LOG6308 - TP 1

Grégoire Chapeaux - 2033122 — Hugo Canneddu - 2096673

15 Février 2021

Dans le cadre de ce TP nous avons cherché via diverses approches, basées sur des filtres collaboratifs, à prédire le vote d'utilisateurs sur des films grâce aux données de GroupLens.

1 Seuil de comparaison - 3 pts

Afin d'avoir un point de comparaison nous avons dans un premier temps effectué une approche naïve basée sur la moyenne : pour effectuer une prédiction, on calcule la moyenne des votes de l'utilisateur, la moyenne des votes pour l'item, et on effectue la moyenne des deux.

On valide l'approche par une validation croisée avec 10 replis, qui nous donne une erreur quadratique moyenne de **0.96** pour cette approche.

2 Item-item - 11 pts

On applique ici une approche item-item, avec pour fonction de distance et pour fonction de similarité le cosinus. On va analyser certains résultats intermédiaires :

2.1 Distribution des similarités et proportion de poids nuls

On obtient, en calculant les similarités entre un item du dataset et ses 10 voisins les plus proches la distribution de similarités suivante :

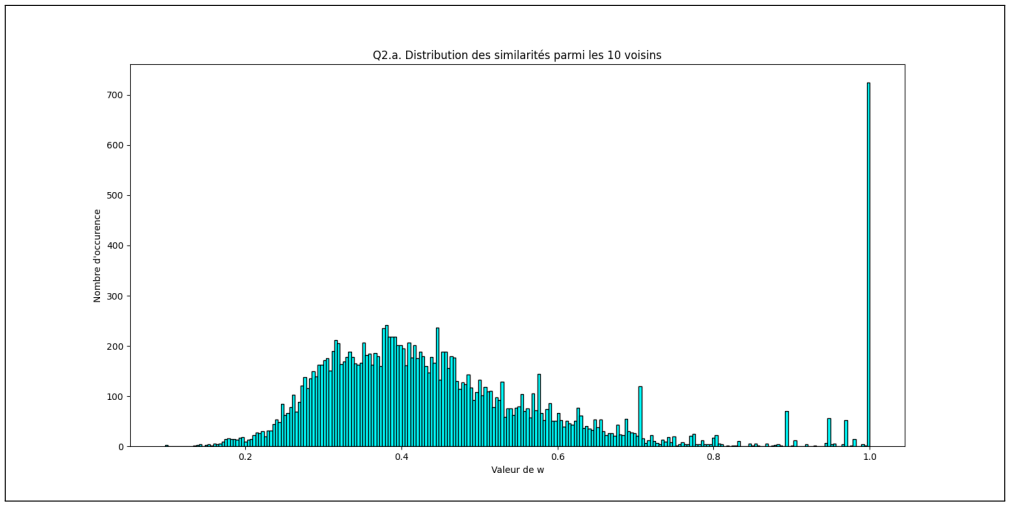

Figure 1: Distribution des similarités pour l'approche item-item

Nous remarquons un pic de valeur pour w = 1 : ce pic est du à un certain nombre de films qui n'ont eu que très peu de votes, donnés à chaque fois par un même utilisateur ce qui entraîne une similarité de 1 entre ces deux item.
Par ailleurs, on peut mesurer la proportion de poids nuls dans cette situation : en considérant tous les items voisins, chaque item a en moyenne une similarité nulle avec **30.4%** des autres items, mais on observer une proportion de poids nuls avec les 10 plus proches voisins de **0%**. En d'autres termes, on constate que si l'on choisit les 10 plus proches voisins de chaque item, on n'aura jamais de voisin avec une similarité strictement nulle.

2.2 Nombre de voisin avec votes communs et proportion de votes manquants

On calcule, pour chaque item, le nombre de voisins avec votes communs : en d'autres termes, pour un item donné, on cherche le nombre d'items présentant le vote d'au moins un utilisateur ayant voté pour l'item courant. On reporte dans la figure suivante la distribution de ce nombre de voisins avec vote commun :

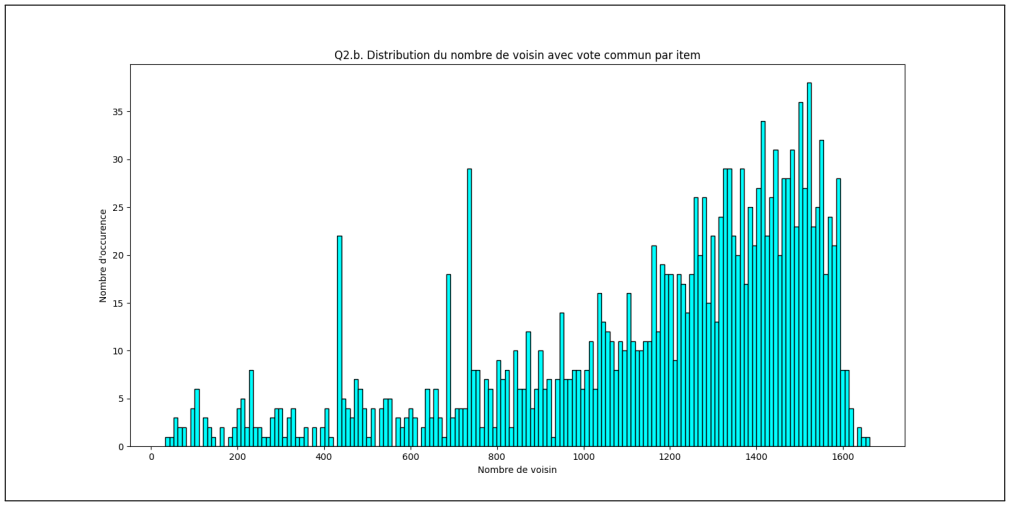

Figure 2: Distribution du nombre de voisin avec votes communs par item

Par ailleurs, on calcule la proportion de votes manquants, soit, pour une prédiction donnée, la proportion moyenne de voisins de l'item concerné qui n'a pas de vote de l'utilisateur en question. Pour cela, on itère sur toutes les prédictions, on calcule le nombre de voisins de l'item (nombre d'items ayant un vote en commun), on calcule le nombre d'items pour lesquels l'utilisateur a voté (qui sont donc, a priori, des voisins de l'item), et on obtient donc la proportion d'items voisins ayant un vote de l'utilisateur en commun, et donc a fortiori la proportion d'items voisins n'ayant pas de vote de l'utilisateur en commun : on trouve ainsi que **87.5 %** des voisins en moyenne sont perdus lors de la prédiction.
Pour résumer, on n'aura en moyenne que 12.5 % des voisins qui seront conservés lors d'une prédiction. Si l'on veut considérer 10 valeurs pour notre calcul de prédiction, il nous faut donc pour chaque item un total de $\frac{10}{12.5\%}$ soit 80 voisins par item. Le graph de distribution nous dit que dans la majeure partie des cas, cette condition sera vérifiée (9 items ont moins de 80 voisins), il est donc pertinent d'effectuer une approche par validation croisée en considérant les 10 plus proches voisins.

2.3 Erreur quadratique moyenne

On peut désormais utiliser l'approche item-item avec les 10 voisins les plus proches pour effectuer des prédictions de vote. Pour les votes disposant déjà d'une entrée, on va calculer l'erreur quadratique moyenne de notre prédiction, et l'on va récupérer la moyenne par item de ces prédictions. La distribution de cette erreur quadratique moyenne par item est représentée dans la figure suivante :

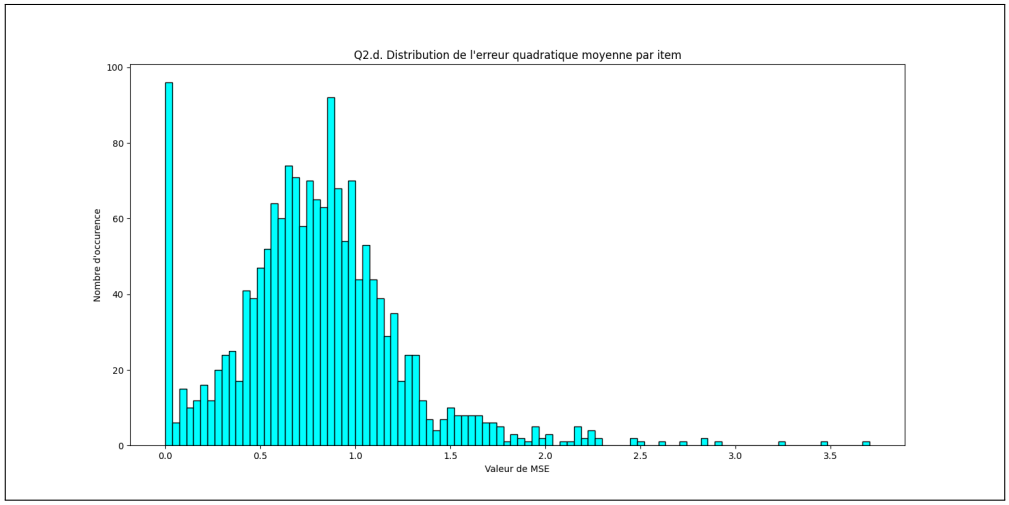

Figure 3: Distribution de l'erreur quadratique moyenne par item

Finalement, on arrive donc à une erreur quadratique moyenne de l'approche égale à **0.800**, bien meilleure que l'approche naïve de la partie 1.

3 Approche SVD - 5 pts

On effectue une décomposition SVD de la matrice des votes. Pour cela, il va tout d'abord falloir remplir les cases de la matrice de vote correspondant aux données manquantes : on va s'inspirer de la question 1, en imputant pour chaque vote manquant la moyenne des moyennes de l'utilisateur et de l'item concernés. Nous allons par ailleurs choisir cette grandeur comme biais des votes, et la soustraire avant décomposition SVD puis la rajouter à la matrice décomposée.
L'implémentation de cette approche est inspirée de l'article **Beginner's Guide to Creating the SVD Recommender System** *(https://towardsdatascience.com/beginners-guide-to-creating-an-svd-recommender-system-1fd7326d1f65)*

On va chercher de manière empirique le nombre de dimensions à conserver : pour cela, on va effectuer un test par validation croisée avec 5 replis, pour un nombre k de dimensions conservées variant de 5 à 20. On reporte les observations dans le graph suivant :

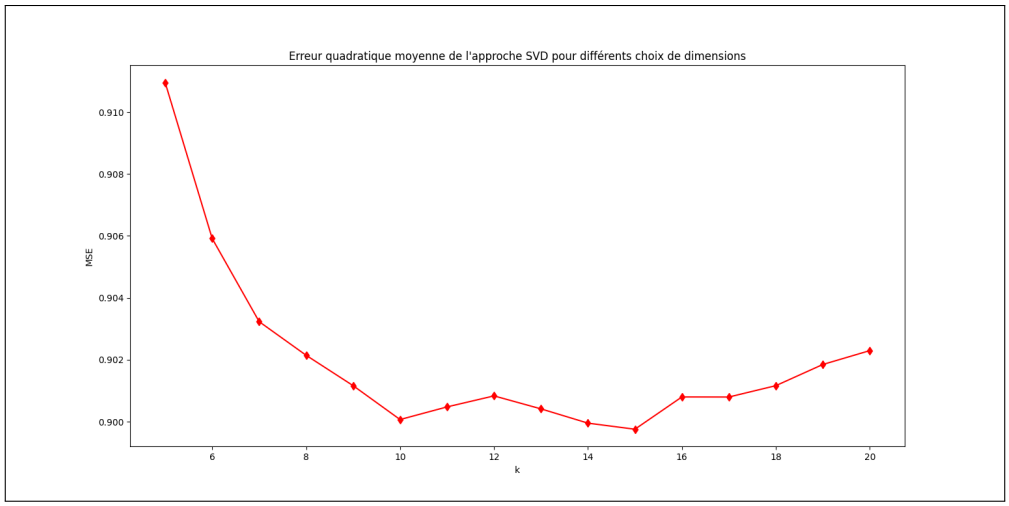

Figure 4: Erreur quadratique moyenne relativement au nombre de dimensions conservées pour SVD

On va ainsi choisir de conserver les **15 premières dimensions**, et on a alors une erreur quadratique moyenne de **0.902** pour ce choix de 15 facteurs latents, donc une performance meilleure que notre seuil.

4 Approche par agglomération - 5 pts

Dans cette partie nous avons cherché à prédire les votes en se basant sur une approche par agglomération. Afin de pouvoir calculer la MSE nous avons effectué une validation croisée de 5 replis.

Dans un premier temps nous avons calculé la matrice de corrélation qui nous a ensuite permis de déterminer nos clusters grâce à l'algorithme K-means avec différentes valeurs de nombre de classes qui sont : 5, 10, 20, 40,80. Une fois les clusters déterminés nous avons pu calculer les moyennes par item et par cluster, pour ensuite pouvoir effectuer nos prédictions. Dans l'état actuel la meilleure mse obtenue était de 1,08 avec 5 clusters. Cependant dans chaque cluster nous n'avons pas forcément de données pour tous les items ce qui entraîne des valeurs manquantes pour les prédictions, qui sont alors ignorées.

Afin de pallier ce problème nous avons dans un premier temps remplacé les valeurs manquantes par la moyenne générale de l'item. Malheureusement cette méthode n'a pas donné de très bon résultat, la meilleure MSE obtenue était alors de 1,13 avec 5 clusters.

Ensuite, nous avons mis en place une deuxième approche. Elle consiste à calculer à l'avance l'ensemble des valeurs de notre matrice de votes, en utilisant la même approche que dans la question 1. Une fois le calcul des clusters effectué, nous avons pu calculer la moyenne pour chaque item, en se basant sur notre matrice de votes pleine et non la matrice de votes creuse. Grâce à cela nous obtenons bien dans chaque cluster une moyenne pour chaque item. De plus c'est grâce à cette approche que nous avons obtenu notre meilleur résultat qui est une **MSE de 1.047 pour 20 clusters**.

Il est à noter qu'en fonction de la seed des fonctions aléatoires utilisées la meilleure valeur obtenue varie entre 20 et 40 clusters.

Clusters	5	10	20	40	80
MSE	1.051	1.049	1.047	1.051	1.056

Table 1: MSE en fonction du nombre de clusters

Nous remarquons cependant que cette approche ne donne pas de meilleurs résultat que celui obtenu dans la question 1. Il n'est donc pas intéressant de faire une approche par agglomération qui est bien plus longue en temps de calcul.