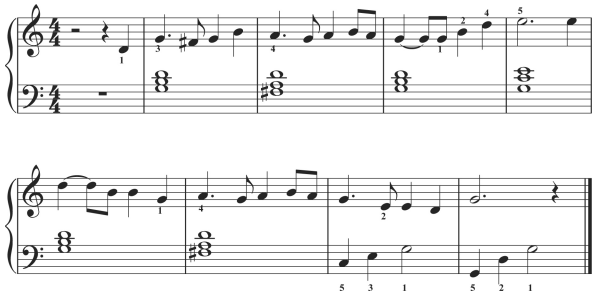
Factorisation matricielle non-négative pour la classification des instruments de musique



Objectif

Ce projet a pour but d'appliquer la NMF à l'analyse des instruments de musique. Supposons que l'on possède un ensemble de morceaux musicaux (en format *.wav) où chacun d'entre eux correspond à un des instruments de musique (piano, flûte, violon etc).

1. Créez une matrice dans laquelle chaque ligne correspond à un des morceaux musicaux.

   (a) Sauvegardez quelques examples de chaque instrument de musique dans les variable séparées en utilisant la fonction **wavread**. Combien de canaux audio est-ce qu'on retrouve dans chaque fichier?

   (b) Concatenez toutes les variables dans une seule matrice en utilisant la concatenation verticale [**variable_1;variable_2**].

   (c) Créez un vecteur des étiquettes **instr_labels**.

   (d) Visualisez dans une figure séparée les données qui correspondent à deux instruments différents en utilisant la fonction **plot**. Qu'est-ce qu'on peut en déduire?

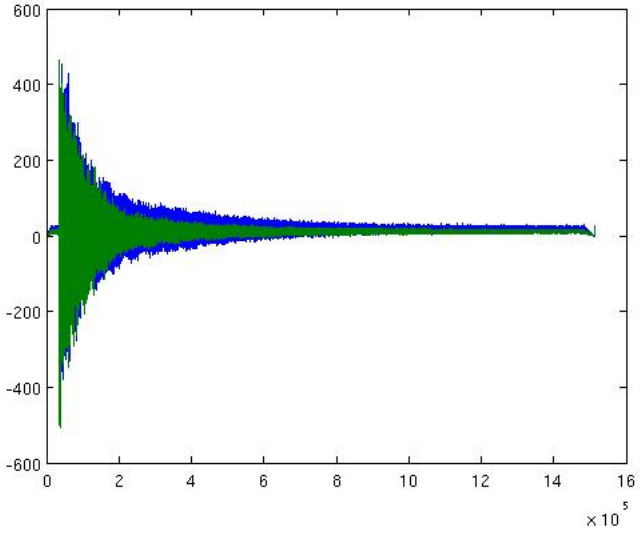


Figure 1: Un des fichiers qui correspond au piano

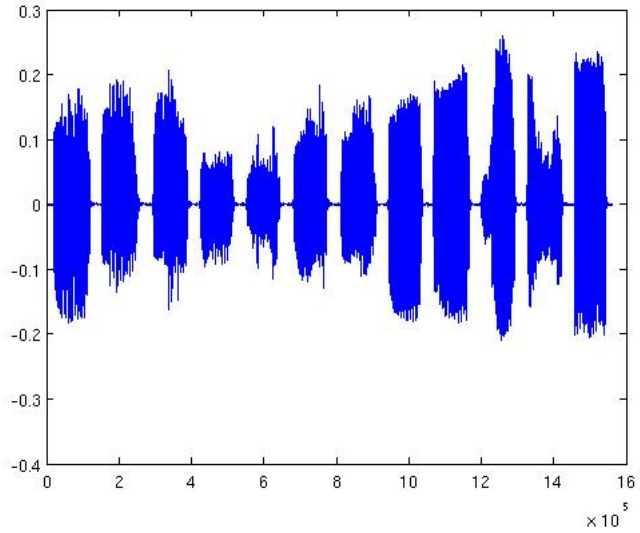


Figure 2: Un des fichiers qui correspond à la flûte

2. Pour appliquer la NMF à un ensemble de données en grandes dimensions on effectue la selection de "features".

(a) Calculez les caractristiques du format audio telles que: "energy entropy", "short time energy", "spectral rolloff", "spectral centroid" et "spectral flux" pour chaque fichier en utilisant la commande **computeAllStatistics**.

(b) Visualisez la vraie partition de données avec des étiquettes sauvegardées dans **instr_labels**. Utilisez l'ACP (commande **pcaproj**) pour réduire le nombre de dimensions à 3.

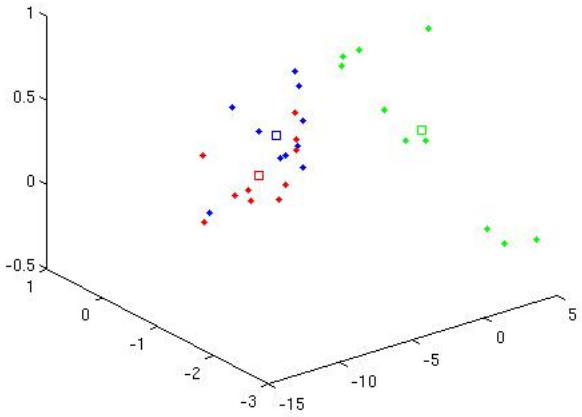


Figure 3: La partition initiale en 3D

(c) Appliques la NMF à la matrice obtenue précédemment en utilisant la fonction **nmfrule**. Interpretez les résultats de NMF.

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

(e) Quelle est la signification de ce remplacement?

(f) Calculez la pureté et la matrice de confusion en utilisant les matrices I et **instr_labels**.

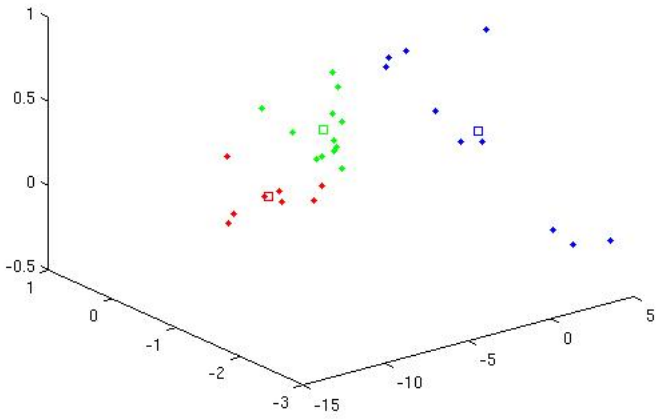


Figure 4: La partition obtenue en 3D

3. Analysez les résultats obtenus et faites les conclusions concernant des avantages et des inconvenients de NMF appliquée à la classification des instruments de musique.

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.

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 la fonction **PlotClusters**?

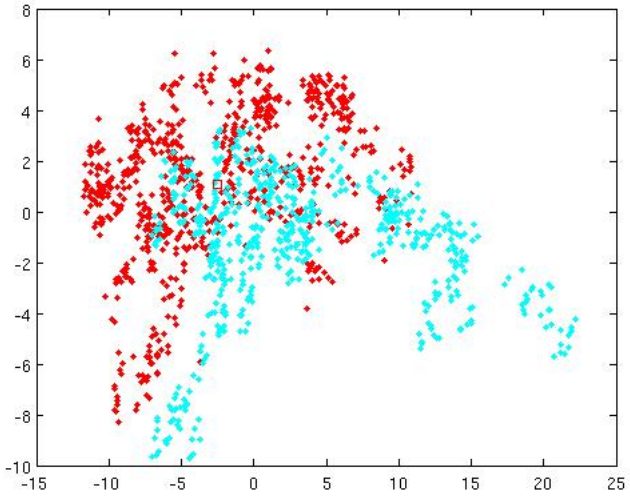


Figure 1: La partition initiale en 2D

2. Appliquez la Semi-NMF à la matrice **bank_train** en utilisant la fonction **seminmfnnls**. 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 (la commande **pinv**).

   (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**.

   (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 **valid_internal_deviation**.

(e) Visualisez les données avec les étiquettes obtenues en utilisant la fonction **PlotClusters**.

3. Faites la séquence de commandes (2a)-(2e) avec la NMF (commande **nmfrule**).

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

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

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

   (c) Visualisez les données avec les étiquettes obtenues en utilisant la fonction **PlotClusters**?

   (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.

Factorisation Matricielle Non-Négative pour la reconnaissance de visages

Objectif

Ce projet a pour but d'appliquer la NMF à l'analyse des images de Yale Faces Dataset.

1. Téléchargez la base de données Yale Database

   (http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html).

   On va travailler avec des images 32x32 (32x32 Data File).

2. Sauvegardez cet ensemble de données dans deux variables : **yale_data** pour les données et **yale_labels** pour les étiquettes.

3. Créez une fonction **show_face** qui visualise des images de toutes les personnes de telle façon qu'on ait 11 images par ligne (chaque ligne correspond aux photos d'une seule personne). Pour cela, on utilise les commandes **reshape**,**imagesc** et **colormap**.

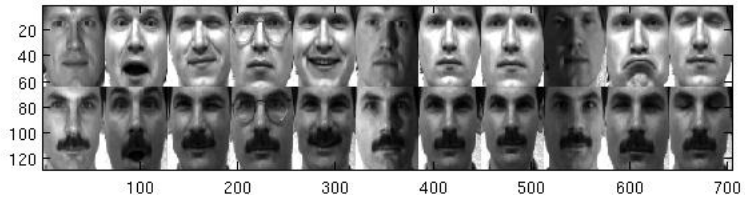


Figure 1: Exemple avec des photos de deux personnes

4. Appliquez PCA à la matrice **yale_data**.

   (a) Calculez les moyennes $\{M_i\}_{i=1}^{15}$ pour toutes les photos de chaque personne.

   (b) Créez une matrice A en soustrayant les moyennes calculées précédemment de chaque ligne de la matrice **yale_data**.
   (i.e. $A(:, 1:11) =$ **yale_data**(:,1:11) - M_1)

   (c) Utilisez la commande **pcaproj** pour calculer les vecteurs propres de **A**.

(d) Visualisez le résultat obtenu en utilisant la fonction **show_face**.

(e) Sauvegardez l'image obtenue dans un fichier **pca_eigenfaces.jpg**.

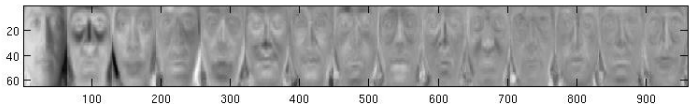


Figure 2: PCA "eigenfaces"

5. Appliquez NMF à la matrice **yale_data** en utilisant la commande **nmfrule**. Visualisez la matrice de prototypes obtenue en utilisant la fonction **show_face**.

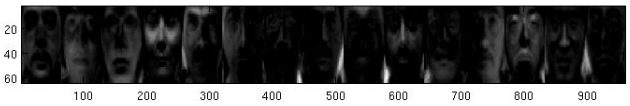


Figure 3: Vecteurs-prototypes obtenus par NMF

6. Appliquez Projective NMF à la matrice **yale_data**. Visualisez la matrice de prototypes obtenue en utilisant la fonction **show_face**.

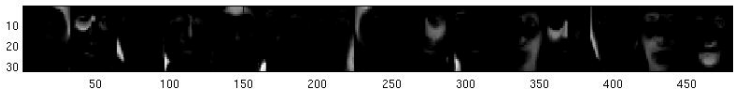
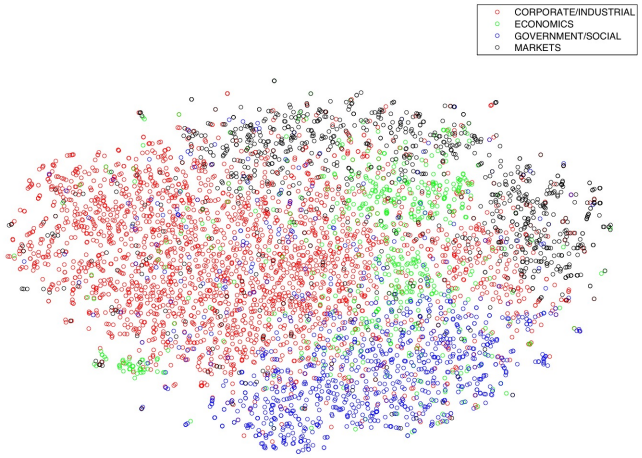


Figure 4: Vecteurs-prototypes obtenus par Projective NMF

7. Comparez les résultats obtenus. Que remarquez-vous?

Factorisation Matricielle Non-Négative pour le Text Mining



Objectif

Ce projet a pour but d'appliquer la NMF au Text Mining.

1. Supposons que l'on possède un ensemble des émissions de télévision Reuters. Après avoir appliqué la lemmatisation ("Porter stemming") et l'élimination des mots fréquents ("stop words"), on obtient une matrice où chaque ligne est décrite par 18933 caractèristiques: chaque caractèristique est le nombre d'apparition d'un terme dans une émission de télévision.

   (a) Téléchargez la base de données Reuters21578. (http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html).

   (b) Sauvegardez cet ensemble de données dans deux variables : **reuters_data** pour les données et **reuters_labels** pour les étiquettes. Utilisez

la fonction **tfidf** pour transformer les données au format "term frequency-inverse document frequency".

(c) Créez deux matrices **reuters4** (avec 400 objets de 4 premières classes(100 par classe)) et **labels4** (avec des étiquettes qui correspondent aux données sauvegardées dans reuters4).

(d) Ensuite, créez deux matrices **reut_train** (75 objets par classe) et **reut_test** (le reste). Sauvegardez galement les tiquettes pour les deux matrices.

(e) Ecrivez un programme qui retourne des indices de k éléments les plus fréquents dans des colonnes d'une matrice. Utilisez ce programme pour réduire le nombre de dimensions à 3.

(f) En utilisant la matrice réduite visualisez les données dans une figure séparée en utilisant la fonction **PlotClusters**?

2. Appliquez la NMF à la matrice **reut_train** en utilisant la fonction **nmfrule**. 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 **reut_test** en utilisant la matrice de prototypes **W_train** apprise précédemment (**H_test** = **W_train**$^{-1}$∗**reut_test**). Attention! Au cas où la matrice **W_train** est une matrice non carrée, on utilise le pseudo-inverse de Moore-Penrose (la commande **pinv**).

(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 **entropyCluster**.

(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 **valid_internal_deviation**.

(e) Visualisez les données avec les étiquettes obtenues en utilisant la fonction **PlotClusters**.

3. Appliquez la tri-NMF à la matrice **reuters4** en utilisant la fonction **orthnmfrule**.

   (a) Faites la séquence de commandes (2c)-(2e) pour les résultats retournés par **orthnmfrule**.

   (b) Analysez la matrice F. (La matrice F répresente les clusters de variables trouvés par la NMF Orthogonale)

   (c) Faites la séquence de commandes (3a)-(3b) en imposant les contraintes d'orthogonalité seulement à la matrice F.

   (d) Faites la séquence de commandes (3a)-(3b) en imposant les contraintes d'orthogonalité seulement à la matrice G.

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

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

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

   (c) Visualisez les données avec les étiquettes obtenues en utilisant la fonction **PlotClusters**?

   (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.