Factorisation Matricielle Non-Négative pour la détection de faux billets

Objectif

Ce projet a pour but d'appliquer la NMF à la détection de faux billets. Il vous est demandé de produire un code-notebbok en Python-Jupiter. Vous pouvez utiliser pour certaines fonctions Python les ressources existantes sur Internet, en idiquant bien sur vos sources. Il existe plusieurs libraries comme par exemple :

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html (*).

1. Supposons que l'on possède un ensemble de photos où chacune d'entre elles correspond à un des billets (vrai ou faux). Après avoir appliqué la transformée en ondelettes aux images de billets, on obtient une matrice où chaque image est décrite par 4 caractèristiques: la variance, l'asymétrie, le kurtosis et l'entropie.

   (a) Sauvegardez cet ensemble de données dans deux variables : **bank_data** pour les données et **bank_labels** pour les étiquettes.

   (b) Créez deux matrices **A_faux** (que des faux billets) et **A_vrais** (que des vrais billets).

   (c) Ensuite, créez deux matrices **bank_train** (500 objets de **A_vrais** et 600 objets de **A_faux**) et **bank_test** (le reste).

   (d) Visualisez les données dans une figure séparée en utilisant une fonction de visualization de l'archive (*) par example fit transform.

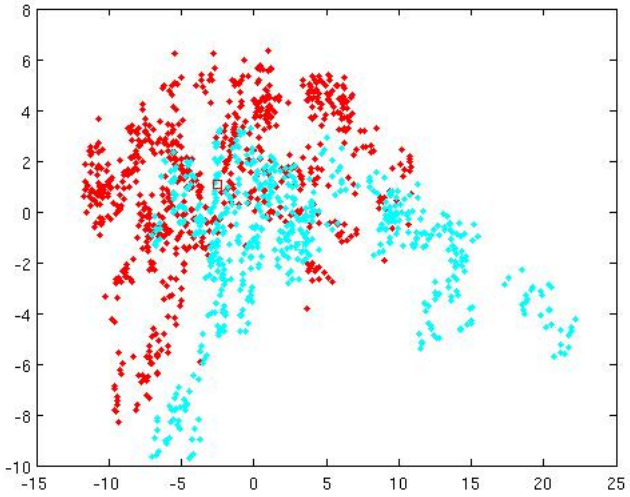

Figure 1: La partition initiale en 2D

2. Appliquez la Semi-NMF à la matrice **bank_train** en utilisant la fonction correspondante de l'archive (*) (example nmf). Sauvegardez la matrice de prototypes obtenue dans **W_train**.

   (a) Ecrivez un programme **show_clusters** pour transformer la matrice de partition obtenue à une vraie matrice de partition I (On cherche un élément maximal dans chaque ligne et on le remplace par 1. Tous les autres éléments sont remplacés par 0.). Calculez la pureté pour la matrice de partition obtenue précédemment.

   (b) Classifiez les objets sauvegardés dans **bank_test** en utilisant la matrice de prototypes **W_train** apprise précédemment (**H_test** = **W_train**$^{-1}$∗**bank_test**). Attention! Au cas où la matrice **W_train** est une matrice non carrée, on utilise le pseudo-inverse de Moore-Penrose (cherchez la commande correspondante dans scikit-learn).

   (c) Calculez les indices externes (la pureté et l'entropie pour la matrice de partition **H_test**). Pour cela, on utilise les commande **purity** et **entropy** de scikit-learn.

   (d) Calculez les indices internes (l'indice DB de Davies et Bouldin, l'indice CH de Calinsky et Harabsz, l'indice KL de Krzanowski

et Lai et l'indice de Dunn) pour la matrice de partition **H_test**. Pour cela, on utilise la commande corresponadante scikit-learn

(e) Visualisez les données avec les étiquettes .

3. Faites la séquence de commandes (2a)-(2e) avec la NMF classique.

4. Appliquez Symmetric NMF à la matrice **K_test**.

   (a) Calculez la matrice de Gram **K_test** en utilisant la commande corresapondante **kernelRBF** avec $\sigma = 1$.

   (b) Calculez les indices externes et internes pour la matrice de partition.

   (c) Visualisez les données avec les étiquettes obtenues ?

   (d) Faites la séquence de commandes (4a)-(4c) avec la matrice de Gram d'un noyau polynomiale avec les paramètres [1;0;2].

5. Comparez les résultats obtenus.