Partie III

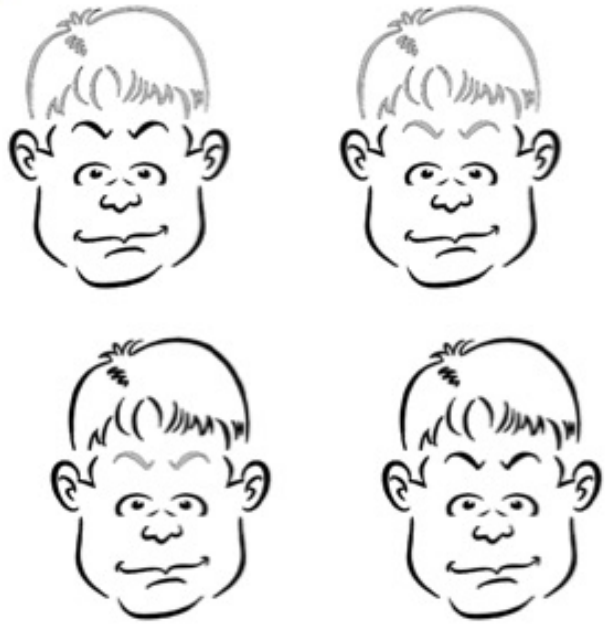
- **Partie III : Factorisation matricielle non négative**
  Nonnegative Matrix Factorization
  - Objectif de la NMF
  - Pourquoi la nonnégativité ?
  - NMF de base
  - Semi-NMF
  - Orthogonal-NMF
  - Clustering par K-means vs NMF
  - Convex-NMF
  - Projective-NMF
  - Tri-NMF
  - NMF symétrique
  - Kernel NMF
  - Multi-Layer-NMF
  - Simultaneous-NMF
  - Convolutive-NMF
  - Multi-View-NMF

Objectif de la Non-negative Matrix Factorization (NMF)

Supposons que les dessins sont donnés à une méthode non supervisée pour l'analyse du problème.

Les figures sont composées de trois objets de base à savoir les **cheveux**, les **sourcils** et le **reste du visage**.

On peut voir que les cheveux et les sourcils peuvent être soit de couleur noire soit gris.



La tâche consiste à construire des algorithmes qui sont en mesure de **trouver ces trois objets** de base et sont également en mesure de **déterminer l'intensité** de ceux-ci dans chaque dessin.

Pourquoi la non-négativité ?

Beaucoup de données du monde réel sont positives ou nulles et les composantes cachées correspondantes ont une signification physique seulement lorsqu'elles ne sont pas négatives.

Certains jeux de données sont intrinsèquement non-négatifs :

- Compteurs (occurrences de chaque mot dans un document texte)
- Quantités (quantité de chaque ingrédient dans une expérience chimique)
- Intensités (intensité de chaque couleur dans une image)

Les décompositions de type SVD peuvent comporter des valeurs négatives sans aucune interprétation naturelle !

Pourquoi la « sparsity » ?

Une représentation parcimonieuse des données en un nombre limité de composants (cachés) est un problème de recherche très important en apprentissage artificiel.

Les contraintes de parcimonie "sparsity" peuvent augmenter l'efficacité et la qualité d'un dictionnaire de composants (prototypes), tandis que la positivité accroît le sens physique ou physiologique des traits calculés.

NMF de base

NMF a été étudiée par de nombreux chercheurs, par exemple, **Paatero et Tapper** [1], mais il a gagné en popularité à travers les œuvres de **Lee Seung** publiés dans *Nature* et *NIPS* [2,3].

Le problème de **NMF** de base peut être énoncé comme suit :
Soit une matrice **X** de n vecteurs colonnes de données positives

$$X = (x_1, x_2, \ldots, x_n) \quad X \in \Re_+^{p \times n}$$

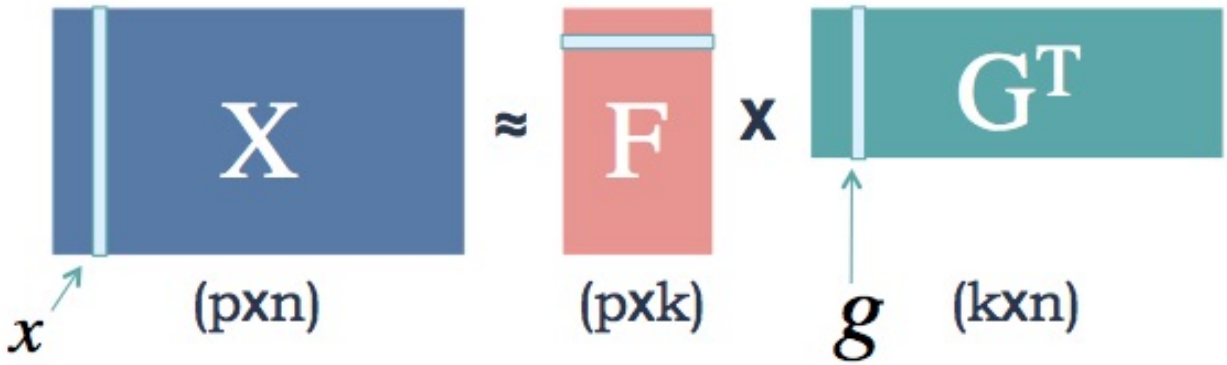
Trouver deux matrices **F** et **G** positives tel que :

$$X \approx F\,G^T \qquad X \in \Re_+^{p \times n}, F \in \Re_+^{p \times k}, G \in \Re_+^{n \times k}$$
$$k << \min(p,n)$$

1. P. Paatero and U. Tapper. Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values. *Environmetrics, 5:111–126, 1994.*
2. D.D. Lee and H.S. Seung. Learning of the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.
3. D.D. Lee and H.S. Seung. *Algorithms for Nonnegative Matrix Factorization, volume 13. MIT Press, 2001.*

NMF de base

$$X_+ \approx F_+ G_+^T \qquad X \approx F G^T \qquad X \in \Re_+^{p \times n}, F \in \Re_+^{p \times k}, G \in \Re_+^{n \times k}$$
$$k << \min(p,n)$$



Quelle est la signification de cette approximation ?

$$x \approx F\, g$$

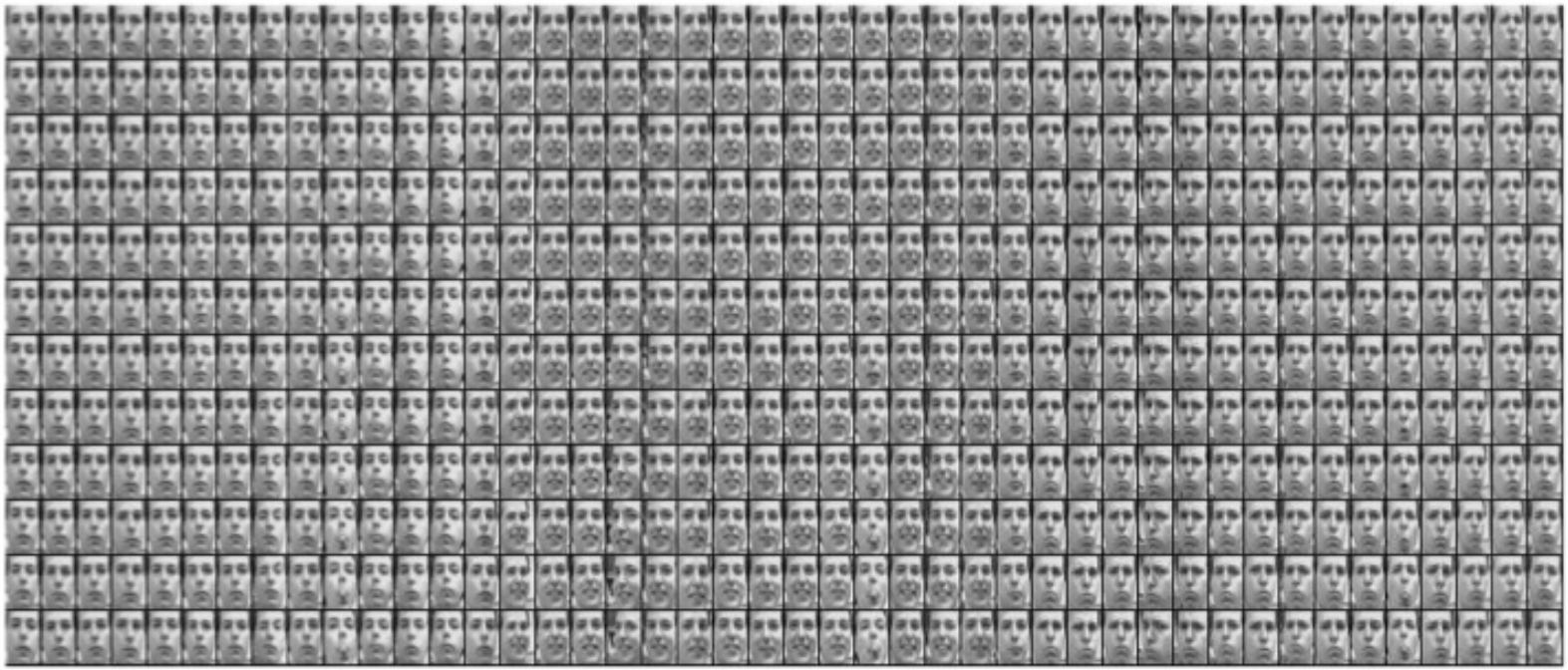
Chaque donnée **x** est approximée par une combinaison linéaire des colonnes de **F** (prototypes), pondérée par les composantes de **g** élément de **G** (partition).

Exemple illustratif de NMF

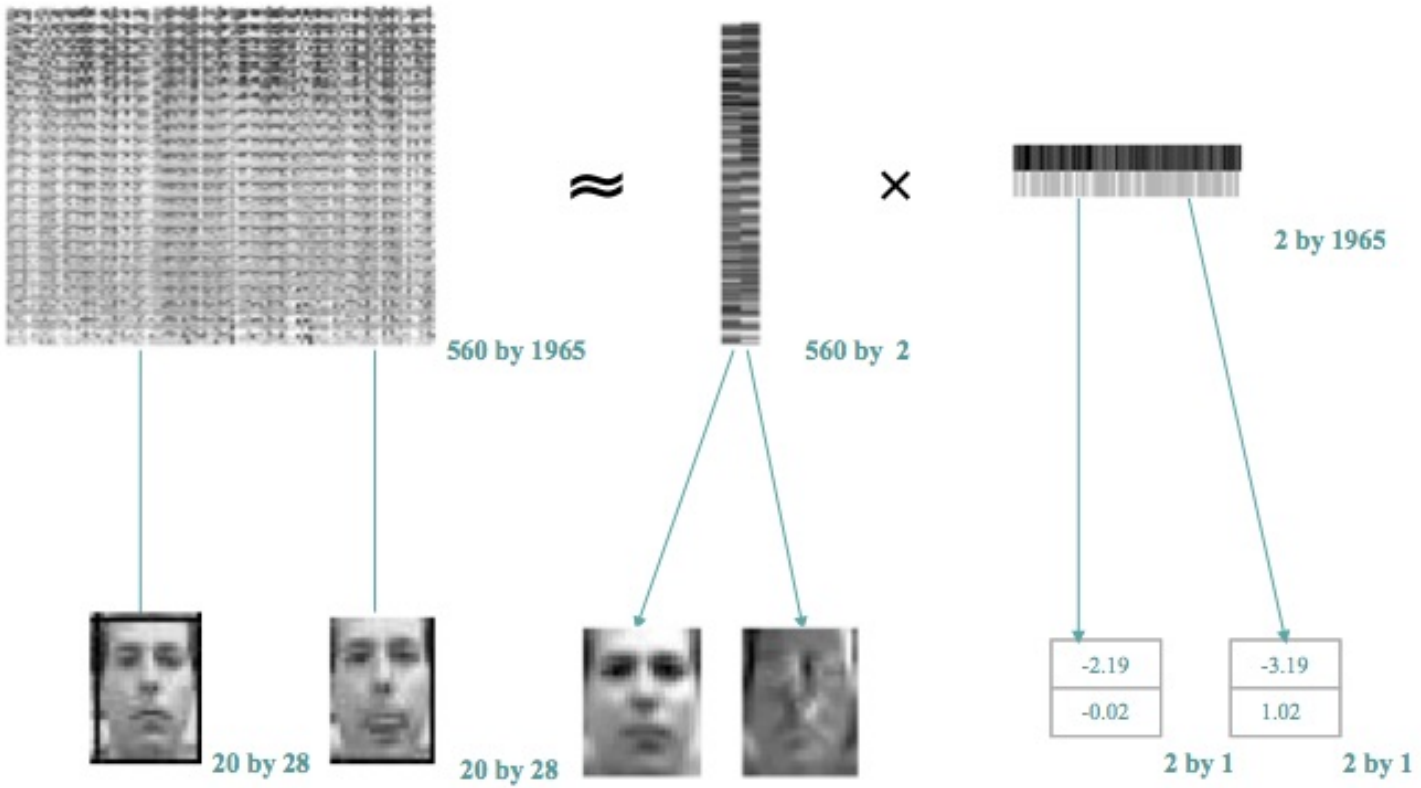
$$\begin{pmatrix} 0.185 & 0.326 & 0.761 & 2.799 & 2.375 & 2.970 & 2.585 \\ 0.508 & 0.380 & 0.884 & 2.134 & 2.374 & 2.342 & 2.524 \\ 0.452 & 0.887 & 0.457 & 2.065 & 2.484 & 2.253 & 2.163 \\ 1.486 & 1.843 & 1.858 & 0.566 & 0.103 & 0.417 & 0.269 \\ 1.496 & 1.806 & 1.610 & 0.612 & 0.158 & 0.560 & 0.784 \end{pmatrix} \approx \begin{pmatrix} 0.0403 & 0.3695 \\ 0.0889 & 0.3149 \\ 0.1033 & 0.2945 \\ 0.3882 & 0.0002 \\ 0.3794 & 0.0210 \\ \hline 16.83 & 30.64 \end{pmatrix} \times \begin{pmatrix} 0.234 & 0.287 & 0.259 & 0.080 & 0.012 & 0.063 & 0.065 \\ 0.006 & 0.014 & 0.040 & 0.223 & 0.238 & 0.244 & 0.236 \end{pmatrix}$$

$$X_+ \approx F_+ G_+^T$$

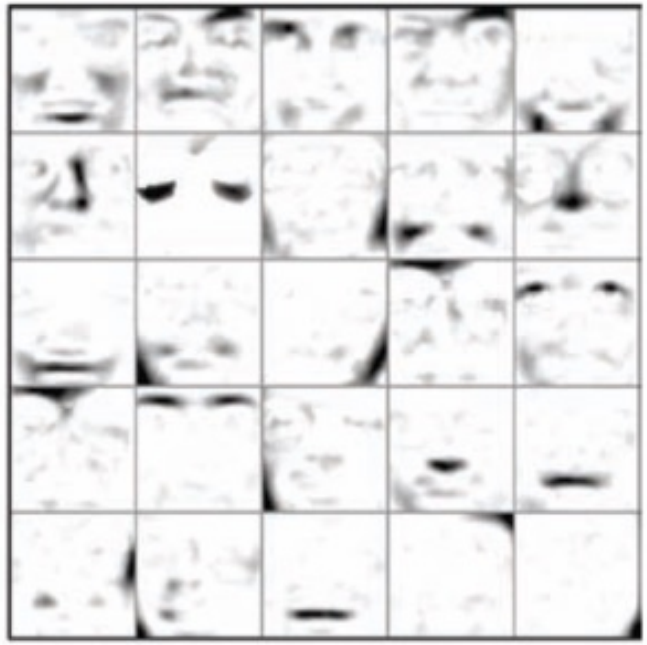
NMF de base



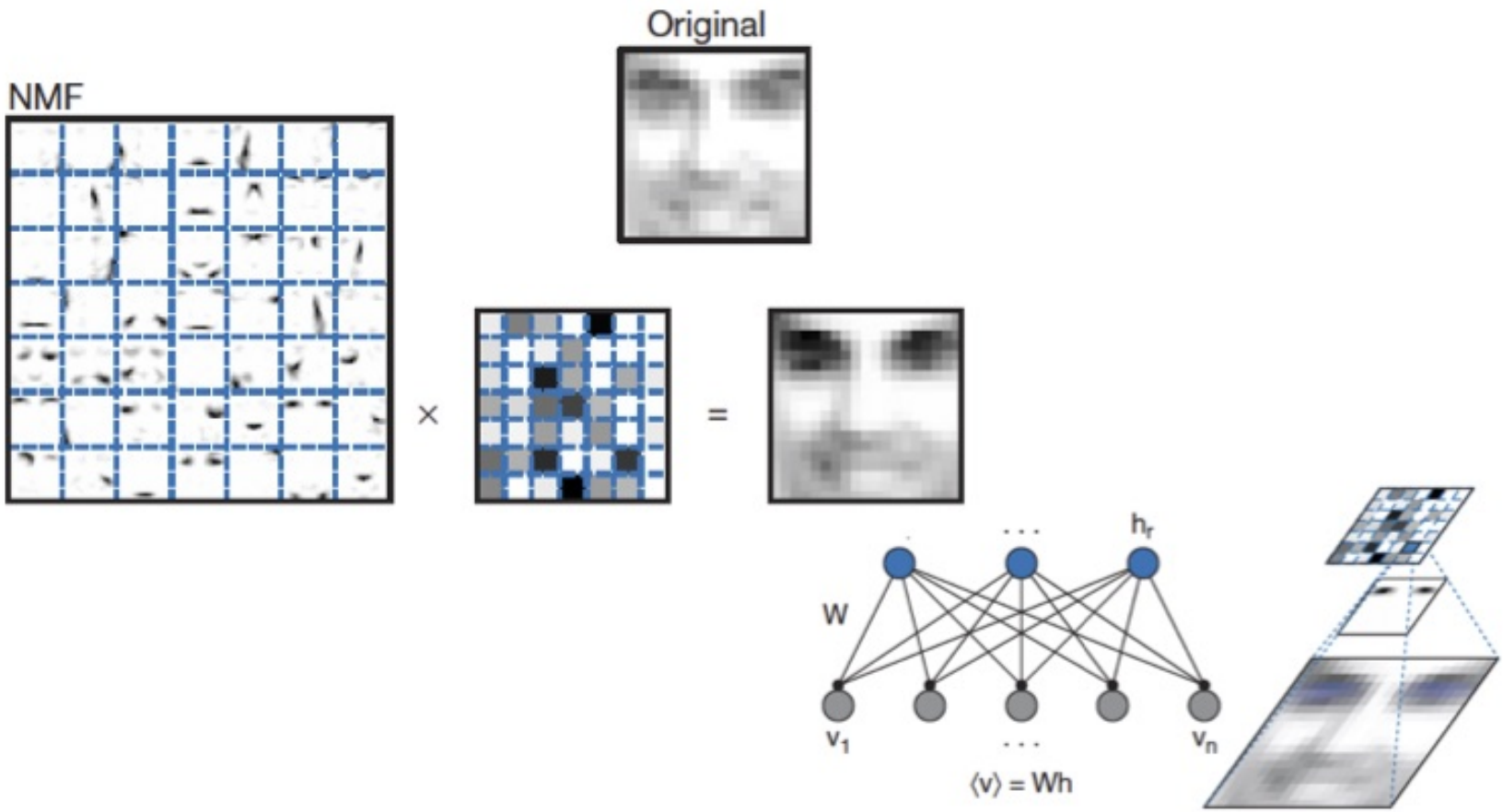
NMF de base


560 by 1965
560 by 2
2 by 1965
20 by 28
20 by 28
-2.19
-0.02
-3.19
1.02
2 by 1
2 by 1

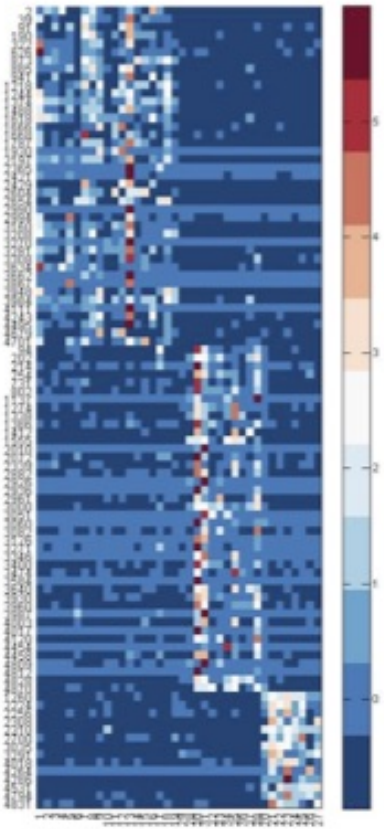
NMF de base

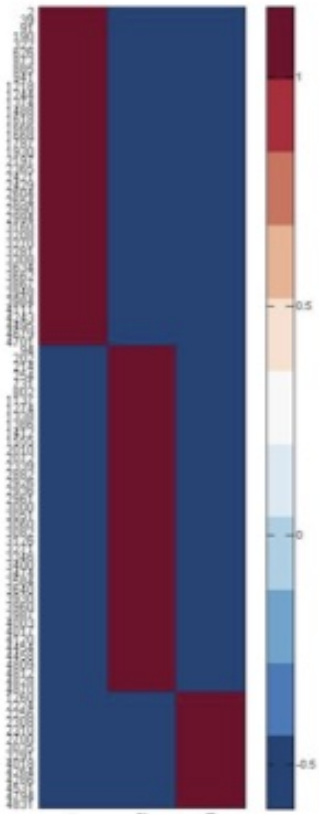


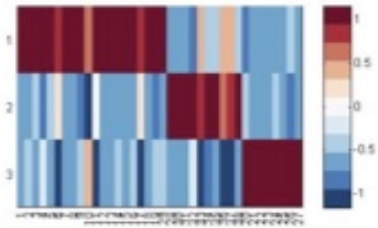
NMF de base



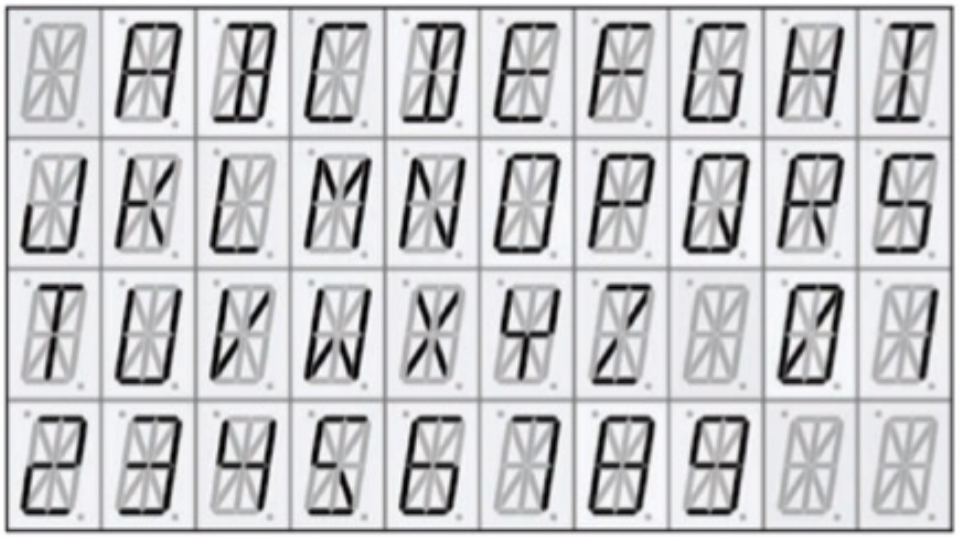
NMF de base

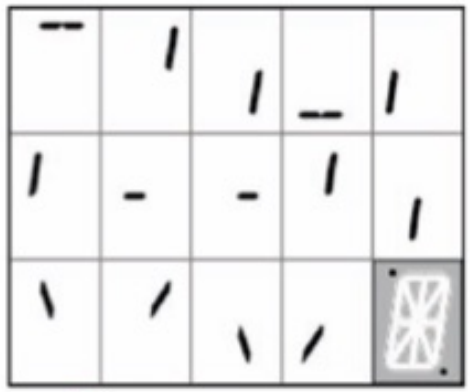






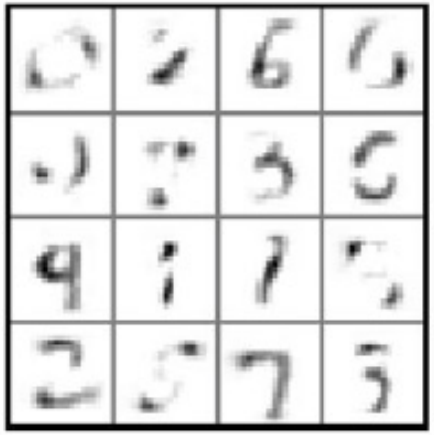
NMF de base

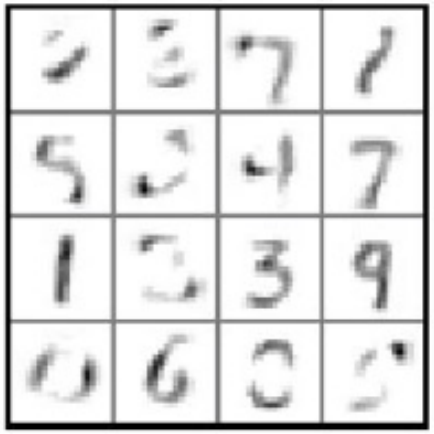




$$X \approx F\,G^T$$

NMF de base





Fonction de coût de NMF

La norme matricielle de Frobenius :

$$
\begin{cases}
\{F,G\} = \arg\min\limits_{F,G} D_F\left(X \| FG\right) \\
\qquad\quad = \arg\min\limits_{F,G} \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{m} \left(x_{ij} - \left(FG^T\right)_{ij}\right)^2 \\
\qquad\quad = \arg\min\limits_{F,G} \left\| X - F\ G^T \right\|_F^2 \\
s.c \quad F \geq 0, G \geq 0
\end{cases}
$$

Règles d'adaptation : méthode multiplicative

Différentes implémentations itératives de ce critère sont possibles : les plus connues sont les méthodes multiplicatives de mise à jour.

Lee et al [*] mentionnent que la mise à jour suivante constitue un bon compromis entre rapidité et facilité d'implémentation.

Théorème : La norme de Frobenius est noncroissante avec les règles de mise à jour multiplicatives suivantes :

$$F_{ik} \leftarrow F_{ik} \otimes \frac{(XG)_{ik}}{\left(FG^T G\right)_{ik}} \qquad G_{jk} \leftarrow G_{jk} \otimes \frac{\left(X^T F\right)_{jk}}{\left(GF^T F\right)_{jk}}$$

Où X⊗Y et X/Y désignent respectivement le produit de Hadamard et la division élément par élément. Avec cette mise à jour, la norme de Frobenius est noncroissante, au pire elle devient invariante si l'on a atteint un point stationnaire qui cependant pas une garantie d'un minimum global du critère.

* Lee Daniel D. et Sebastian Seung H. Learning the parts of ob-jects by non negative matrix factorization.Nature. vol. 401, n °6755, pp. 788-791, 1999.

Règles d'adaptation : méthode multiplicative

L'algorithme précédent, souvent appelé l'algorithme de Lee-Seung peut être considéré comme une extension naturelle de l'algorithme **ISRA** (Image Space Reconstruction Algorithm) proposé par Daube-Witherspoon et Muehllehner (*) et étudié par de nombreux chercheurs, en particulier, De Pierro et Byrne (**).

L'algorithme ISRA est relativement lent, et de nombreuses approches heuristiques ont été proposées pour l'accélérer. Afin de réaliser une convergence plus rapide, une approche de relaxation augmente les coefficients multiplicateurs avec une puissance $\omega \in (0, 2]$ qui est :

$$F_{ik} \leftarrow F_{ik} \otimes \left(\frac{(XG)_{ik}}{(FG^T G)_{ik}} \right)^{\omega} \qquad G_{jk} \leftarrow G_{jk} \otimes \left(\frac{(X^T F)_{jk}}{(GF^T F)_{jk}} \right)^{\omega}$$

* M.E. Daube-Witherspoon and G. Muehllehner. An iterative image space reconstruction algorthm suitable for volume ECT. *IEEE Transactions on Medical Imaging, 5:61–66, 1986.*

** A.R. De Pierro. On the relation between the ISRA and the EM algorithm for positron emission tomography. *IEEE Transactions on Medial Imaging, 12(2):328–333, June 1993.*

Règles d'adaptation : méthode multiplicative

Pour comprendre l'origine des règles de mise à jour précédentes, considérons les conditions d'optimalité du premier ordre de Karush-Kuhn-Tucker (KKT) :

$$\begin{cases} F \geq 0 & G \geq 0 \\ \nabla_F = \dfrac{\partial D_F}{\partial F} \geq 0 & \nabla_G = \dfrac{\partial D_F}{\partial G} \geq 0 \\ F \otimes \nabla_F = 0 & G \otimes \nabla_G = 0 \end{cases}$$

$$\begin{cases} \dfrac{\partial D_F}{\partial F} = FG^T G - XG \\ \dfrac{\partial D_F}{\partial G} = GF^T F - X^T F \end{cases} \longrightarrow \begin{cases} F \otimes FG^T G = F \otimes XG \\ G \otimes GF^T F = G \otimes X^T F \end{cases}$$

$$\begin{cases} F_{ik} \leftarrow F_{ik} \otimes \dfrac{(XG)_{ik}}{(FG^T G)_{ik}} \\ G_{jk} \leftarrow G_{jk} \otimes \dfrac{(X^T F)_{jk}}{(GF^T F)_{jk}} \end{cases}$$

Les conditions Karush-Kuhn-Tucker (KKT) sont contenues dans un système d'équations et d'inégalités que la solution d'un problème de programmation non linéaire doit satisfaire lorsque la fonction objective et les contraintes sont différentiables. Les conditions KKT sont nécessaires pour qu'une solution en Programmation non linéaire soit optimale. Il s'agit d'une généralisation de la méthode des multiplicateurs de Lagrange pour des contraintes d'inégalité qui fournit une stratégie de recherche du minimum d'une fonction de coût soumis à des contraintes.

Règles d'adaptation : Méthode ALS

Une variante est la méthode des moindres carrés alternés (**ALS** : Alternative Least Squares). Il s'agit de trouver **F** (resp. **G**) en fixant **G** (resp. **F**).

$$\begin{cases} F^{(k+1)} = \arg\min_{F} \left\| X - F\, G^{(k)T} \right\|_F^2 \\ G \text{ fixe} \\ s.c \quad F \geq 0 \end{cases} \qquad \begin{cases} G^{(k+1)} = \arg\min_{G} \left\| X - F^{(k+1)} G^{T} \right\|_F^2 \\ F \text{ fixe} \\ s.c \quad G \geq 0 \end{cases}$$

$$\begin{cases} \dfrac{\partial D_F}{\partial F} = FG^TG - XG = 0 \\ \dfrac{\partial D_F}{\partial G} = GF^TF - X^TF = 0 \end{cases} \longrightarrow \begin{cases} F \leftarrow XG\left(G^TG\right)^{-1} \\ G \leftarrow X^TF\left(F^TF\right)^{-1} \end{cases}$$

NB : La résolution de ce problème est en général plus gourmande en temps de calcul que la résolution par mise à jour multiplicative.

Autres fonctions de coût de NMF

D'autres fonctions de coût sont aussi populaires en particulier celles utilisant la divergence de Kullback-Leibler :

$$\begin{cases} \{F,G\} = \arg\min_{F,G} D_{KL}\left(X \| FG\right) \\ \qquad = \sum_{ij}\left(x_{ij} \ln \frac{x_{ij}}{[FG]_{ij}} - x_{ij} + [FG]_{ij} \right) \\ s.c \quad F \geq 0, G \geq 0 \end{cases}$$

Règles de mise à jour

Théorème : La divergence de Kullback-Leibler est noncroissante avec les règles de mise à jour multiplicatives suivantes :

$$F_{ik} \leftarrow F_{ik} \otimes \frac{\sum_{\mu} G_{k\mu} X_{i\mu} / (FG)_{i\mu}}{\sum_{\nu} G_{k\nu}}$$

$$G_{k\mu} \leftarrow G_{k\mu} \otimes \frac{\sum_{i} F_{ik} X_{i\mu} / (FG)_{i\mu}}{\sum_{j} F_{jk}}$$

Convexité de NMF

Bien que le problème d'optimisation **NMF** n'est pas convexe, les fonctions objectives sont séparément convexe dans chacun des deux facteurs **F** et **G**, ce qui implique que la recherche de la matrice de coefficients optimale **F** correspondant à une matrice fixe **G** se réduit à un problème d'optimisation convexe et vice-versa.

Cependant, la convexité est perdue dès que nous essayons d'optimiser les deux matrices de facteurs en même temps.

Non unicité de NMF

La factorisation **n'est pas unique** : une matrice et son inverse peuvent être utilisées pour transformer les deux matrices de factorisation par :

$$\mathbf{F}\mathbf{G}^{\mathbf{T}} = \mathbf{F}\,\mathbf{B}\mathbf{B}^{-1}\,\mathbf{G}^{\mathbf{T}}$$

Si les deux nouvelles matrices **FB** et $\mathbf{B}^{-1}\mathbf{G}^{\mathbf{T}}$ sont non-négatives, elles forment une autre solution de la factorisation.

Le non-négativité de **FB** et $\mathbf{B}^{-1}\mathbf{G}^{\mathbf{T}}$ est vérifiée si **B** est une matrice non-négative.

Plus de contrôle sur la non-unicité de NMF est obtenu avec des contraintes de parcimonie (sparsity).

Initialisation de NMF

La solution et la convergence fournies par des algorithmes **NMF** dépendent fortement des conditions initiales, surtout dans un contexte multivarié.

Ainsi, il est important d'avoir des moyens cohérents et efficaces pour initialiser les matrices **F** et/ou **G**.

En d'autres termes, l'efficacité de nombreuses **NMF** est affectée par le choix des matrices de départ.

Des initialisations pauvres donnent souvent lieu à une convergence lente, et dans certains cas, peut même conduire à une solution erronée, ou non pertinente.

Initialisation de NMF

Listing Basic initializations for NMF algorithms.

```
1 function [Ainit ,Xinit] = NMFinitialization(Y,J,inittype)
2 % X : nonnegative matrix
3 % J : number of components
4 % inittype 1 { random}, 2 {ALS}, 3 {SVD}
5 [I,T] = size(X);
6 Finit = rand(I,J);
7 Ginit = rand(J,T);
8
9 switch inittype
10       case 2 % ALS
11               Finit = max(eps,(X*Ginit ')*pinv(Ginit*Ginit '));
12               Ginit = max(eps,pinv(Finit '*Finit)*(Finit '*X));
13       case 3 %SVD
14               [Finit ,Ginit] = lsvNMF(X,J);
15 end
16 Finit = bsxfun(@rdivide ,Finit ,sum(Finit));
17 end
```

Semi-NMF

Lorsque les données d'entrée présentent des signes positifs et négatifs, nous pouvons imposer à **G** la positivité mais pas à **F** :

$$X = (x_1, x_2, \ldots, x_n) \quad X \in \Re_+^{p \times n}$$

$$X_{\pm} \approx F_{\pm} G_+^T$$

$$X \in \Re_{\pm}^{p \times n}, F \in \Re_{\pm}^{p \times k}, G \in \Re_+^{n \times k}$$

$$k << \min(p,n)$$

Règles d'adaptation : Semi-NMF

Les règles d'adaptation dans le cas de la **Semi-NMF** se présentent de la façon suivante :

$$F = XG\left(G^T G\right)^{-1}$$

$$G_{ik} \leftarrow G_{ik}\sqrt{\frac{\left(X^T F\right)_{ik}^{+} + \left[G\left(F^T F\right)^{-}\right]_{ik}}{\left(X^T F\right)_{ik}^{-} + \left[G\left(F^T F\right)^{+}\right]_{ik}}}$$

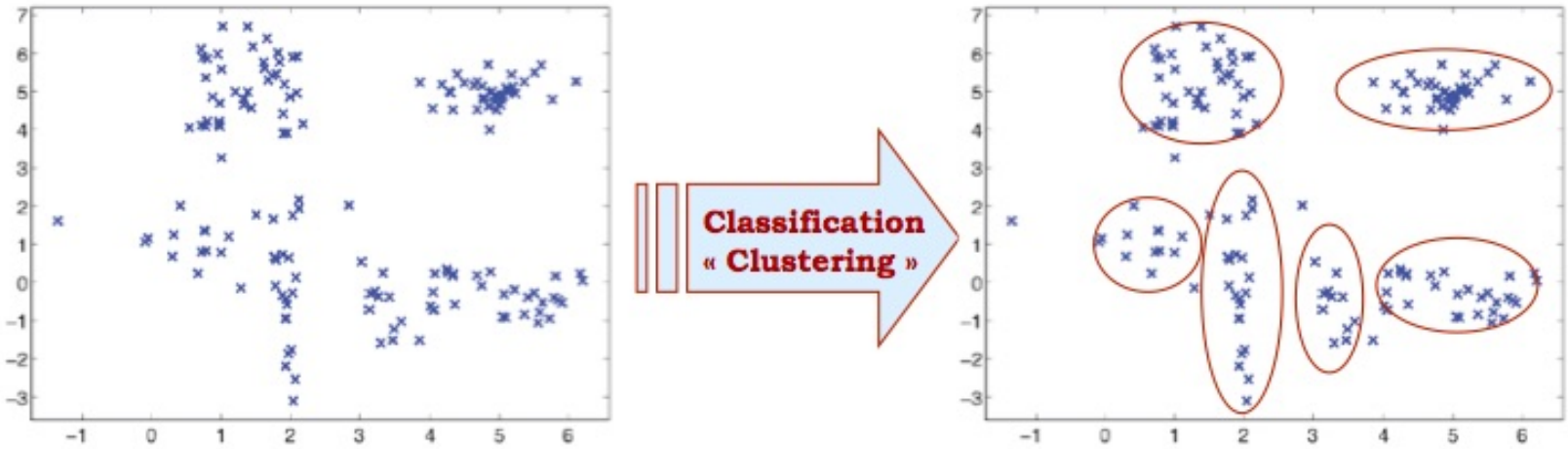
Où les parties positive et négative d'une matrice A sont définies par :

$$A_{ik}^{+} = (|A_{ik}| + A_{ik})/2,\ A_{ik}^{-} = (|A_{ik}| - A_{ik})/2$$

Clustering par K-means vs NMF

Objectif du clustering :

produire des groupements homogènes à partir d'un ensemble d'observations (formes) ou d'une matrice de (di)ssimilarités.



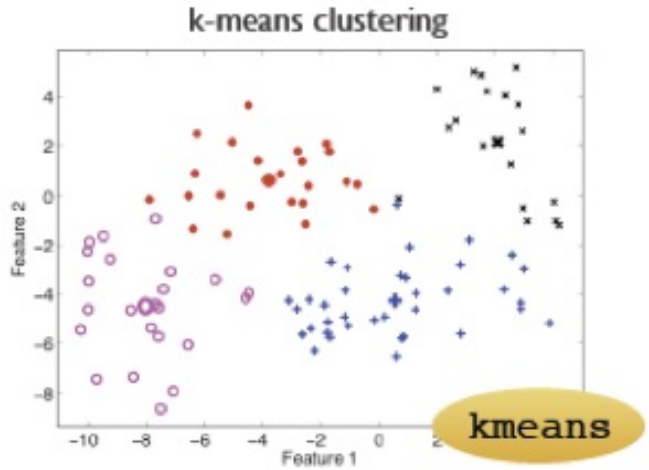
Clustering par K-means vs NMF

Fonction de coût K-means :

$$D_{K-means} = \sum_i \|x_i\|^2 - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,j \in C_k} x_i^T x_j$$

$$G = (g_1, g_2, \ldots, g_K)$$

$$g_k = \left(0 \ldots 0, \overbrace{1 \ldots 1}^{n_k}, 0 \ldots 0\right)^T / n_k^{1/2}$$



$$D_{K-means} = \sum_i x_i^2 - \sum_{k=1}^{K} g_k^T X^T X g_k$$

$$\longrightarrow D_{K-means} \begin{cases} \max_{G \geq 0} Tr\left(G^T X^T X G\right) \\ s.c \quad G^T G = I \end{cases}$$

Clustering par K-means vs NMF

Théorème : Orthogonal-NMF,

Cluster indicators

$$\begin{cases} \{F,G\} = \arg\min_{F,G} \left\| X_{\pm} - F_{\pm} G_{+}^{T} \right\|_{F}^{2} \\ s.c \quad G \geq 0, G^{T}G = I \end{cases}$$

Cluster centroids

est équivalente à K-means.

Démonstration :

$$D_F(X\|FG) = \left\|X - F\,G^T\right\|_F^2 = Tr\left(X^TX - 2F^TXG + F^TF\right)$$

$$\frac{\partial D_F(X\|FG)}{\partial F} = -2XG + 2F = 0 \Rightarrow F = XG$$

Cte

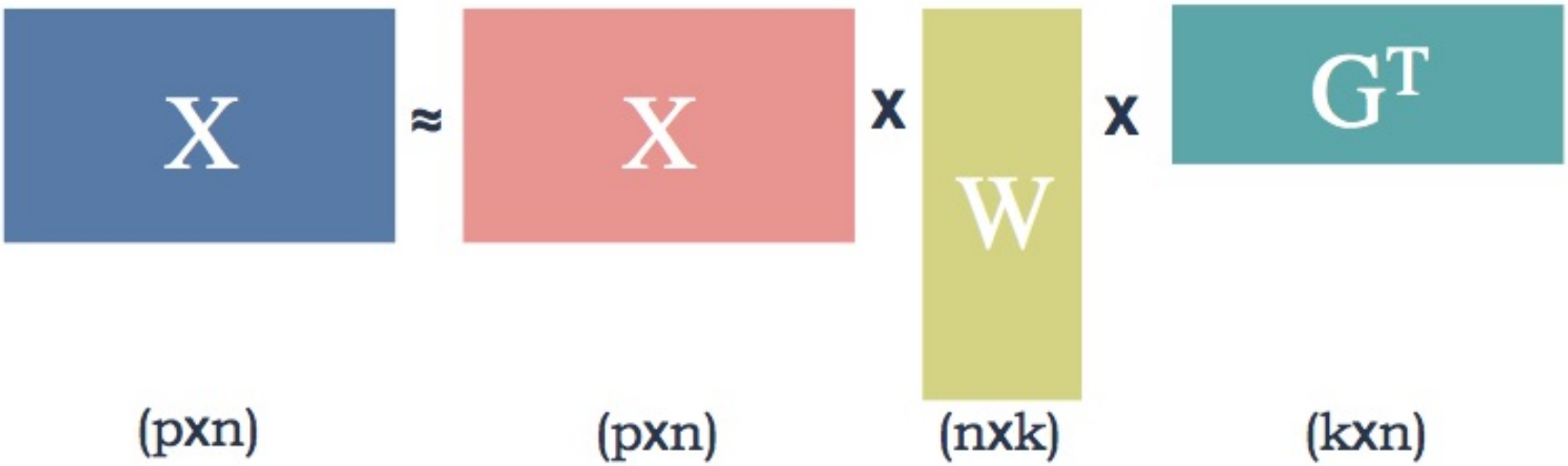
$$D_F(X\|FG) = Tr\left(X^TX - G^TX^TXG\right)$$

$$D_{K-means} \begin{cases} \max_{G\geq 0} Tr\left(G^TX^TXG\right) \\ s.c \quad G^TG = I \end{cases}$$

Convex-NMF

Pour que les vecteurs de **F** puissent capturer la notion de centres de gravité des clusters, nous forçons la forme des éléments de **F** à des combinaisons convexes des données **X** :

$$F = X\ W \qquad X_{\pm} \approx X_{\pm} W_{+} G_{+}^{T}$$



Convex-NMF s'applique à la fois sur des données positives ou nulles et sur des données d'entrée mixtes.

Fonction de coût de Convex-NMF

La norme matricielle de Frobenius :

$$\begin{cases} \{W,G\} = \arg\min_{F,G} D_F\left(X \| XWG\right) \\ \qquad\qquad = \arg\min_{F,G} \left\| X - X\,W\,G^T \right\|_F^2 \\ s.c \quad W \geq 0, G \geq 0 \end{cases}$$

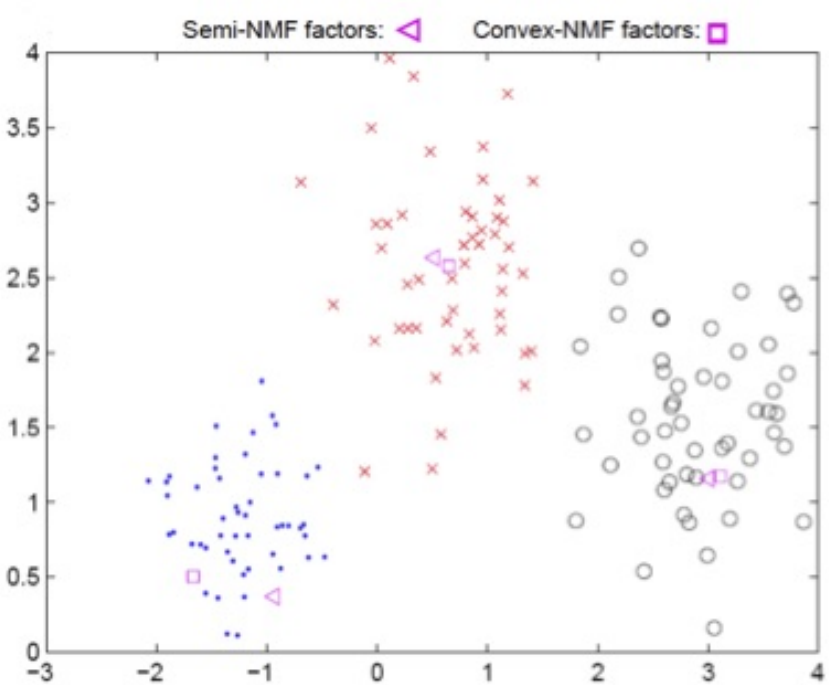
Règles d'adaptation : Convex-NMF

Les règles d'adaptation dans le cas de la **Convex-NMF** se présentent de la façon suivante :

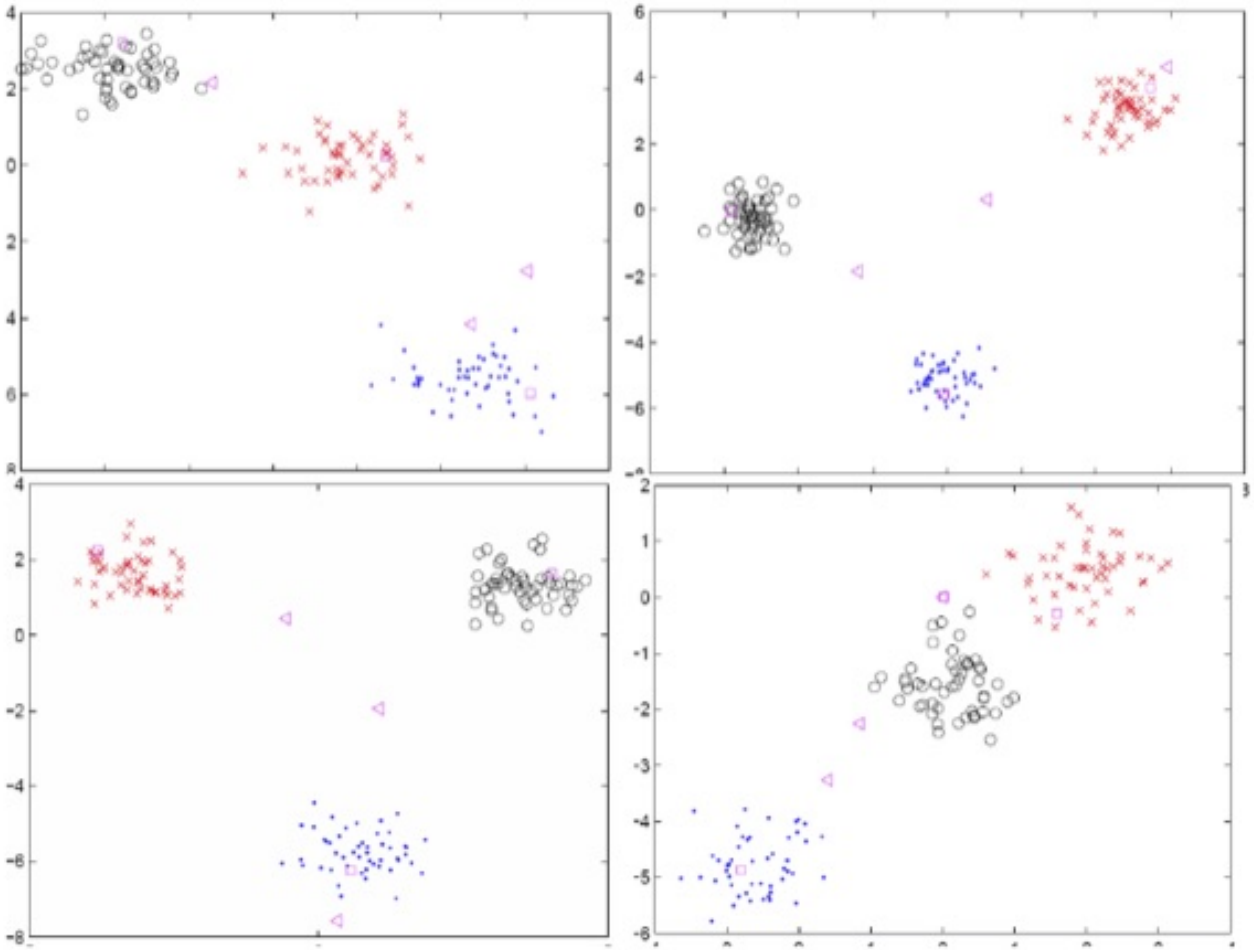
$$G_{ik} \leftarrow G_{ik} \sqrt{\frac{\left[\left(X^T X\right)^+ W\right]_{ik} + \left[G W^T \left(X^T X\right)^- W\right]_{ik}}{\left[\left(X^T X\right)^- W\right]_{ik} + \left[G W^T \left(X^T X\right)^+ W\right]_{ik}}}$$

$$W_{ik} \leftarrow W_{ik} \sqrt{\frac{\left[\left(X^T X\right)^+ G\right]_{ik} + \left[\left(X^T X\right)^- W G^T G\right]_{ik}}{\left[\left(X^T X\right)^- G\right]_{ik} + \left[\left(X^T X\right)^+ W G^T G\right]_{ik}}}$$

Convex-NMF vs Semi-NMF



Convex-NMF vs Semi-NMF



Convex-NMF vs Semi-NMF

Cluster 1 Cluster 2

$$X = \begin{pmatrix} 1.3 & 1.8 & 4.8 & 7.1 & 5.0 & 5.2 & 8.0 \\ 1.5 & 6.9 & 3.9 & -5.5 & -8.5 & -3.9 & -5.5 \\ 6.5 & 1.6 & 8.2 & -7.2 & -8.7 & -7.9 & -5.2 \\ 3.8 & 8.3 & 4.7 & 6.4 & 7.5 & 3.2 & 7.4 \\ -7.3 & -1.8 & -2.1 & 2.7 & 6.8 & 4.8 & 6.2 \end{pmatrix}$$

$$F_{svd} = \begin{pmatrix} -0.41 & 0.50 \\ 0.35 & 0.21 \\ 0.66 & 0.32 \\ -0.28 & 0.72 \\ -0.43 & -0.28 \\ \hline 25.5 & 15.6 \end{pmatrix},$$

$$F_{semi} = \begin{pmatrix} 0.05 & 0.27 \\ 0.40 & -0.40 \\ 0.70 & -0.72 \\ 0.30 & 0.08 \\ -0.51 & 0.49 \\ \hline 20.3 & 23.0 \end{pmatrix}, F_{conv} = \begin{pmatrix} 0.31 & 0.53 \\ 0.42 & -0.30 \\ 0.56 & -0.57 \\ 0.49 & 0.41 \\ -0.41 & 0.36 \\ \hline 31.0 & 39.3 \end{pmatrix},$$

$$G_{svd} = \begin{pmatrix} 0.25 & 0.05 & 0.22 & -0.45 & -0.44 & -0.46 & -0.52 \\ 0.50 & 0.60 & 0.43 & 0.30 & -0.12 & 0.01 & 0.31 \end{pmatrix}$$

$$G_{semi} = \begin{pmatrix} 0.61 & 0.89 & 0.54 & 0.77 & 0.14 & 0.36 & 0.84 \\ 0.12 & 0.53 & 0.11 & 1.03 & 0.60 & 0.77 & 1.16 \end{pmatrix}$$

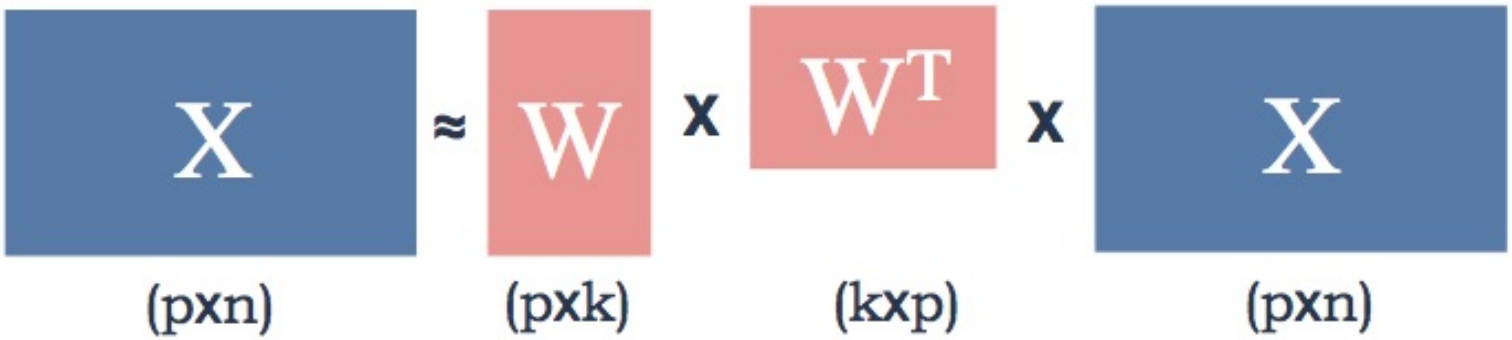
$$G_{conv} = \begin{pmatrix} 0.31 & 0.31 & 0.29 & 0.02 & 0 & 0 & 0.02 \\ 0 & 0.06 & 0 & 0.31 & 0.27 & 0.30 & 0.36 \end{pmatrix}$$

$$\|X - FG^T\| = 0.27940, 0.27944, 0.30877$$

Projective-NMF

Une **NMF** projective peut être formulée comme l'estimation d'une matrice **W** non négative qui satisfait l'équation matricielle :

$$X_{\pm} \approx W_{+} W_{+}^{T} X_{\pm}$$



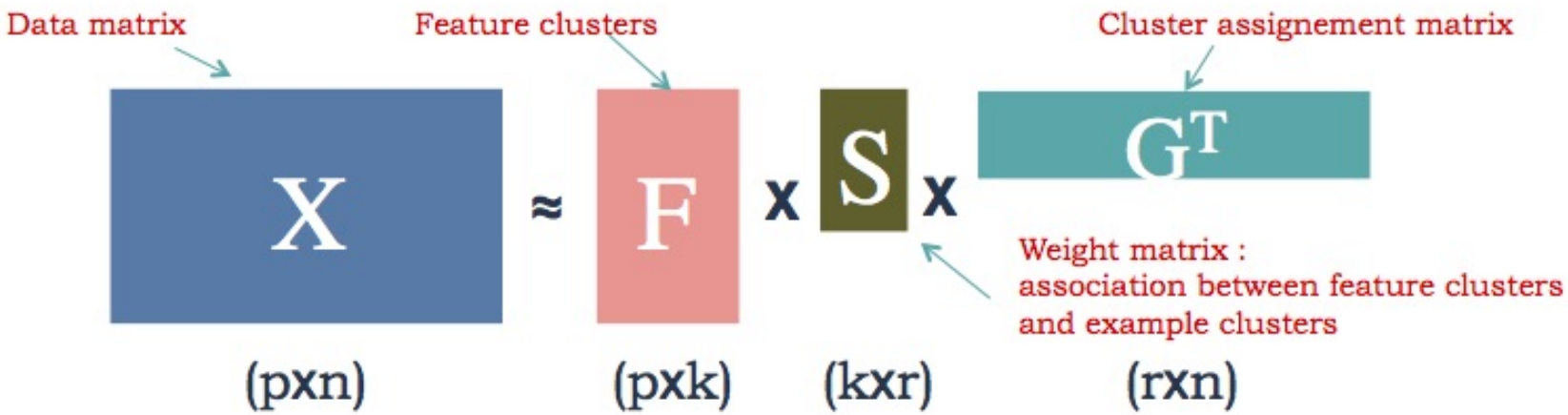
Dans une forme non symétrique plus générale, la **NMF** projective implique l'estimation de deux matrices non négatives **A** et **B** :

$$X_{\pm} \approx A_{+} B_{+}^{T} X_{\pm}$$

Tri-NMF

Pour regrouper simultanément les lignes et les colonnes de la matrice des données **X**, nous considérons la décomposition non négative à trois facteurs suivants :

$$X \approx F\ S\ G^T$$



Tri-NMF

Pour cette tri-factorisation $X \approx F\ S\ G^T$
nous résolvons le problème suivant :

$$\begin{cases} \{F,S,G\} = \arg\min_{F,S,G} D_F\left(X \| FSG\right) \\ \qquad\qquad = \arg\min_{F,S,G} \left\| X - F\ S\ G^T \right\|_F^2 \\ \\ s.c \quad F \geq 0, S \geq 0, G \geq 0 \quad et \quad F^T F = I, G^T G = I \end{cases}$$

Règles d'adaptation : Tri-NMF

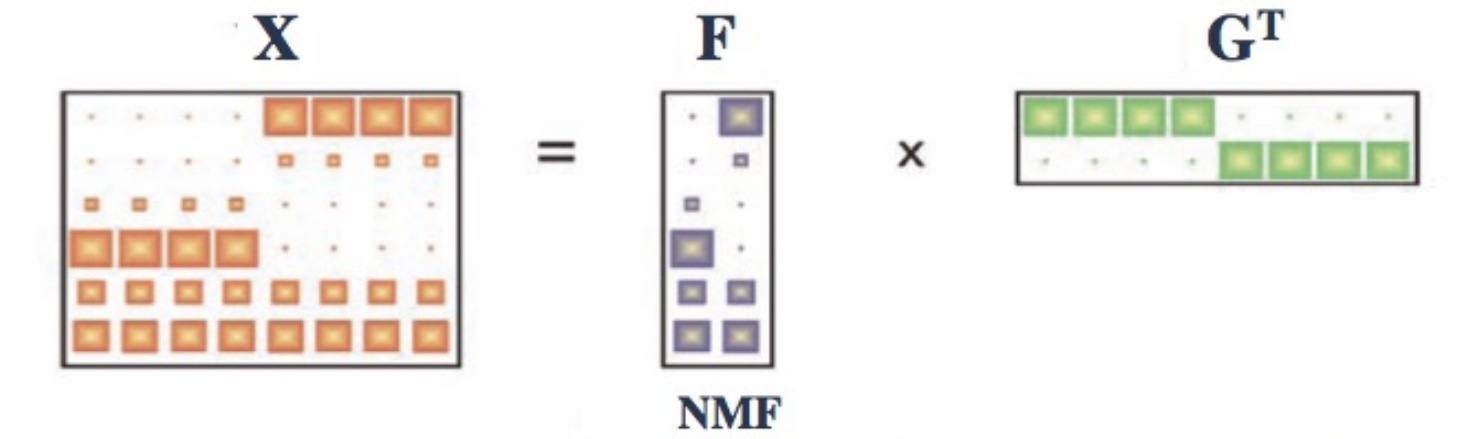
Les règles d'adaptation dans le cas de la **Tri-NMF** se présentent de la façon suivante :

$$G_{jk} \leftarrow G_{jk} \sqrt{\frac{\left(X^T FS\right)_{jk}}{\left(GG^T X^T FS\right)_{jk}}}$$

$$F_{ik} \leftarrow F_{ik} \sqrt{\frac{\left(XGS^T\right)_{ik}}{\left(FF^T XGS^T\right)_{ik}}}$$

$$S_{ik} \leftarrow S_{ik} \sqrt{\frac{\left(F^T XG\right)_{ik}}{\left(F^T FSG^T G\right)_{ik}}}$$

Tri-NMF vs NMF



Tri-NMF vs NMF

Cluster 1 Cluster 2

$$\mathbf{X} \begin{pmatrix} 0.185 & 0.326 & 0.761 & 2.799 & 2.375 & 2.970 & 2.585 \\ 0.508 & 0.380 & 0.884 & 2.134 & 2.374 & 2.342 & 2.524 \\ 0.452 & 0.887 & 0.457 & 2.065 & 2.484 & 2.253 & 2.163 \\ 1.486 & 1.843 & 1.858 & 0.566 & 0.103 & 0.417 & 0.269 \\ 1.496 & 1.806 & 1.610 & 0.612 & 0.158 & 0.560 & 0.784 \end{pmatrix}$$

$$F_{nmf} = \begin{pmatrix} 0.0403 & 0.3695 \\ 0.0889 & 0.3149 \\ 0.1033 & 0.2945 \\ 0.3882 & 0.0002 \\ 0.3794 & 0.0210 \\ \hline 16.83 & 30.64 \end{pmatrix}, F_{Tri} = \begin{pmatrix} 0.0000 & 0.3704 \\ 0.0215 & 0.3228 \\ 0.0320 & 0.3068 \\ 0.4773 & 0.0000 \\ 0.4692 & 0.0000 \\ \hline 2.6172 & 3.4036 \end{pmatrix}$$

$$G_{nmf} = \begin{pmatrix} 0.234 & 0.287 & 0.259 & 0.080 & 0.012 & 0.063 & 0.065 \\ 0.006 & 0.014 & 0.040 & 0.223 & 0.238 & 0.244 & 0.236 \end{pmatrix}$$

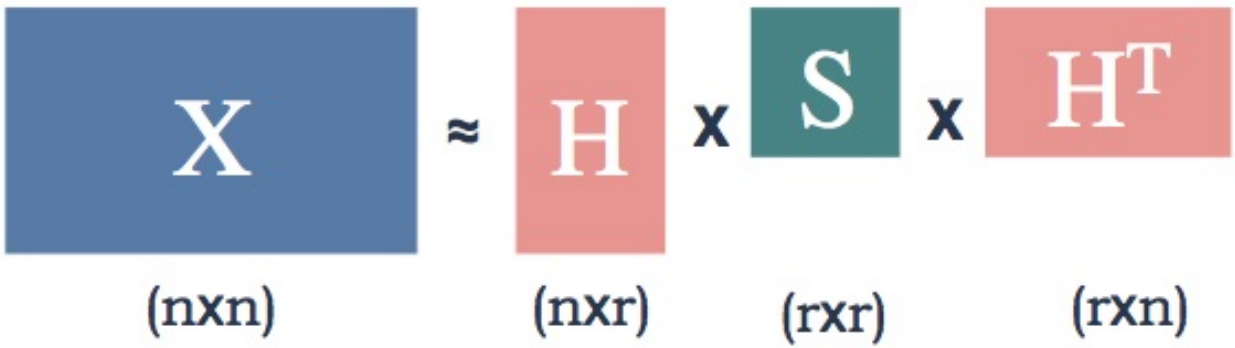
$$S_{Tri-factor} = \begin{pmatrix} 4.3626 & 1.0136 \\ 1.4824 & 8.4000 \end{pmatrix}$$

$$G_{Tri} = \begin{pmatrix} 0.270 & 0.335 & 0.333 & 0.034 & 0.000 & 0.009 & 0.020 \\ 0.000 & 0.000 & 0.000 & 0.239 & 0.248 & 0.264 & 0.250 \end{pmatrix}$$

NMF symétrique

Un cas particulier et important de la **Tri-NMF** est le cas où l'entrée **X** représente une **matrice de similarités** par paires : $\mathbf{X} = \mathbf{X}^T = \mathbf{W}$.

$$X_+ \approx H_+ S_+ H_+^T$$



NMF symétrique

$X = X^T = W$ $\qquad X_+ \approx H_+ S_+ H_+^T$

Dans ce cas, **F=G=H**, nous optimisons la **NMF symétrique** :

$$\begin{cases} \{H,S\} = \arg\min_{H,S} D_F\left(X \| HSH^T\right) \\ \qquad = \arg\min_{F,S,G} \left\| X - H\ S\ H^T \right\|_F^2 \\ \\ s.c \quad H \geq 0, S \geq 0 \quad et \quad H^T H = I \end{cases}$$

Règles d'adaptation : Sym-NMF

Les règles d'adaptation dans le cas de la **Sym-NMF** se présentent de la façon suivante :

$$H_{ik} \leftarrow H_{ik}\left(1-\beta+\beta\frac{(XHS)_{ik}}{(HSH^{T}HS)_{ik}}\right)$$

$$0<\beta<1$$

$$S_{ik} \leftarrow S_{ik}\frac{(H^{T}XH)_{ik}}{(H^{T}HSH^{T}H)_{ik}}$$

Kernel NMF

Considérons une fonction de projection :

$$\mathbf{x_i} \rightarrow \phi\,(\mathbf{x_i}),\ \mathbf{ou}\ \mathbf{X} \rightarrow \phi\,(\mathbf{X}) = (\phi\,(\mathbf{x_1}), \cdots, \phi\,(\mathbf{x_n})).$$

Une **NMF** standard ou **Semi-NMF** comme $\phi\,(\mathbf{X}) \approx \mathbf{FG^T}$ seraient difficiles puisque **F**, **G** dépendraient explicitement de la fonction $\phi(\cdot)$.

Cependant, une **Convex-NMF** offre une meilleure possibilité :

$$\phi\left(X_{\pm}\right) \approx \phi\left(X_{\pm}\right) W_{+} G_{+}^{T}$$

Fonction de coût de Kernel-NMF

La norme matricielle de Frobenius :

$$
\begin{cases}
\{W,G\} = \arg\min_{W,G} D_F\left(\phi(X)\|\phi(X)WG\right) \\
\qquad = \arg\min_{W,G} \left\|\phi(X) - \phi(X)W\ G^T\right\|_F^2 \\
\qquad = Tr\left[\phi(X)^T\phi(X) - 2G^T\phi(X)^T\phi(X)W + W^T\phi(X)^T\phi(X)WG^TG\right] \\
\qquad = Tr\left[K - 2G^TKW + W^TKWG^TG\right] \\
avec\ K = \phi(X)^T\phi(X) \\
\\
s.c \quad W \geq 0, G \geq 0
\end{cases}
$$

Règles d'adaptation : Kernel-NMF

Les règles d'adaptation dans le cas de la **Kernel-NMF** se présentent de la façon suivante :

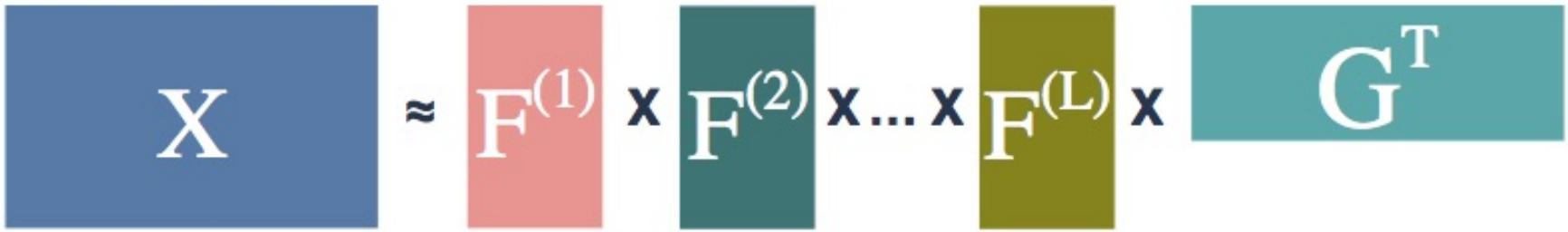
$$G_{ik} \leftarrow G_{ik} \sqrt{\frac{\left[\left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^+ W\right]_{ik} + \left[G W^T \left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^- W\right]_{ik}}{\left[\left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^- W\right]_{ik} + \left[G W^T \left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^+ W\right]_{ik}}}$$

$$W_{ik} \leftarrow W_{ik} \sqrt{\frac{\left[\left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^+ G\right]_{ik} + \left[\left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^- W G^T G\right]_{ik}}{\left[\left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^- G\right]_{ik} + \left[\left(\left\langle \phi(X)^T \phi(X) \right\rangle\right)^+ W G^T G\right]_{ik}}}$$

Multi-Layer-NMF

Dans **multi-layer-NMF** (NMF-multicouche) la matrice de base **F** est remplacée par un ensemble de matrices en cascade (facteurs). Ainsi, le modèle peut être décrit comme :

$$X \approx F^{(1)} F^{(2)} \dots F^{(L)} G^T$$



Multi-Layer-NMF

$$X \approx F^{(1)} F^{(2)} \ldots F^{(L)} G^T$$

$$\begin{cases} X \approx F^{(1)} G^{(1)T} \\ G^{(1)} \approx F^{(2)} G^{(2)T} \\ \ldots \\ G^{(L-1)} \approx F^{(L)} G^{(L)T} \end{cases} \longrightarrow \begin{cases} F = F^{(1)} F^{(2)} \ldots F^{(L)} \\ G = G^{(L)} \end{cases}$$

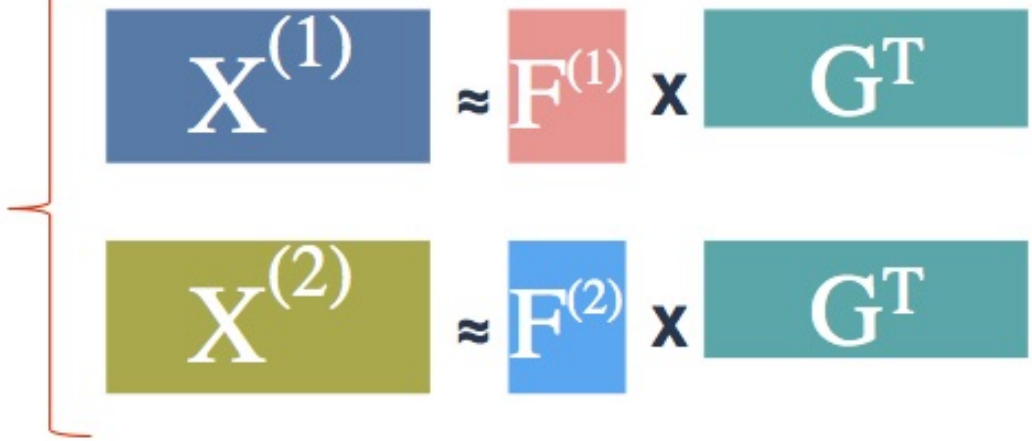
Le point clé de cette approche est que le processus d'apprentissage (mise à jour) pour trouver les paramètres des matrices $G^{(L)}$ est effectué séquentiellement, couche par couche, où chaque couche est initialisée de façon aléatoire avec des conditions initiales différentes.

On constate que l'approche multi-couche hiérarchique peut améliorer les performances de la plupart des algorithmes **NMF**.

Simultaneous-NMF

En **NMF** simultanée (**Si-NMF**), nous disposons de deux ou plusieurs matrices de données d'entrée liées (disons, $\mathbf{X}^{(1)}$ et $\mathbf{X}^{(2)}$) et l'objectif est de les décomposer en matrices de facteurs positifs ou nuls de telle sorte que l'un des facteurs soit commun :

$$\begin{cases} X^{(1)} \approx F^{(1)} G^T \\ X^{(2)} \approx F^{(2)} G^T \end{cases}$$



Convolutive-NMF

La **NMF Convolutive** est une extension naturelle et une généralisation de la **NMF** standard.
Dans la **NMF Convolutive**, on traite un ensemble de matrices ou de motifs non négatifs qui sont des versions horizontalement décalées (ou différées dans le temps) de la matrice **X** (notée G précédemment) :

$$Y = \sum_{p=0}^{P-1} A_p \overset{p\rightarrow}{X}$$

où **A** (notée F précédemment) est un ensemble de matrices de base non négatives inconnues et **X** est une matrice représentant des sources ou des motifs primaires.

avec $\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, $\overset{1\rightarrow}{\mathbf{X}} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 4 & 5 \end{bmatrix}$, $\overset{2\rightarrow}{\mathbf{X}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 4 \end{bmatrix}$, $\overset{\leftarrow 1}{\mathbf{X}} = \begin{bmatrix} 2 & 3 & 0 \\ 5 & 6 & 0 \end{bmatrix}$

Convolutive-NMF

leads to simple multiplicative update rules

$$a_{ijp} \leftarrow a_{ijp} \frac{\sum_{t=1}^{T} (y_{it}\ \hat{y}_{it}^{\beta-1})\ x_{j(t-p+1)}}{\sum_{t=1}^{T} \hat{y}_{it}^{\beta}\ x_{j(t-p+1)}}, \qquad x_{jt} \leftarrow x_{jt} \frac{\sum_{i=1}^{I}\sum_{p=0}^{P-1} a_{ik(p+1)}\ y_{i(t+p)}\ \hat{y}_{i(t+p)}^{(\beta-1)}}{\sum_{i=1}^{I}\sum_{p=0}^{P-1} a_{ik(p+1)}\ \hat{y}_{i(t+p)}^{\beta}},$$

for which the compact matrix form is:

$$\mathbf{A}_p \leftarrow \mathbf{A}_p \circledast \left(\bar{\mathbf{Y}}_\beta \overset{p\rightarrow}{\mathbf{X}}^{T}\right) \oslash \left(\hat{\mathbf{Y}}^{\cdot[\beta]} \overset{p\rightarrow}{\mathbf{X}}^{T}\right),$$

$$\mathbf{X} \leftarrow \mathbf{X} \circledast \left(\sum_{p=0}^{P-1} \mathbf{A}_p^T \overset{p\leftarrow}{\bar{\mathbf{Y}}}_\beta\right) \oslash \left(\sum_{p=0}^{P-1} \mathbf{A}_p^T \overset{p\leftarrow}{\hat{\mathbf{Y}}}^{\cdot[\beta]}\right),$$

where $\bar{\mathbf{Y}}_\beta = \mathbf{Y} \circledast \hat{\mathbf{Y}}^{\cdot[\beta-1]}$.

In practice, we adopt a slightly different procedure for estimation of $\mathbf{X}$

Multi-View-NMF

De nombreux ensembles de données dans le monde réel sont naturellement composés de différentes représentations ou vues.

La **NMF multi-vues** est une extension naturelle et une généralisation de la **NMF** standard :

$$\sum_{v=1}^{n_v} \left\| X^{(v)} - U^{(v)} \left(V^{(v)} \right)^T \right\|_F^2 + \sum_{v=1}^{n_v} \lambda_v \left\| V^{(v)} - V^* \right\|_F^2$$

où **V*** est le consensus des différentes matrices de coefficients.

Règles de mise à jour Multi-View-NMF

$$U_{i,k} \leftarrow U_{i,k} \frac{(XV)_{i,k} + \lambda_v \sum_{j=1}^{N} V_{j,k} V^*_{j,k}}{(UV^TV)_{i,k} + \lambda_v \sum_{i=1}^{M} U_{i,k} \sum_{j=1}^{N} V^2_{j,k}}$$

$$V_{j,k} \leftarrow V_{j,k} \frac{(X^TU)_{j,k} + \lambda_v V^*_{j,k}}{(VU^TU)_{j,k} + \lambda_v V_{j,k}}$$

$$V^* = \frac{\sum_{v=1}^{n_v} \lambda_v V^{(v)} Q^{(v)}}{\sum_{v=1}^{n_v} \lambda_v}$$

Algorithm 1 Multi-View NMF (MultiNMF)

Input: Nonnegative Matrix $\{X^{(1)}, X^{(2)}, \ldots, X^{(n_v)}\}$, parameters $\{\lambda_1, \lambda_2, \ldots, \lambda_{n_v}\}$

Output: Basis Matrices $\{U^{(1)}, U^{(2)}, \ldots, U^{(n_v)}\}$, Coefficient Matrices $\{V^{(1)}, V^{(2)}, \ldots, V^{(n_v)}\}$ and Consensus Matrix V^* [1]

```
Normalize each view X^(v) such that sum_j X_j^(v) = 1
Initialize U^(v), V^(v) and U* (1 ≤ v ≤ n_v)
repeat
    for v = 1 to n_v do
        repeat
            Fixing V* and V^(v), update U^(v) by Eq. 6
            Normalize U^(v) and V^(v) as in Eq. 7
            Fixing V* and U^(v), update V^(v) by Eq. 8
        until Eq. 5 converges.
    end for
    Fixing U^(v) and V^(v) (1 ≤ v ≤ n_v), update V* by Eq. 9.
until Eq. 4 converges.
```