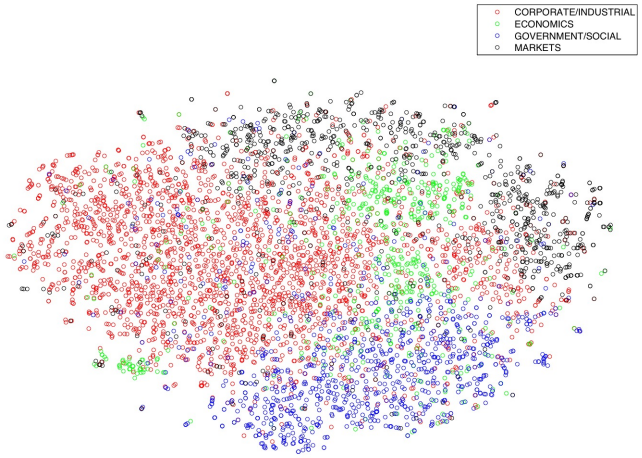
Factorisation Matricielle Non-Négative pour le Text Mining



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.