment la solution optimale que MU.

Conclusion et Perspectives

- Conclusion :
  - Etude du problème de trifactorisation matriciel symétrique non négative avec la contrainte orthogonale sur W.
  - Nouvelle méthode de descente de coordonnées et son implémentation en Julia en n^2r^2
  - Démonstration de l'efficacité de cet algorithme sur des données synthétiques.
- Perspectives : Tests sur données réelles

Merci!

Contact: alexandra.dache@student.umons.ac.be

UMONS

University of Mons